Télécom Paris (ENST)
Institut Eurécom

THÈSE

Pour Obtenir le Grade de Docteur

de l'Ecole Nationale Supérieure

des Télécommunications

Spécialité: **Réseaux et Informatique**

Shiyi WU

Protocoles de diffusion dans les Réseaux Ad Hoc sans Fil

Date de Soutenance: 13/10/2004
Composition du Jury:

	Président
Prof. Andrzej Duda	Rapporteur
Prof. Eric Fleury	Rapporteur
Prof. Guy Pujolle	Examinateur
Prof. Gwendal Le Grand	Examinateur
Doct. Stéphane Amarger	Examinateur
Prof. Christian Bonnet	Directeur de Thèse

Acknowledgment

First and foremost, I would like to thank my supervisor Christian Bonnet for his encouragement and support over the past years. He gave me a lot of freedom with my research as well as the possibility to present my work in several international conferences.

I am grateful to the members of my doctoral committee for their time, support and useful comments.

My work with Christian Bonnet, Navid Nikaein, Houda Labio, Masato Hayashi, and Yukihiro Takatani gave me the chance to gain a valuable experience.

I also want to thank all the staff in the Mobile Communications Department and Institut Eurécom for their warm reception. We have spent so much wonderful time together. I will never forget them.

I owe a special gratitude to my family. I want to thank my sister, my father and mother who support me all the time.

Finally, my wife Liu Xiang gave me her infinite support and love. I am grateful to her and my son Wu Haotian for enriching my life.

Abstract

The advances in portable computing and wireless technologies are opening up exciting possibilities for the future of wireless mobile networking. A Mobile Ad hoc NETwork (MANET) is a collection of wireless mobile nodes forming dynamical and temporary network without the use of any existing network infrastructure or centralized administration. Its capability of providing rapidly deployable communication makes it an ideal choice for consumer, company and public uses. Most applications are characterized by a close degree of collaboration. Multicasting could prove to be an efficient way of providing necessary services for these kinds of applications. However, due to the limited transmission range of wireless network interfaces, multiple network "hops" may be needed for one node to exchange data with another one across the network. Consequently, the extra challenges such as frequent topology change and limited network resources are introduced in multicasting protocol design.

In this dissertation, we first examine different techniques and strategies which are used by current multicast routing protocols for MANETs. Then, we present our proposition, multicast routing protocol with dynamic core (MRDC), to provide best effort multicast routing. This protocol addresses the issue of how to optimize packet delivery success rate and overhead. This protocol gives a trade-off between forwarding overhead and routing overhead but also an optimization between delivery success rate and overhead regarding application requirement and network situation. For the applications which require 100% percent packet delivery, we study a reliable multicasting protocol which activates intermediate nodes to assist retransmission. All these works have the same goal: optimize packet delivery ratio and overhead to satisfy application requirement with good utilization of network resources especially bandwidth.

This dissertation also includes our experiences in implementing a mobile ad hoc testbed. We developed this testbed by implementation of DDR, a unicast routing protocol and MRDC so that the testbed can support both one-to-one communications and many-to-many communications. This testbed will allow us to analyze the performance of routing protocols in real network. It can also be used to study protocols and new MANET applications.

Résumé

Les avancées dans le domaine de l'informatique personnelle et des technologies sans fil ouvrent des possibilités passionnantes pour le futur de la gestion des réseaux mobiles. Les réseaux mobiles "ad-hoc" sont créés par un ensemble de terminaux sans fil qui communiquent entre eux. Les nœuds d'un réseau "ad hoc" forment dynamiquement un réseau à façon sans utilisation de quelconque infrastructure existante ou administration centralisée. Ses capacités à fournir rapidement et flexiblement des moyens de communication font des réseaux "ad hoc" un choix idéal pour certaines applications personnelles, publiques ou d'entreprise. Beaucoup de ces applications sont caractérisées par un degré étroit de collaboration. Le multicast peut s'avérer être une manière efficace de fournir les services nécessaires pour ce genre d'application. En raison de la limitation de la couverture radio de l'interface sans fil, le relayage par sauts multiples peut être nècessaire pour qu'un nœud puisse échanger des données avec les autres à travers le réseau. En conséquence, les défis supplémentaires tels que le changement fréquent de topologie et les ressources limitées de réseau sont à relever dans la conception de protocole multicast.

Dans cette dissertation, nous examinons d'abord les techniques qui sont employées par des protocoles courants de routage de multicast. Ensuite, nous présentons en détail notre proposition, Multicast routing protocol with dynamic core (MRDC), pour fournir un routage de multicast de "best effort". Ce protocole adresse le problème de comment optimiser le taux de succès de la livraison de paquets tout en réduisant le coût de signalisation du protocole. Il donne une compromis entre les surcharges liées au routage et les surcharges de transmission mais également une optimisation entre le taux de succès de la livraison et les surcharges en regardant des exigences des applications et les conditions du réseau. En outre, pour les applications qui exigent la livraison fiable de paquets (cent pour cent de réussite), nous proposons un protocole fiable de multicast, Active Reliable Multicast Protocol with Intermediate node support (ARMPIS), qui active des nœuds intermédiaires pour aider les retransmissions. Tous ces travaux ont le même but : optimiser le taux de livraison de paquets pour répondre aux exigences des applications avec la bonne utilisation des ressources du réseau et notamment la bande de passante.

Cette dissertation inclut également notre experience de la construction et de la validation d'un banc de test de réseau ad-hoc. Nous avons développé ce banc de test par l'implementation de DDR, d'un protocole de routage d'unicast et de MRDC de sorte que le banc de test puisse supporter des communications point-à-point et aussi des communications multipoint. Ce banc de test nous permettra d'analyser les performances de MRDC dans un vrai réseau. Il peut également être employé pour étudier des protocoles et de nouvelles applications de réseaux ad hoc san fil.

Contents

List of Figures

List of Tables

Glossary

ACK	Positive acknowledgment
ABAM	Associativity-Based Ad hoc Multicast
ADMR	Adaptive Demand-driven Multicast Routing
AMRIS	Ad hoc Multicast Routing protocol utilizing Increasing id-numberS
AMRoute	Ad hoc Multicast Routing protocol
API	Application Interface
AQL	Average Queue Length
ARQ	Automatic Repeat Request
BGMRP	Border Gateway Multicast Routing Protocol
CAMP	Core-Assisted Mesh Protocol
CBR	Constant Bit Rate
CBT	Core-Based Tree
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DCF	Distributed Coordination Function
DDM	Differential Destination Multicast
DVMRP	Distance Vector Multicast Routing Protocol
FGMP-RA	Forwarding Group Multicast Protocol - Receiver Advertising
FGMP-SA	Forwarding Group Multicast Protocol - Sender Advertising
ID	Identifier
ICMP	Internet Control Message Protocol
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAM	Lightweight Adaptive Multicast Algorithm
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
MAODV	Multicast operation of Ad-hoc On-demand Distance Vector routing protocol
MCEDAR	Multicast Core Extracti on Distributed Ad hoc Routing
MOR	Medium Occupation for Reception
MOSPF	Multicast Open Shortest Path First
MRDC	Multicast Routing Protocol with Dynamic Core
NACK	Negative Acknowledgment
NSMP	Neighbor Supporting ad hoc Multicast routing Protocol
ODMRP	On-Demand Multicast Routing Protocol
OSI	Open System Interconnection
PIM-DM	Protocol-Independent Multicast - Dense Mode
PIM-SM	Protocol-Independent Multicast - Sparse Mode
RGMP	Receiver-initiated Group-membership Protocol
RTS/CTS	Request to Send/Clear to Send
TCP	Transport Control Protocol
TTL	Time To Live
UDP	User Datagram Protocol

Chapter 1

Introduction

The advances in wireless communication and economical, portable computing devices have made mobile computing possible. The mobile terminal (Portable PC, PDA) as well as the mobile telephone becomes ubiquitous not only in business scene but also in our daily life. People require more from the communications network. Communications with anybody anytime anywhere in whatever forms - data, voice or video - is being envisaged, and automatic roaming of hosts in the network also seems not so “distant”. However, the liberty of communication is limited by the network infrastructure. Whenever anyone wants to send information to someone else, he must use network infrastructure. If any entity of a communication pair is out of the coverage range of the network infrastructure, the communication cannot be established even if they are in face-to-face distance from each other. However, seamless coverage is expensive and sometimes impossible. With the above drawkback, one research field to cope with this issue has recently attracted more and more interest. It concerns the design of mobile ad hoc networks (MANET)[1]. A MANET is an autonomous system formed by a collection of mobile nodes equipped with wireless interface(s). These nodes communicate with each other without the intervention of any existing network infrastructure or centralized administration. In such a network, each node acts as a host, and may act as a router if it volunteers to carry traffic. If two nodes are out of their radio transmission range, the network is able to establish communication between them with the help of some other nodes. As a consequency the literature sometimes uses the term “multihop networks“ for MANETs.

MANETs, having self-organizing capability, can provide rapidly deployable communication in the place where network connectivity is not attainable or expensive. They are considered suitable systems to support a number of applications [2]. Some of them are listed below and more can be found in [3].

- Virtual classrooms,
- Military communications,
- Emergency search and rescue operations,

- Data acquisition in hostile environments,
- Audio/Video conference,
- File distribution and
- Internet games etc.

These applications cover consumer uses, company uses and also public uses. Gaming is one of the typical applications for consumer usage where two or several players gather and form a game group somewhere. Email and file transfer are also considered easily deployable within an ad hoc network environment for personal or business uses. There is no need to emphasize the wide range of military and public applications possible with ad hoc networks since the technology was initially developed with them in mind, such as emergency rescue operations after an earthquake wherein existing network infrastructures are completely destroyed.

By analyzing the properties of the potential applications of MANETs, [3] indicates that most of the examples mentioned in that paper include one-to-one, one-to-many and many-to-many communication model. For example, an Internet game can contain only two players (chess) or several players (card games). In the former case, game information has a unique destination - the other player. In the later case, any player is a potential information sender which sends game information to all the other players in which case many-to-many communication is required. This kind of application can also be characterized as group oriented since information exchange takes place among a group of users. One-to-one communication is well studied through unicasting all kinds of networks including wired, wireless and ad hoc networks. However, aiming to support one-to-many and many-to-many communication, which requires transmitting packet(s) to multipoint, the research is far from being accomplished in mobile environment.

In general, networks have three methods to transmit a packet addressed to multiple receivers: unicasting, broadcast and multicasting. In unicast, the sender duplicates the packet and sends separately a copy to every receiver. The same copy will appear on some links. In broadcast, the sender and intermediate nodes send a copy to each outgoing link. As a result, even nodes that do not require a copy will get the packet. Multicasting can efficiently support them. In multicast, the sender makes only one transmission. At each intermediate node, copies are made and sent to outgoing links as required. At most, one copy is required on each physical link. Therefore, multicasting is able to deliver packets to multipoint in an efficient and scalable way, which is more important in MANET where bandwidth is scarce. A protocol is called scalable if it works efficiently even as the size of the network increases. Without any surprise, current multicasting protocols for MANET are inspired by Internet.

The first Internet multicast paradigm, the “host group” model, was proposed in 1989 [4]. In this model, a single class D IP address identifies the hosts participating in the same multicast session form a host group . A host may join and

leave the group at any time and may belong to more than one group at a time. To send datagrams to a group, a host does not need to know the membership of the group, or be a member of the group. Data delivery in the host group model is best effort. Senders multicast to and receivers receive from their local links and it is the multicast routers that have the responsibility of delivering the multicast datagrams. The Internet multicast architecture is largely evolved from this model. It consists of the group management protocols, the IP multicast routing protocols, and multicast transport protocols. The group management protocols (IGMP [5], [6], [7] and RGMP [8]) are used for group member hosts to report their group information to the multicast routers on the subnet. Multicast routing protocols on the Internet deal with the problem of efficiently transmitting multicast datagrams from the source(s) to the destinations. Although multicast routing protocols provide best effort delivery of multicast datagrams on the Internet, many multicast applications have requirements beyond this. Therefore, various multicast transport protocols are proposed on top of the multicast routing protocols to meet the needs of different applications. Multicast transport protocols serve two major functions, namely, providing reliability and performing flow and congestion control.

In ad hoc network environment, the "host group" model should be slightly modified since nodes act as host and router at the same time. The router to which a group member host should report its group information is probably the node itself. And it is the nodes themselves that have the responsibility of delivering the multicast datagrams. That makes multicast routing a more important issue on providing multicasting for ad hoc networks. This dissertation focuses on the problem of multicast routing in mobile ad hoc networks

1.1 Motivation and Objectives

Multicast routing protocols have twin design goals: high delivery success rate and low overhead. The former is important because it is the principle aim of multicasting - transmitting packets to their receivers. The overhead is very critical since the efficient utilization of network resources is concerned. In fact, there are two kinds of overhead in multicasting: control overhead and forwarding overhead. Control overhead is generated during routing information collection and update. Forwarding overhead is the result of delivering multicast packets. In the Internet, the network topology is usually stable, few control messages are needed to update routing information. Furthermore, the lower layer protocols in use in the network can efficiently support multicast transmission. These factors permit researchers to focus their interest on the problem of reducing forwarding overhead. They use either a source-based tree or group-shared tree as routing structure to provide best effort multicast delivery. Multicast tree contains the unique path from source(s) to receivers to guarantee transmission efficiency. As for the problem of packet loss during transmission, which is usually caused by congestion, researchers prefer to resolve it in transport protocols. The transport protocols make congestion control in

order to reduce packet loss or retransmit lost packets to provide delivery guarantee. However, providing multicasting in mobile ad hoc networks is a more challenge task than that in wired networks due to frequent changes in network topology as well as the nature of the network wireless interface. While, the frequent changes in network topology implies a short validation time of routing information, wireless nature of the interface implies the limited bandwidth capacity. Thus, mobile and wireless environments exhibit opposite requirements. On one hand, node's mobility requires a high degree of routing information updates in order to maintain connectivities among senders and receivers. On the other hand, the wireless medium has low capacity and hence cannot be used for additional control traffic that is needed to continually update stale information. These properties make multicast routing protocols for Internet not adapt for MANET environment and introduce special challenges in multicasting protocol designing. Facing the challenges of bandwidth limitation and frequent topology changes, most researchers study how to reduce control overhead to maintain the connection of multicast routing structure in MANET environment. They usually reach their goals in three ways: (a) Try to limit the effect of topology change in a small range by doing local repair, (b) Introduce more routes into the routing structure in order to make the structure robust against topology changes, (c) Reduce as many as possible the routing information which should be maintained for multicast routing. Based on these ideas, several multicast routing protocols are proposed recently.

However, multicasting is far from being well established for MANET. First, the principle idea of these three ways is to reduce the control overhead with the cost of transmission efficiency. However, we cannot consecrate too much transmission efficiency to reduce control overhead. An extreme example is to deliver multicast packets by flooding them in the network. This method does not generate any control overhead, but produces an important forwarding overhead. Therefore, a certain trade-off should be found between control overhead and forwarding overhead in order to reduce total overhead for multicast routing. Another important issue that is usually ignored during multicast routing protocol designing is the cooperation with the other layers. For example, if the MAC layer protocol in use cannot efficiently support multicast transmission, there will be significant packets lost during delivery. Bad network situation can worsen this problem. In the ad hoc networking scenario, it is difficult to maximize delivery success rate and minimize overhead simultaneously. Thus, there is also a trade-off between them. From the applications point of view, they do not always require the maximum delivery success rate. Some applications can tolerate a certain degree of packet loss but require short transmission delay while some others are not sensitive to transmission delay but do expect that receivers can receive as many as possible packets. Different strategies can be applied according to the diverse requirement of these applications in order to find the trade-off between multicast delivery and total overhead. Thus, we give another definition of "best effort" multicasting in MANET: to deliver multicast packets with the lowest cost and as close as possible to the application's requirement.

Therefore, our research goal is to design a best-effort multicast routing protocol. This protocol is able to provide an optimal trade-off between efficient delivery and total overhead while at the same time meet the requirements of most multicast applications. Considering that some applications of MANET do require a guarantee of transmitting packets to receivers, we also discuss the reliable multicasting in MANET. We believe that with these efficient multicasting protocols, it will be possible to realize a number of envisioned group-oriented applications in MANET environment.

1.2 Structure of Dissertation

This Dissertation focuses on providing multicasting in mobile ad hoc network in an efficient way. Multicasting is divided into two sub issues: delivery structure management and multicast packet forwarding. The remainder of this dissertation is organized as follows.

Chapter 2 provides an overview of providing multicasting in MANET. It includes the design challenges and different techniques aiming to resolve these problems. We also present our contribution and compare it with other well-known multicast routing protocols.

Chapter 3 describes our multicast routing protocol with dynamic core (or MRDC in abbreviation) in detail. This protocol contains two plans: control plan and forwarding plan. We introduce our strategy to construct and maintain a multicast tree on traffic demand in control plan. The root of a multicast tree (also called core) may change during the multicast session to adapte to network situations. The forwarding plan delivers multicast packets in a *best-effort* way. Considering that IEEE802.11 DCF, a widely used MAC layer protocol in MANET, does not transmit multicast packets in a suitable way, we integrate our solution in the forwarding plan. This solution chooses a suitable transmission method to deliver multicast packets by taking network situation and application requirement into account.

Chapter 4 analyzes the performance of MRDC in a network simulator ns2 [9]. The simulation first tests the correctness and robustness of this delivery structure under different environment (group configuration, network load, node's mobility). We then demonstrate that our multicast forwarding mechanism is adaptive and can achieve better performance in low loaded networks.

Chapter 5 presents our scalable reliable multicasting protocol to aiming to offer multicast delivery guarantee in MANET. This protocol distributes multicast packet cache and retransmission tasks among intermediate nodes that overhear multicast packets. Simulation results show that our reliable multicasting protocol has a packet delivery rate close to 100% and maintains a low bandwidth consumption facing frequent topology change.

Chapter 6 introduces our experiences in implementation an ad hoc testbed. We implemented not only MRDC but also a unicast routing protocol in the testbed. This testbed allow us to evaluate the performance of our multicasting protocol in

real world. It can also be employed to concept new MANET applications since it can support one to one and many to many communications.

Finally in chapter 7, we outline remarks and summaries of our research work and contribution. We discuss the general experience learned about on designing multicasting protocols for mobile ad hoc networks and outline some directions for future research.

Chapter 2

Study of Multicast Routing Protocols for Ad Hoc Network

Multicasting is the transmission of datagrams to a group of hosts identified by a single destination address [10]. To realize this mode of transmission, three types of protocols are needed. They are group management protocols, the IP multicast routing protocols and multicast transport protocols. Since the first Internet multicast paradigm appeared in the later 1980's, numerous multicast routing protocols were well designed to offer efficient multicasting service in conventional wired networks. These multicast routing protocols, having been designed for stationary networks, may be unsuitable for mobile ad hoc networks due to the special properties of MANETs. Facing the design challenges, several multicasting routing protocols are proposed during the last few years. These protocols coped with group membership dynamic and topology dynamic by using diverse strategies and techniques. One thing is clear that none of them is suitable in all cases. Therefore, before designing a new multicast routing protocol for MANET, it is important to:

1. Understand the properties of MANETs,
2. Study the common design challenges of providing multicast in networks and the dedicate challenges in MANET,
3. Analyze the advantage and the inconvenience of each strategy and technique and
4. Choose suitable strategies and techniques.

This chapter covers these above steps.

2.1 Issues in providing multicasting in MANETs

In this section, we first outline the specific properties of mobile ad hoc networks. After a short discussion of the multicast routing protocols for Internet, we list the design challenges in providing multicast routing for this type of networks.

2.1.1 Mobile Ad hoc Network properties

Compared to other networks, ad hoc networks have the following special features that need to be addressed for routing protocol designing:

Infrastructureless - There is no fixed backbone infrastructure for network management and packet relay. Therefore, mobile nodes become the potential network infrastructure and they must act cooperatively to handle network functions. This property yields some other special properties of ad hoc networks such as frequent topology change and limited resources.

Frequent topology change - The network topology may change randomly and rapidly over time since nodes are free to move in an arbitrary manner. Furthermore, networks may be partitioned and merged from time to time because of node's movement and environment change.

Wireless communication - Nodes use wireless interface to communicate. Wireless communication implies limited bandwidth capacity in comparison with wired communication, and differs from those by the respect that electromagnetic waves propagate in free air instead of inside cables. Many issues, which emerge from this fact such as multipath, pathloss, attenuation, shadowing, noise and interference on the channel, make the radio channel a hostile medium whose behavior is difficult to predict. Therefore, wireless communication potentially has low capacity, high collision probability, and high bit error rate.

Limited node resources - Node resources, which include energy, processing capacity, and memory, are relatively abundant in the wired networks, but may be limited in ad hoc networks and must be preserved. For instance, limited power of the mobile nodes and lack of fixed infrastructure restrict the transmission range and create the need for effective multihop routing in mobile ad hoc networks.

These features and their associated challenges make the multicast routing protocol design a very difficult task in such an environment. The most important characteristics which should be considered during multicasting protocol design are broadcast capacity, topological changes, high message loss rate and limited bandwidth and node resources.

2.1.2 The Design Challenges of Multicast routing for Ad hoc networks

The primary goal of multicast routing is to direct and transport packets through the network from the source node(s) to the destination node(s). To realize this operation, it contains the following functionalities:

- Multicast translation - which discovers all receivers behind a multicast address in the network.
- Routing structure construction and maintenance - which establishes and maintains routes among group members.
- Packet forwarding - which transports packets from sender to these receiver(s).

Multicast routing protocols for the Internet address the issue of routing structure which connects source(s) and destinations and forward data packets using this structure. Multicast translation is accomplished during the structure construction phase when group receivers join the structure. Through this way, these protocols focus on the problem of the minimum cost structure for packet forwarding. Here, the cost could be distance, delay, and so on. A natural routing structure for multicasting is a tree. The multicast routing protocols differ in how the multicast trees are constructed and what IP unicast routing algorithms are used when constructing the trees. Currently, there are mainly two kinds of multicast trees: source-based shortest path tree and group-shared tree. DVMRP [11], MOSPF [12] and PIM-DM [13] [14] use shortest path trees rooted at source, while CBT [15], BGMP [16] and PIM-SM [17] use group-shared tree. The shared tree in PIM-SM can be switched to a shortest path tree when needed.

Multicast routing protocols do not perform well in wireless ad hoc networks because tree structures are fragile and must be readjusted as connectivity changes. These methods generally assume that the topology of networks is stable and routing information only account for a small portion of the network bandwidth. These protocols may fail to keep up with topology changes in a MANET. The frequent exchange of routing information triggered by continuous topology changes yields excessive channel and processing overhead. For example DVMRP meets *data flooding overhead* problem when it is in ad hoc networks [18]. However the limited bandwidth and node resources on one side prevent routing protocol sending too much control messages and on the other side demand protocols to well choose routers in order to reduce resource consumption and fairly use node's resource. Furthermore, some protocols especially those protocols based on group shared tree use a single special node (called core or Rendez-Vous node) to construct tree. These protocols may not correctly operate in MANET due to single node failure problem which is usually caused by network partitions, node turned off, and so on.

Therefore, multicast routing for mobile ad hoc network should efficiently deal with both group member dynamics and topology dynamics. The construction and maintenance of routing structure should be done with a reasonable routing overhead in both low and high mobility networks, while providing efficient data transmission. It is also desirable that a routing protocol to be simple, distributed, adaptive, and dynamic. In brief, in order to achieve transmission efficiency, the design challenges of providing multicast routing for ad hoc networks cover not only group membership changes, data transmission efficiency, which exist in wired networks too, but also frequent topology changes, high message loss rate, limited bandwidth and node resources, which are relative to the properties of MANETs. The transmission efficiency means to consume as less as possible network resource while the results of multicast delivery meet as much as close to the application's requirement.

2.2 Techniques in providing multicast in a MANET

Designing a new multicast routing protocol may necessitate examining the main strengths and weaknesses of each approach in mobile ad hoc network environment and comparing the different existing approaches [10], [19] according to the properties of MANETs. These approaches are around four issues:

- Initiate condition issue, the moment to run protocol;
- Multicast Structure issue, the form to connect group members;
- Routing Philosophy issue, the dependence on a unicast routing protocol;
- Forwarding issue, how to forward multicast packet

2.2.1 Initiate condition issue

This issue replies the question of which triggers the running of multicast routing protocol operations. All researchers declare that their multicast routing protocols are on demand ([20], [21], [22], [23], [24], [25], ...). However they are on demand in two different fashions, which we call on group demand and on traffic demand respectively. In on group demand fashion, so far as group members exist in the network, routing protocol runs to handle membership and/or topology changes. Because group members are kept connected, there is small discovery latency when a sender begins a new packet transmission. However, control messages are injected into network when there is no multicast packet being sent. That could be considered inconvenient in MANET where bandwidth is limited. On the other hand, on traffic demand fashion reduces this overhead by in the way that the operation of the protocol is driven by the presence of packets being sent. In this fashion, nodes keep silent when they become group receivers. When a sender begins to transmit multicast packet, routing protocol starts to discover group receivers and then delivers packets. Once the transmission terminates, routing protocol stops too. Through this way, routing overhead is limited around traffic. But on traffic demand fashion introduces discovery latency. Due to the distances from sender to receivers are various, it is difficult to decide when to begin packet delivery. If the delivery starts too early, some receivers may not receive first packets for being far from sender. There are some mechanisms to alleviate this problem. For example some protocols broadcast the first packet instead of waiting until all receivers are discovered.

2.2.2 Multicast Structure issue

This issue discusses which type of structure is used to connect group members. Up to now there are two types of structure: tree and mesh. Tree structure is well used in wired network. It involves as less as possible intermediate nodes and contains unique path between sender and receiver pair. This structure can thus provide high data forwarding efficiency. There are two kinds of multicast trees: source-based

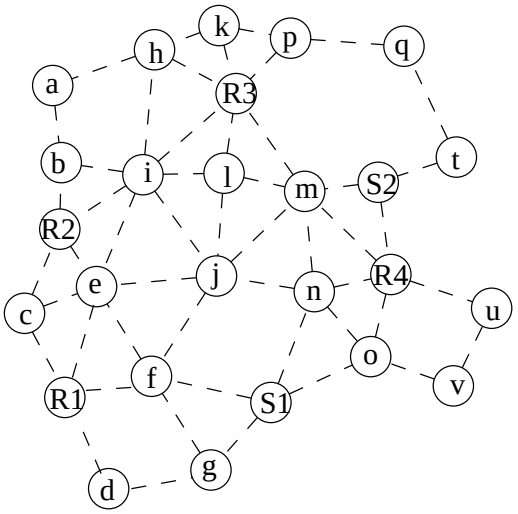
tree and group shared tree. Source-based tree means for each group, source pair there is a multicast tree which is rooted at the source and normally constructed by the shortest from the source to the destinations. This type of tree is efficient in data transmission since packets are delivered along the shortest path to the destinations. On the other hand, a core node serves as the root of the group shared tree. Group shared tree is not efficient in packet transmission since usually source should first transmit packets to core node then core node deliver them to group receivers. However this type of tree is more efficient in the point view of routing. Multicast source just needs to explore the route to the core instead of to all group receivers in source-based tree. Only one tree need to be maintained, control overhead is relatively low. While, the idea of using the least connectivity for multicast delivery from tree structure is not necessarily best suited for multicast in a MANET. In such an environment, nodes may move in an unpredictable way which causes network topology changes frequently. Because no alternative path between a sender and a receiver, every link broken takes place on tree trigger a reconfiguration. Compared to tree, mesh structure is robust against topology changes. Mesh is a subset of graph that may have multiple paths between any source and receiver pair. These alternative paths allow multicast packets to be delivered to the receivers even if links fail. Link failure may not trigger a reconfiguration. In brief, tree-based approaches provide high data forwarding efficiency at the expense of low robustness, whereas mesh-based approaches provide robustness and low routing overhead at the expense of higher forwarding overhead and increased network load.

Figure2.1 illustrates how tree and mesh connect group members in a MANET. This group contains two senders (S1 and S2) and four receivers (R1, R2, R3 and R4) distributed in a MANET. In Figure2.1(b) a tree rooted at S1 consists of 4 intermediate nodes connects these six group member. And in Figure2.1(c), a mesh which contains the shortest path between any pair of sender and receiver involves 7 nodes to connect group members. If any node among four intermediate nodes of tree fails, tree should be reconfigured. However, mesh has not this problem. If we delete anyone of these 7 nodes, mesh keeps connecting.

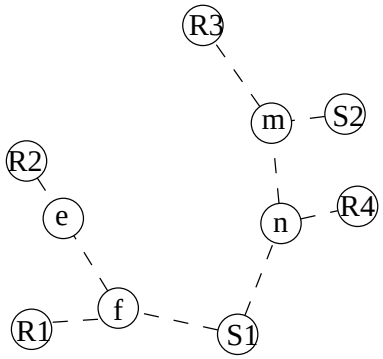
During structure construction and maintenance, intermediate nodes are explicitly invited to join structure. While for leaving they have two solutions, using soft state, in which if the structure membership is not updated before a timer out, node leaves structure automatically, or using hard state, in which node deactivates its structure membership upon receiving certain control packets. Control packets might be lost during their transmission. Thus, a more general way is to use the combination of these two solutions for deactivating structure membership

2.2.3 Routing Philosophy issue

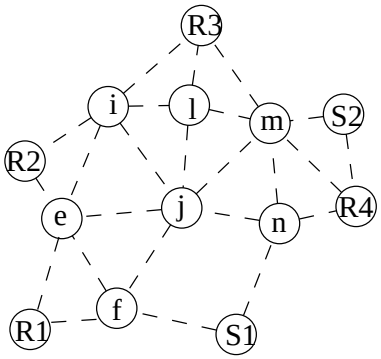
Multicasting in MANET should face membership dynamics and topology dynamics. Different to the case of Internet, where unicast routing protocols have been well studied before multicast conception was proposed, in MANET the hot research of these two kinds of routing begins at nearly the same time. That is why we can



(a) A multicast group in a MANET



(b) Tree, rooted at S1



(c)Mesh, contains shortest path

Figure 2.1: Example of tree and mesh to connect group members

find some protocols rely on unicast routing protocol hoping this protocol suitable in most case while others are independent. According to the dependence on an underlying unicast routing protocol, we can classify multicast routing protocols to simple multicasting (full dependent), median multicasting (median dependent) and all-in-one multicasting (independent).

The idea of simple multicasting is that multicast routing protocol discovers group receivers and unicast routing protocol provides the routes to concerned receivers in intermediate nodes. Aggregating these routes, multicast protocol deliver packet. This solution simplifies the protocol design in terms that multicast protocols focus on group membership dynamic and requires no multicast routing information other than group state information to be maintained in network.

On the contrary of simple multicasting, all-in-one multicasting is completely independent of any unicast routing protocol and realize all functionalities by multicast routing protocol itself. When discovering group members, protocol can probe the route to these members and construct a delivery structure at the same time. Thus the control overhead can be reduced via aggregating messages of these two functionalities. Instead of needing unicast routes to certain destinations, intermediate nodes just needs to know its delivery structure neighbors or its state on delivery structure. Then, the multicast packet is forwarded on this delivery structure. Topology changes on one side cause link failure which multicast protocol should repair, and on the other side create better routes which multicast protocol should include into structure to get better performance. For this aim, this solution requires that multicast protocols periodically probe network topology. However, periodical probe reacts slowly to topology changes and cannot adapt to various frequency of topology change. Furthermore, some link failures generate important effect on multicast delivery. Local recovery mechanism is studied to overcome this shortcoming. In this mechanism, structure members survey the links to its structure neighbors and immediately repair a link failure in local area.

A compromise of simple multicasting and all-in-one multicasting can be found in median multicasting. This solution needs unicast routing protocol to construct and maintain delivery structure, while packet forwarding is done via delivery structure. This solution requires less unicast routing information than first solution in intermediate nodes (normally only route to core or sender). This solution is simpler than the second solution since unicast routing protocol shares routing function. It just needs to modify structure according to route changes (detected by unicast routing protocol) and membership change. However, the protocol's performance is greatly depends on the correctness and efficiency of the unicast routing protocol in use.

2.2.4 Forwarding issue

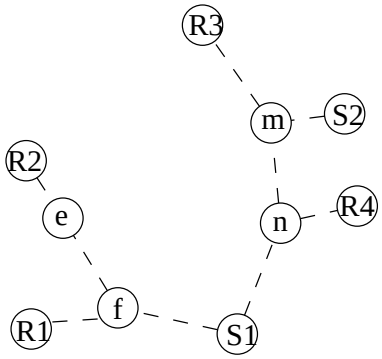
This issue addresses how to deliver multicast packets by using the multicast structure or the unicast routing protocol. In a tree structure, the general way is that a node takes packets from nodes with whom a tree branch has been established and

forwards it to other branches than the incoming one. On the contrary packets are normally flooded in mesh in the way that a mesh member can accept unique packets coming from any neighbor in the mesh and them send them. Unique packets mean the packets that a node never received before. We will explain the mechanism to judge unique packet later. As we explained in section 2.2.2, tree structure generates less forwarding overhead while is not robust against topology changes and mesh can resist to link failure but is not efficient in data forwarding. Some techniques emerge to improve packet delivery in these two structures respectively. For example, one method to improve robustness of tree structure is to flood packet on tree to benefit broadcast nature of wireless interface. The idea is, instead of limiting packet forwarding along tree branches, that nodes forward packet on broadcast and tree members accept unique packets from any neighbor node to achieve alternative path. Figure2.2 compares these two forwarding method on tree. Flooding on tree implicitly adds three links into the tree: link between node m and R4, between S2 and R4 and between node e and R1. If the link between node n and R4 breaks, due to the link between node m and R4 and link between S2 and R4, R4 can still receive packet from S1and S2. While for mesh structure, the question is how to reduce forwarding overhead. Thus one possible way is to select routes on the mesh to form a sub graph and then send packet in this sub graph. Figure2.3 illustrates this idea. In this example, four nodes (node e, f, m and n) are chosen to forward traffic sent by S1 and nodes e, i, l and m are invited to transmit packet generated by S2. Thus, only four intermediate nodes are needed instead of seven nodes in the case of using mesh directly.

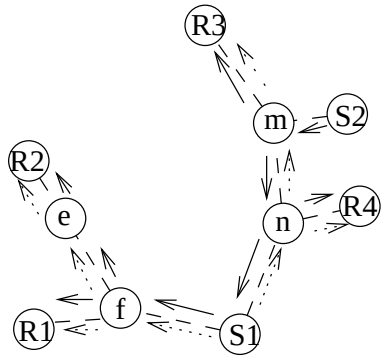
When forwarding multicast packets in MANET, one difficulty is to assure the packet in process is unique because multicast packet forwarding in MANET is different to that in traditional networks. In traditional networks, router receives multicast packet from one network interface and sends them to another interface(s). This is not the case with MANET. One node in MANET can use the same interface talking to any neighbor on the same wireless channel. This property imposes another method to avoid sending same packets multiple times. This is more important when protocol floods packet in delivery structure. The general method is to utilize sequence numbers for duplication detection. When the sender generates a new packet, it assigns a sequence number to the packet. This sequence number along with source and destination identifications uniquely identifies a packet in the network. A node stores temporarily this information of packets it has proccessed to detect duplication.

2.3 Current ad hoc multicast routing protocols

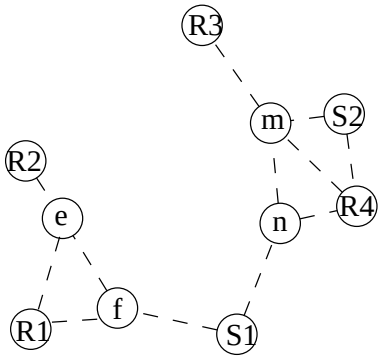
Although the advent of Defense Advanced Research Projects Agency (DARPA) packet radio networks addressed in the early 1970s [26], multicasting for mobile ad hoc networks becomes a topic of active research only during last five years. The main reason appears to be a popular belief that similar to the evolution of Internet



(a) Tree

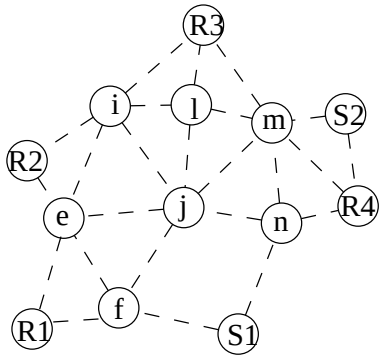


(b) Forwarding restrict on tree

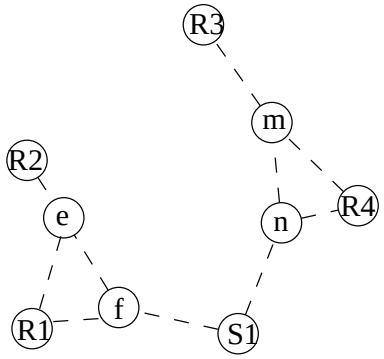


(c) Flooding on tree

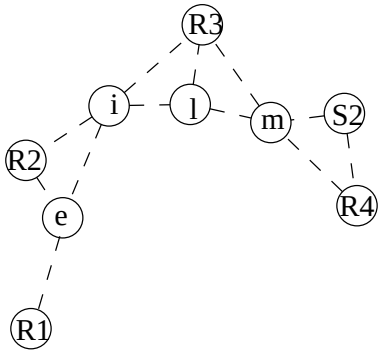
Figure 2.2: Packet forwarding in tree-based approaches



(a) Mesh, base graph



(b) Subgraph for S1



(b) Subgraph for S2

Figure 2.3: Example of subgraphs of mesh

routing multicast routing in MANET will be built on top of the unicast routing protocols. For this reason, most research has focused on solving the unicast routing issues in mobile ad hoc networks and then, based on these unicast routing infrastructures, multicast routing protocols are proposed. They are Ad hoc Multicast Routing protocol (AMRoute)[27], Core-Assisted Mesh Protocol (CAMP) [28], [29] Differential Destination Multicast (DDM)[30] and Lightweight Adaptive Multicast Algorithm (LAM)[31].

However, other researchers believe that because of the broadcast capacity of wireless nodes, mobile ad hoc networks are better suited for multicast, rather than unicast, routing and, that it is more effective to solve the multicast routing problem separately [10]. They designed their mutlicast protocols independent of any unicast routing protocol. They either extends the principle ideas of unicast routing protocols for MANET, which is the case of Associativity-Based Ad hoc Multicast (ABAM) [32] (from Associativity-Based Routing (ABR) [33]), Multicast operation of Ad-hoc On-demand Distance Vector routing protocol (MAODV) [23] (from Ad-hoc On-demand Distance Vector routing protocol (AODV) [34]) and Multicast Core Extraction Distributed Ad hoc Routing (MCEDAR) [35] (from Core Extraction Distributed Ad hoc Routing (CEDAR) [36] [37]), or develop a new multicast routing protocol, which is the case of Adaptive Demand-driven Multicast Routing (ADMR) [38], Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS)[22], Forwarding Group Multicast Protocol - Receiver Advertising and Sender Advertising (FGMP-RA, FGMP-SA) [18], On-Demand Multicast Routing Protocol (ODMRP) [24], [39] and Neighbor Supporting ad hoc Multicast routing Protocol (NSMP) [40].

2.4 Our Contributions

A new multicast routing protocol Multicast Routing protocol with Dynamic Core (MRDC) [41], is proposed. In this protocol, a hybrid multicast tree is constructed on the demand of traffic. By rooting tree at the first source of a multicast session, the multicast tree becomes hybrid: in single source group, it is source-based tree; while in multiple sources group, it is group-shared tree. The core of a multicas-tree may change in a passive or active way during a multicast session according to the traffic transmission and node's conditions. Tree structure faults are temporary tolerated and periodical tree refreshing removes these errors and adapts tree to current topology. Through this design principle, the control overhead to construct and maintain tree structure remains reasonable low, while the transmission efficiency of multicast tree is maintained. As a result low total potential bandwidth consumption for multicast delivery could be obtained.

MRDC use an adaptive mechanism to forward multicast packets according to network situation and application requirements. In fact two transmission modes are defined in MRDC. One transmission mode is similar to flooding in the tree. In this mode, MRDC considers tree structure as mesh and the interior nodes forms

forwarding group. Then, multicast packets are flooded in the structure without respecting the tree structure. In the second transmission mode, multicast packets are transmitted along tree edges with certain degree of reliability and with the cost of bandwidth and transmission delay. The former mode is suitable in congested networks and applications which are sensitive to transmission delay but can tolerate transmission errors such as voice conference. The later mode is preferable in non-congested networks for applications which are not sensitive to transmission delays such as news group. In this way, MRDC provides the best effort multicast routing by considering both application requirements and network condition.

As for the applications that require a multicast delivery guarantee, a reliable multicasting protocol, named active reliable multicast protocol with intermediate node support (ARMPIS) [42], [43], is designed on the top of MRDC. This protocol distributes the responsibility of multicast packet storage and retransmission to group receivers as well the multicast routers.

2.5 Classification

Table 2.1 classifies the current multicast routing protocol according to the criteria proposed in Section 2.2. Because the third issue, routing philosophy issue, discusses how to construct and maintain multicast structure, we employed the term (Re)Configuration in the table, which abbreviates configuration and reconfiguration. As demonstrated by this classsification, none technique can outperform than others and adapt to all situation. Most protocols are designed with a predefined situation (for example: node's mobility and multicast group size) and then choose or develop suitable techniques corresponding to that situation. However, as the definition of ad hoc network, the network situation can be changed arbitrarily and the same as for group configuration. We think it is important to study an optimal multicasting protocol which can smartly choose techniques to adapt to most situations.

Protocols	Initiate	Structure	(Re)Configuration	Forwarding
ABAM	Traffic	Source Tree	Independent	On tree
ADMR	Traffic	Source Tree	Independent	Flooding
AMRIS	Group	Group Tree	Independent	On tree
AMRoute	Group	Group Tree	Full-dependent	On tree
CAMP	Group	Mesh	Demi-dependent	Flooding
DDM	Traffic	Source Tree	Full-dependent	On tree
FGMP-RA	Group	Mesh	Independent	Flooding
FGMP-SA	Traffic	Mesh	Independent	Flooding
LAM	Group	Group Tree	Demi-dependent	On tree
MCEDAR	Group	Mesh	Independent	On forwarding tree
MAODV	Group	Group Tree	Independent	On tree
NSMP	Traffic	Mesh	Independent	Flooding
MRDC	Traffic	Hybrid Tree	Independent	Adaptive
ODMRP	Traffic	Mesh	Independent	Flooding

Table 2.1: Classification of current multicast routing protocols

Chapter 3

Multicast Routing Protocol with Dynamic Core (MRDC)

We study a multicast routing protocol called multicast routing protocol with dynamic core (MRDC) [41] for MANETs. MRDC consists of two planes, a control plane and a forwarding plan. The control plane constructs and maintains a structure to connect multicast group members on traffic demand. The forwarding plane transmits multicast packet using this structure to provide best effort delivery. Here, the best effort delivery means that routing protocol delivers packets as close to the application requirements as possible regarding network situation. Since nodes could act as host and traffic forwarder as well, in the sequel, we define a router as a node which transmits the traffic packets generated by itself or other nodes. A multicast group member may be at the same time a multicast router for that group.

This chapter is organized as follows: Section 3.1 introduces our assumptions during protocol design. Section 3.2 presents the protocol architecture and the main design principles of MRDC. Section 3.3 describes in detail the control plane of MRDC including the creation and maintenance of a multicast tree. In section 3.4, we discuss some problems of multicasting in IEEE802.11 and propose some solutions. Then, Section 3.5 presents in detail the forwarding plane of MRDC with a special emphasis on an adaptive forwarding mechanism for IEEE802.11 ad hoc networks. Finally, Section 3.6 closes this chapter with concluding remarks.

3.1 System model

During the design process, we will suppose that nodes in MANET are uniquely identified by some kind of identification such as their IP address. All nodes communicate on the same shared wireless channel and wireless links among nodes are symmetric, which means that the transmission characteristics of the transmitter and receiver of data on the link are identical. For instance, if node a can hear node b, node b can also hear node a. Some MAC-level protocols such as MACA [44] or MACAW [45] can ensure this bidierctional transmission.

Moreover, each multicast source binds on a communication port for transmission and assigns a reference number to each multicast packet it sends. This reference number, including source address and port number, can be considered as a packet's unique identifier in the network.

3.2 MRDC Architecture and Design Principles

3.2.1 MRDC Architecture

Contrarily to most multicast routing protocols which combine multicast packet forwarding with delivery structure construction and maintenance, Multicast routing protocol with dynamic core (MRDC) makes a distinction between routing, which is making the decision which routes to use, and forwarding, which is what happens when a packet arrives. One can think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is **forwarding**. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm (or control part) comes into play. This distinction leads to that MRDC is internally divided into two planes: a control plane and a forwarding plane, as shown in Figure 3.1. The control plane deals with the construction and maintenance of multicast delivery structures, while the forwarding plane copes with how to forward multicast packets hop by hop to their destinations. This architecture allows us to concentrate on studying an optimal routing strategy to reduce global bandwidth consumption while adapting to network topology changes, and then design an adaptive multicast transmission policy regarding network situation and application requirements. The control plane works in a passive fashion and is driven by the forwarding plan. In fact, the forwarding plane triggers the control plane to collect and update multicast routing information. Thanks to this routing information, the forwarding plane is able to deliver multicast packets generated by any node to their final destinations. The control plane is somewhat lower layer independent in the sense that physical layer and MAC layer have little influence on the result of delivery structure. Conversely, the question of how to forward multicast packets hop by hop to their receivers is closely relative to the MAC layer in use. Considering IEEE 802.11 [46] is preferred by MANETs, we develop an adaptive multicast forwarding mechanism in the forwarding plane of MRDC that provides a best effort multicast delivery in IEEE 802.11 ad-hoc networks.

3.2.2 Control Plane Design Principles

The bandwidth consumption of a multicast routing protocol comes from routing overhead and transmission overhead. The aim of the MRDC control plane is to find a trade-off between routing overhead and forwarding overhead in a way to improve bandwidth utilization efficiency. The optimal strategy to get to this goal is to use the most efficient data transmission technique since traffic packets are usually bigger

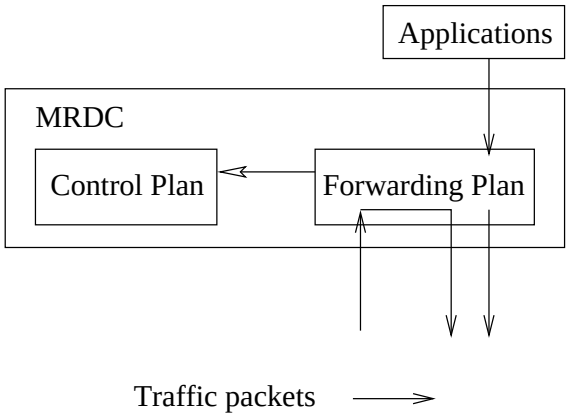


Figure 3.1: MRDC architecture

and more numerous than control messages and consequently consume much more bandwidth. Therefore, we try to reduce the routing overhead of this solution at a limited cost of traffic transmission efficiency.

In terms of traffic transmission efficiency, which means generating as little multicast forwarding overhead as possible, MRDC uses a tree structure in a way to limit the number of routers involved in the delivery structure. Extra nodes are invited to join the delivery structure only when the topology changes require the protocol to do so. The tree is periodically reconfigured to better fit to current topology.

MRDC uses a group-shared multicast tree on which the root, which we call core in this dissertation, is the first source of the multicast session or the source which wins the core competition after the disappearance of the core. Group-shared tree is less efficient than source-based tree in terms of multicast packet delivery but needs less control overhead for group members to join a multicast group and the maintenance of the delivery structure. On consequence the multicast tree is optimal for the core (the first or winner source) but suboptimal for the other sources. However, the control overhead can be significantly reduced. Multicast tree is constructed on traffic demand so that control overhead is limited around the traffic. Another factor in reducing multicasting control overhead is that MRDC temporarily tolerates faults in the multicast tree and leaves periodical tree reconstruction to remove these faults.

MRDC uses core to limit control overhead. However, the core concept used in MRDC differs from other group-shared tree-based multicast routing protocols for MANETs, which also use this concept (AMRIS [22], CAMP [28] [29], LAM [31] and MAODV [23]). Instead of select a core for a multicast session, in our protocol, core is initially the first sender of a multicast session and then may transfer to another sender under certain conditions. This choice on one side guarantees that tree is constructed and maintained on traffic demand, core is a router (it should at least send multicast packets generated by itself) and on the other side provides

flexibility facing the number of sources in a multicast group and adapts to network topology. In a single-sender multiple-receiver session, it creates a source-oriented tree. For a multiple-sender multiple-receiver application, it offers a group-shared tree. If MRDC detects that core has ended the transmission or any other cases (e.g. battery level of core is too low to allow it to relay multicast packets or heavy congestion observed around the core), MRDC can transfer the core role to another sender. Therefore even when the network is partitioned, receivers can always receive packets sent by the senders in the same partition because MRDC designates a sender as core in each partition. In this way, MRDC can also prevent single node failure problems. Once these partition merge into one, the corresponding trees also merge into one tree which root is designated through a core competition among senders.

Most tree-based multicast protocols demand that tree structure be kept coherent and loop-free so that transmission efficiency can be preserved. This requirement implicitly increases control overhead. However, MRDC proposes to temporarily tolerate faults in the multicast tree as a way to reduce such routing overhead. For example, in CBT and MAODV, a branch break results in the dissolution of the concerned sub-tree and all receivers in the sub-tree should rejoin the multicast tree individually. In MRDC, when such a break is detected, routers try to find another route to replace the broken one without taking care of fault forming in the tree or losing a little transmission efficiency compared to the mechanism used in CBT and MAODV. In a tree structure, fault usually means tree fragmentation or loop. If the tree is logically fragmented but is still physically connected, multicast forwarding can continue on it. As for duplicate forwarding caused by loops, it can be avoided through a duplication table. Furthermore, a periodical tree refresh mechanism removes faults through destroying old trees and constructing new ones.

3.2.3 Design Principles of the Forwarding Plane

The aim of the forwarding plane or in a more general term, the multicasting protocol is to deliver multicast packets in a best effort way. Yet, this does not depend only the delivery structure but also on the MAC layer protocol in use.

Most mobile ad hoc networks give preference to IEEE 802.11 as MAC sublayer protocol since it is regarded as a standardized protocol which allows terminals to share the wireless channel through ad hoc configuration. This protocol primarily targets unicast communications and, up to this time, does not efficiently support multicast transmission. According to the IEEE 802.11 specification, MAC protocol broadcasts multicast packets. The multicast sender simply listens to the channel and then transmits its data packet when the channel remains free for a period of time. There is neither MAC-level acknowledgment nor recovery procedure for a multicast packet. As a result, once a collision occurs due to problems such as hidden terminal, etc., the packet cannot be recovered, which degrades the performance of multicast routing protocol even when the network load is low.

In order to improve multicast packet transmission using current IEEE 802.11

standard, one method is that members of a delivery structure unicast multicast packets when it is possible. By doing so, the network layer transmits multicast packets point-to-point to selected neighbors using RTS/CTS option in order to avoid the hidden terminal problem. However, this method consumes more bandwidth since a node sends multiple copies instead of a single one in the broadcast way. This method is useful when the medium is not congested and with the condition that it should not create congestion. Therefore, a mechanism is needed to smartly choose the forwarding method. Following the above ideas, we design an adaptive data forwarding mechanism in forwarding plane for IEEE 802.11 MANETs. Two forwarding modes are defined in this mechanism: unicast mode and broadcast mode. Unicast mode consists in treating one multicast packet as multiple unicast packets and sending them with IEEE 802.11's RTS/CTS option, thus avoiding the hidden terminal problem. The broadcast mode is to pass multicast packets directly to IEEE 802.11 layer to reduce bandwidth consumption. This forwarding mechanism is adaptive because it makes a choice between two forwarding modes according to network load and tries to avoid congestion. To achieve this goal, our approach primarily uses the Average Queue Length (AQL) as its mode selection metric. AQL is the mean MAC queue length, it represents the difficulty of sending packets into the network. Because queue increase is usually the result of high network load and since mode change has an important effect on queue length, using AQL can control but cannot prevent congestion. The forwarding mechanism employs another metric called Medium Occupation for Reception (MOR) using some MAC layer statistics provided by most 802.11 implementations. MOR is defined as the percentage of the time where MAC is busy in reception over a period. Accordingly, MOR does not include the traffic generated by a node itself in order to avoid the impact of any forwarding mode change. Thus MOR serves as the network load criterion and AQL as congestion level criterion.

3.3 MRDC Control Plane Description

In this section, we introduce in detail the control plane of MRDC. MRDC adopts an on-traffic-demand tree to connect group members. A multicast tree is rooted at the first source of a multicast session. The control part of MRDC consists of two aspects: Tree construction and Tree maintenance. Tree construction is the aspect by which a core is selected and advertised to the network. Nodes that are interested in the multicast session are able to join the tree. During the communication, nodes may move, appear, disapear in the network. Radio environement may also change. These factors lead to topology change to which MRDC should react in order to maintain transmission efficiency. This burden is called tree maintenace. The tree maintenance aspect contains a reactive maintenance procedure and a proactive maintenance procedure. In reactive maintenance procedure, tree members detect broken branches and attempt to repair them in order to continue multicast traffic delivery in multicast tree. This procedure is also called local tree recovery since

the recovery procedure is limited in local region. In the proactive tree maintenance procedure, MRDC periodically reconfigurates the whole multicast tree. Due to its behavior, this procedure is named periodical tree refresh. Multicast tree maintenance also takes care of receivers eager to leave the group. Nodes use MRDC messages to exchange routing information, which is stored in MRDC tables. The following sections provide a closer view of the MRDC control plane.

3.3.1 Messages and Tables

The messages used by MRDC to exchange multicast routing information among nodes have the format shown in Figure 3.2. Thus, we will consider them as MRDC message. A MRDC message contains five fields: the type of the MRDC message (Type), a reserved field (Reserved) for future use, reference number (REF), group ID number (GID) and node ID number (NID). The field $< Type >$ indicates which kind of MRDC control message the packet carries. The field $< Reserved >$ permits different type of MRDC message has their own usage or for future extension. The field $< REF >$ contains a reference number which is assigned by the sender and used for duplication detection if the message is a broadcast one. GID represents the ID number of the group that this message concerns. NID is the ID number of some node, which depends on the message type.

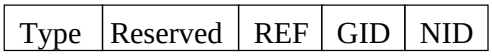


Figure 3.2: Structure of a MRDC message.

Each node in MANET possesses four tables: multicast routing table (denoted as MRTable), unicast routing table (denoted as URTable), duplication table and active neighbor table. The multicast routing table stores multicast routing information. It contains six fields: group ID number (GID), core ID number (CID), reference number (REF), upstream node ID number (UID), downstream node ID numbers (DIDs), state and last update time (LUT). GID represents the ID number of the group that this entry concerns. CID is the ID number of the core. REF stores the last reference number assigned by the core. UID is the ID number of the direct upstream node on the tree. DIDs saves the ID number of direct downstream node(s) on the tree. The state field indicates the current state of a multicast routing entry. It commands the behavior of that node for multicast forwarding. A multicast routing entry has three states: on-tree, tree-fault and non-forwarder. If a node has an on-tree state multicast routing entry, it means that the node belongs to the multicast tree and all branches to its direct tree neighbors are correct. Tree-fault also means that the node is a multicast tree member but some branches might contain contain errors. Non-forwarder state signifies that the node is not a multicast tree member. This is the default state of a multicast routing entry. For a non-tree-member node, the UID and DIDs fields are empty. The LUT represents the last update time of a

multicast routing entry. It is used to purge any expired information out of the table. The multicast routing table of node x is shown in Table 3.1, and it is denoted by $MRTable_x$. A multicast routing entry of group G is denoted by $MRTable(G)$

GID	CID	REF	UID	DIDs	STATE	LUT

Table 3.1: Multicast Routing Table

A unicast routing table maintains the routing information necessary to reach other nodes in the network. Initially, it is used to forward some unicast MRDC messages to their destination. However if any other unicast routing protocols for ad hoc networks ([34], [47], etc.) are present, they are also given read and write rights in order to share routing information. This table holds three fields: destination address (dst@), next hop address (hop@) and last update time (LUT). Similar to the MRTable, LUT is used to purge the expired information out of the table. Unicast routing tables are generally modified and used by MRDC.

dst@	hop@	LUT

Table 3.2: Unicast Routing Table

A duplication table, illustrated in Table 3.3, saves the packet header information for duplication detection during broadcast and multicast packet forwarding. It contains three fields: source address (src@), port number (port #) and reference number. The reference number is assigned by the source before sending the packet. These three fields uniquely identify a multicast packet in the network. As for MRDC messages, only source address and reference number are enough since only one MRDC process runs on each node.

src@	port #	REF

Table 3.3: Duplication Table

An active neighbor table is used to restore one hop neighbor nodes which are currently active. This table has two fields (see Table 3.4): a neighbor node address (nid@) and a last update time (LUT). When a node receives a control message or a traffic packet from a neighbor node (which means the neighbor node is active to send, receive and/or forward), it updates the corresponding entry and set the LUT as the current time. If an entry is not updated in a given time, the entry is removed from the active neighbor table.

nid@	LUT

Table 3.4: Active Neighbor Table

3.3.2 Tree construction

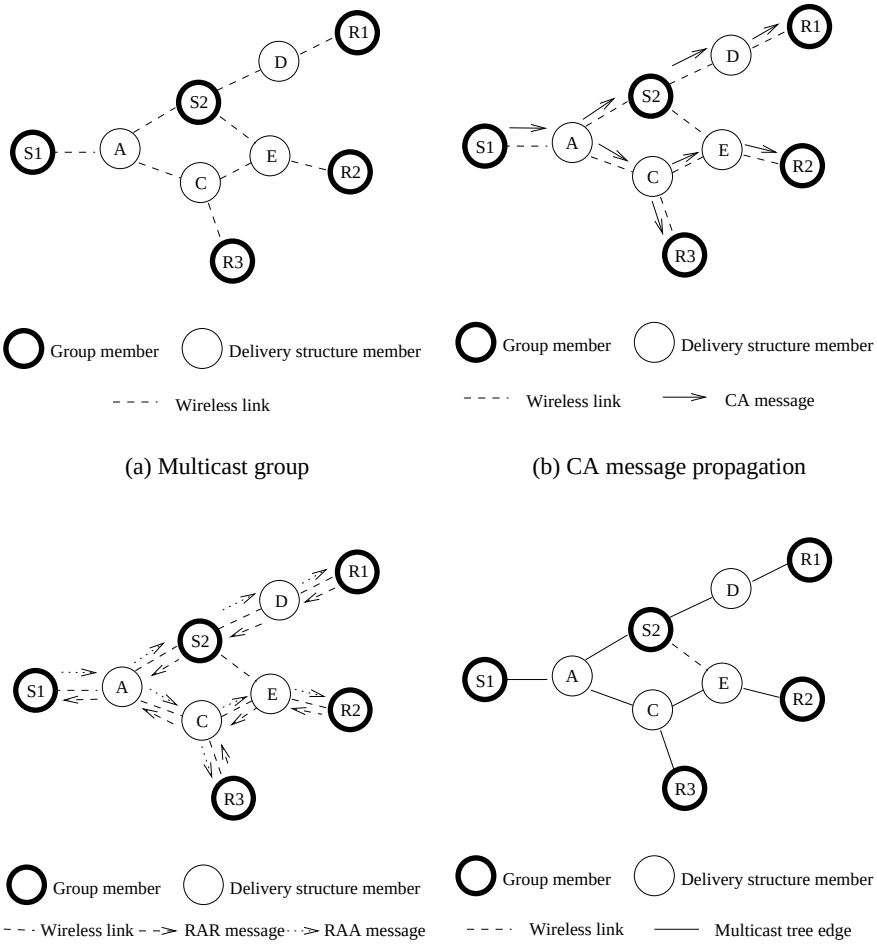
Multicast tree construction is initiated when the first source of a multicast session appears in the network. This source becomes core and broadcasts a Core Advertisement (CA) message to the network. Upon receiving this message, other group members send back a Route Active Request (RAR) message through the reverse path. When receiving RAR message, core or an active tree member replies to the request with a Route Active Acknowledge (RAA) message. When RAA message is transmitted to its destination, the nodes on the route become active tree members. In this way, the multicast tree is constructed.

Figure 3.3 gives an example of multicast tree construction under MRDC. Initially there are two multicast sources and three multicast receivers in the network (Figure 3.3(a)). Source S1 appears earlier than source S2. Therefore S1 becomes core and broadcasts a CA message to the network as shown in Figure 3.3(b). Upon receiving the CA message, group members (S2, R1, R2 and R3) sends back RAR message and core replies with RAA messages to activate the corresponding links. This procedure is illustrated in Figure 3.3(c). In fact, during RAR message forwarding this type of message is aggregated at nodes S2, C and A to reduce bandwidth consumption. Finally the multicast tree is constructed as demonstrated in Figure 3.3(d).

CA, RAR and RAA messages are MRDC messages. A CA message is a MRDC message in which the $< Type >$ field is set to CA. The $< Reserved >$ field is not used in this type of message and consequently is set to zero. The REF field contains the reference number assigned by the core. The GID field is the ID number of the concerned group and the NID field contains the ID number of the core. The reference number, together with the core ID number, uniquely identifies a CA message in the network. RAR and RAA messages are similar to CA messages with the $< Type >$ field set to RAR and RAA respectively. The difference lies in the REF field and the NID field. The REF field of RAR messages and RAA messages is the reference number stored in the multicast routing entry $MRTable(GID)$. In other words, the last reference number assigned by core that the node is aware of. The NID field of these two types of MRDC messages is the ID number of the node which generated the message or the last node which processed the message.

A multicast routing entry in the MRTable means the presence of multicast traffic and multicast tree for the group GID. Its existence determines the behavior of a node when this node activates the membership of group GID. There are four possible cases:

1. A node becomes group receiver when the corresponding entry does not exist.



(a) Multicast group

(b) CA message propagation

(c) RAR and RAA messages transmission

(d) Final multicast tree

Figure 3.3: Multicast tree construction

In this case the node remains silent remains silent until the corresponding entry is created.

2. A node becomes group source when the corresponding the entry does not exist. In this case, this node considers that it is the first source of the multicast session. It becomes core and initializes the multicast tree construction.

3. A node becomes group receiver when the corresponding entry exists.

4. A node becomes group source when the corresponding entry exists.

In the two later cases, the node sends a RAR message to join the multicast tree.

When a multicast source of group GID becomes core, it generates a reference number and creates an entry $MRTable(GID)$ in the MRTable with the necessary information. Then core initiates a CA message and broadcasts this message to the network. Upon receiving a CA message, any node first passes the core ID number and reference number through the duplication table in order to test whether the message is a duplicate or an original. In the former case, the node discards the duplicated CA message. In the later case, the duplicate table stores the core address and the sequence number. Then the node creates an entry $MRTable(GID)$ in its MRTable with all essential information such as Core ID number and reference number. It records the ID number of the node from which it received the CA message as the next hop node towards the core in the unicast routing table. This route would be used to send or forward RAR messages. Then it propagates the CA message to its neighbors. Therefore, other nodes are able to hear of the creation of a multicast session and are able to establish a reverse path to the core.

When it is a group member that receives a CA message, this group member generates a RAR message with the NID field set to its ID number. Then it sends this message to the next hop on the reverse route to the core. When receiving a RAR, a node first compares the reference number of the RAR message to that in the corresponding multicast routing entry $MRTable(GID)$. The difference of reference numbers indicates incoherent routing information and the node will stop processing the RAR message in order to avoid the formation of an erroneous tree. On the contrary, if these reference numbers are identical, the node adds NID into the DIDs field of the multicast routing entry to register this potential downstream node. Then, it replaces the NID field of RAR message with its ID number and sends this RAR message to the next hop towards the core. In this way, a RAR message is forwarded hop by hop till reaches the core or a active multicast tree member. As a RAR message propagates through the network towards the core, a potential multicast tree branch is formed. This type of branch begins at a potential multicast tree leaf that is a group member and ends by either the core, or an active multicast tree member, or a node which is waiting for active acknowledgment.

When the core or a multicast tree member (the nodes possede an active multicast routing entry of the corresponding group) receives a RAR message, it processes the message as previously described. However, instead of forwarding the

RAR message, it replies with a RAA message to activate route entries of nodes belonging to the potential branches. When a node receives a RAA message while it was waiting for such acknowledgment, it changes the state of $MRTable(GID)$ from *non-forwarder* to *on-tree*. It takes the node from which it received the RAA message as its upstream node by recording the ID number of that node in the UID field of $MRTable(GID)$. Then the node replaces the NID field of the RAA message with its ID number and forwards the RAA message to all nodes registered in the $< DIDs >$ field of $MRTable(GID)$. At last the RAA message arrives at the potential multicast tree leaf. Thus the multicast delivery tree is constructed and the source can use this tree to transmit packets.

The state change of multicast routing entry in the core is different from other nodes since it does not receive RAA message. Core uses the first received RAR message to activate the corresponding entry. However instead of doing it immediately, core waits for a while before changing the state of $MRTable(GID)$ from *non-forwarder* to *on-tree*, when it receives the first RAR message. This period of time permits multicast tree to be completely constructed before the beginning of multicast packet delivery.

In order to aggregate RAR messages, after sending a RAR message, each node starts a timer and waits for a RAA message. The node does not send or relay any further RAR messages before this timer expires. However it still processes the RAR messages by recording their NID fields into the $< DIDs >$ field of $MRTable(GID)$. The timer is stopped when the node receives a RAA message. Due to unpredictable reasons, for example RF interference, congestion, topology change during control message propagation, RAR and RAA messages might be lost. In this case, the node has not received any RAA message when the timer expires. Then if the node is not a group member, it resets the timer to be ready to forward another RAR message. As for a group member, it makes a second attempt by re-sending a RAR message. If the attempt still fails, the member will wait for the next incoming CA to join the multicast tree.

3.3.3 Tree maintenance

MRDC tree maintenance is composed of four parts: local tree recovery, periodical multicast tree refresh, group members departures and core migration. Local tree recovery reacts quickly to link failures and demands little routing overhead. On the other hand, periodical multicast tree refresh overcomes the link failures that cannot be solved by local tree recovery and gives an optimal tree structure. Periodical tree refresh gives members a chance to implicitly leave group. However they can explicitly leave by sending a message. Core migratoin refers to the movement of core from one source to another during a multicast session. It contains a core competition algorithm which deals with multiple core existing in the network which is normally a result of the merger of several network partitions. Core migration also occurs in a passive way when the source of the core finishes packet generation or in an active way in order to reduce traffic congestion or fair use node's energy.

Local tree recovery

Multicast tree members may become disconnected from their upstream node or from some downstream node when nodes move in the network or when wireless transmission conditions change. Once a disconnection is detected, tree members execute local tree recovery to resume interrupted multicast delivery. In the local tree recovery procedure, the upstream node broadcasts a Joining Invitation (JI) message to *n-hops* away to discover a recovery route to the lost downstream node. Then the lost downstream node uses this route to rejoin multicast tree when receiving the JI message.

Tree members use *active neighbor table* to detect disconnection. After having forwarded a multicast packet, a tree member checks whether its upstream node and downstream nodes are all in its active neighbor table. If a tree neighbor is not in the table, a disconnection is detected. Recall that an entry of the active neighbor table is updated when a node receives a traffic packet or a control message from the corresponding neighbor. If an entry is not updated for a given time, it is removed. Since traffic usually flows from the root to the leaves in a tree structure, a tree member just needs to notify its presence to the upstream node. For this aim, every NEIGHBOR_HELLO period, tree members check whether they have broadcast some packet or successfully sent at least one packet to their upstream nodes during the last period. If it is not the case, they broadcast a hello message to make their upstream nodes aware of their presence.

When a broken tree edge is detected, a node runs a different sort of local tree recovery to handle this problem depending on its level in the tree structure.

If an upstream node detects a downstream link broken, this node sends a JI message to explore a route through which the lost downstream node can rejoin the multicast tree. A *Joining Invitation* (JI) message is a MRDC message with the $< Type >$ field set to JI and the NID field set to the lost downstream node's ID number. It is then broadcast *n-hops* away using an Expanded Ring Search (ERS) procedure [34] to control the message propagation. Initially, n is set to 2. This means only nodes within two hops away from the sender will receive the JI message. After a certain elapsed time, if the node does not hear from the addressed node, it increases n by one and rebroadcast the JI message. The node repeats this procedure until either it receives a recovery message from the addressed node or n reaches a maximum value, called *Greatest-Range*. Similar to the CA message, when a JI message propagates through the network, a reverse path to the message's sender is established. Upon reception of the first JI message, the lost downstream node replaces the upstream field (UID) of the respective multicast routing entry with the next hop on the reverse path, sends a recovery message and discards subsequent JIs message. The recovery message addresses the JI message's sender and its NID field is set to the lost downstream node's ID number. When a node receives a recovery message, it extracts the next hop information and compares it with the upstream field of the correspond entry $MRTable(GID)$. There are three cases. The first case happens when the node is not a multicast tree member

(the state of $MRTable(GID)$ is *non-forwarder*), it therefore adds the ID number stored in the NID field of the recovery message into the $< DIDs >$ field, sets the UID field to the next hop node and activates the entry as *on-tree* state. The second case takes place when the node is a tree member and the upstream node coincides with the next hop node, the node simply adds NID into the $< DIDs >$ field. In the last case, when the node is a multicast tree member but the upstream node is not equal to the next hop node, the node sets the entry's state to *tree-fault* and changes the type of the MRDC message to *recovery_fault* type. Upon reception of a *recovery_fault* MRDC message, further nodes will only set their state of $MRTable(GID)$ to *tree-fault* and will not add the NID of the MRDC message to the $< DIDs >$ field. Finally, the node replaces the NID with its ID number and forwards the message to the next hop. This step is repeated until the recovery message arrives at its destination.

If it is the downstream node that detects at first the link failure, it triggers a timer before taking any action. If it does not receive any JI message before the expiration of the timer, this tree member removes the upstream node from the multicast routing entry and sets the state to *tree-fault*.

Let's see why we need the *tree-fault* state. The discovered routes which would be used to replaces broken tree edges can be classified into three cases, as illustrated in Figure 3.4.

- The first case (Figure 3.4 (a)) is when no node belong to the current multicast. In this case, MRDC can safely add the route to the multicast tree by setting the state of these node to *on-tree*.
- The second case (Figure 3.4 (b)) is when some nodes on the route are also tree members but none of them belong to the sub-tree rooted at the downstream node of the broken edge.
- The third case (Figure 3.4 (c)) is when at least one node is a member of the sub-tree.

In the two later cases, instead of sending extra control message to maintain a correct tree structure, MRDC tolerates these errors by setting the state of corresponding nodes to *tree-fault* and not adding the corresponding nodes to the downstream node list of their upstream node. The multicast tree is logically fragmented but physically connected with the *tree-fault* state since they are still within coverage range of their upstream nodes, therefore allowing multicast delivery to continue.

Periodical multicast tree refresh

Local tree recovery could provide non-optimal routes or may fail if either the lost downstream node is farther away than the *Greatest-Range* of the upstream node, or if the recovery message is lost. MRDC periodically reconfigures multicast trees to remove these kinds of problems and also give a chance to better adapt the tree to the current topology. The core is in charge of initiating a periodical multicast

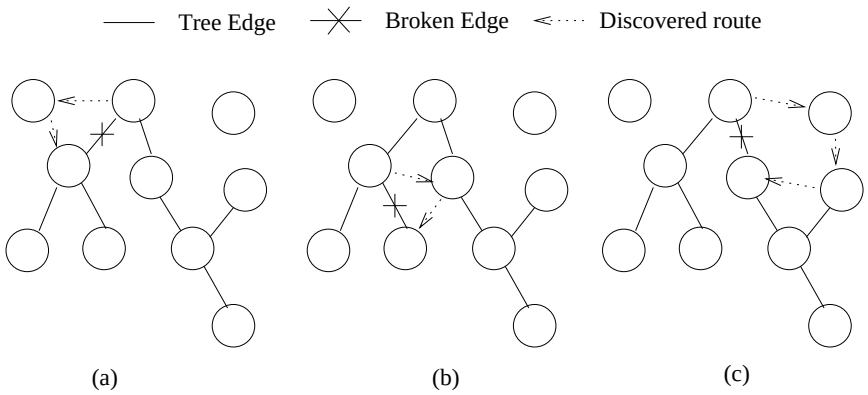


Figure 3.4: Possible local recovery scenarios

tree refresh. Every period (called *PERIOD_REF*), the core changes the state of the entry $MRTable(GID)$ to *non-forwarder*, computes a new reference number and broadcasts a CA message. This message will refresh the multicast tree and at the same time, the corresponding reverse path to the core. Once a node receives a non-duplicated CA message, it updates the corresponding fields of the multicast routing entry $MRTable(GID)$, empties the $< DIDs >$ field and sets the state of the entry to *non-forwarder*. In this way, the multicast tree is destroyed and group members run a RAR/RAA procedure to construct another multicast tree.

Group member departures

This periodical multicast tree refresh procedure gives multicast group members the possibility to quietly leave the group. These nodes can first check whether the next multicast tree refresh time will come soon. If so, they do not run the RAR/RAA procedure upon receiving the CA message so that the branch(es) will be pruned automatically. Otherwise, they might choose to explicitly leave the tree by sending a message to their upstream node. In the case when the core ends the transmission and wants to leave the group, it checks whether there is another source in the multicast tree that could become the new core. If it is the unique source in the group, it dismisses the tree. Otherwise the new core will be in charge of sending periodical CAs.

Core migration

Because of network partition, or any other reason, a source may choose to become a core without hearing any CA message from the core. Hence, there may be more than one core node existing in the network. After the network converges, a core can hear the CA messages of other core(s). These cores use a **core competition** algorithm to decide which source is the winner and it should continue acting as

core. Cores can use their IP addresses, identification or any other information to compete. The losers will stop sending periodical CA messages and will not react to group member join packets such that their trees will be quietly dismissed after the multicast route entries have expired.

Another case of core migration happens when the core finish its packet generation. This core should choose another source to play the role by using thecore competition algorithm since it knows IP addresses of all the other sources when receiving traffic packets from those sources. A special message or an extra field of CA message can be used to inform the sources or the entire network about this designation. Then the designated source is in charge of the initiation of period tree refresh in following periods.

One important shortcoming of group-shared multicast tree is the traffic concentration at the core. It may provoke congestioin around the core. On the other side, core should use more energy than other tree members to receive and forward multicast packets. MRDC can relieve this problem by making core migrate among sources time to time.

3.3.4 MRDC control overhead discussion

The control overhead of MRDC comes from the periodical tree refresh and local tree repair procedures. In periodical tree refresh, CA messages are broadcast by flooding. Each node sends at least once the CA message. Then, every group member except the core sends a RAR message and should receive a corresponding RAA message. Thus, if a network consists of n nodes and a multicast group contains m members, the control overhead of the periodical tree refresh per seconds is

$$\frac{n + 2 * (m - 1 + x)}{PERIOD\_REF}, \tag{3.1}$$

where x is the number of non group member nodes on the tree. The number of non group member tree node, x, is determined by the distribution of group members in the network. In the ideal case, where all the group members are within the coverage range of core, x reaches it minimum value, zero. On the opposite, in the worst case where group members are distributed at the bound of the network and multicast tree contains all nodes in the network, $x = n - m + 1$.

Consequently, the total control message rate of MRDC per second is:

$$\frac{n + 2 * (m - 1 + x) + y}{PERIOD\_REF}, \tag{3.2}$$

where y is the amount of control messages involved in local tree repairs during a period. The parameter y is function of both node distribution and topology change speed. In a stable network where there is no topology change, y is zero. While in an extremely dynamic network, y could reach a significant value.

With the same parameters, we can estimate the control overhead of ODMRP, a mesh-base multicast routing protocole, and ADMR, a multicast routing procoled

with source-based tree. The total control message rate of ODMRP per second is:

$$\frac{n + (m - 1 + x)}{PERIOD\_REF} * s \tag{3.3}$$

And that of ADMR is:

$$(\frac{n + (m - 1 + x)}{PERIOD\_REF} + OH_{local\_recovery}) * s \tag{3.4}$$

Compared to these two protocoles, the control overhead of MRDC is not a function of the number of sources in a group. As a result, its control overhead keeps stable as the number of sources increasing in the group. This behavior makes MRDC a suitable protocol to support both one-to-many and many-to-many communications.

To summaries, the control overhead of MRDC depends on the network size, the group size, group member distribution and the topology change frequency. It is not smaller than $\frac{n+2*(m-1)}{PERIOD\_REF}$ for a given tree refresh period *PERIOD_REF*. For a given network and multicast group, we can reduce the control overhead through increasing the tree refresh period. However, a big refresh period can lead to an unusable tree in high mobility networks. One trade-off should be found between the controle overhead and the reliability of the multicast. We will study this trade-off through simulations.

3.4 IEEE 802.11 Background and Multicast in IEEE 802.11

Before introducing the forwarding plane of MRDC in detail, we would like to discuss multicasting in IEEE 802.11 and then justify that an adaptive forwarding mechanism is indeed necessary

IEEE 802.11 Distributed Coordination Function (DCF) is often used as MAC sub-layer protocol for mobile ad hoc networks [48], [49], [50], since DCF is the mode which allows terminals to share the wireless channel in an ad hoc configuration. Unfortunately, until now, IEEE 802.11 DCF is almost identical to the basic Carrier Sense Medium Access/Collision Avoidance (CSMA/CA) when it comes to send multicast packets. The multicast sender simply listens to the channel and then transmits its data frame when the channel becomes free for a period of time. The sender does not receive any transmission acknowledgment message from the receivers.

Multicast transmission not only suffer from wireless interface problems but also from the well known hidden terminal problem in the CSMA/CA protocol [51], [52]. Hidden terminals appear when the network simultaneously transmits different multicast packets. For example, Figure 3.5 shows a simple IEEE 802.11 ad hoc network. In this network, source S wants to send multicast packet to three receivers R1, R2 and R3 through five delivery structure (either mesh or tree) members (nodes A, B, C, D and E). If the source generates packets at a high rate, inter-packets collision might happen. Supposing that multicast source S generates packet p_n when

node B or C is sending packet p_{n-1}, S sends packet p_n and node A cannot get this packet. The same problem exists when multiple multicast/broadcast traffic is present in the network at the same time. This kind of hidden terminal problem is generally related to traffic load in the network. The more multicast/broadcast traffic flows in the network, the more hidden terminals will be generated. Yet, there exists another kind of hidden terminal in multicast communications which has no relationship with traffic load. Hidden terminal problem may occur even when delivering one multicast packet. In the same network, let us suppose that source S has only one multicast packet to send. Among delivery structure members, node C and node B cannot hear each other while node E is unfortunately placed in the coverage range of both node B and C. When receiving the multicast packet from node A, node B and C might begin to transmit the same packet simultaneously or before the other node finishes transmission because they do not have any information from the other side. In other words, nodes B and C becomes hidden terminal for each other when they transmit the multicast packet to node E. The packet coming from node B and C collide at the wireless interface of node E prevents this node from receiving this packet from either node B or node C. As a result, the multicast transmission fails at node E. We name this kind of packet collision as "identical packet collision". The frequency of *inter-packet collision* is a function of network load while *identical packet collision* rate is independent of network load and is determined by the delivery structure. That also can be used to explain why all multicast routing protocols analyzed in [19] except AMROUTE [27] multicast protocols do not achieve 100 percent packet delivery ratio in stable and low load networks.

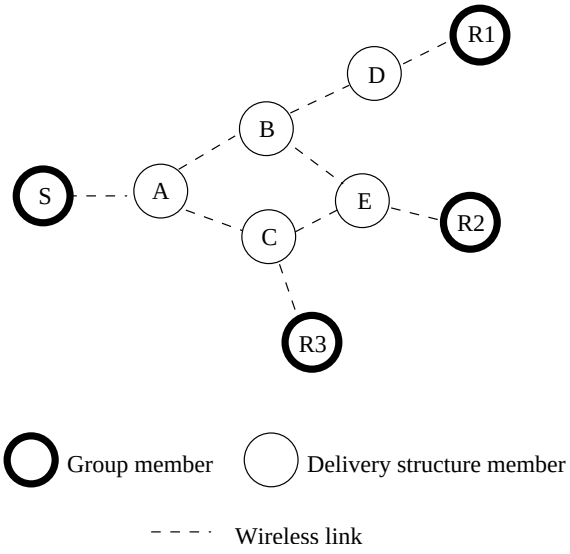


Figure 3.5: Multicast packet delivery in a simple IEEE 802.11 ad hoc network

To reduce multicast packet delivery failure, one potential solution is inspired from the mechanism used in unicast packet transmission. For unicast packets, the

specific access scheme of IEEE 802.11 DCF is CSMA/CA with acknowledgments. To mitigate collisions caused by hidden terminals, the nodes can send a unicast packets with RTS/CTS option that is based on two control frames: Request-To-Send and Clear-To-Send for virtual carrier sensing. RTS/CTS option uses a four-way RTS/CTS/DATA/ACK exchange. Before sending a unicast packet, a node first sends an RTS (Request To Send) packet to the destination. If the destination believes that the medium is idle, it responds with a CTS (Clear To Send). The sender then transmits the data packet, and waits for an ACK (Acknowledgment) from the receiver. If a node overhears an RTS or CTS, it knows the medium will be busy for some time, and avoids initiating new transmissions or sending any CTS packets. Thus for multicast communication using current IEEE802.11 MAC protocol, the simplest way is to treat a multicast packet as multiple unicast packets and send them individually to each direct delivery structure neighbors, which we call unicasting multicast packets or **unicast fashion**. On the other side, we define the way of using CSMA/CA to forward multicast packets as broadcasting multicast packets or **broadcast fashion**.

Unicasting multicast packets introduces extra forwarding overhead and transmission delay. In a multicast tree, the forwarding overhead of the ”broadcast fashion” is the number of interior nodes on the tree and that of ”unicast fashion” is the number of edges on the tree. Depending on the multicast delivery structure, the difference between these two fashions is the number of leaf nodes on the tree which varies from 0 (chain) to n-1 (star). Therefore, the unicast fashion is preferable when the network load is not high and at the same time this fashion does not generate congestion. The transmission delay is defined as the difference between the moment that a packet is generated and the moment an application receives it. In broadcast fashion, all delivery structure neighbors simultaneously receive the packet. Thus, the transmission delay of each destination depends mostly on the distance to the source. In unicast fashion, transmission delay is not only a function of the distance but also of the number of branches on the path from source to destination, since tree members send multicast packets to theirs direct tree neighbors one after another. Compared to the broadcast fashion the delay could be increased by $(n-2)*T$ if the delay of one hop transmission is T, and without considering extra delay due to RTS/CTS/ACK and retransmission.

Therefore, broadcast fashion is suitable for applications which are sensitive to transmission delays and jitter but can tolerate packet loss such as voice. Conversely, for applications which are not sensitive to delays (for example, newsgroup), unicast fashion can be used in low load network to achieve a better delivery success rate. Moreover, when the network load increases or congestion appears, a node can switch to broadcast fashion to alleviate congestion so as to provide a best effort delivery. Bearing this idea in mind, we developed the forwarding plane of MRDC.

3.5 MRDC Forwarding Plane Description

In this section, we introduce in detail the forwarding plane of MRDC. The forwarding plane triggers the functionalities of control plane and delivers multicast packets to their destinations. In this section, we assume that all the nodes communicate on IEEE802.11 in ad-hoc mode. If the MAC layer protocol used differs from IEEE 802.11, the forwarding plane supposes the MAC layer support efficient multicast transmissions and will simply pass multicast packets to the MAC layer. The MAC layer accepts all multicast packets and forwards them to the network layer. Then, network layer decides whether to drop the packets or to forward the packet.

In the adaptive multicast forwarding mechanism, we defined two transmission modes: unicast mode and broadcast mode. In the unicast mode, forwarding mechanism treats a multicast packet as multiple unicast packets and sends a copy to every delivery structure neighbor with the RTS/CTS option. In the broadcast mode, multicast packets are sent directly using CSMA/CA protocol. A structure neighbor list, which contains a node list and a broadcast flag, is defined in order to permit the forwarding plane to obtain essential routing information from the control plane.

This mechanism contains two procedures: mode selection and data forwarding. The mode selection procedure is executed periodically. In this procedure, nodes compute two metrics and select a suitable transmission mode. The data forwarding procedure transmits multicast datagrams in the selected mode. The following sections present these two procedures in detail.

3.5.1 Mode Selection

If the underlying MAC layer protocol is IEEE 802.11, nodes initially operate in unicast mode and periodically (every MS_PERIOD seconds) calculate two metrics: Average Queue Length (AQL) and Medium Occupation for Reception (MOR) and compare them respectively to their thresholds: Queue LENgth threshold (QLEN) and INcoming occupation ThReshold (INTR). The comparison result defines which transmission mode will be used in the next period. Figure 3.6 shows the mode selection procedure at time t. The mode selection procedure contains two steps: metrics computation and mode selection. We will present this procedure in detail.

When the current IEEE 802.11 specification is applied to transmit multicast packets, the hidden terminal problem creates severe transmission failures. In order to reduce them, one potential solution is to deliver multicast packets point-to-point to selected neighbors with RTS/CTS option. However, the obvious disadvantage of unicasting multicast packet is that this method generates more forwarding traffic than CSMA/CA. The extra forwarding traffic becomes annoying as network load increases. Therefore, unicasting multicast packets is feasible only in low load networks. Hence, we chose the Average Queue Length (AQL) as the key metric to choose transmission mode. AQL is the mean number of packets in queue that are awaiting transmission by the network interface. Because queue lengths change greatly as transmission mode and/or traffic pattern changes, the following formula

```
Algorithm Mode_selection(t)
Notation: AQL: Average Queue Length
          MOR: medium occupation for reception
          QL: Queue Length
          #B(t): Number of bytes received till moment t
          C(t): Transmission speed at moment t
          QLEN: Queue LENgth threshold
          INTR: INcoming occupation ThReshold
Begin
  Get QL(t), #B(t) and C(t);
 calculate AQL(t);
 calculate MOR(t);
 if mode==unicast then
   if AQL>QLEN or MOR>INTR then
     mode=broadcast;
 else
   if AQL<QLEN/2 and MOR<INTR then
     mode=unicast;
 store AQL(t) and #B(t) for the calculation of next period
End
```

Figure 3.6: Mode_selection() at time t

is used to calculate current average queue length AQL(t):

$$AQL(t) = \alpha * AQL(t-1) + (1-\alpha) * QL(t) \tag{3.5}$$

where $QL(t)$ is the queue length at time t and AQL(t-1) is the average queue length of the last period. This metric reflects the node's difficulty to send packets into the network. At the beginning where there is no traffic in the network and consequently no packet waits in the queue, the initial average queue length $AQL(0) = 0$. As traffic increases in the network, nodes have more packet to send or forward. Then their average queue length increase. When AQL exceeds a certain threshold, called Queue LENgth threshold (QLEN), the broadcast mode should be employed in order to reduce bandwidth consumption. After a node switches from unicast mode to broadcast mode, the length of the queue will significantly get reduced because it sends less copies and broadcast packets are sent faster (no four handshakes at the MAC layer for example). Therefore to avoid ping-pong switch, nodes should wait until their AQL become smaller than another threshold (e.g. half of QLEN) before they can switch back to unicast mode.

Furthermore, some implementations of 802.11 provide some statistics such as the number of bytes received or sent during the last period, etc. These counters can be utilized together with AQL to make a smarter choice in avoiding congestion. Here we define a metric called Medium Occupation for Reception (MOR) to reflect network load. MOR is defined as the MAC busy for receiving over a period. With this metric, a node estimates the bandwidth occupied by its neighbors and consequently the bandwidth it can use. Nodes use the statistics of received bytes provided by the underlying IEEE802.11 layer to calculate MOR as follows: The MAC layer counts the number of bytes it receives until time t: $\#B(t)$. Then the nodes are able to compute MOR using the formula:

$$MOR(t) = (\#B(t) - \#B(t-1))/(C * MS\_PERIOD) \tag{3.6}$$

where C represents the MAC layer transmission rate during the last period or at time t. The accuracy of this metric depends on how many neighbor nodes use unicast mode to transmit multicast packets since the estimation does not take into account RTS, CTS, ACK and retransmission. Thus, the more neighbors operate in broadcast mode, the more accurate MOR a node can obtain. The forwarding mechanism can estimate the available bandwidth for the node through the formula:

$$(1 - MOR(t)) * TOTAL\_BANDWIDTH \tag{3.7}$$

When MOR is bigger than a threshold, called INcoming occupation ThReshold (INTR), a node considers that it is in hot spot and probably has not enough bandwidth to unicast multicast packets and should switch to the broadcast mode. Otherwise, the node is not in hot spot and can use unicast mode if it does not create congestion. In case the MAC layer counters are not available, the algorithm always has $\#B(t) = \#B(t-1) = ... = 0$, which leads to $MOR = 0$ and consequently dis-activates this metric automatically.

In brief, if any one of AQL and MOR is superior to their respective thresholds (QLEN for AQL and INTR for MOR), the node is considered either in hot spot or having sent too many packets. It will choose broadcast mode as the multicast transmission mode to be used in the next MS_PERIOD. On the other hand, if AQL is smaller than half of QLEN and MOR is inferior to INTR, the node considers that the traffic it sent is relatively low and the medium occupation rate permits to send more traffic. It can therefore use unicast mode to transmit multicast packets in the next period. Multicast routers will forward multicast data packets according to the current transmission mode in use.

3.5.2 Multicast Data Forwarding

Routers deliver multicast packets according to the transmission mode in use and the state of correspond multicast routing entry.

When the state of the multicast routing entry is set to *on-tree*, sources begin to send their multicast packets. Intermediate routers should detect duplications before relaying a multicast packet. When a node receives a multicast packet, it consults its Duplication Table to see if the packet has been processed before. If so, it discards the packet. Otherwise, it updates the Duplication table to reflect the packet header information (source address, port number and reference number). After ensuring that the packet is non-duplicate, the forwarding plane asks the control plane to fill the structure neighbor list of group G. It decides to forward the packet, drop the packet or start any action depending on the content of this list.

If the multicast routing entry does not exist, the control plane returns an error. The forwarding plane checks whether the packet's source is the node itself. If it is the case, which means a new session begins and the node is the first source, the node should act as a core. The forwarding plane triggers a multicast tree construction. Otherwise, the forwarding plane simply drops the packet.

From now on, let us consider the case where the multicast routing entry exists. As mentioned in Section 3.3.1, a multicast routing entry has three states: *on-tree*, *tree-fault* and *non-forwarder*. While non-forwarder state indicates that a node is not a delivery structure member and as a result returns an empty list and does not set the broadcast flag, the other two states are reserved for multicast tree members. In these cases, when receiving a packet for some group G, the control plane fills in the structure neighbor list of group G with the list of all direct tree neighbors stored in the multicast routing entry and sets an indication of broadcast requirement if the entry's state is tree-fault. The control plane takes out of the list the node from which the packet has been received before passing the result to the forwarding plane. If the structure neighbor list does not contain any node and the broadcast flag is not set, the forwarding plane drops the multicast packet. Otherwise, the forwarding plane sends the packet according to the transmission mode.

If a node is in broadcast mode, the forwarding plane passes the packet directly to the MAC layer and this packet will be sent with CSMA/CA mechanism without acknowledgment. On the other hand, if a node is in unicast mode, the forwarding

plane sends a copy of the packet to each member in the list. Then, if the broadcast flag is set, forwarding plane broadcasts the packet.

Let us now see how the unicast mode works in the former example (Section 3.4). S explicitly sends a multicast packet (or encapsulates the multicast packet in a unicast packet and then sends it) to node A, then node A duplicates the multicast packet and sends a copy to node B and C. This process continues until the multicast packet reaches R1, R2 and R3. As a result, the multicast transmission failure caused by hidden terminal can be greatly reduced thanks to four-way handshaking.

3.5.3 Forwarding overhead discussion

The forwarding overhead of MRDC at network layer depends on the multicast tree structure and the forwarding mode of the routers contained in the tree. If all nodes operate in broadcast mode, the number of forwarding events is equal to the number of interior nodes on the tree:

$$\#of forwarding = \#\ of\ interior\ nodes \tag{3.8}$$

On the other hand, if routers use unicast mode to transmit multicast packets, the number of forwarding events is the number of edges on the tree, which is equivalent to the number of nodes on the tree (interior nodes plus leaf nodes) minus one.

$$\#of forwarding = \#\ of\ nodes\ on\ tree - 1 \tag{3.9}$$

Therefore, the forwarding difference of these two modes is the number of leaf nodes minus one.

3.5.4 Related Works

To efficiently support multicast transmission, a few multicast MAC protocols [53], [54], [55], [56] have been proposed to extend the IEEE 802.11 broadcast/multicast protocol with RTS/CTS handshaking. In [53], once a sender gains access to the medium, it transmits an RTS packet to its neighbors and waits for CTS packet for WAIT_FOR_CTS time units. If a node receives an RTS packet when it is not in the YIELD phase, it sends back a CTS and then waits for the data packet for WAIT_FOR_DATA time units. If the sender does not receive any CTS packet before its WAIT_FOR_CTS timer expires, it backs off and enters the contention phase again to retransmit the broadcast/multicast data packet. If the sender receives any CTS packet before its WAIT_FOR_CTS timer expires, it transmits the data packet and waits for WAIT_FOR_NAK time units for any possible transmission problem reported by the neighboring nodes. If a receiver does not receive the data packet after it has transmitted the CTS packet for WAIT_FOR_DATA time units, it transmits NAK packet. If the sender does not receive any NAK packet before its WAIT_FOR_NAK timer expires, the broadcast/multicast service is complete. Otherwise, the sender backs off and enters the contention phase again to retransmit

the data packet. Broadcast Medium Window (BMW) [54] mainly considers supporting reliability for broadcast but it can also support multicast. The basic idea of BMW is that a node reliably transmits a broadcast packet to each of its neighbors in a round robin fashion. The neighbor list is obtained by both periodical HELLO messages and overhearing. Once a packet other than HELLO message is transmitted by a node, the node will suppress its next HELLO message, assuming neighbor nodes can know its presence by overhearing this packet. The main drawback of the BMW is that it uses at least m rounds of unicasts for a broadcast/multicast packet addressed to its m neighbors, which does not only introduce at least m rounds of contention phases, but also makes no use of the broadcast nature of the wireless channel. This protocol is very similar to our unicast forwarding mechanism except that it operates at the MAC layer while ours is located in Network layer.

Considering the problem of BMW, Batch Mode Multicast MAC protocol (BMMM) [55] proposes to consolidate the m contention phases into a single one and transmits the data packet only one time before the ACK collecting. To reliably transmit a multicast packet in BMMM, a sender first uses its RTS packets to request intended receivers one by one to reply with a CTS. If the sender receives at least one CTS, it transmits the data packet. After the data packet transmission, it uses a new control packet called RAK (Request for ACK) to request ACKs from the intended receivers one by one. In case of missing ACKs, the sender will do retransmission. All the intervals between the above sequence of packets are set to a value less than DIFS, so once the sender grabs the channel, the reliable multicast operation will not be interrupted by other transmissions. It introduces m rounds of RTS/CTS exchange and RAK/ACK exchange, in which any of the $4m$ packets missing will cause retransmission.

BMMM uses less medium than the unicast transmission mode of our adaptive forwarding mechanism. However, it obtains a certain reliability at the cost of bandwidth consumption and bigger transmission delays. This protocol uses extra bandwidth for channel reservation and transmission acknowledgment compared to the plain CSMA/CA mechanism defined in the IEEE 802.11 specification. Furthermore, if these protocols operate alone, they might provide unnecessary reliability in some cases. For example, in Figure 3.2, node E is a multicast receiver for both node B and C at the MAC layer point of view . Thus nodes B and C could reliably send multicast packets to node E since reliable transmission from one node is enough. However, if BMMM cooperates with MRDC by replacing the unicast transmission mode, we can achieve a much better multicasting performance. Suppose that the MRDC constructs a multicast tree to connect source and receivers in Figure 3.2 and that node E is node B's downstream node on the tree. Node B can explicitly inform the MAC layer that receivers are node E and D. Node C does not need to reliably reach node E but node E is always able to receive multicast packet from node C. In this way, we can reduce bandwidth consumption and obtain redundant transmissions at the same time.

3.6 Conclusion

In this chapter, we introduced a best effort multicast routing protocol, called the Multicast Routing protocol with Dynamic Core (MRDC) for mobile ad hoc networks. The aim of this work is to reach a compromise between forwarding overhead and routing overhead so we can minimize the total bandwidth consumption. Because traffic packets are generally much more numerous and bigger than control messages, our strategy chooses the techniques which are most efficient for data transmissions while usually creates heavier control overheads. We developed some techniques which significantly reducing the control overhead with the cost of loosing some data transmission efficiency to obtain an optimal bandwidth utilization. Finally the routing protocol uses the best effort approach to deliver packets.

According to this idea, MRDC is split into two planes: the control plane and the forwarding plane. The control plane chooses tree structure to achieve the best data transmission efficiency. As nodes move and the network configuration changes, the tree is periodically reconfigured to maintain efficiency. A multicast tree is rooted at the first source of a multicast session. Therefore, since the tree is source-based, MRDC achieves the best data transmission efficiency for in a single source multicast session. In multiple source applications the tree becomes group-shared to reduce control overhead. Another improvement to reduce the control overhead is that the construction and maintenance of any multicast tree are triggered in an on-traffic-demand basis at the cost of introducing transmission delays for the first packets. Faults are tolerated in the trees, which might affect packet delivery on the concerned sub-tree but can avoid heavy control overheads to maintain a strictly correct tree. MRDC only needs a small quantity of control overhead but can provide a delivery structure which generates less forwarding overhead, therefore reducing the total bandwidth consumption.

The forwarding plane is in charge of delivering multicast packets on a best effort basis. Considering the shortcoming of the current 802.11 standards in multicast packet transmission, we introduced an adaptive multicast forwarding mechanism in MRDC's forwarding plane for 802.11 wireless networks. This mechanism makes a transmission mode choice based on two metrics: Average Queue Length (AQL) and Medium Occupation for Reception (MOR). If both metrics are smaller than their respective thresholds, the forwarding mechanism treats a multicast packet as a set of unicast packets and delivers them with the RTS/CTS option of IEEE 802.11. If it is not the case, the forwarding mechanism passes a multicast packet directly to the MAC layer. Through this mechanism, MRDC provides some degree of reliability in one hop multicast delivery. Thus, we can offer a better service for upper layer applications. If they are sensitive to delay or benefit from some other mechanisms to recover delivery failures, MRDC can always use the broadcast mode to achieve short delays and low bandwidth consumption. For the other applications, MRDC is able to offer an optimal packet delivery ratio with respect to the network load. That is what we call **best-effort multicasting**.

We estimated the routing overhead and forwarding overhead of MRDC. This

overhead is a function of the multicast tree size which is further decided by the distribution of group members in the network and their distance to the core. In the next chapter, we will study the performance of MRDC in a packet level simulator. After selecting the suitable key metrics for MRDC such as the period of tree refresh (PERIOD_REF) and the thresholds of mode selection procedure, we evaluate the size of MRDC multicast tree and discuss the efficiency and robustness of MRDC under different movement and traffic scenarios. We then compare its performance to other multicast routing protocols.

Chapter 4

Performance Analysis of Multicast Routing Protocol with Dynamic Core (MRDC)

In chapter 3, we introduced our proposal: Multicast Routing Protocol with Dynamic Core (MRDC), and briefly analyzed its performance in terms of routing overhead and forwarding overhead. In this chapter, we evaluate the performance of MRDC through a detailed packet level simulation using the network simulator, ns-2 [9] in order to obtain better understanding of this protocol. This performance analysis has two goals: to select key MRDC parameters (e.g. period of multicast tree refresh and threshold for average queue length) and to analyze its performance for different traffic loads and mobility patterns. The performance analysis is further divided into two parts: multicast tree analysis and protocol comparison.

The rest of this chapter is organized as follows. Section 4.1 introduces the simulation environment followed by Section 4.2 in which we present the movement pattern and traffic pattern that will be used in the simulations. Section 4.3 describes the chosen implementation. Section 4.4 chooses optimal parameters of MRDC. With these parameters, we first analyze the correctness and robustness of multicast tree in Section 4.5. And then section 4.6 evaluates the performance of MRDC in comparison with some other multicast routing protocols. Finally, Section 4.7 concludes this chapter.

4.1 Simulation Environment

We conduct our simulations using the *ns-2* network simulator [9], with MONARCH project wireless and mobile extension ([48] and [57]). *Ns-2* is a publicly available discrete, event-driven simulator developed by the University of California at Berkeley and the VINT project. MONARCH project at Carnegie Mellon University extended ns-2 to provide support for simulating multi-hop wireless networks completed with signal strength, radio propagation, data link layer and IEEE802.11

MAC protocol[46]. A comparison of ns-2 with other popular simulators such as OPNET [58] and GloMoSim (QualNet) [59] can be found in [60] and [61].

Our simulation models a network of 50 mobile nodes placed randomly within a 1000mx1000m flat space. The physical radio characteristics of each mobile node's network interface, such as the antenna gain, transmit power and receiver sensitivity, are chosen to approximate the Lucent/Agere WaveLAN [62] direct sequence spread spectrum radio characteristics. The nominal bit-rate is 2 Mb/s and the nominal radio range is 250 meters, depending on the capture effect and packet collision. The link layer model is the Distributed Coordination Function (DCF) of the IEEE 802.11 wireless LAN standard. We have extended the existing simulation modules to enable multicast simulations with MRDC. Each node has a priority queue, called an interface queue, for packets awaiting transmission of the network interface. This queue gives priority to routing messages. It holds up to 64 packets and is managed in a drop-tail fashion.

4.2 Simulation Scenarios

A number of movement scenarios and traffic scenarios are generated and used as inputs to the simulations. Each movement scenario file determines movements of 50 nodes. The movement model of nodes is the random waypoint model [48] without pause. Each node begins the simulation by selecting a random destination in the 1000mx1000m space and moves to that destination at a speed distributed uniformly between 0 and a maximum movement speed. Upon reaching the destination, the node selects another destination, and moves there as previously described. Nodes repeat this behavior for the duration of the simulation. Each simulation runs for 900 seconds of simulation time. Movement patterns are generated for different maximum speed. When maximum speed equals to 0, nodes do not move during a simulation which represents stable networks. A low maximum speed results in a low relative movement speed of nodes and corresponds to low mobility scenarios. On the contrary, a high maximum speed means high relative movement speed among nodes and corresponds to high mobility. Because the performance of the protocols is very sensitive to node position and movement pattern, we generated 10 movement scenarios for each value of maximum speed. Thus, each collected data in figures and tables presents an average of these 10 movement scenarios with the same maximum speed. Network partition is tolerant in mobility scenarios while excluded in stable networks.

Traffic scenarios determine the number of groups, group members and multicast traffic. A number of nodes are chosen as multicast group members. To reduce side effects, membership control features are turned off. All group members join the multicast session at the beginning of the simulation and remain as members throughout the simulation. Multicast traffic is generated by constant bit rate (CBR) sources. Each source sends 4 packets per second. The size of data payload is 512 bytes. The transmissions start at times uniformly distributed between 30 and 60

simulation seconds and continue till the end. These sources are attached to nodes which were chosen among multicast members. The number of groups is mode two of the number of sources. For example a 5-source traffic scenario defines 3 multicast groups among which 2 groups have respectively 2 sources and the third one has one source. This configuration forms not only inter-group competition but also intra-group inter-sources competition.

4.3 Implementation Decisions

In implementing the MRDC in *ns-2*, we made following decisions. The Greatest-Range of JI message propagation is 4 hops. Upstreams wait for 0.5 seconds before broadcasting another JI message. Downstreams set the multicast routing entry to *tree-fault* state 1.5 seconds after detecting edge broken.

NEIGHBOR_HELLO period is set to 0.5 second and the timer of the active neighboring entry is set to 1 second in the simulations. In order to improve bandwidth efficiency, MAC layer cooperation is used in updating active neighbor table. When a node successfully sends or receives a packet to/from a neighbor, it updates the corresponding entry in active neighbor table because the MAC layer control message (RTS, CTS and ACK) is received from the neighbor.

4.4 Parameter Selection

The simulations in this step attempt to achieve a suitable period value for multicast tree refresh and optimal thresholds for transmission mode selection. These parameters will be used in the simulations of the performance analysis.

4.4.1 Period of multicast tree refresh

The period of multicast tree refresh is an important parameter of MRDC, which has a direct impact on the performance of protocol. The longer the period is, the more slowly MRDC reacts to topology changes and thus more faults might exist in the multicast tree. That reduces the number of packets delivered to receivers. On the other hand, a shorter period means frequent network range broadcast which increases routing overhead significantly. Therefore, an ideal refresh period (PERIOD_REF) should permit this protocol to deliver as many multicast packets as possible without creating significant routing overhead. For this reason, the following two metrics are employed to select the period of tree refresh.

- **Packet delivery ratio**: the ratio of the number of multicast data packets correctly delivered to the receivers versus the number of multicast data packets supposed to be received.

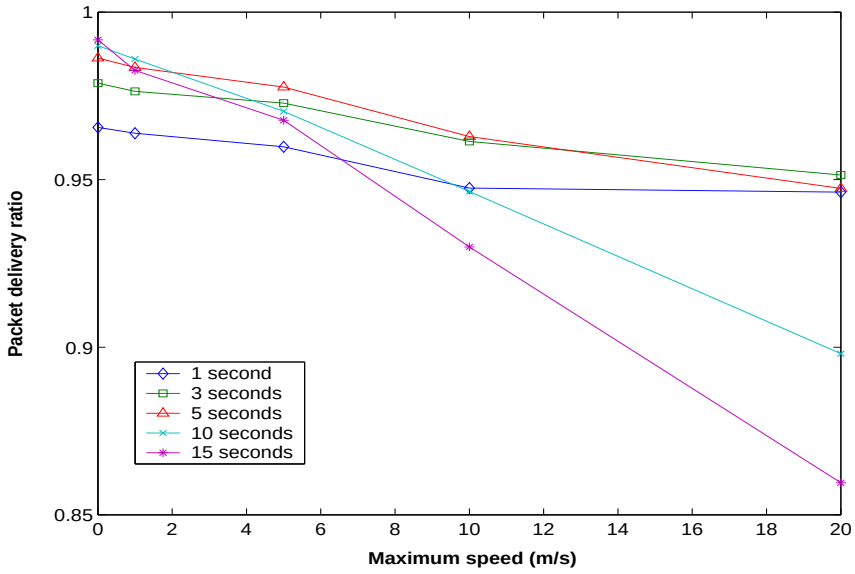
- **Number of control messages per second**: The rate of MRDC control messages transmitted for multicast tree construction and maintenance. This metric is used to investigate the resource consumed by multicast routing protocol.

Because periodic tree refresh mainly addresses topology changes, we use different movement scenarios without changing the traffic scenario in this step. The maximum movement speed is varied from 0m/s (stable networks) to 20m/s (high mobility networks). A traffic scenario in which one multicast group contains 10 members and two traffic senders is chosen to simulate a group-shared case. One sender plays the role of core and the other one acts as normal group member. Mode selection is disabled in the simulations. All routers broadcast multicast packets.

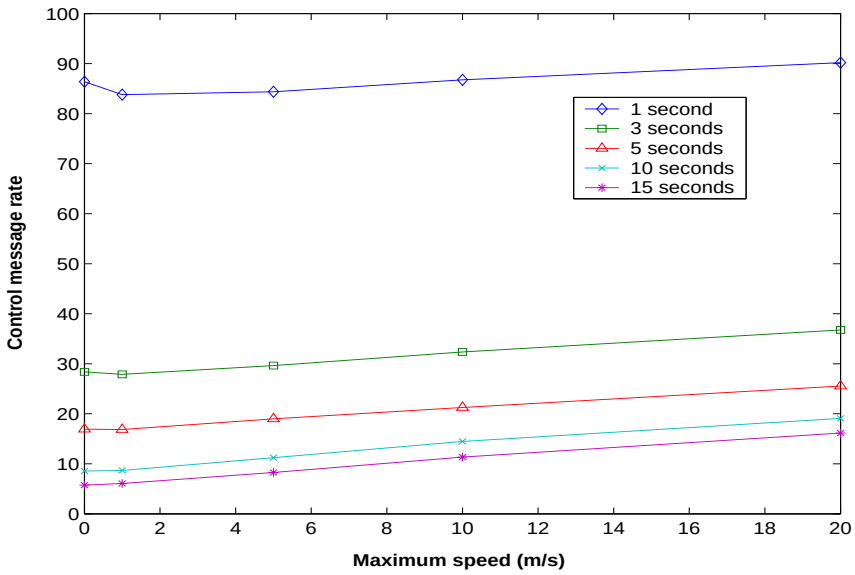
The simulation results are shown in Figure 4.1. The packet delivery ratio decreases with the increase of mobility speed but in shorter periods it resists better than in longer ones, as illustrated by Figure 4.1(a). A tree structure offers the unique route to distribute data packet from sources to receivers. Once topology changes touch the multicast tree, packets transferred on the broken branch(es) will be dropped. High relative movement speed causes a high degree of topology changes that in turn gives a high tree break rate. A shorter tree refresh period produces more frequently reconfiguration and consequently can react more quickly to topology changes. That is why short PERIOD_REF is robust against topology. We will study more deeply the impact of mobility on a multicast tree in the performance analysis section.

In terms of achieving a better packet delivery ratio, Figure 4.1(a) shows a contradiction that low mobility networks favor long periods while short periods are preferable in high mobility networks. After analyzing the reasons for packet delivery failure, we find the reason for this contradiction. Besides low layer transmission failure and routing protocol, packet delivery failure is also caused by the bad cooperation between the control plan and the forwarding plan. In period tree refresh, MRDC first destroys the old tree and then constructs a new one. Therefore, muulticast packets cannot be correctly delivered to all receivers before a new tree is completely constructed. More frequent tree refresh causes more delivery failure relative to this fact. Believing that a smart forwarding mechanism can greatly reduce this type of delivery failure, short PERIOD_REF is preferred in all mobility cases.

As shown in Figure 4.1(b), bigger PERIOD_REF values generate a smaller number of routing messages to construct and maintain multicast tree, while their control overhead increases more quickly than that of smaller ones with the increase of mobility. A high degree of topology changes makes MRDC generate more control messages for local tree recovery. Frequent tree reconfiguration alleviates this requirement. Thus node mobility has less of an effect on control overhead of short PERIOD_REF than long ones. However, in all the cases, shorter PERIOD_REFs generate more overhead.



(a) Packet delivery ratio as a function of Maximum speed and PERIOD_REF



(b) Routing overhead as a function of Maximum speed and PERIOD_REF

Figure 4.1: MRDC's performance under different period of multicast tree refresh

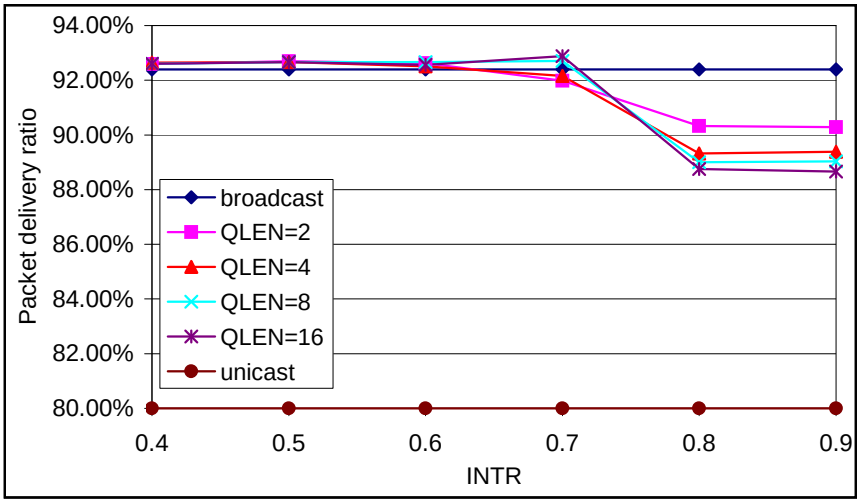
Short PERIOD_REF makes the protocol robust against topology changes. While, long PERIOD_REF makes protocol efficient with low control overhead. In the rest of simulations, we use 5 seconds as PERIOD_REF since in this case MRDC can deliver more than 94% data packet and create less than 5% routing overhead.

Thresholds of Mode Selection Procedure

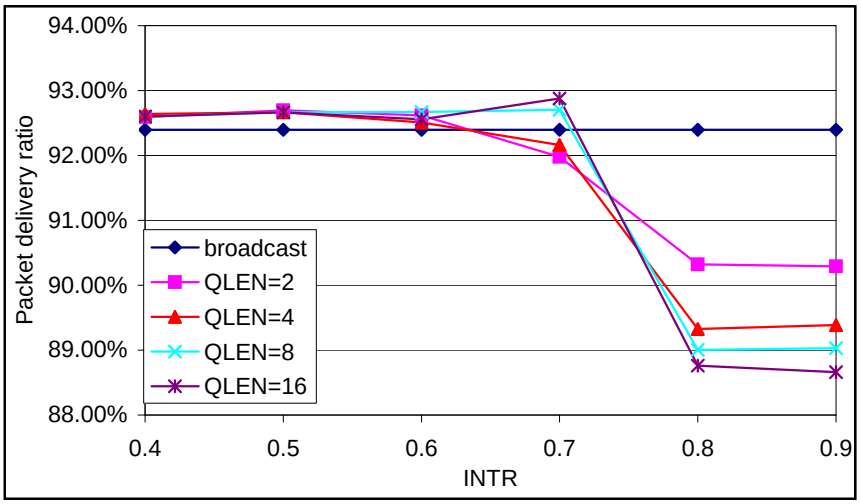
In this section, we study the impact of QLEN and INTR on the performance of adaptive multicast forwarding mechanism to obtain optimal thresholds. We set node's maximum movement speed to 5 m/s and choose 6-source traffic scenario, because this is the traffic scenario in which broadcast mode begins to outperform unicast mode (see Figure 4.4 in Section 4.6). This scenario defines three multicast groups and each group has 10 members and two CBR sources. We vary the QLEN from 2 to 16 and INTR from 0.5 to 1.0. MOR is always less than 1.0 because it does not consider medium occupied by a node itself for sending packets. Thus, by setting INTR to 0.9, which makes the metric MOR always smaller than its threshold, we simulate the case where MAC layer counters are unavailable. For comparison reason, we also test the performance of MRDC in the cases in which all nodes operate in broadcast mode (set INTR=0 for example) or in unicast mode (QLEN=65 and INTR=1.0). The former case is denoted as *broadcast* and later as *unicast*. Following two metrics are employed in the simulation of mode selection threshold:

- **Packet delivery ratio**: Same as that in the Section 4.4.1
- **Average end-to-end delay**: the average time between that a packet is initiated by a source and that it is received by multicast receiver. This metric is important for some applications sensitive to delay. This metric can also demonstrate how much transmission delay is introduced by unicast mode.

Figure 4.2 (a) shows that the packet delivery ratio of the adaptive multicast forwarding mechanism as a function of INTR and QLEN. In order to show better the details of the performance curves, we enlarge the y-axis scale range from 88% to 94% and show the result in Figure 4.2 (b). The simulation results show that the adaptive multicast forwarding mechanism provides the best packet delivery ratio when INTR equals to 0.7 and QLEN is 16. If the MAC layer counter is not available, the QLEN should be set to 2. Small INTR and/or QLEN makes nodes easily switch from unicast mode to broadcast mode and stay in broadcast mode. Thus the corresponding results are similar to those of broadcast case, in which all nodes operate in broadcast mode. However, the nodes which do not locate in hot spot continue to use unicast that improves the packet delivery ratio compared to broadcast to reduce multicast transmission failure. As INTR and QLEN increase, more and more nodes tend to operate on unicast transmission mode, which gives a performance comparable to that of unicast case.

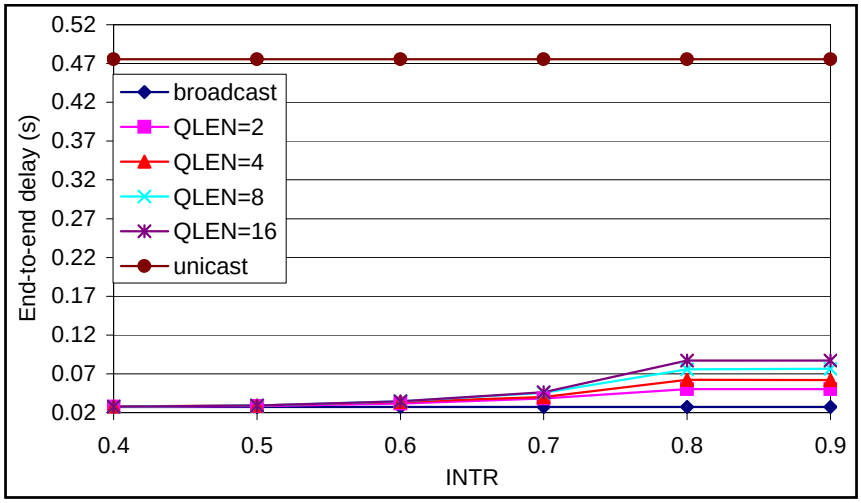


(a) Original (including unicast case)

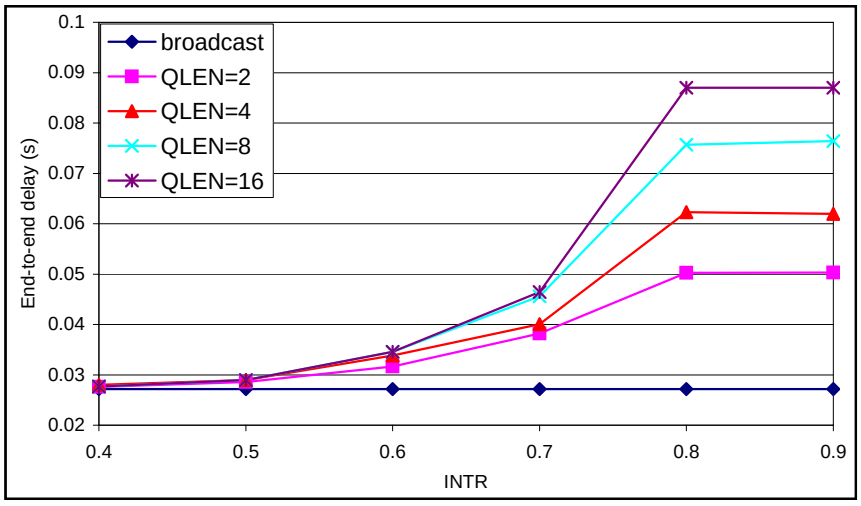


(b) Enlarged (without unicast case)

Figure 4.2: Packet delivery ratio v.s. QLEN and INTR



(a) Original (including unicast case)



(b) Enlarged (without unicast case)

Figure 4.3: End to end delay v.s. QLEN and INTR

In general, when QLEN remains unchanged, packet delivery ratio increases first and then decreases as INTR increases. While, for a given INTR, the smaller the QLEN is, the better the packet delivery ratio we get. But this observation is no longer true when INTR equals to 0.7, smaller QLENs yield worse packet delivery ratios than bigger QLENs.This phenomenon is relative to a node's position and its transmission mode. When INTR and QLEN are small, only the nodes which are located at the network border operate on unicast mode. The increase of INTR and/or QLEN favors nodes located in central region or hot spot to stay on unicast mode instead of switching to broadcast mode. This tendency has two effects. On one hand, packet delivery ratio is improved by reducing transmission failure caused by the hidden terminal problem. On the other hand, this change degrades the performance because unicast mode prevents a delivery structure member getting packets from other links than those defined by the tree. When the members of the delivery structure located in the region far from the center, there is little possibility to have redundant transmission. Thus unicast mode is preferable. On the contrary, for the center nodes which have larger possibility to possess redundant transmission and normally at same time have more traffic to forward, broadcast mode is appreciated. For this reason, packet delivery ratio increases first and then decreases.

Figure 4.3 (a) illustrates end-to-end delay evolution as a function of INTR and QLEN. Figure 4.3 (b) is just an enlargement of the bottom part of Figure 4.3 (a). The results prove that unicast mode creates more delay than broadcast mode but has different consequences depending on which nodes practice this mode. When only border nodes employ unicast mode, the number of nodes in the structure neighbor entry remains small and the extra delay is tolerable. On the contrary, center nodes usually have a big structure neighbor list. Sending a multicast packet one by one to these neighbors creates a large transmission delay. Thus, in terms of small transmission delay, INTR should be small. However, a small QLEN is appreciated in both packet delivery and delay.

INTR plays a more important role on the performance (Figure 4.2 and 4.3) when it is less than 0.7, and after that QLEN has an effect on both packet delivery and delay. A small INTR makes node easily think that it is in a hotspot before packets accumulate in the queue and as a result the influence of QLEN is reduced.

Although the ($INTR = 0.7, QLEN = 16$) pair gives the best packet delivery ratio, it generates much greater delivery delay. To get a compromise between delivery ratio and delivery delay, we use the ($INTR = 0.6, QLEN = 8$) pair as the thresholds of the mode selection in the performance comparison simulations.

4.5 Multicast Tree Analysis

We evaluated the performance of MRDC multicast tree in a variety of mobility and communication scenarios. Because we focused on the control plan of MRDC, the performance in this chapter relates to the efficiency and robustness of the multicast tree. Mode selection of forwarding plan is disabled and multicast routers broadcast

multicast packets. Although the aim of this simulation is to evaluate the robustness and efficiency of multicast tree, we still introduce multicast traffic to test the performance for different network loads.

4.5.1 Simulation Metrics

Tje performance analysis aims to demonstrate the robustness and efficiency of MRDC multicast tree. The robustness is to test whether the multicast tree keeps connecting and covers all reachable group members when the network topology changes or control messages are lost. On the other hand, the efficiency means whether the potential forwarding overhead and routing overhead of MRDC multicast tree scale well with different mobility and traffic scenarios. The following metrics are chosen:

- **Average number of multicast routers**: This metric counts the average number of nodes in the multicast tree which transmit multicast packets during a simulation. It allows us to estimate the forwarding overhead in terms of the number of packets forwarded to deliver a multicast packet to receivers in broadcast mode and under an ideal condition (for example without transmission loss). Thus this metric provides the scalability and efficiency of multicast routing protocol.

- **Average number of non-member routers**: It measures the means of the number of nodes which are on the multicast tree but at the same time not the group member. This metric can be used to compute routing overhead in a periodic tree refresh but also the forwarding overhead of unicast transmission mode. This metric gives the value of parameter x in Formula 3.1. Applying the other two predefined parameters, number of mobile nodes and number of group members, we can calculate the routing overhead of periodic tree refresh in a simulation. The number of non-member routers plus the number of group members gives the total number of nodes in the tree. That is the forwarding overhead in unicast transmission mode.

- **Number of tree repair times**: This metric counts the number of local tree repair times initiated by MRDC. MRDC's routing overhead comes from periodic tree refresh and local tree recovery. For a given simulation time, the routing overhead generated for periodic tree refresh can be calculated from Formula 3.1. The routing overhead of local tree recovery varies scenario to scenario. Therefore, this metric allows us to estimates the variation of routing overhead of MRDC in different scenarios.

- **Tree broken times**: It counts how many times the simulator detects that the multicast tree is broken during a simulation. Supposing that all multicast routers operate on broadcast transmission mode to deliver multicast packets, the simulator checks whether all group members within the same network

(partition) as the core are reachable through the multicast tree. In other words, tree broken here means physical fragmentation of multicast tree, since the simulator does not verify logical relationship among tree members. This metric reflects the robustness of MRDC.

To calculate these metrics, after multicast sources begin their transmission, the simulator reports every half second the number of total tree nodes and interior tree nodes and whether the tree covers all reachable group members. Reachable group members are the group members which are within the same network (partition) as the core or in other words core can reach these members directly or through some other nodes. There are in total 1696 such reports during a simulation. Interior tree nodes are tree members that have downstream nodes. The number of total tree nodes minus reachable group members gives the number of non-member routers. A multicast tree covers all reachable group members if the core can reach all other group members within the same network partition through the tree. In the other case, the multicast tree is called broken.

4.5.2 Performance analysis

Mobility Speed

In this experiment, the maximum movement speed is varied from 0 m/s to 20 m/s to examine the robustness of the protocol against topology changes. One multicast group containing 20 members is simulated. The network load is set to very light (1 source) to exclude as much as possible the influence of traffic packets on control message transmission.

Maximum speed (m/s)	Average non-member routers	Average interior node	# of tree repair times	# of tree broken times
0	8.24	14.81	0	8
1	7.23	12.61	51	9
2	7.05	12.51	91	19
5	7.10	12.72	205	44
10	7.10	13.06	344	74
15	6.48	12.44	466	94
20	6.64	12.84	596	122

Table 4.1: Performance of MRDC multicast tree as a function of Maximum mobility speed

Table 4.1 illustrates the performance of MRDC multicast tree as a function of maximum movement speed. It shows that the MRDC multicast tree is scalable in terms of forwarding overhead and remains reasonably correct as topology change increases. The forwarding overhead in broadcast mode remains stable since the

number of interior node slightly changes in dynamic networks. For the forwarding overhead of unicast mode, node mobility even decreases slightly the size of multicast tree. This can be obtained by adding the number of group members to the number of non-member routers. The result decreases from 28.23 (=20+8.23) to 26.64 (=20+6.64). One reason is that movement makes node uniformly distributed in network, and as a result, the distances, in terms of number of hops, from group members to the core are reduced as shown in Table 4.2. In dynamic networks, we do not exclude network partitioning. Network partitioning makes some group member temporarily unreachable (see Table 4.2) which consequently also reduces the requirement of multicast router.

Maximum speed (m/s)	Distance (# of hops) from members to core	Times of members unreachable to core
0	63.3	0
1	49.6	0
2	49.1	0.0428
5	47.8	0.1694
10	47.0	0.1623
15	45.5	0.2077
20	45.8	0.1201

Table 4.2: Multicast group in mobility simulations: distance and unreachable time

Table 4.2 also demonstrates the advantage of multicast comparing with unicast and broadcast in delivering a packet to multiple receivers. The distance in terms of the number of hops from the core to a multicast member is exactly the forwarding overhead of sending a packet to that member. The second column of Table 4.2 gives the forwarding overhead of unicast. This column divided by the third column of Table 4.1 gives the gain of multicast method. That of broadcast method is 50, the number of nodes in the network. Table 4.3 compares the forwarding overhead of unicast, multicast and broadcast methods and shows that multicast can at least reduce 3 times the forwarding overhead.

Both the number of local tree repair and the number of trees broken detected by simulator increase with the node mobility, which increases the control overhead for local tree recovery, but with different speeds. The number of tree repair increases more quickly than tree broken times does. In fact, the number of tree repair times is determined by the frequency of link changes during simulations as shown in Table 4.4. The number of link changes is about 14-16 times of tree repair times. Therefore, the cost of maintaining a multicast tree in dynamic networks increases as the number of link changes in the network.

We observe that the simulator detects physical tree segmentation in stable network simulations. This phenomenon is due to control messages, especially CA messages lost during their transmission. The multicast tree is consequently not

Maximum speed (m/s)	Forwarding overhead (unicast/multicast)	Forwarding overhead (broadcast/multicast)
0	4.3	3.4
1	3.9	4.0
2	3.9	4.0
5	3.8	3.9
10	3.6	3.8
15	3.7	4.0
20	3.6	3.9

Table 4.3: Multicast group in mobility simulations: distance and unreachable time

Maximum speed (m/s)	Tree repair times	Link changes	Link changes to tree repair ratio
0	0	0	-
1	51	711	13.94
2	91	1285	14.12
5	205	2810	13.71
10	344	4999	14.53
15	466	7339	15.75
20	596	9474	15.90

Table 4.4: Tree repair times v.s. the number of wireless link changes in mobility simulations

correctly constructed and does not cover all group receivers for that period. In dynamic networks, local recovery abandons branch repair after several attempts and leave periodical tree refresh to overcome such errors. This leads to an augmentation of tree fragmentation. However, when maximum movement speed is 20 m/s, simulator detects about 122 broken times over a total 1696 reports during a simulation. This means that even in high mobility scenarios, MRDC multicast tree provides connectivity in 92% of time.

Multicast Group Size

We varied the number of multicast group size from 5 to 40 members to investigate the scalability of the protocol. The number of sources is fixed at 1 and maximum speed at 1m/s to reduce as much as possible the affect of node's movement.

Group size	Average non-member routers	Average interior node	# of tree repair times	# of tree broken times
5	4.23	5.63	17	5
10	6.90	9.33	30	6
15	7.32	11.10	40	8
20	7.23	12.61	51	9
25	6.80	13.97	57	11
30	6.02	15.14	64	15
35	4.88	16.16	72	11
40	3.61	16.90	76	10

Table 4.5: Performance of MRDC multicast tree as a function of Multicast group size

The performance of the multicast tree as a function of group size is shown in Table 4.5. These results show that MRDC multicast tree scales well and is robust facing group growth. The number of non-member routers increases firstly and then decreases. When the group size is small, group members may position in different direction from the core. To cover group members placed in different directions, MRDC should involve more nodes in multicast routing. But as the number of routers reaches a certain value, the multicast tree can cover most of the network. Only a small number of extra nodes are needed to connect the new members which are in an uncovered area. As a result, the number of average interior nodes increases logarithmically with the number of group members. To deliver multicast packet to 5 additional receivers, only one more forwarding is sufficient when the group already contains 30 members. And from 35 members to 40 members, the forwarding overhead increases by less than one.

As the group size increases, the size of the multicast tree increases. As a result, the multicast tree suffers more and more of wireless link changes. MRDC should run more local recovery to maintain these connectivities. On the other hand, if

the trunk of multicast tree, which is formed by interior nodes, keeps physically connecting and covering most part of the network, the movement of node creates less tree segmentation. Some leaf nodes may just move from the coverage range of one interior node to another interior node. Or when an interior node moves away, other interior nodes will cover its leaf nodes. Therefore, the frequency of tree broken times decreases after 30-member scenarios since the coverage of multicast tree increases.

Number of Senders

In this experiment, we examine the performance of the multicast tree constructed by MRDC with a different number of senders which ranges in the set 1,2,3,4,6,8,12. The approach used in the simulations to transmit broadcast messages (CA messages, JI messages) and multicast packets on top of an IEEE 802.11 protocol is by flooding. Naively broadcasting by flooding may cause serious redundancy, contention, and collision in the network, which is called the broadcast storm problem in [63]. We analyzed the collision problem in Section 3.4. The higher the network load is, the greater the possibility of control message loss. However, the contention problem is also important because it delays control message transmission. This experiment thus demonstrates the robustness of MRDC facing control message loss or delays. The multicast group size is set to 20. Node mobility speed is less than 1m/s so that the impact of topology change can be ignored.

Number of Senders	Average non-member routers	Average interior node	# of tree repair times	# of tree broken times
1	7.23	12.61	51	9
2	7.18	12.57	49	25
3	7.20	12.57	49	30
4	7.21	12.59	47	32
6	7.19	12.53	48	42
8	7.20	12.55	46	86
12	7.44	12.74	45	169

Table 4.6: Performance of MRDC multicast tree as a function of Number of senders

Table 4.6 demonstrates the performance of the MRDC multicast tree as a function of the number of multicast senders. It shows that MRDC tree is sensitive to control message loss. Tree broken times increase dramatically when the number of senders changes from 6 to 8 then 12. At the same time, average interior node number and tree repair times remain nearly unchanged. Control message and multicast packets contend on medium access because they share the same wireless channel. Heavy traffic load causes high loss rate due to collision for the broadcast control messages (CA message, JI message) and delays the transmission of the unicast con-

trol messages (RAR, RAA and Recovery messages) because of heavy contention. Losing or delaying a control message has a direct effect on the performance of MRDC. For example, if a node fails in transmitting a CA message to its neighbors, it might make the constructed multicast tree incorrect and/or sub-optimal. The multicast tree does not contain the necessary nodes to covers all group members, which reduces the number of routers in the tree. On the other hand the shortest path is not discovered and tree is constructed by a longer path, which increases the number of routers in the tree. As a result, we can observe that both the number of interior nodes and average non-member routers remain stable till 8-sender traffic scenarios and finally slightly increase in 12-source scenarios. CA message transmission failure and delayed RAR and RAA message transmission increases the time for multicast tree re-construction. The long duration of tree reconfiguration results in tree broken times increasing quickly in high load networks because simulator considers the tree is broken if it runs the tree test before MRDC finishes the multicast tree reconfiguration.

The above simulation results show that MRDC provides correct multicast tree most of the time. It can efficiently support multicast delivery in most cases. However, the trees become fragile when nodes' mobility increases. This protocol relies greatly on the correctness and promptness of routing message transmission. These points should be taken into account in future work.

4.6 Protocol Comparison

After choosing the key parameters of MRDC and analyzing the efficiency and robustness of MRDC multicast tree, in this section, we study the performance of MRDC with some other multicast routing protocols to understand the behavior of MRDC under different mobility and traffic scenarios. We set PERIOD_REF to 5 seconds, INTR to 0.65 and QLEN to 6 because they give an optimal performance.

4.6.1 Comparison Multicasting Protocols

For comparison, we consider two versions of MRDC: MRDC-unicast, which forwards packets only in unicast mode, and MRDC-broadcast, which forwards packets only in broadcast mode. The MRDC integrated with adaptive forwarding mechanism is denoted as MRDC-adaptive in the simulations.

The Rice Monarch project [57] at Rice University has made multicast extensions for ns2 [9]. The extensions include implementations of the Adaptive Demand-Driven Multicast Routing protocol (ADMR) [38] and the On-Demand Multicast Routing Protocol (ODMRP) [24] for routing in wireless multi-hop ad hoc networks [1]. We take these two protocols as references in protocol performance comparison. Here, we give a brief overview of these two protocols.

[1]The source code can be found from http://www.monarch.cs.rice.edu/multicast_extensions.html

Adaptive Demand-Driven Multicast Routing protocol (ADMR)

ADMR constructs a source-oriented loose multicast tree on traffic demand. In ADMR, receivers must explicitly join a multicast group. Sources periodically send a network-wide flood, but only at a very low rate in order to recover from network partitions. In addition, forwarding nodes in the multicast tree may monitor the packet forwarding rate to determine when the tree is broken or the source has become silent. If a link is broken, a node can initiate a repair on its own; if the source has stopped sending any forwarding state is silently removed. Receivers likewise monitor the packet reception rate and can rejoin the multicast tree if intermediate nodes have been unable to reconnect the tree.

To join a multicast group, an ADMR receiver floods a MULTICAST SOLICITATION message throughout the network. When a source receives this message, it responds by sending a unicast KEEP-ALIVE message to that receiver, confirming that the receiver can join that source. The receiver responds to the KEEP-ALIVE by sending a RECEIVER JOIN along this same unicast path. In addition to the receiver's join mechanism, a source periodically sends a network-wide flood of a RECEIVER DISCOVERY message. Receivers that get this message respond to it with a RECEIVER JOIN if they are not already connected to the multicast tree.

Each node which acts as a receiver or forwarder maintains a counter of recently received packets, and if a certain number of consecutive packets are not received by a receiver, it concludes that it has become disconnected from the group and it starts a repair process. A node that is a pure receiver (and not a forwarder for that source/group) simply rejoins the group by sending a MULTICAST SOLICITATION message. A node that is only a forwarder sends a REPAIR NOTIFICATION message down its subtree to determine whether it is the closest node to where the packet loss is occurring. Any downstream nodes cancel their own disconnect timers when they get this notification. Once a node has determined that it is the most upstream node that has been disconnected, it transmits a hop-limited flood of a RECONNECT message. Any forwarder which receives this message forwards the RECONNECT up the multicast tree to the source. The source in return responds to the RECONNECT by sending a RECONNECT REPLY as a unicast message that follows the path of the RECONNECT back to the repairing node.

A receiver keeps track of how many times it has had to initiate a repair due to a disconnection timeout. If this number reaches a certain threshold then the receiver believes that it has encountered a situation of high mobility. In the next RECEIVER JOIN message sent to the source, the receiver sets a high mobility flag as a signal to the source indicating that the network is encountering high mobility. When the source receives a particular number of join messages with the high mobility flag on, then it switches to flooding for a limited period. During flooding, all the data packets are sent as network-wide flood and all repair messages are suppressed.

In multicast forwarding, it obtains redundant retransmission by flooding multicast datagram in the tree.

On-Demand Multicast Routing Protocol (ODMRP)

ODMRP creates a mesh on traffic demand which contains the selected (shortest) path of each source destination pair, thus provides path redundancy. Nodes on the mesh form a "forwarding group" which forwards multicast packets via flooding (within the mesh), thus providing further redundancy. A soft state approach is taken in ODMRP to maintain multicast group members. Thus, no explicit control message is required to leave the group.

In ODMRP group membership and multicast routes are established and updated by the source on demand. When multicast sources have packets to send, but do not have routing or membership information, they broadcast a Join-Query control message to the entire network. When a node receives a non-duplicate Join-Query, it stores the source ID and the sequence number in its message cache to detect any potential duplicate. The routing table is updated with the upstream node ID and the node rebroadcasts the message. When the Join-Query message reaches a multicast receiver, it creates and broadcast a Join-Reply to its neighbors. When a node receives a Join-Reply, it checks whether the next node ID of one of the entries matches its own ID. If it does, the node realizes that it is on the path to the source and thus it is part of the forwarding group and sets the forwarding group flag. It then broadcasts its own Join-Reply built on matched entries. The next hop node ID field contains the information extracted from its routing table. In this way, each forwarding group member propagates the Join-Reply until it reaches the multicast source via selected path. This process constructs (or updates) the routes from sources to receivers and builds a mesh of nodes. Multicast senders refresh the membership and updates routes by sending Join-Query control message periodically.

ADMR and ODMRP Simulation Parameters

In performance comparison simulations, the parameters of ADMR and ODMRP are identical to those used in [38]. This means in ADMR, the periodic data flood interval is 30 seconds and 1.2 for multiplicative factor of the average inter-packet time in the absence of data, and 2 missing packets to trigger disconnection detection. For ODMRP, the Join-Query flood interval is 3 seconds, and a forwarding state lifetime of 3 times this interval (a total 9 seconds).

4.6.2 Metrics

We have used the following metrics in comparing protocol performance. Some of these metrics are suggested by the IETF MANET working group for routing/multicasting protocol evaluation [64]. We use the following four metrics in performance comparison:

- **Packet delivery ratio**: Identical to that in section4.4.1

- **Average end-to-end delay**: Identical to that in section4.4.1
- **Number of data packets transmitted per data packet delivered**: "Data packets transmitted" is the count of every individual transmission of data by each node over the entire network. This count includes transmission of packets that are eventually dropped and retransmitted by the intermediate nodes. Note that in unicast protocols, this measure is always equal to or greater than one. In multicast, since a single transmission (broadcast) can deliver data to multiple destinations, the measure may be less than one.
- **Number of control bytes transmitted per data byte delivered**: Instead of using a measure of pure control overhead, we chose to use the ratio of control bytes transmitted to data bytes delivered to investigate how efficiently control messages are utilized in delivering data. Note that not only bytes of control messages (e.g. beacons, route updates, join requests, etc.), but also bytes of data packet headers are included in the number of control bytes transmitted.

The two latter metrics concern bandwidth utilization. The number of bytes transmitted per data byte delivered can be considered as uniform forwarding overhead and the number of bytes transmitted per data byte delivered as uniformed control overhead. The sum of these two metrics is the uniform bandwidth consumption of each protocol.

4.6.3 Analysis Results

This section introduces the simulation results of the protocol comparison. We firstly test the performance of the multicast routing protocols in moderate dynamic networks with different traffic scenarios. And then we choose the 4-source traffic scenario to obtain the behavior of these protocols in different network mobility condition. The results with confidence can be found in Annex A.1.1.

Number of sources and groups

In a first step, we set the maximum speed of mobile nodes to 5 m/s and each group contains 10 members. We vary the number of sources from 2 to 8 to see the performance of these five protocols. We choose 8 sources as the maximum traffic load because this traffic scenario reaches the maximum network capacity of MRDC-unicast. According to the calculation of [65], the maximum throughput of a node in a regular lattice network is $1/12$ of the channel capacity. For 512-byte packets transmitted with RTS/CTS option, this is $\frac{1}{12} * \frac{512}{512+40+39+47} * 2 = \frac{1}{12} * 1.6$ Mbps, or 0.13 Mbps. Supposing a node is located in a hot spot where all traffic travels through it with a bit rate of $4 * 512 * 8 = 16$kbps, in 8-source case, its maximum capacity is reached. As the number of sources increases, the medium access contention and bandwidth consumption also increases. This contention comes from different groups, the same group but different sources and also the competitions

between multicast packets and control messages. Thus, this simulation tests the efficiency of these protocols in different network load cases and at the same time their robustness against control message error.

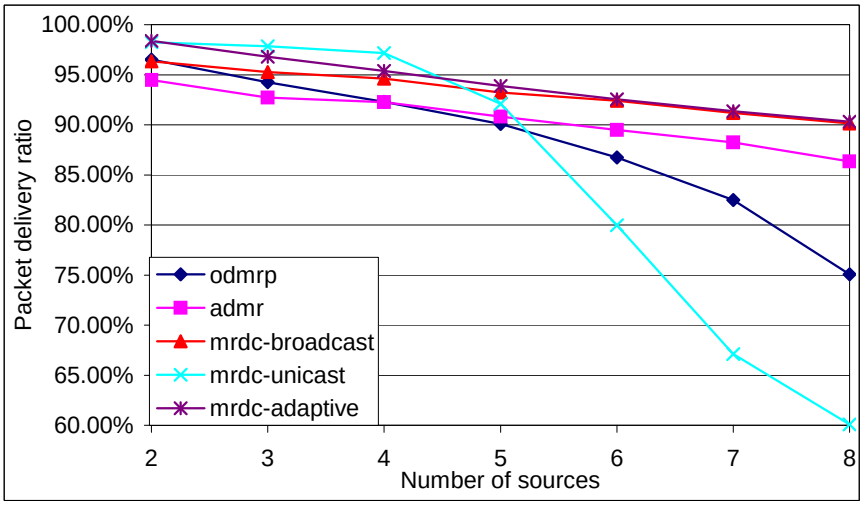


Figure 4.4: Packet delivery ratio v.s. Number of source

The packet delivery ratio of these protocols is illustrated in Figure 4.4. In low load networks in which the number of source is less than 5, MRDC-unicast delivers the most packets and ADMR gives the worst delivery ratio. MRDC-unicast and ODMRP degrade more quickly than other protocols. MRDC-broadcast has the most stable delivery ratio. On the other hand, except two points (3-source and 4-source cases), MRDC-adaptive is the best protocol in terms of packet delivery ratio.

In terms of transmission delay, as shown in Figure 4.5, MRDC-broadcast performs the best. In fact this protocol maintains the end-to-end delay at about 30 ms. MRDC-adaptive and ADMR generates a little more transmission delay than MRDC-broadcast. MRDC-unicast creates the most transmission delay in the scenarios of less than 7 sources and then is overtaken by ODMRP.

Forwarding overhead which is represented by number of data packets transmitted over data packet delivered is demonstrated in Figure 4.6. MRDC-broadcast provide the best performance in terms of forwarding overhead.

The control information efficiency of five protocols are compared in Figure 4.7. All three approaches of MRDC generate about three times less control overhead than ODMRP and ADMR.

As the network load increases, the performance metrics of all protocols degrade with a different trend. MRDC-broadcast is the most stable protocol in these simulations. It creates the least transmission delay, forwarding overhead and control overhead to provide a reasonably good packet delivery. The performance variation

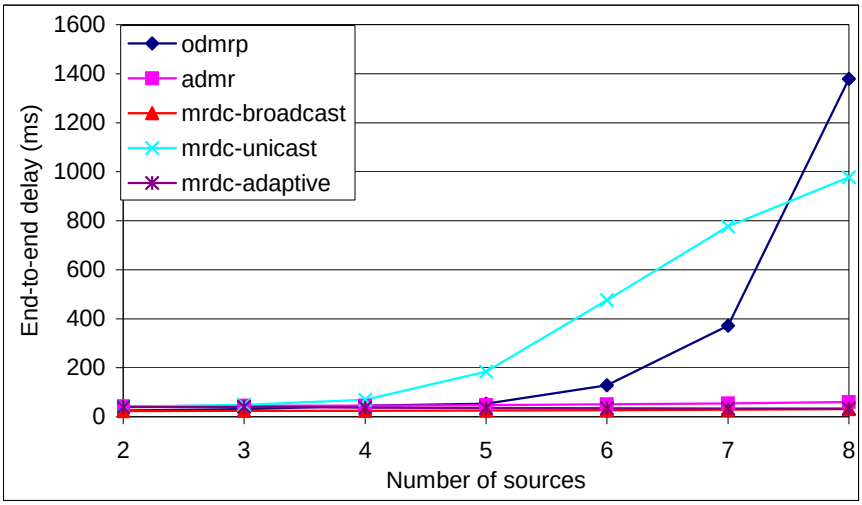


Figure 4.5: End to End delay v.s. Number of sources

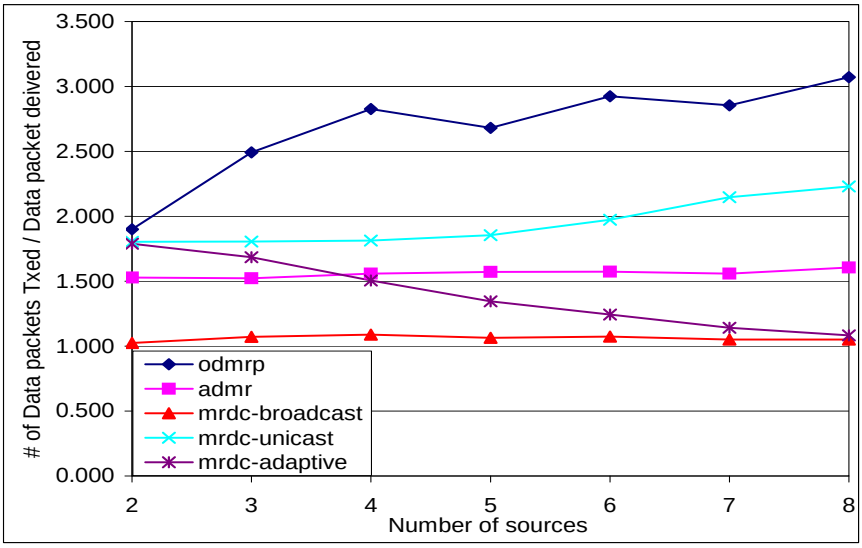


Figure 4.6: Number of data packets transmitted per data packet delivered v.s. Number of sources

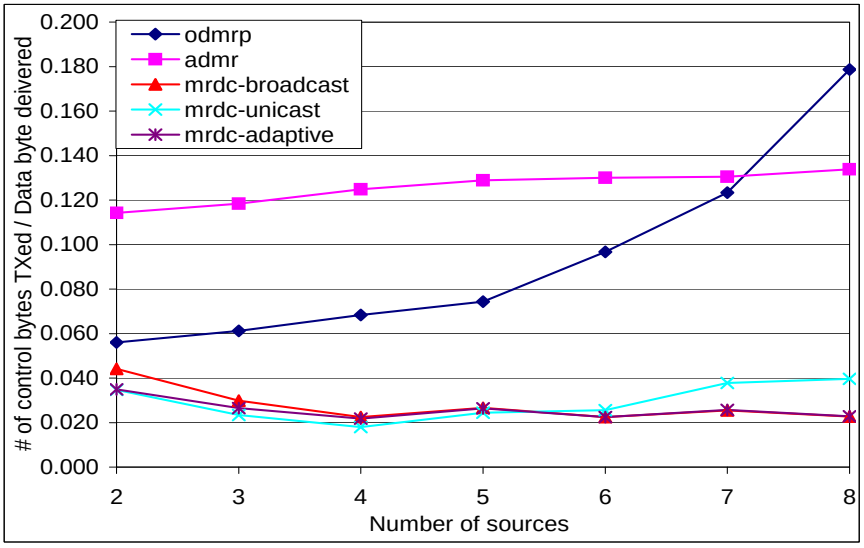


Figure 4.7: Number of control bytes transmitted per data byte delivered v.s. Number of sources

of ADMR is also limited but that of MRDC-unicast and ODMRP is much greater than others. We now study these protocols one by one to explain these phenomena.

ODMRP constructs mesh by including the shortest path of every source-destination pair. Each source periodically refreshes the path to the destinations. Routing overhead therefore is a function of the source number (Figure 4.7). ODMRP creates a forwarding state within nodes in the network, that is expired after a fixed timeout. This timeout is set to a multiple of the periodic JOIN QUERY flood interval in order to ensure that loss of the flood packets will not cause disruptions in the delivery of multicast data. However, this mechanism leads to the creation of redundant states in the network, since new nodes may become forwarders for a group, while forwarders created during a previous periodic flood still have a set forwarding flag and may overhear packets for that group. While the redundancy that ODMRP creates increases its resilience to losses, it significantly increases the load on the network (see Figure 4.6). As a result of the high load, overall network performance degrades and transmission delay goes up (Figure 4.4 and 4.5).

ADMR generates the most routing overhead according to Figure 4.7. This is on one side because it adds on each traffic packet a packet header which contains 32 bytes and sometimes 40 bytes. On the other side, ADMR constructs source-based tree. The fact that each source reconfigures its tree periodically results in routing overhead being a function of source number. ADMR also creates redundant states in the network when, as a result of tree breakage and repair, the forwarding tree no longer includes certain nodes that were part of the tree before the breakage of tree. However, nodes that forward for a source in ADMR expire their forwarding state

when there are no downstream nodes that are interested in receiving the multicast packets through them. It exhibits much less forwarding overhead than ODMRP (Figure 4.6) and delivery delay (Figure 4.5). However, the long period of source tree reconfiguration (30 seconds) compared with ODMRP and MRDC makes the tree structure non-adaptive to topology change in time and consequently delivers fewer packets than MRDC does in all cases and ODMRP does in low load cases (see Figure 4.4).

MRDC generates the least routing overhead because it uses group-shared multicast tree (see Figure 4.7). The routing overhead is mainly a function of group number. Adding new sources in a multicast group does not introduce important extra routing overhead. On the contrary, it improves the utilization of control messages since more traffic are delivered. That is why the ratio between control byte and data bytes delivered of MRDC-broadcast behaves in a wave-like manner. Theoretically, a group-shared tree generates more forwarding overhead than source-based tree in delivering traffic of non-core sources. However, MRDC does not create redundant states in the network and this fact permits MRDC-broadcast become the protocol which generates the least forwarding overhead (see Figure 4.6). The forwarding overhead difference between MRDC-unicast and MRDC-broadcast is about 0.75 in the simulations where MRDC-unicast correctly maintains multicast trees (when the number of sources is smaller than 5). Using the result of 2 sources in Table 4.7, the forwarding overhead of the unicast case is about $(10 + 6.66 - 1)/9 = 1.74$ and that of broadcast case is about $9.21/9 = 1.02$. This result confirms our previous discussion of forwarding overhead in these two modes. No redundant path in the delivery structure makes the multicast trees of MRDC sensitive to control message loss. MRDC-unicast aggravates this sensibility since this protocol delivers multicast packets respecting the tree structure. High forwarding overhead generated by MRDC-unicast delays control message transmission and even causes some messages loss. As a result the observed tree broken times in MRDC-unicast increases dramatically as the network load increases, while the multicast trees of MRDC-broadcast are well maintained (see Table 4.7). Thus, we can see that when the number of sources is less than 5, multicast tree can be correctly established so that MRDC-unicast gives the best packet delivery ratio (see Figure 4.8) due to RTS/CTS option. when greater than 5, the performance of MRDC-unicast degrades quickly and MRDC-broadcast outperforms all the other protocols in both packet delivery ratio and end-to-end delay. The simulation results also show that MRDC-adaptive not only provides a compromise between MRDC-unicast and MRDC-broadcast but also improves the packet delivery ratio of MRDC-broadcast by 3% in low load cases.

MRDC-broadcast resists much better facing network load increase and sometimes offers the best performance. MRDC-unicast is the most sensitive to network load increase. Unicast mode requires more bandwidth than broadcast mode. In high load networks, this mode generates congestion and delays the transmission of not only multicast data packet delivery but also routing messages. Table 4.7 shows that the number of multicast tree breaks under MRDC-unicast dramatically

Number of Sources	Average non-member routers	Average interior node	Tree broken times (broadcast)	Tree broken times (unicast)
2	6.66	9.21	40	30
3	6.74	9.26	39	31
4	6.75	9.27	35	39
5	6.81	9.29	42	90
6	6.77	9.29	38	160
7	6.75	9.30	41	218
8	6.78	9.30	41	247

Table 4.7: Multicast tree of MRDC-broadcast and MRDC-unicast as a function of Number of sources

increases as the network load does. Here, we only count physical tree fragmentation, while the number of logical fragmentation is much bigger than the number given in this table. Tree-based protocols depend more than mesh-based ones on the correctness and punctuality of routing message transmission to maintain delivery structure. Therefore, the performance of MRDC-unicast degrades quickly when network load increases. MRDC-broadcast delivers the most packets to receivers in high load network. On one hand, a tree structure contains fewer routers than mesh structures, and hence, creates less collision and congestion than mesh when routers operate in broadcast mode. On the other hand, MRDC is more active than ADMR in terms of tree maintenance. MRDC-adaptive yields a compromise on the packet delivery ratio in this simulation.

Mobility Pattern

In the second step, we vary the maximum speed of mobile nodes from 0 m/s (stable network) to 20 m/s and choose a 4-source-traffic scenario in which two multicast group are defined and each has 10 members. The aim of this simulation is to evaluate the performance of these multicast routing protocols due to topology changes and a certain degree of medium access contention (e.g. among control messages and multicast packets).

The results of packet delivery ratio (Figure 4.8) show that under low medium contention environment, MRDC-unicast and MRDC-adaptive have almost the same results and both of them outperform other three protocols. MRDC-broadcast has a peak at 1 m/s scenarios and then slightly decreases to 94%. ADMR provides the worst packet delivery when maximum movement speed exceeds 5m/s.

The average end-to-end delay of five protocols is illustrated in Figure 4.9. MRDC-broadcast creates the least transmission delay (smaller than 30 ms), while MRDC-unicast creates the most sometimes reach 70 ms.

Figure 4.10 compares the number of data packet transmission to successfully

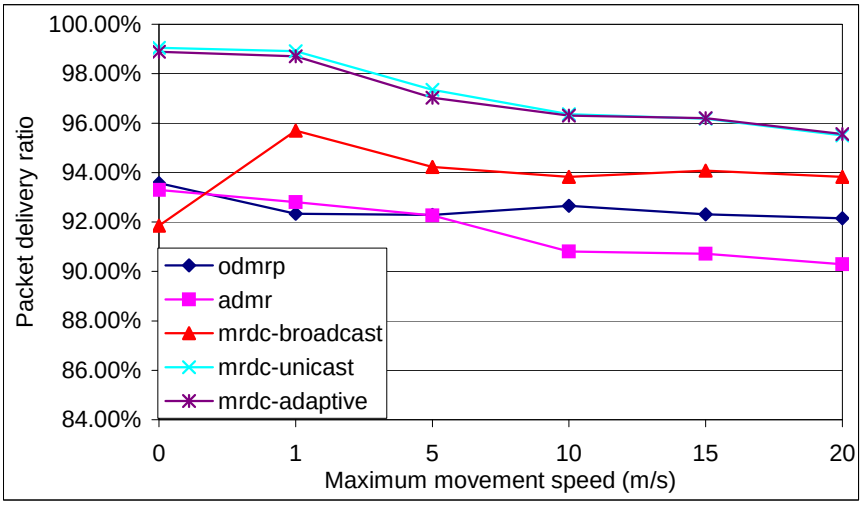


Figure 4.8: Packet delivery ratio v.s. Maximum movement speed

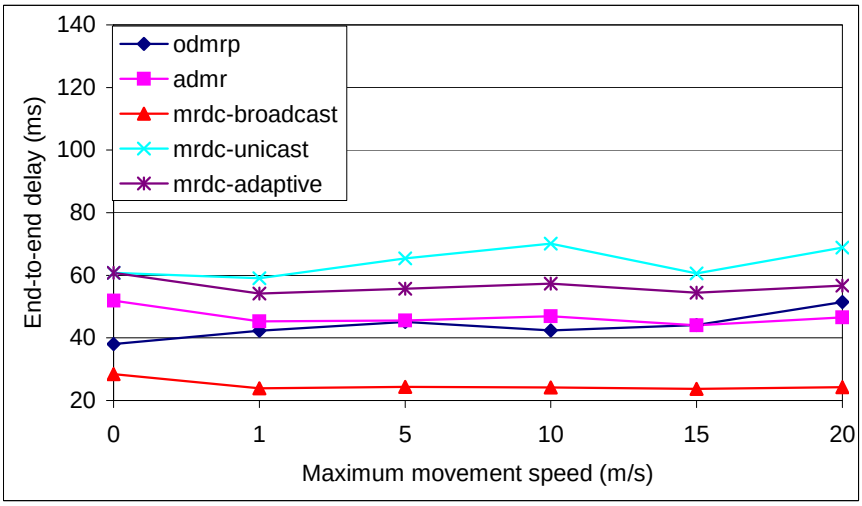


Figure 4.9: End to End delay v.s. Maximum movement speed

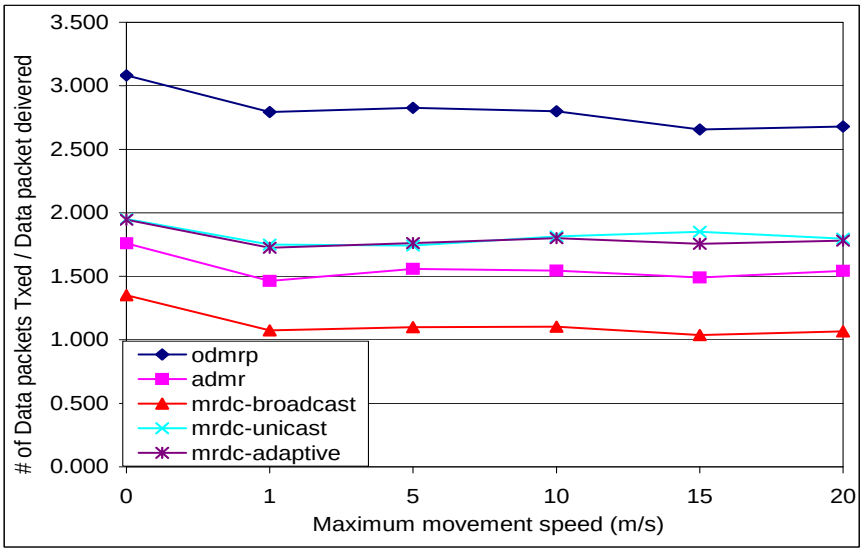


Figure 4.10: Number of data packets transmitted per data packet delivered v.s. Number of sources

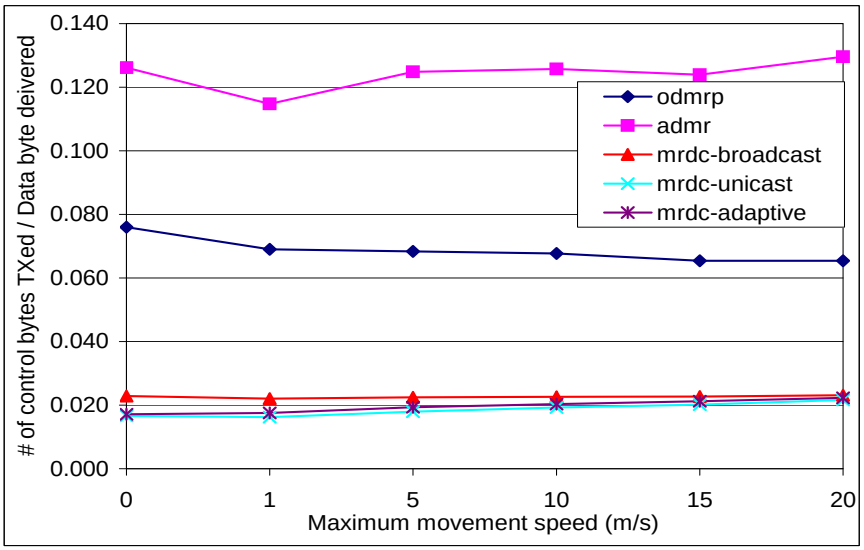


Figure 4.11: Number of control bytes transmitted per data byte delivered v.s. Number of sources

deliver a data packet to a receiver under these five protocols. MRDC-broadcast requires the minimum number of packet transmission while ODMRP needs 3 times more.

The number of control bytes transmitted per data byte delivered as a function of maximum movement speed of five protocols is demonstrated in Figure 4.11. ADMR produces the most uniform control bytes. It is MRDC-unicast that makes the least. The uniform control bytes under MRDC-broadcast remains stable but under MRDC-unicast that increases with the movement speed. In 20 m/s these two protocols have almost the same results.

To understand the above behavior, we should analyze the impact of nodes' mobility on the performance of these protocols.

ODMRP is source-based in the control part but group shared in the forwarding part. When a source reconfigures routes to receivers these routes are also available for delivery packet from other sources. More sources in a group implicitly increase the frequency of mesh reconfiguration, that makes mesh structure more robust against topology changes. This protocol depends on the periodic Join-Query and does not do local structure repair. Recall that the Join-Query flood interval used in the simulations is 3 seconds. Here two sources per group means a part of the mesh is updated every 1.5 seconds. As a result, the node's movement has nearly no effect on its performance. However, the transmission delay increases because it buffers multicast packets before completing mesh reconfiguration.

ADMR employs mainly local recovery against topology change since it has fewer redundant forwarding nodes. Figure 4.11 shows that ADMR gives the biggest the control bytes over data byte but this ratio keeps stable. However, we observed an increase of control messages as node's mobility increases in relation to tree maintenance. This increase is flushed by packet headers. In ADMR, a tree member should wait for a while before it affirms link break and runs local recovery after topology changes. This mechanism degrades the packet delivery ratio as network becomes more and more dynamic. Other metrics remain stable.

MRDC offers a better packet delivery ratio than ADMR and ODMRP in dynamic networks. The multicast tree maintenance mechanism used by MRDC is similar to ADMR. However, MRDC reconfigures multicast trees more frequently than ADMR does. This provides a tree adapting better to network topology. MRDC-broadcast remains stable and offers the best performance in terms of the shortest transmission delay and the lowest forwarding overhead. Although the tree structure does not contain redundant paths, the broadcast mode exploits the broadcast capacity of the wireless interface to create redundant transmission so that it could improve the packet delivery ratio. On the contrary, unicast mode does not benefit from this advantage, which causes its packet delivery ratio to decrease more quickly than in broadcast mode. MRDC-adaptive has almost the same behavior as MRDC-unicast since the forwarding mechanism takes only traffic related metrics into account and the traffic scenario used in simulations favors unicast mode. However, some minor differences demonstrate that certain nodes operate in broadcast mode, which degrades slightly the packet delivery ratio due to unreliable trans-

mission but improves slightly the same metric in high dynamic networks thanks to redundant transmission.

In stable networks, when all protocols forward multicast datagram on broadcast (ODMRP, ADMR and MRDC-broadcast), MRDC-broadcast provides the worst packet delivery ratio. This is caused by group-shared tree structure with significantly packet collision in MAC layer. Using IEEE802.11, the longer distance a packet goes through, the greater possibility there is it to be lost on the route. The delivery structure of ODMRP and ADMR contains shortest path for each source receiver pairs. Sources can transmit multicast packets directly through these paths to destinations. The group-shared tree used by MRDC only contains the shortest path from receivers to the core (the first source in MRDC) but not to non-core source(s). Non-core sources should send their packets to core in order to reach those group receivers which are not in their sub tree. This tree structure increases the path length from non-core source to receivers. For example, in figure 4.12, all receivers are situated no more than two hops far away from S1 in the tree. However, packets from S2 should travel five hops to reach R2 and four hops to R1 after getting through S1, nevertheless three hops are enough in ADMR. Therefore, in MRDC the traffic generated by non-core source suffers more transmission failure which degrades total packet delivery ratio. The position of nodes in a stable network aggravates this problem because the distances from group members to the core are longer and consequently the multicast trees are larger than that in dynamic networks as shown in Table 4.8. Therefore MRDC-broadcast delivers the least packets to receivers. However, since the multicast trees are correctly established in these simulations, unicast mode can improve the packet delivery ratio significantly by reducing packet collision in MAC layer.

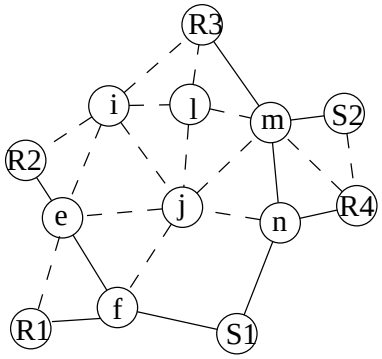


Figure 4.12: MRDC group-shared multicast tree

4.6.4 General Discussion

In this section, we compare the performance of ODMRP, ADMR and MRDC operating in different transmission modes (broadcast, unicast and adaptive). These protocols use different types of delivery structure. ODMRP constructs a group

Maximum speed (m/s)	Distance from members to core	Average interior node
0	30.8	11.3
1	23.8	9.4
5	22.8	9.4
10	22.6	9.3
15	21.6	8.9
20	21.4	9.0

Table 4.8: Distance and Multicast tree size under MRDC-broadcast as a function of Maximum speed

shared mesh. ADMR uses source-based tree. MRDC provides group-shared tree. During packet delivery, the former two create redundant forwarding state within nodes in the network against routes broken caused by topology change or control message loss. Whereas MRDC does not use this technique. The simulation results show that the greatest difficulty for these protocols to use redundant forwarding state is how to find the compromise between robustness and efficiency. Rich redundant forwarding state permits routing protocols to maintain its performance in dynamic networks without the requirement of extra routing overhead. However, the forwarding overhead as a result of redundant forwarding state degrades protocol's performance in high load cases. As for MRDC, it provides the best performance since this protocol uses the least connectivity and in most cases control messages are correctly and duly transmitted. However, the group-shared trees do not provide the same performance for non-core sources. When group members are badly distributed in the network, MRDC provides a worse packet delivery compared to other two protocols under the condition that all of them broadcast multicast packets.

Unicast mode creates more forwarding overhead and depends more on the correctness of the multicast tree than broadcast mode since multicast transmission strictly respects tree structure. Thus MRDC-unicast degrades more quickly than MRDC-broadcast as network load and mobility increase. MRDC-adaptive sometimes outperforms both MRDC-unicast and MRDC-broadcast because it takes advantage of unicast mode and broadcast mode at the same time. In fact, when MRDC broadcasts multicast packets, a tree member can receive multicast packets from neighbors which are not listed in its multicast routing entry. That provides a certain redundancy to improve packet delivery. Unicast multicast packets offers certain degree of reliability for transmission but deprive transmission redundancy. In ideal cases, MRDC-adaptive uses broadcast mode in hot spot to create transmission redundancy and avoid congestion and uses unicast mode to assure transmission in other regions so that this protocol can provide the highest packet delivery.

4.7 Conclusion

In this chapter, we studied the performance of MRDC in ns2. At first, we selected key parameters of MRDC: tree refresh period PERIOD_REF and thresholds of mode selection. A longer period generates less routing overhead for multicast tree refresh but reduces the robustness and efficiency of MRDC since the packet delivery ratio decreases as nodes' mobility changes. On the other hand, a shorter refresh interval makes MRDC robust against topology changes at the cost of high control overhead. The simulation results show a 5-second period yields a compromise between robustness and low routing overhead. As for mode selection thresholds, a small INTR and QLEN make nodes easily think that they are in a hotspot and switch to broadcast mode. This is appreciated in high load networks to avoid congestion created by unicast mode. However, it is not welcome in low load networks because broadcast mode generates significant MAC layer packet collision. ($INTR = 0.6, QLEN = 8$) pair shows a good trade-off between delivery ratio and delivery delay, and is used in performance comparison simulations.

Then, we evaluated performance of MRDC under different movement and traffic scenarios. The evaluation contains two parts: the first, the characteristics of MRDC multicast tree and the second, the performance comparison. The simulation results of MRDC multicast tree such as the average number of interior nodes and the average number of non-group-member routers allow us to estimate the routing overhead and forwarding overhead of MRDC. The results also show that MRDC multicast tree scales well in terms of tree size as the group size increases, while the number of tree repair times is proportional to link changes during a simulation. On the other hand, the correctness of MRDC greatly depends on the transmission of control message. As the network load increases, tree fragmentation becomes more frequent owing to the loss of control messages. In the performance comparison, MRDC operating in different transmission modes (broadcast, unicast and adaptive) are compared with other two multicast routing protocols, ODMRP and ADMR using four metrics: packet delivery ratio, transmission delay and routing and forwarding overhead. Thanks to the tree structure, MRDC-broadcast generates the least forwarding overhead. MRDC-adaptive provides optimal results and sometimes the best multicast packet delivery since it yields redundent transmission in high load network and offers reliable delivery in low load networks.

From the simulations we find that the packet loss is due to different effects. The first is physical condition, for example network partition makes some group members unreachable by other members. Routing protocols can do nothing to solve this problem. The second is low layer transmission failures such as packet collision occurring in the MAC layer. Routing protocols can alleviate this problem through redundant transmission. However too much redundancy can aggravate packet collision and even create congestion. The third is due to routing protocol problems for example if delivery structures are not constructed in time or the routing protocol does not repair the fragmentation in time, multicast packets cannot be delivered to the involved receivers during that period. MRDC does suffer from this kind of

problem because it does not preserve the forwarding state during tree refresh. One solution is to introduce a *forwarding* state into the states of multicast entry. Multicast routers firstly degrade from tree member to forwarding group member when receiving a CA message. Forwarding group members continue forwarding multicast packets during tree refresh. The forwarding group membership expires after a short while and the node either becomes multicast tree member or leaves multicast tree by setting entry state to *non-forwarder* after tree refresh. In this way, multicast delivery continues even during multicast tree reconfiguration.

Chapter 5

RELIABLE MULTICASTING FOR AD HOC NETWORKS

While MRDC provides best-effort non-guaranteed multicast delivery, some applications of MANETs have requirement beyond this. For example file distribution, whiteboard and Internet games are sensitive to packet loss and require reliable data transfer to group receivers. To achieve reliability some error recovery mechanism for lost packets has to be implemented. Automatic repeat request (ARQ) is one widely used mechanism. This mechanism makes the sender or some other nodes to retransmit lost packet. In protocols designed for small (local area) multicast groups, the ARQ mechanism is usually realized at the sender, which is responsible for processing positive or negative acknowledgements (ACKs/NAKs) and for retransmitting packets. Examples of such sender-originated reliable multicast protocols are MTP [66] or AMTP [67]. However, with increasing size or geographic spread of the multicast group the performance of these protocols gets worse, and more scalable protocols are required. Some reliable multicast protocols have been developed for that purpose. Besides the sender these protocols allow either dedicated receivers (e.g. RMTP [68], SRM [69], TMTP [70]) or routers (e.g. AER [71], ARM [72], PGM [73]) to handle ACKs/ NAKs and to retransmit packets for members in their local environment (we will call these protocols receiver-assisted and router-assisted, respectively). Thus the cost for retransmissions can be decreased and the ACK processing load on the sender is relieved [74].

The properties of MANETs listed in Chapter 2 make the design of a reliable multicast protocol for MANET a challenging task. Such a protocol should consume little bandwidth and node resources (e.g. processing, energy, memory space) but guarantee reliable delivery in a high packet loss rate environment. Some reliable multicasting protocols are studied: Reliable Broadcast (RB) [75], APRM [76], Family ACK Tree (FAT) [77], [78], Reliable Adaptive Lightweight Multicast Protocol (RALM) [79], Reliable Multicast Alogrithm (RMA) [80], ReMHoc [81], Anonymous Gossip [82] and Route Driven Gossip (RDG) [83]. The six former protocols are called deterministic protocols because they provide “all-or-

nothing" delivery guarantees for the delivery of packets to a group of nodes. The two latter protocols are probabilistic protocols since they guarantee delivery with a certain probability. Deterministic protocols have bad tradeoffs between reliability and scalability/mobility, while probabilistic protocols do not provide deterministic delivery guarantees [84]. Seeing the drawback of deterministic protocols, we focus our research interesting on retransmission method and design a reliable multicasting protocol, called Active Reliable Multicast Protocol with Intermediate Node Support (ARMPIS). This protocol extends both receiver-assisted and router-assisted scheme to MANET in order to improve the scalability of deterministic protocols in mobile environments.

Our main contribution is that ARMPIS distributes packet storage and retransmission responsibility to all nodes which overhear multicast packets. These nodes are called intermediate nodes. According to this definition intermediate nodes include not only group members and nodes which forward multicast packets but also the neighbors of multicast packet forwarders. In reliable multicast, retransmission load of sender is a function of link loss rate, size of network and group. In MANET, link loss rate is relatively high due to wireless interface and node mobility. Thus, we think it is necessary to make intermediate nodes share retransmission tasks. Retransmission made by intermediate nodes allows recovery packets to travel a shorter route than the originals and consequently achieves a higher recovery success and lower bandwidth consumption. Intermediate nodes need to store multicast packets for retransmission while limited memory prevents them from storing all packets. Our strategy is that in ARMPIS, intermediate nodes randomly store overheard multicast packets to reduce duplicated cache among neighbors and a node queries its neighbors about the request packets before forwarding a retransmission request. Furthermore, this protocol needs no other control packet than negative acknowledge message (NACK) and is independent of unicast routing protocols. The route to the sender is established by on-going traffic, and retransmission paths are established during NACK forwarding. ARMPIS is initially designed for MRDC, which is a tree-based multicasting protocol, but it can also cooperate with mesh-based multicast routing protocols, such as ODMRP, CAMP and NSMP. In the rest of this chapter, we do not specify which kind of underlying multicast routing is used and employ multicast delivery structure to indicate the multicast tree and mesh in the rest paper.

The rest of this chapter is organized as follows. In Section 5.1, we first present in detail the mechanism of ARQ and analyze the source retransmission load to demonstrate the necessity of making nodes other than sender do retransmission in MANETs, which are generally not large. Section 5.2 briefly introduces some current reliable protocols for MANET and their shortcomings. Then Section 5.3 discusses ARMPIS in detail including system model, design principle and ARMPIS' procedures. Section 5.4 simulates the performance of ARMPIS on top of MRDC. Final Section 5.5 concludes this chapter by summerizing remarks and pointing out future work.

5.1 ARQ Mechanism and Retransmission Load Analysis

ARQ is a mechanism used in reliable multicast protocols to provide error control. In this section, we briefly review this mechanism and then analyze the source retransmission load of different retransmission schemes on binary trees.

5.1.1 An Overview of ARQ mechanism

ARQ is a retransmission on demand mechanism, where the sender is alerted to packet losses through feedback from receivers and lost packets are retransmitted by either the sender or other nodes [85]. An ARQ scheme can either be sender- or receiver-initiated. In a sender-initiated scheme, the sender maintains state information of receivers and detects packet losses. Receivers need to acknowledge every received packet by ACK to the sender. If the sender does not receive the ACK for a packet after time out, it will assume that the packet is lost and a retransmission or a congestion avoidance mechanism will be triggered. In a receiver-initiated scheme, receivers have the responsibility of detecting losses, e.g., by observing gaps in received packets. After a loss is detected, a NACK will be issued to report the loss and request retransmission. Usually, in multicast transmission, receiver-initiated schemes are more scalable than sender-initiated schemes [74], since the burden of maintaining reliability is distributed among receivers and NACKs are only issued when packet losses occur.

Since ARQ consists of feedback and retransmission, researchers discuss the efficiency of an ARQ reliable multicasting protocol from these two dimensions: the scalability of ACK or NACK message in both network view and individual router view and the retransmission load. Feedback is generated by receivers and sent to the sender. It increases with the number of receivers and multicast packets. To reduce bandwidth consumption, feedback is aggregated to present the reception of a set of packets at receivers and downstream nodes at the router. As for the retransmission load which is a function of the size and geographic spread of the multicast group, the receiver-assisted and router-assisted retransmission schemes are proposed. In a receiver-assisted retransmission scheme, when a router receives a feedback it forwards this feedback to some other receiver in its sub-network instead of forwarding the feedback directly to the sender. If the receiver has the required packets, it retransmits the packets. In order to reduce further bandwidth consumption and reduce recovery latency,a router-assisted retransmission scheme requires the routers store the multicast packets they forward for future retransmission.

5.1.2 Mathematic analysis of ARQ's Retransmission Load

Now we examine the source retransmission load of three retransmission schemes: sender-originated, receiver-assisted and router-assist protocols. We suppose that a source-based multicast tree delivers packets to destinations, which are all located

at leaves. After one (re)transmission, a sub tree is constructed based on the orignal tree for the next retransmission. This sub multicast tree contains only those destinations which have not received the packet. All links have the same packet loss probability which is denoted as γ to facilitate the analysis.

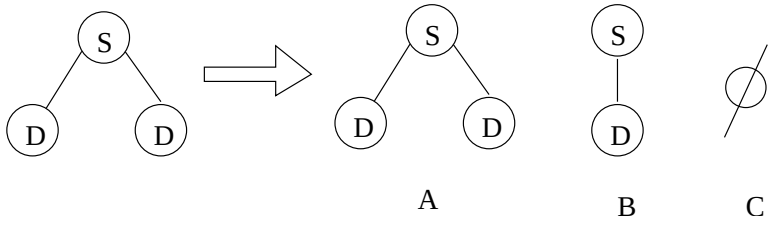


Figure 5.1: One-level binary tree and its sub trees

First, we consider a one-level binary tree as shown in Figure 5.1 where one source delivers multicast packets to two destinations which are located in the source's coverage range. The right part of this figure lists all possible retranmission trees. Tree A is the result of transmission failure to both destinations, while tree B is the case where one destination does not receive the packet and tree C is empty which represnts the successful delivery to two destinations. The probability distribution function of retransmission tree (number of retransmissions) can be determined based on a homogeneous discrete-time Markov chain (DTMC) as shown in Figure 5.2.

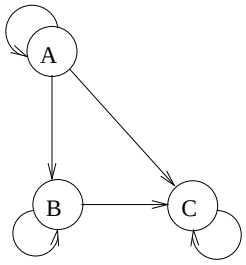


Figure 5.2: DTMC for one level binary trees

The corresponding transition probabilities can be written as the matrix

$$\mathbf{P} = \begin{pmatrix} \gamma^2 & 2(1-\gamma)\gamma & (1-\gamma)^2 \\ 0 & \gamma & (1-\gamma) \\ 0 & 0 & 1 \end{pmatrix}$$

The initial state distribution of the DTMC is given by $\pi(0)$=($\pi_A(0), \pi_B(0), \pi_C(0)$)= $(\gamma^2, 2(1-\gamma)\gamma, (1-\gamma)^2)$, since the original transmission is already finished before retransmissions begin, i.e. we just have to consider the transition probabilities originating from state A. The probability of being in a certain state of the DTMC after

n steps $\pi(n)$ can be determined simply by calculation of the *n*th power of P. We are only interested in being in state C, the case of empty tree, which indicates that all receivers have obtained the packet. Hence, we get

$$\pi_C(n) = \sum_{i=A,B} p_{i,C}^{(n)} \pi_C(0) = (1-\gamma^{n+1})^2$$

The probability f_n that exactly n retransmissions are required is given by $f_n = \pi_C(n) - \pi_C(n-1)$ for $n > 0$ and $f_n = \pi_C(0)$ for n = 0. The expected source retransmission load can be written as

$$E[R](L=1) = \sum_{n=1}^{\infty} n f_n \tag{5.1}$$

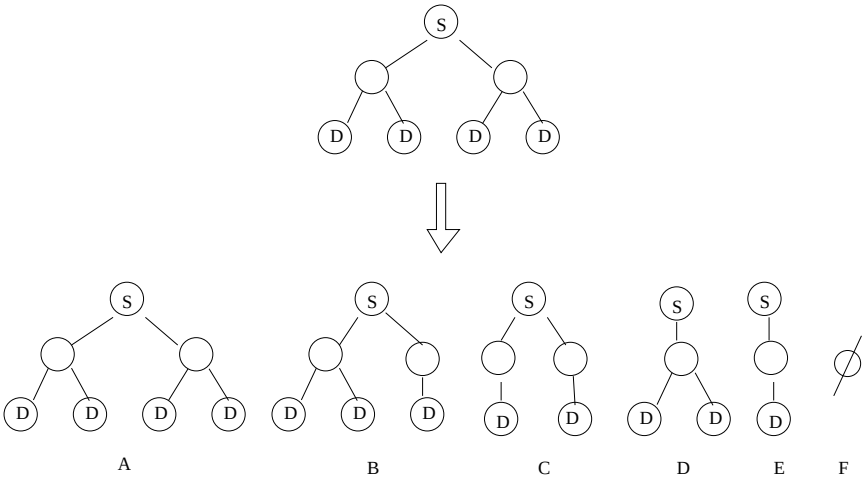


Figure 5.3: Two-level binary tree and its sub trees

Now, we analyze the source retransmission load in a two-level binary tree as shown in Figure 5.3 under a sender-originated scheme. This multicast tree has six possible retransmission sub trees. The corresponding transition probabilities can be written as the matrix

$$\mathbf{P} = \begin{pmatrix} P_{AA} & P_{AB} & P_{AC} & P_{AD} & P_{AE} & (1-\gamma)^6 \\ 0 & P_{BB} & P_{BC} & P_{BD} & P_{BE} & (1-\gamma)^5 \\ 0 & 0 & P_{CC} & 0 & P_{CE} & (1-\gamma)^4 \\ 0 & 0 & 0 & P_{DD} & P_{DE} & (1-\gamma)^3 \\ 0 & 0 & 0 & 0 & P_{AB} & (1-\gamma)^2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The transition probabilities are given by
$P_{AA} = \gamma^2 + 2(1-\gamma)\gamma^3 + (1-\gamma)^2\gamma^4$

$P_{AB} = 4((1-\gamma)^2\gamma^2 + (1-\gamma)^3\gamma^3)$
$P_{AC} = 4(1-\gamma)^4\gamma^2$
$P_{AD} = 2((1-\gamma)^3\gamma + (1-\gamma)^4\gamma^2)$
$P_{AE} = 4(1-\gamma)^5\gamma$
$P_{BB} = \gamma^2 + (1-\gamma^2)\gamma^2 + (1-\gamma)^2\gamma^3$
$P_{BC} = 2(2-\gamma)(1-\gamma)^2\gamma^2$
$P_{BD} = (1-\gamma)^2(\gamma + (-1\gamma)\gamma^2$
$P_{BE} = (1-\gamma)^3\gamma + 3(1-\gamma)^4\gamma$
$P_{CC} = \gamma^2 + 2(1-\gamma)\gamma^2 + (1-\gamma)^2\gamma^2$
$P_{CE} = 2(1-\gamma)^2(2-\gamma)\gamma$
$P_{DD} = \gamma + (1-\gamma)\gamma^2$
$P_{DE} = 2(1-\gamma)^2\gamma$
$P_{EE} = 1-(1-\gamma)^2$
The expected source retransmission load can be derived from Formula 5.1.

For the receiver-assisted scheme we first have to divide the multicast group into subgroups. Let us assume that a receiver-assisted scheme would use three subgroups in our example (see Figure 5.4): $Group_1$ consisting of S, D_2 and D_3 (the source being responsible for retransmissions), $Group_2$ consisting of D_1 and D_2 (D_2 being the dedicated receiver) and $Group_3$ consisting of D_3 and D_4 (D_3 being the dedicated receiver). As a result, from the point of view of the source, there are only three forms of sub multicast tree which corresponding to sub-tree C, E and F in the sender-originated case. The transition probabilities matrix becomes

$$\mathbf{P} = \begin{pmatrix} P_{CC} & P_{CE} & (1-\gamma)^4 \\ 0 & P_{EE} & (1-\gamma)^2 \\ 0 & 0 & 1 \end{pmatrix}$$

The router-assisted scheme can be seen as the source reliably transmits packets to its downstream nodes and then these downstream nodes guarantee packet delivery to their downstream nodes. Therefore, the source retransmission load is the same as the case of one-level binary tree.

Figure 5.5 illustrates how the expected retransmission load for sender-originated, receiver-assisted and router-assisted schemes varies with the loss probabilities γ. The diagram on the right is just an enlargement of the bottom left corner of the left one. Obviously the load of sender-originated scheme soon becomes unacceptable in two-level binary tree. Even for small loss probabilities the retransmission load is rather high. The improvement achieved by the receiver-assisted scheme is only marginal. In comparison to this, router-assisted scheme performs the best. This scheme results in acceptable load even for loss probabilities higher than 50%.

5.1.3 Discussion

When packet loss rate of wireless links is small, a two-level binary tree can be seen as a one level tree in which two leaves are a sub one-level tree. Thus, we can use the following formula to approximately calculate the retransmission load

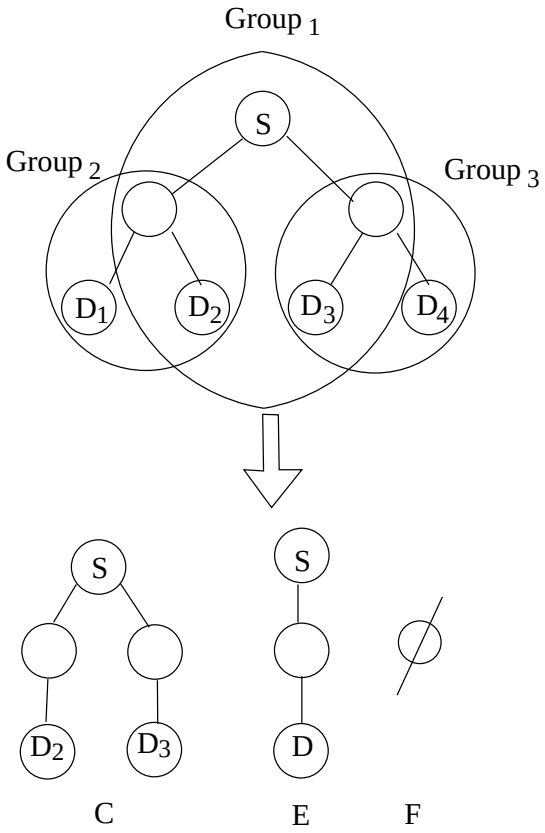


Figure 5.4: Subgroups for receiver-assisted retransmission

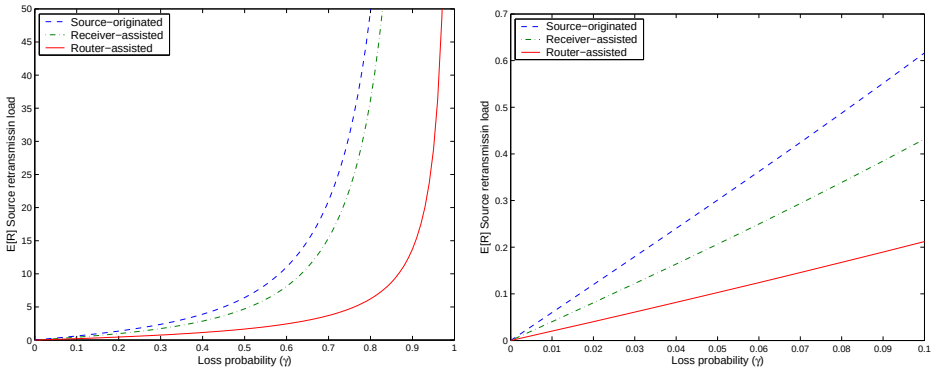


Figure 5.5: Retransmission load when varying γ

$E[R](L = 2) = 3\frac{2\gamma+\gamma^2}{1-\gamma^2}$. Consequently, as network and group size increase, the retransmission load of source in a L level tree is $E[R](L) = (2^L - 1)\frac{2\gamma+\gamma^2}{1-\gamma^2}$. This formula demonstrates that the source retransmission load of ARQ is a function of the multicast tree size and the packet loss rate on links. In a wired network, packet loss rate is relatively small and it is network size that plays a key role in retransmission overhead. That is why improvements address the issue of maintaining the scalability of reliable multicasting to reduce retransmission in large scale networks [72]. Their common idea is to distribute retransmission task to nodes other than the sender through local recovery. According to the type of nodes which send recovery packets, the reliable protocols using local recovery techniques can be further classified into receiver-assistance retransmission schemes and router-assistance retransmission schemes. However, the packet loss rate in MANET is relatively high due to wireless interface and unpredictable topology changes. Consequently, even in a small network, the retransmission load might be important. As show in Figure 5.5 if the packet loss rate on each link is 0.08, the retransmission load of source (tree root) exceeds 0.5 in a two-level binary tree under sender-originated scheme. It is necessary to employ receiver-assistance retransmission scheme and/or router-assistance retransmission scheme in reliable multicasting protocol for MANETs.

5.2 Current Reliable Multicasting protocol for MANET

The first reliable multicast protocols for MANET ([76]) tries to make the ARQ mechanism adapt to a mobile multihop environment. They just use the sender to retransmit lost packets. However, the high retransmission load and bandwidth consumption make researchers attempt to extend receiver-assisted scheme ([82]) or router-assisted scheme ([77],[78]) to reliable multicasting for MANET. The rest of this section gives a brief introduction some of these works.

5.2.1 Adaptive Protocol for Reliable Multicast in MANET (APRM)

In [76], the authors proposed an adaptive protocol for reliable multicast to a set of predefined group members against topology change, which we call APRM. A core-based multicast tree is constructed to delivery messages reliably. In the case of fragmentation due to node movement, a ”forward region” is introduced to glue together the fragmented tree and messages are flooded in this region. However, this protocol requires that each recipient sends feedback directly to the sender and gets recovery messages from the sender. Neither the receiver nor the router assists in retransmission. Thus, this protocol uses a sender-originated retransmission scheme and is not efficient as the size of the multicast group or transmission failure increases.

5.2.2 Anonymous Gossip

Receiver-assistance retransmission schemes for the Internet require that router know the group receivers in its sub-network. As group membership or topology change, control messages should be sent to update the information in routers. Anonymous Gossip [82] employs the receiver-assistance retransmission scheme for MANETs with efforts to reduce this kind of control overhead.

Each multicast receiver periodically generates a retransmission request, called a gossip message, which lists lost packets. Then the node forwards this message to a node randomly chosen from its delivery structure neighbors. Upon receiving a gossip message, the router randomly select another delivery structure neighbor as next hop and forwards the message. This procedure is repeated until the gossip message arrives at a group member. In this way, a receiver establishes connection with a randomly selected group member and tells this member about packets it has not received. The group member checks to see if it has the required packets and retransmits those that it finds.

In their paper, the authors announce that anonymous gossip favors tree structure to prevent gossip messages from reaching the same nodes twice. Consequently, the sub tree of a router can be seen as the sub-network in a receiver-assistance retransmission scheme. If the selected next hop is a downstream node, it means that router executes local recovery. But the difference is that in case of local recovery failure, the router will not do another attempt to address other receivers in its sub tree or send the feedback to the sender. If the selected group member has no packets or less than all of the required packets, the initiator of the gossip request will make another attempt with another group member. Therefore, the performance greatly depends on the selected group member, which may generate significant overhead and recovery latency.

5.2.3 Family ACK Tree (FAT)

Router-assistance retransmission schemes are developed in stable networks. If a router is no longer the multicast relayor, the stored packets become of no use. In MANETs, multicast delivery structure may change frequently due to the node mobility, which results in worse utilization of router's buffer and the degradation of the retransmission scheme to sender-originated scheme.

To overcome this shortcoming and also to solve the scalability problem of source-based retransmission, Family ACK Tree (FAT) [77],[78] extends the router-assistance retransmission scheme to adapt MANET environment. In FAT, a tree, called a family ACK tree, is constructed to assume the reliability multicasting. Each node on the tree knows its parent, grand parent and children. Children confirm the reception of multicast packets to their parents. This protocol requires that nodes temporarily store packets. In case of transmission failure, the parent looks for the packet in its cache and retransmits to the corresponding child. If a node decides to leave family ACK tree, it transfers its children to its parent and also

any packets waiting to be acknowledged. On the other side, those children contact their grandparent for recovery packets when they discover the disappearance of their parents. However this protocol becomes inefficient in high mobility networks due to the difficulty of ACK tree maintenance.

5.3 Active Reliable Multicast Protocol with Intermediate node Support

5.3.1 System model

Other than the assumption mentioned in 3.1, we make the following further supposition for reliable multicast protocol designing. The reference assigned by the source to multicast packets is consecutive so that receivers can detect losses primarily by reference gap in the data packets. During a multicast session, sources have all packets that they have sent and receivers have all the packets they have received. We consider a scenario where there are n sources and m receivers in the multicast group sharing the same multicast delivery structure.

5.3.2 ARMPIS Protocol Design Principle

The active reliable multicast protocol with intermediate node support (or ARMPIS in abbreviation) involves both receiver and router in retransmission. Nodes cach multicast packets for retransmission. In the case that the required packets are not found in cache, routers look for them in their "sub-network", which is their neighborhood. In ARMPIS, intermediate nodes are group members as well as nodes which convey multicast traffic and the neighbors of these conveyors, in brief, all nodes that overhear multicast traffic. These nodes are active in the sense that they cache multicast packets and perform retransmission. When a multicast traffic conveyor forwards packets, the broadcast nature of the air interface permits its neighbors to overhear the packets. Thus, these neighbor nodes can help to cache data packets for future retransmission. For example, Figure 5.6 illustrates a simple MANET where source S sends packets to three receivers R_1, R_2, R_3. When $nodeA$ forwards multicast packets, its neighbor $nodeY$ can receive those packets at same time. Then $nodeY$ can store and participate in retransmission if there is delivery failure to R_2 and R_3.

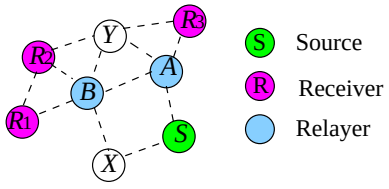


Figure 5.6: Multicast packet delivery

Intermediate nodes store packets with a certain probability (denoted by p) to realize distributed multicast data cache. There are some further reasons why we use such a probability.

- The memory capacity of mobile nodes is limited. If nodes store every data packet they receive, they can only keep the newest packets.
- It is unnecessary to store all packets. Simulation results ([23], [24] and 4.6) show that multicast routing protocol can deliver safely most of the traffic. Storing successfully delivered packet wastes memory capacity.
- Nodes mobility causes frequent changes in their roles. A node may be a multicast traffic conveyor at one moment and then become a neighbor or may be far away from the structure at the next moment.

5.3.3 ARMPIS Protocol Description

ARMPIS is a receiver-initiated, NACK-based scheme in which receivers are responsible for detecting multicast packet losses and initiating retransmission request. This protocol contains two phases: data delivery phase and data repair phase. In the data delivery phase, when MRDC deliver data packets, intermediate nodes randomly cache these packets and fill in the duplication table of MRDC to avoid process duplications. In the data repair phase, nodes aggregate NACKs and try to get the requested packets locally. In case of local repair failure, nodes delete the information of request packets from MRDC's duplication table so that the node is ready for retransmission. Then NACKs is forwarded to the next hop along the reverse path to the source. At last multicast routing protocol delivers the recovered packet. Each node in ARMPIS reserves a memory space as a multicast packet caching buffer, which behaves in a FIFO fashion. ARMPIS defines two kinds of NACKs: local broadcast NACKs which are sent to neighbors for local inquiry, and unicast NACKs which are addressed to the request packet's source. A NACK message contains group identification, source identification and a reference list, each reference corresponds to a retransmission request. A NACK message can contain at most x requests for the same group and source pair. During data forwarding, a header is added into traffic packets which contains a field to indicate whether the packet is original or retransmitted. For each multicast flow, identified by the <@group, @source> pair, nodes aggregate NACKs using three following sequence number arrays:

1. local repair array, which contains the sequence numbers that will be sent to neighbors;
2. request array, which contains the sequence numbers that have been sent to neighbors and will be sent to the next hop towards the source;
3. sent array, which contains the sequence numbers that have already been sent to the next hop towards the source.

These three arrays do not contain duplicate sequence numbers. Nodes delete the sequence number from these arrays when they receive the corresponding packet.

In the data delivery phase, the source assigns consecutive sequence numbers into data packets before sending them and MRDC use broadcast mode to deliver these packet to group receivers. Thus, no delivery guarantee is provided during transmission. When intermediate nodes receive a non-duplicate multicast packet, the MRDC forward plan fills in the duplication table of MRDC to avoid processing the same packet the second time. Then, the MRDC forward plan passes the packet to ARMPIS. If the packet is original, the node generates a reverse path to the source by recording the node from which the packet comes. The path is stored in the URTable of MRDC. Group receivers cache all multicast packets during the session. While, non-receiver nodes use the following method to realize cache overheard packets with probability p. Upon receiving a multicast packet, the node asks a uniform distribution random value generator to generate a random number between 0 and 1. If the random number is smaller than p, the node stores this packet.

In the data repair phase, receivers detect losses primarily by sequence gap in multicast packets. If such a gap is found, the receiver waits for a short moment to make sure that the gap is not produced by disorderly delivery. If there are still some packets which are not received after the time out, the receiver considers these packets to be lost and inserts their sequence numbers into the local repair array. Each node periodically checks its local repair array. If the array is not empty, it initiates a local negative acknowledgment message (local broadcast NACK) for the first L sequence numbers and puts these L sequence numbers into the request array. The local broadcast NACK message is sent one hop away to see whether some neighbor has the lost packets. After waiting for a while, if a node still has some sequence numbers in its request array, it generates a unicast NACK message containing these sequence numbers and appends these sequence numbers to the sent array. The unicast NACK message is sent to the next hop on the reverse path to the source. When receiving such a NACK message, the node checks whether it has some of requested packets in its buffer because it is possible that this node did not receive the broadcast NACK message. Then, it deletes the sequence numbers which also appears in its ree sequence number arrays and puts the rest of the sequence numbers into the local repair array and erases the corresponding packet information from the MRDC's duplication table. In this way, the node is ready to forward the recovered packets. These steps are repeated until all requested packets are found or unicast NACK message reaches the source. Before transmission, the node marks in the packet header that this packet is a retransmitted packet. Then, these requested packets are delivered by multicast routing protocol as a normal multicast packet and are forwarded only by the multicast routers which do not have the relevant packet information in their duplication table. In this way, retransmitted packets flow on the sub-tree where transmission failure occurred and will not continue to the other part of the multicast tree. Intermediate nodes periodically delete the eldest L sequence numbers from their sent array. While receivers move these sequence numbers from their sent array to their local repair array. Thus the

data repair phase continues in case of retransmission failure.

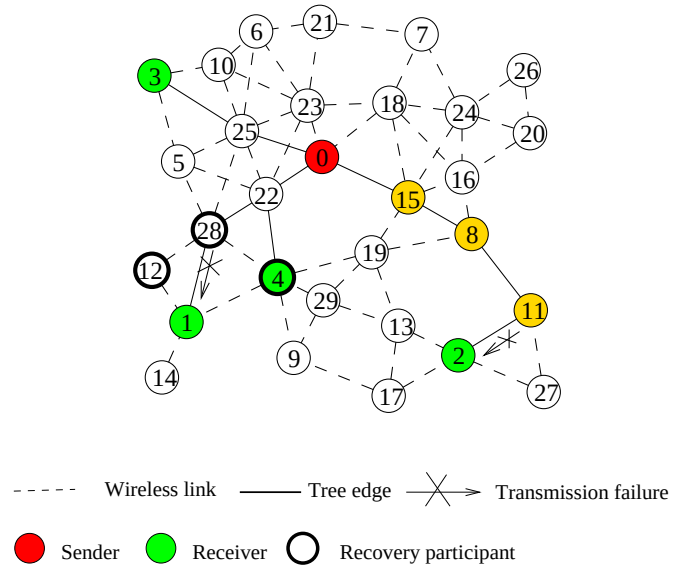


Figure 5.7: Multicast packet (re)transmission in a MANET

Let's consider an example that illustrates how ARMPIS works. Figure 5.7 shows a 30-node mobile ad hoc network. Nodes are differenced by their identification. If two nodes are in each other's coverage range, a dotted line connects them. In this network, there is a multicast group contains one source, node 0, and four receivers, nodes 1, 2, 3, and 4. MRDC constructs a multicast tree to connect these group members. The tree is rooted at node 0 and contains 6 routers: nodes 8, 11, 15, 22, 25 and 28. Eight nodes are tree neighbors, they are nodes 5, 10, 12, 16, 18, 19, 23 and 27. If node 28 fails to transmit a multicast packet p_i to node 1, node 1 sends a broadcast NACK to its neighbors without knowing there is another group receiver node 4 in its neighborhood. Even if node 4 does not have p_i, node 1 can still expect to get the packet from node 28 or even node 12 if they chose to cache p_i. If none of these three nodes has packet p_i in their cache, node 1 then sends a unicast NACK to node 28. Node 28 in its turn inquires its neighbors for p_i. If none of its neighbors has p_i, node 28 removes packet information from duplication table and sends NACK to node 22. In this way, NACK is sent to source, node 0 if none intermediate nodes has p_i. Node 0 retransmits packet p_i. This packet flows on the branch of node 22 to node 1 and will not go to branches of node 15 and 25 since they consider it as duplication.

After several seconds, the network topology has been changed as a result of node's mobility as shown in Figure 5.8. The node's movement results in the path from node 0 to node 3 changing because the link between node 3 and 25 is broken. On the other side movement creates a shorter path from node 0 to node 2 owing

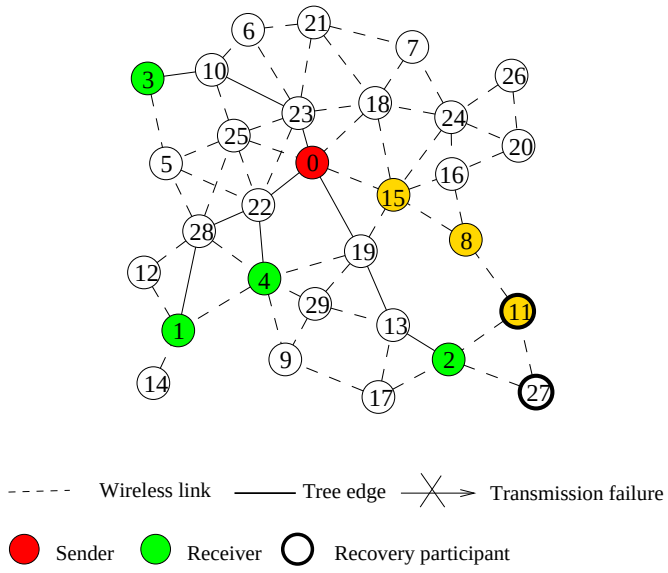


Figure 5.8: Multicast packet (re)transmission after topology change

to a new link established between node 0 and 19. MRDC reacts to these topology changes by reconfiguring the multicast tree which leads to a change of node roles. Old neighbors, nodes 19, 10 and 23, become routers and old routers, node 15 and 25, become tree neighbors. However the multicast packets cached by these nodes are still available for local recovery.

5.4 Performance analysis

We evaluate the performance of our reliable multicasting protocol in terms of delivery guarantee and bandwidth consumption in ns2.

5.4.1 Simulation Environment and Implementation Decision

The simulation environment is identical to that of Section 4.6. We use movement scenarios which contain more network partitions. As for the traffic scenarios, they differ slightly from those used in previous simulations. Instead of sending packets until the end of simulation, in these reliabilities experiments, each source transmitted 3200 packets during a simulation. With a speed of 4 packets per second, sources finish their transmission in 800 seconds. The rest of simulation time (about 20 seconds) permits retransmissions to finish their work.

5.4.2 Protocols and parameters

The capacity of buffer is set to permit a node store 64 data packets. The time to check local repair array is every 1 second. Nodes wait for 1 second before sending a unicast NACK. And a sequence number rests in sent array for 3 seconds. So, the total time for NACK suppression is 5 seconds. The probability p of nodes storing a packet is 10%.

We studied the performance by varying three parameters: the probability p to choose a suitable cache probability, the maximum movement speed and the number of sources to test the performance. For reasons of comparison, we develop a reliable multicast protocol similar to [76] in which nodes do not cache packets and feedbacks are sent directly back to the source. This protocol is denoted as APRM in simulations.

Three metrics are used during the performance analysis:

- **Packet delivery ratio:** the percentage of data packets correctly delivered to receivers over the number of data packets that should have been received. The goal of our reliable multicast protocol is to provide 100% delivery ratio in most cases.

- **Total network load:** the total number of bytes sent during simulation, it includes both control messages and traffic packets.

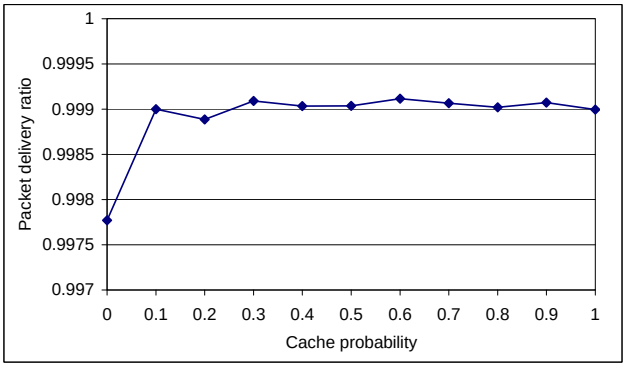
- **Source retransmission load:** the number of data packets retransmitted by sources. This metric counts the extra load of sources when providing transmission guarantee using ARQ technique.

The rest of this section presents the simulation results in detail. The simulation results with confidence can be found in Annex A.1.2

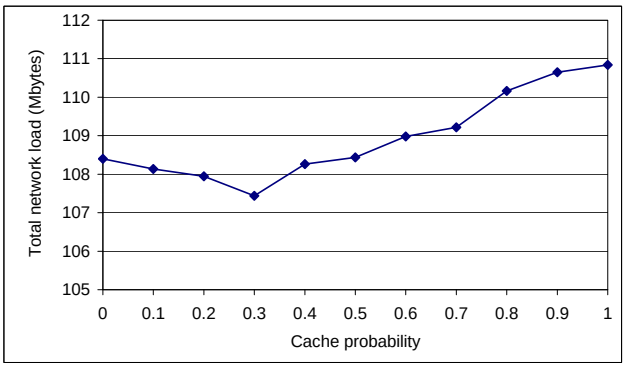
5.4.3 The choice of cache probability p

First, we set the number of sources to 6 (three groups and two sources per group) and maximum movement speed to 5 m/s while vary the cache probability from 0 to 1 to see the behaviors of ARMPIS. When p equals to 1, the nodes store all packets they overhear. This results to only the newest packets being stored in cache. On the contrary, when p is set to zero, nodes do not cache any packets.

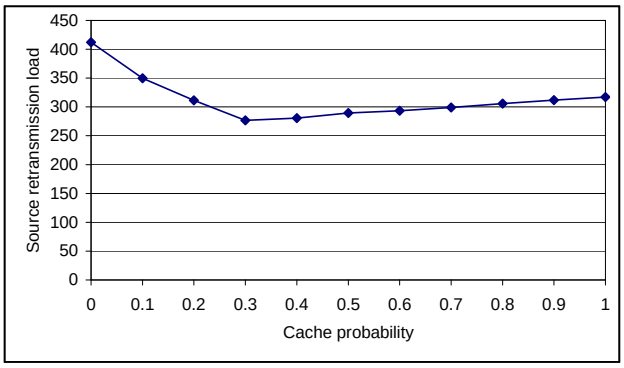
Figure 5.9(a) shows that packet delivery ratio is improved when cache probability passes from 0 to 0.1 and then remains nearly stable. Thus, increase cache probability cannot enhance the packet delivery ratio. ARMPIS cannot provide 100% delivery because there are some packets that are not retransmitted before the simulation terminates. Therefore, the packet delivery ratio reflects how many packets are waiting to be retransmitted at the end of simulations. In our simulations, the lifetime of the node's buffer, which means the eldest possible packet cached in nodes, covers the last $T = L/(p * r * s) = 10/(6 * 4 * p)$ simulation seconds. If the original packet is generated during that time, the recovery packet



(a) Packet delivery ratio v.s. cache probability



(b) Total network load v.s. cache probability



(c) Source retransmission load v.s. cache probability

Figure 5.9: The performance behavior as a function of cache probability p

might be found on the route to the source with a local recovery success probability of $p_s = 1 - (1 - p)^n$ if there are n neighbors. Otherwise, the request should be sent back to the source. A high probability leads to a high local repair success rate with the cost of shortening the lifetime of node's buffer. Consequently, a quick response may be given to the retransmission request of the latest packets, while a slow response to earliest packets. As we indicated in Chapter 4, three cases in which MRDC is prevented from delivering multicast packet to their destinations. Routing protocol failure, lower layer protocol failure and network partition. Lower layers protocol failures are usually represented by MAC layer packet collision in simulations. Routing protocol failure means that MRDC cannot react to topology changes in time or does not correctly construct multicast tree during tree refreshing due to control message loss. MAC layer packet collision and temporary tree fragmentation usually produce a short term packet loss and can be detected quickly, while network partition and tree fragmentation which persist during one or several periods may cause packet loss during a longer period. When cache probability increases, ARMPIS deals more and more efficiently with transmission failure related to packet collision or temporary tree fragmentation by quickly recovering more packet loss, but copes less and less efficiently with network partition or long duration tree fragmentation by leaving more packets recovered by sources. As a result, the number of non-repaired packets keeps constant. The case in which cache probability equals to 0 can be seen as an extreme case of high probability where lifetime of buffers is zero. Thus, there is no recovery acceleration through router assistance. It gives the biggest non-repaired packet list and the worst packet delivery ratio.

Total network load as a function cache probability is illustrated in Figure 5.9(b)). For a give movement and traffic scenario, the difference of total network load is usually the result of retransmission overhead. The results show that total network load first decreases and then rises after it reaches minimum value along with the increase of cache probability. When nodes begins to cache packets, retransmission overhead is reduced through local recovery. As cache probability increases, the distribution of multicast packets among neighbors becomes worse and the probability of neighbor nodes caching the same packets also increases. The probability of duplicate cache is $p_d = 1 - (1 - p)^n - np(1 - p)^{(n-1)}$. When a node runs local recovery for a latest packet, it might get the same recovery packet from multiple neighbors. These duplications consequently increase retransmission overhead.

The same behaviors can be observed in source retransmission load as shown in Figure 5.9(b). The retransmission load of source quickly decreases and then slightly increases after it reaches the smallest value. The local recovery assisted by routers reduces the retransmission load of the source, while the increase of cache probability makes local recovery concentrate more on the latest packets and the sources, as a result, should deal more and more with the earliest packets.

This simulation shows that ARMPIS gives the best compromise among packet delivery ratio, bandwidth consumption and source retransmission load when cache probability equals to 0.3. In the following simulations, we choose this value as cache probability.

5.4.4 The impact of node mobility

In this aspect, the maximum movement speed of nodes range in the set {0, 1, 5, 10, 15, 20} m/s. A 4-source scenario is chosen as traffic pattern which defines two groups and two sources per group. Therefore, when reliable multicast protocol deals with topology changes, it should also face slight inter group and intra group competition.

Figure5.10(a) shows the packet delivery ratio with different maximum speed of these three protocols. The results show that ARMPIS is reliable even in the case of frequent topology changes: mobility has nearly no impact on the performance of ARMPIS, which can provide almost 100% packet delivery ratio in all simulations, while the performance of underlying multicast routing protocol has significant changes. APRM gives a worse performance than ARMPIS. In APRM, only source can resend the lost packets, it is worse than the 0-probability of ARMPIS where there are still receivers assisting retransmission. Thereby, this protocol makes the recovery packets have the same loss probability as the primary ones and provides a slow recovery speed. When MRDC provides a bad packet delivery (for example in stable networks), APRM leave a big list of packets not be recovered at the end of simulations. However, the local recovery mechanism helped by routers and receivers accelerates recovery procedure risk by proposing a shorter path for retransmission. Thus the packet delivery ratio is improved.

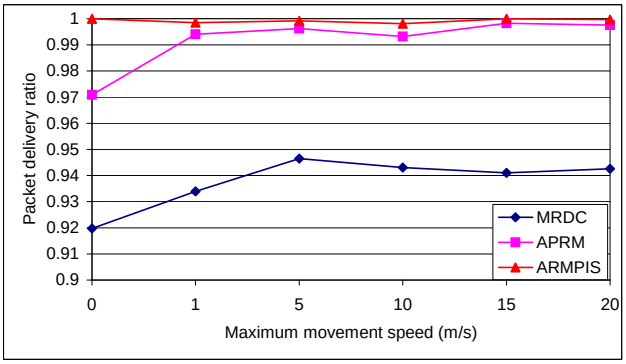
Figure5.10(b) demonstrates the total network load, which includes control messages, original packets and retransmission packets, as a function of node mobility. As topology changes become frequent, the routing overhead of MRDC increases to locally repair multicast trees. However, the high node mobility leads to a fair distribution of group members in the network. which reduces the size of MRDC multicast trees (see the analysis in Chapter 4). As a result, total network load of MRDC remains stable. ARMPIS generates less than 20% extra network load for retransmission while APRM creates more than 30%.

Figure5.10(c) illustrates the average number of packets retransmitted by sources. Sources in APRM should retransmit more packets when MRDC delivers fewer packets. ARMPIS makes sources retransmit five times fewer packets than APRM does. Compared with APRM, ARMPIS distributes retransmission responsibility and has less retransmission failures.

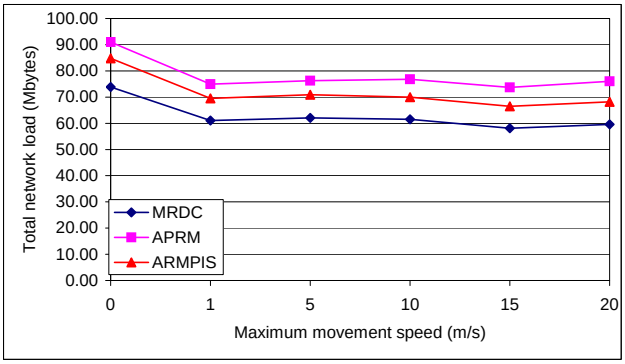
ARMPIS is reliable at facing topology changes and can deliver nearly 100% data packets in all mobility cases. This protocol is also scalable in the sense that it does not generate significant retransmission load as node's movement speed increases.

5.4.5 The impact of traffic load

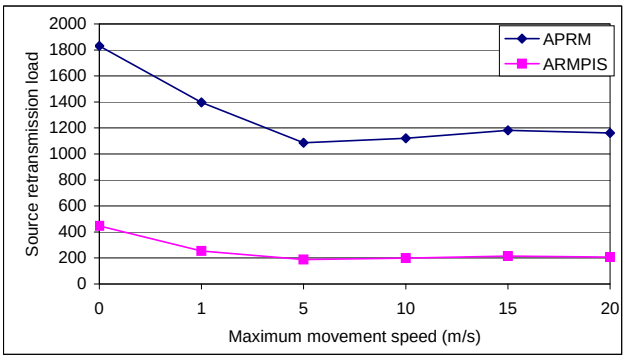
In traffic load experiments, node mobility speed is moderate with maximum speed at 5 m/s. The number of multicast sources is increased from 2 to 8. The number of groups was consequently increased from 1 to 4. The total network load metric is



(a) Packet delivery ratio v.s. Maximum speed



(b) Total network load v.s. Maximum speed



(c) Source retransmission load v.s. Maximum speed

Figure 5.10: The impact of node's mobility

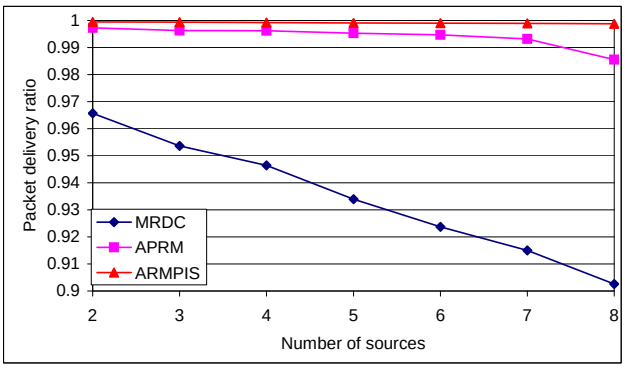
not used in this simulation because low packet delivery ratio in high load network makes comparison unfair.

The packet delivery ratio as a function of the number of sources is presented in Figure5.11(a). ARMPIS maintains nearly 100% packet delivery ratio until seven sources and then appears a little degenerative. However, it can transfer more than 99% data packets to all receivers. This shows that this protocol is reliable when traffic becomes more voluminous. The performance of APRM exponentially degrades. MRDC has a linear degradation even when there is no congestion. This phenomenon is related to the data forwarding fashion employed by MRDC, which works on top of IEEE 802.11. The latter does not offer delivery guarantee for broadcast and multicast packets. When MRDC forwards multicast packets, some packets are lost due to the hidden terminal problem. This problem becomes more and more serious when network load increases. In APRM, retransmission initiated by the original source adds considerable extra traffic to the network (see Figure5.11(b)), which raises collision risk and introduces congestion. That's why the packet delivery ratio decreases more quickly after 7 sources. On the contrary, local recovery mechanism of ARMPIS tries to find the request packet as close as possible to the corresponding receivers. As a result, the retransmission load of ARMPIS is less important than that of APRM that makes ARMPIS outperform APRM. Since there is no retransmission congestion control, when traffic becomes heavy in the network, the performance of ARMPIS degrades only slightly.

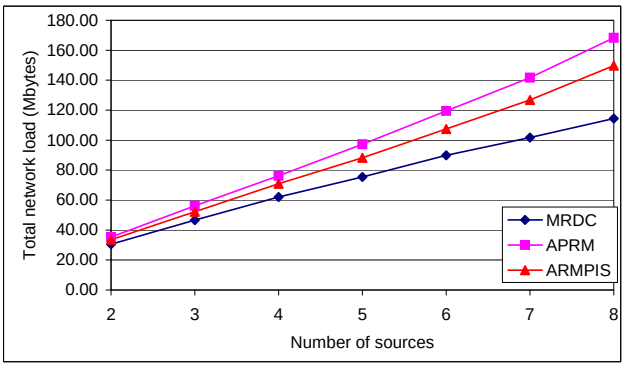
As demonstrated in Figure5.11(c), the packets resent by sources in ARMPIS are much fewer than those in APRM. In the case of 8 sources, each source of APRM retransmits nearly half of the primary packets while retransmission load of sources experience almost no change. This phenomenon can be explained by the fact that wireless channel is saturated around sources which prevent them from receiving further NACKs. These sources do not consequently generate more retransmission load. It also explains why packet delivery ratio of APRM decreases so quickly from 7 sources to 8 sources while at the same time, the degeneration of MRDC is not so significant. On the contrary, in ARMPIS many more NACKs arrive at sources in the case of 8 sources than that of 7 sources. Then, the retransmission load of source is doubled.

Another reason of performance degradation of ARMPIS is due to the lifetime of node's buffer which is inverse by proportional to the traffic load. Recall that the lifetime of node's buffer is $L/(p * r * s)$, where p is the cache probability, r is the transmission rate of source and s is the number of sources. The lifetime of buffer decreases as the number of sources increases. In 8-source scenarios, the buffer of nodes in hot spots can only cover about approximately the amount of data in a 1 second transmission (L = 10 packets, r = 4 packets/source/second and s=8 sources). With so small a lifetime (which is nearly equal to local repair array check period), router's buffer does nearly not help packet caching and retransmission and there are only receivers assist. As a result, both source retransmission load and network load increases, which consequently degrades slightly the packet delivery ratio.

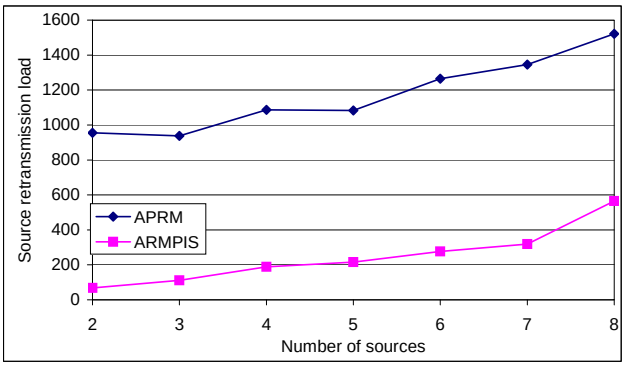
However, low recovery overhead generated by ARMPIS makes this reliable



(a) Packet delivery ratio v.s. Number of sources



(b) Total network load v.s. Number of sources



(c) Source retransmission load v.s. Number of sources

Figure 5.11: The impact of traffic load

protocol scale better than sender-originated retransmission schemes as network load increases.

5.5 Conclusions and Future Work

In this chapter, we introduced our active reliable multicast routing protocol with intermediate node support (ARMPIS) to provide reliable multicasting in mobile ad hoc network. The retransmission load of ARQ is a function of both network size and transmission failure rate on each link. The properties of MANET such as frequent topology changes and wireless interface make multicast packet transmissions fail easily. In order to reduce source's retransmission load and achieve scalability in highly lossy wireless environments, ARMPIS extends the receiver and router assistance retransmission scheme to MANET by distributing retransmission burden to intermediate nodes. When receiving a retransmission request, nodes first check their buffer and also those of their neighbors in order to do a local repair before forwarding the request to the source. A cache probability is employed to decide to store or not a message in each node to reduce the packet cache duplication and stores as many as possible multicast packets among neighbors.

A high cache probability improves local recovery success rate of the latest packets but reduces the lifetime of router's buffer and degrades multicast packet storage distribution among neighbors. The performance evaluations suggest 0.3 as cache probability to achieve an optimal compromise. The simulation results also show that ARMPIS is reliable in both stable and dynamic networks or in a relative high load situations by providing nearly 100% packet delivery to all receivers. And thanks to the local recovery scheme, ARMPIS reduces significantly both network load and source's retransmission load compared to APRM, a source-based retransmission scheme.

Up to now, ARMPIS focus on retransmission scheme by extending both receiver assistant and router assistant retransmission scheme to MANET. However the congestion control is also a key issue in providing reliable multicasting. Future work will seek to introduce flow control in reliable multicasting to avoid retransmission degradation of the network throughput.

Chapter 6

AD HOC NETWORK TESTBED

Software simulations are an excellent choice for the initial algorithm design and performance evaluation. They are cheap and offer a realistic development environment. When implementing MRDC in ns-2, we acquired some experience in designing multicast routing protocol. In particular, Section 6.3.2 involved a lot of trials to simulate the performance of MRDC in 50 node networks by using a software network simulator. Through those experiments, we chose the optimal parameters for MRDC and gain a better understanding of its behaviors in different mobility and traffic patterns. Furthermore, since software simulations can be repeated, software simulations provide a standard way that allows us to directly compare MRDC's performance with some other multicasting protocols. As a result, we can propose the suitable applications and working environment of MRDC. But software simulations have many limitations: First, they do not realistically duplicate the physical layer. Second, a software simulation cannot catch subtle bugs seen in interactions between the operating system, the system hardware, and the real-life design environment. A software simulator also ignores interlayer communication, which is integral to the effectiveness of these protocols. The work in [86] demonstrates a number of significant discrepancies between simulated and real-life results. These limitations make hardware testing essential. In this chapter, we detail our experience in implementation and validation of both unicast routing protocol and multicast routing protocol in an ad hoc testbed. Our goal is not limited to routing protocol performance evaluation. We believe that an ad hoc testbed with suitable routing protocol is very useful for other layer protocol study but also for the design of new ad hoc applications.

Linux [87] was chosen as the operating system in our testbed for its availability and familiarity. It was decided that both unicast and multicast routing protocols run in user space to facilitate cross platform implementation and installation. However a slight difference exist between the architectures used in routing protocol implementation due to the kernel level forwarding support for unicast and multicast packets. Unicast routing protocol is implemented as a routing daemon which runs in user space and maintains kernel level routing tables via system calls [88]. Sev-

eral issues make using the same architecture for multicast routing difficult: First, not all Linux kernels support multicast packet forwarding. Secondly, those kernels with multicast packet forwarding support do not allow single device forwarding. That is to say the kernel will not send a packet to the interface where it received the packet to prevent forming transmission loop. However, in MANET, multicast forwarding happens on the same wireless interface. To overcome these limitations, a procedure containing multicast packet capture, encapsulation, forwarding, decapsulation and delivery is used when we implement multicast routing protocol. This procedure allows multicast routing protocol to forward packets in user space.

We choose the Distributed Dynamic Routing (DDR) algorithm [89] as unicast routing protocol. DDR is a distributed clustering algorithm with deterministic criteria, which deals with the problem of topology management in mobile ad hoc networks. The main idea of the algorithm is to select for each node a neighbor, called *preferred neighbor*, that has a maximum degree of connectivity in the neighborhood (i.e. criteria of election algorithm). This is done using only a periodical beaconing process. It has been proved that irrespective of the network topology connecting each node to its preferred neighbor always yields a forest [89]. In this algorithm, each tree of the forest forms a zone, and each zone is maintained proactively. Zones are connected to each other via the nodes that are not in the same zone but are in the direct transmission range of each other. Therefore, the network is partitioned into a set of non-overlapping zones. As a result, the algorithm combines two notions: forest and zone. Forest reduces the broadcasting overhead by selecting a subset of the set of neighboring nodes for forwarding a packet, and zones are used to reduce the delay due to the routing process and to reach high scalability. The DDR proactive part is validated under different network topologies and movement scenarios without traffic.

On the other side, MRDC is implemented as multicast routing protocol in the testbed [90]. With additional function modules such as a simplified IGMP [6] module and a tree information collection module, we evaluated the bandwidth utilization of MRDC and tested the correctness of the implementation in both topology dynamic scenarios and membership dynamic scenarios using a popular multicast application.

6.1 Implementation Structure

Before developing a routing protocol program, we shall first decide where to locate the program and how it will cooperate with other system components. In this section, we discuss the architecture of the Ad hoc testbed and our choice of Linux as a platform.

6.1.1 Architecture of Ad hoc testbed

An ad hoc testbed consists of at least the following components as shown in Figure 6.1 according to the TCP/IP network model [91]:

- Application layer, where end-user applications reside to send and receive messages.
- Transport layer, which handles communication among programs on a network. It also provides some further functionalities such as reliable transmission, quality of service support, flow control, and so on. TCP and UDP falls within this layer.
- Network layer, which is used for basic communication, addressing and routing. It directs data packets from the sender to the receiver(s).
- Link layer, which defines the network hardware and device drivers. Usually MAC protocols are integrated in device drivers.

According to this structure, routing protocols should lie between the transport layer and the link layer to route packets among different devices.

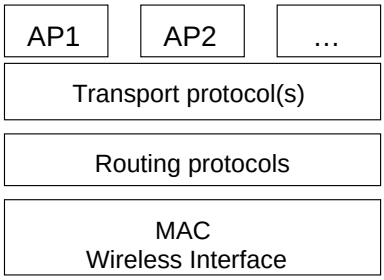


Figure 6.1: System components of an ad hoc testbed

In the current Linux implementation [92], socket layer interface implemented in the kernel provides a standard API which allows user space programs to open a communication endpoint to a remote device. The implementations of transport layer protocols (TCP and UDP) are hidden in the socket layer. Because of this, some researchers developed their routing protocol in the kernel to achieve efficiency. For example, the Mornach project team of Carnegie Mellon University developed Dynamic Source Routing (DSR) [51] inside the network stack [93]. University of Maryland also developed an ad hoc network testbed on Linux by adding a Forwarding Engine (FE) into the kernel [94]. Figure 6.2 summarizes this architecture. This strategy increases the difficulty of implementation and installation in heterogeneous environments since it touches the operating system kernel. This approach is suitable for the final version.

Some researchers prefer not to modify the Linux kernel codes. Modules stay in user space as much as possible. The testbed is implemented as a Linux add-on

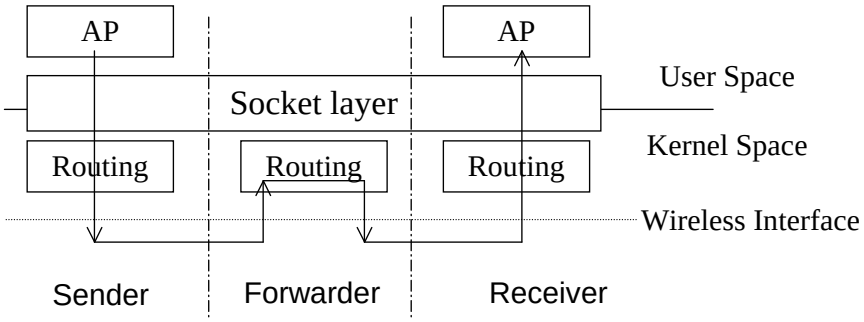


Figure 6.2: Kernel modified routing architecture

rather than a Linux kernel patch. This approach offers the portability and flexibility needed for an experimental system. They usually benefit from the kernel level IPv4 forwarding support built into the Linux operating system to implement routing protocol. An IP forwarding module provides the basic network layer packet routing. This module analyzes the packet's header (destination address, TTL and sometimes source address), and then sends the packet to the corresponding device or drops the packet according to the routing table. The Linux kernel forwards packets according to the following procedures: The network interfaces accept and send all packets to the kernel. The kernel accepts all packets, checks the destination address with the kernel level routing table and decides whether to forward them. A message with a destination not found in the routing table is forwarded to the default gateway. If there is no viable forwarding location, the packet is dropped and an ICMP [95] destination error message is sent. Using this forwarding support, these researchers implemented routing protocol as a user level daemon which updates and maintains the kernel level routing table. Figure 6.3 illustrates this idea. We chose this structure to implement our unicast routing protocol since it on one side reduces developing charge, facilitate program installation and on the other side keeps delivery efficiency by using kernel level forwarding.

However, multicast routing protocol implementation cannot use the same architecture unicast one. First, not all Linux kernels support multicast packet forwarding. Secondly, those kernels with multicast packet forwarding support, do not allow single device forwarding. That is to say the kernel will not forward a packet to the interface where it received the packet to prevent transmission loops. But in MANET, multicast forwarding happens very on the same wireless interface. To overcome these limitations, a structure illustrated in Figure 6.4 is used to implement the multicast routing protocol. In this structure, when an application sends a multicast packet, the Linux kernel, instead of sending it directly to the wireless interface, gives this packet to routing agent running in user space. Then routing agent encapsulates the packet and transmits it hop by hop in a suitable way (broadcast or unicast) till destination routing agent. Finally, the destination routing agent decapsulates the packet deliver this packet to local application. This procedure permits

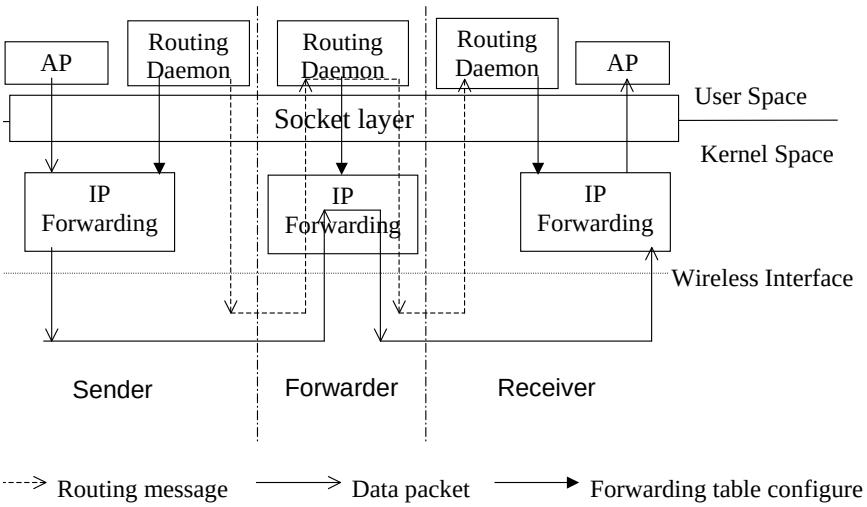


Figure 6.3: User space routing daemon architecture

multicast routing protocol to forward packets in user space.

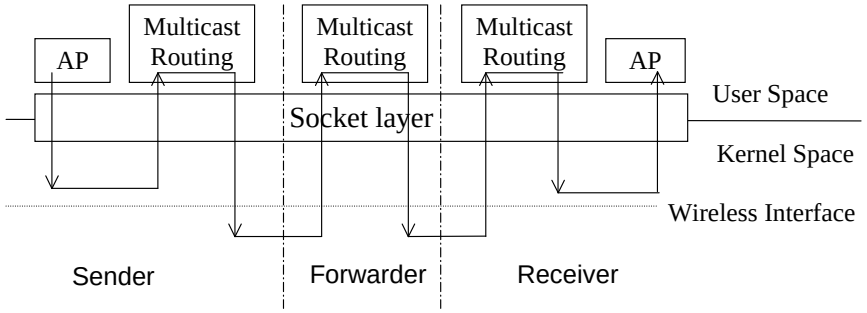


Figure 6.4: User space multicast routing architecture

6.2 DDR Implementation and Validation

In this section, we first give an overview of the Distributed Dynamic Routing algorithm (DDR) and then introduce how we implemented this protocol and validated the implementation.

6.2.1 An Overview of DDR

DDR stands for distributed dynamic routing algorithm, which deals with the problem of topology management and contributes to local routing through clustering in mobile ad hoc networks [89]. To do so, the algorithm selects for each node a neighbor, called *preferred neighbor*, that has a maximum degree of connectivity

in the neighborhood based on the information provided by periodical beaconing exchanged *only* between a node and its neighboring nodes. During the beaconing process, each node gathers the information describing its neighborhood in its neighboring table. This table enables each node to elect their preferred neighbors. The link between a node and its preferred neighbor then becomes a preferred link. The set of preferred links in the neighborhood forms a set of preferred paths in the network, which will be used during the routing process. It has been proved that connecting each node to its preferred neighbor yields to a forest (i.e. no cycle) irrespective of the network topology [89]. Such a forest indeed reduces the broadcasting overhead by selecting a subset of neighboring nodes for forwarding a packet. Finally, each tree of the forest forms a zone, and is maintained proactively. These zones were constructed in order to reduce the delay due to routing process and to reach high scalability. As a result, the algorithm extends the network topology from node level to to zone level (i.e. zone abstraction) by partitioning the network into a set of proactive zones.

In addition to the neighboring table, which maintains the node identifier (NID) and node degree (Deg) about the nodes within the transmission range, the algorithm builds two extra local tables, namely: intra-zone, and inter-zone. Intra-zone table is the table through which a node detects the structure and changes to the tree it belongs to. This table consists of two critical pieces of information: direct preferred neighbor (PN) and the neighboring preferred neighbor(s) learned by the direct preferred neighbor (learned_PNs). In order to construct this table, each node sends a beacon indicating its preferred neighbor, or if the PN remains unchanged a beacon is sent to indicate the learned_PN(s) of the preferred neighbor. Upon receiving such beacons, a node can gather information describing the tree members and the way to reach them. Such information is considered valid for a limited period of time, and must be refreshed periodically to remain valid. Expired information is purged from the table. This table is used to reduce the rebroadcasting overhead to a minimal set of preferred neighbors (in graph theory, this set provides a heuristics to the problem of finding the minimum connected dominating set; MCDS), as well as to provide the local routing information to the routing protocol. Inter-zone table, on the other hand, keeps the information about the connectivity with the neighboring zones of the zone to which the node belongs. It provides the gateways to other zone to the routing protocol.

6.2.2 DDR Software Architecture

The structure of the DDR implementation is shown in Figure 6.5. DDR implementation opens two interfaces for communication. Interface I1 is a UDP socket for sending and receiving beacons and interface I2 is used to modify the IP forwarding table through a netlink socket. Based on the information provided by beacons, DDR establishes a neighbor table and selects the preferred node. Then, DDR constructs an intra-zone table (intra_ZT) and an inter-zone table (inter_ZT) to define spanning tree. DDR sets its routing table and modifies IP forwarding table accord-

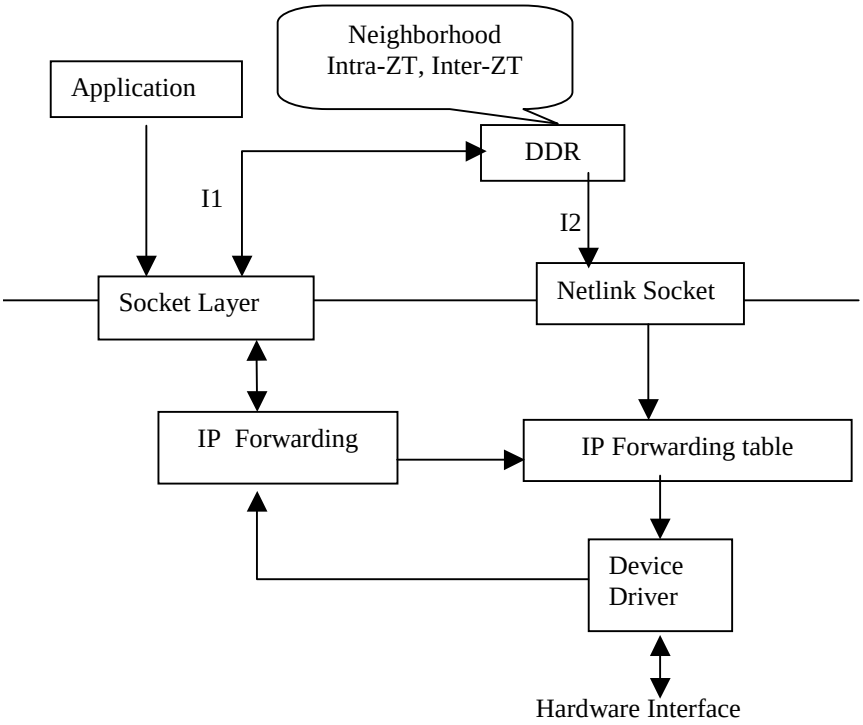


Figure 6.5: DDR implementation structure

ing to the intra-zone table. In this way, we implemented proactive unicast packet routing by using the intra-zone table of DDR.

The flow chart in Annex A.2.1 gives a closer view of DDR's implementation.

6.2.3 Validation of DDR Implementation

To validate the DDR implementation, we used four portable PCs. The operating system is Linux kernel version 2.4.18 provided by Red Hat 7.3. All portable PCs were equipped with IEEE802.11 wireless network cards configured in ad-hoc mode, operating on 2.4 GHz bandwidth and communicating at 2 Mb/s with transmission power of 1mW. The IP address configuration of these four PCs satisfies the relationship: $@Radio1 < @Radio2 < @Radio3 < @Radio4$.

Because we tested the functionalities of DDR such as preferred node selection and intra-zone clustering in this step, there is no need for traffic during the tests. We observed the tables constructed by DDR and compared them to network topology.

We first placed the four PCs to construct an ad hoc network as shown in figure 6.6. Each node is within the transmission range of its direct neighbors. The observed tables of the four portable PCs are illustrated in Tables 6.1, 6.2 and 6.3. The inter-zone table of all four nodes are empty. In this scenario, the degree of Radio1 and Radio4 is 1 and that of Radio2 and Radio3 is 2. Radio1 chose Radio2 as prefer node, Radio2 chose Radio3, Radio3 chose Radio2 and Radio4 chose Radio3.

These four nodes forms a DDR zone. As a result, their Inter-zone tables are empty.

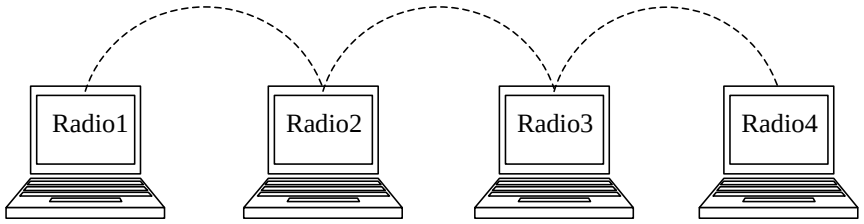


Figure 6.6: DDR validation: line scenario

Table 6.1: Neighbor tables

Radio2

(a) Radio1

Radio1
Radio3

(b) Radio2

Radio2
Radio4

(c) Radio3

Radio3

(d) Radio4

Table 6.2: Intra-zone tables

Radio2	Radio3, Radio4

(a) Radio1

Radio1	
Radio3	Radio4

(b) Radio2

Radio2	Radio1
Radio4	

(c) Radio3

Radio3	Radio1, Radio2

(d) Radio4

Then, we moved Radio4 to the coverage range of all other nodes (Radio1, Radio2 and Radio3) as shown in Figure 6.7. We obtained routing tables of these four nodes as shown in Tables 6.4, 6.5 and 6.6 and inter-zone tables are not listed since they are blank. In network topology, the degree of Radio1 and Radio3 is 2 and that of Radio2 and Radio4 becomes 3. When the maximum degree corresponds to more than one neighbor, node selects the neighbor which has the greatest IP address as the preferred node. According to this rule, Radio1 and Radio3 chose Radio4 as preferred node since $@Radio4 > @Radio2$, while Radio2 and Radio4 chose each other as prefer node. This scenario does not create the second zone either. Consequently the Inter-zone tables of all nodes are empty.

Table 6.3: IP forwarding tables

Dest.	GW.
Radio2	Radio2
Radio3	Radio2
Radio4	Radio2

(a) Radio1

Dest.	GW.
Radio1	Radio1
Radio3	Radio3
Radio4	Radio3

(b) Radio2

Dest.	GW.
Radio1	Radio2
Radio2	Radio2
Radio4	Radio4

(c) Radio3

Dest.	GW.
Radio1	Radio3
Radio2	Radio3
Radio3	Radio3

(d) Radio4

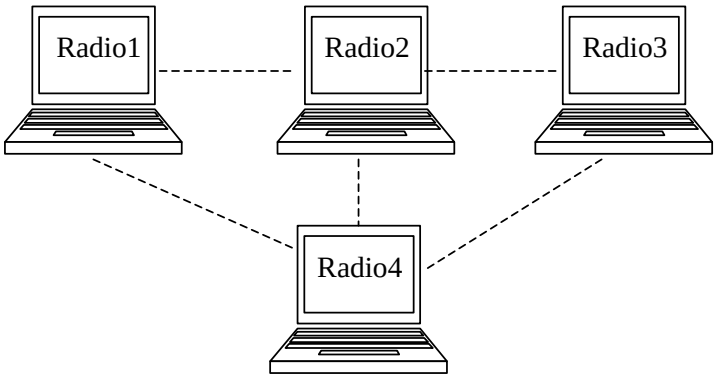


Figure 6.7: DDR validation: star scenario

Table 6.4: Neighbor tables

Radio2
Radio4

(a) Radio1

Radio1
Radio3
Radio4

(b) Radio2

Radio2
Radio4

(c) Radio3

Radio1
Radio2
Radio3

(d) Radio4

Table 6.5: Intra-zone tables

Radio4	Radio2, Radio3

(a) Radio1

Radio4	Radio1, Radio3

(b) Radio2

Radio4	Radio1, Radio2

(c) Radio3

Radio1	
Radio2	
Radio3	

(d) Radio4

Table 6.6: IP forwarding tables

Dest.	GW.
Radio2	Radio4
Radio3	Radio4
Radio4	Radio4

(a) Radio1

Dest.	GW.
Radio1	Radio4
Radio3	Radio4
Radio4	Radio4

(b) Radio2

Dest.	GW.
Radio1	Radio4
Radio2	Radio4
Radio4	Radio4

(c) Radio3

Dest.	GW.
Radio1	Radio1
Radio2	Radio2
Radio3	Radio3

(d) Radio4

These tests validate our DDR implementation and show that DDR can be directly used to route unicast packets in small networks. When the network size increases, DDR will create more than one zone. In this case, a routing protocol (e.g. HARP [97]) is needed to explore route for nodes belonging to different zones.

6.3 MRDC Implementation and Validation

6.3.1 MRDC Software Architecture

According to the implementation structure (see Figure 6.4), MRDC is designed to route packets in user space to simplify installation and test in different systems. The implementation architecture of MRDC is shown in Figure 6.8. Besides the tables defined in MRDC in chapter 3, this implementation contains a group table which records all multicast groups of which this node is a receiver and/or source. Because this implementation is aimed to demonstrate how to support multicast applications, in addition to MRDC core module, the IGMP module and the tree monitoring module are introduced. The MRDC core module is further divided into two parts: a Routing Part (RP) which, as described in Section 3, constructs and maintains a multicast tree on demand and updates the multicast routing table, and a Multicast Forwarding Part (MFP) which forwards multicast datagrams according to the multicast routing table. IGMP module performs as a simplified router to host part of Internet Group Management Protocol (IGMP) version 2 [6]. It detects membership changes on the local host and updates accordingly the group table. The group table can also be updated by MFP and provide group membership states to the MRDC core module. Tree monitoring module is an additional functionality. It collects multicast routing information of a pre-defined group from multicast routing tables in each multicast tree member. Then a tree monitor written in JAVA replays the multicast tree structure based on these information. MRDC opens three sockets for inside and outside communication. IGMP module uses an IGMP socket (named igmp_socket) to send and receive igmp packets. MRDC core module opens a UDP socket (called udp_socket) for inter-node message exchange and a raw socket (denoted as raw_socket) to deliver multicast datagram to local application. The tree monitoring module shares udp_socket for multicast routing information collection. This architecture contains multicast membership detection, multicast tree (re)configuration, multicast packet delivery and tree monitoring. The rest of this section explains the working of these modules in detail.

Tables

The MRDC implementation involves four tables: Group table, Duplication table, Unicast routing table (URTable) and Multicast routing table (MRTable). The three latter tables are identical to those in chapter 3. The group table stores the multicast group memberships of the local host.

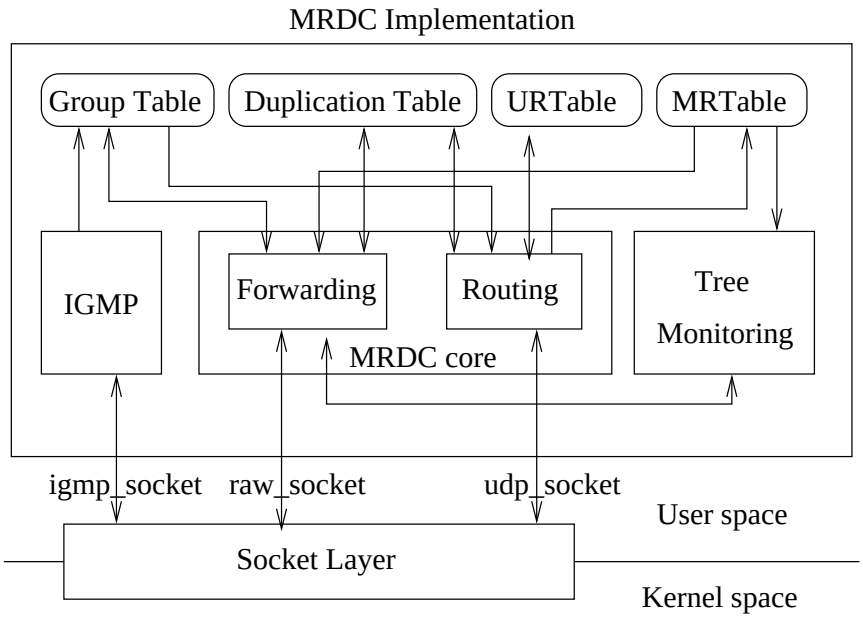


Figure 6.8: MRDC implementation structure

As shown in table 6.7, a group table holds three fields: group ID number (GID) and two membership state fields SENDER and RECEIVER. GID represents the ID number of the group in which this node is a sender and/or receiver. The ID number could be an address of class D in Internet. The group table is modified by the IGMP module and the Forwarding module. When MFP detects that a node is sending multicast packet to the group GID, it sets the SENDER field of that entry. On the other hand, if an application registers to receive packet of multicast group GID, the IGMP module will receive a group join message. Then this module sets the RECEIVER field of the corresponding entry. If the membership is not confirmed before a time out or if the node recognizes a membership change (e.g. the IGMP module receives a group leave message), the relevant field is unset. An entry is removed from the table if both membership state fields are unset.

Table 6.7: Group Table

GID	SENDER	RECEIVER

Routing Part

The Routing Part (RP) of MRDC core module creates and maintains multicast trees on demand. The RP starts to create a multicast tree when the multicast forwarding

part (MFP) detects that local host sends a packet to a multicast group which does not exist in the multicast routing table. At the same time, this node becomes the core. The routing information is stored in a multicast routing table. After the tree construction phase, the MFP begins to deliver datagrams. Nodes can join or leave the multicast group at any time during the session. The core maintains the multicast session by refreshing the tree periodically, but if the source status is timeout which means no multicast datagram is sent during a period of time, the core stops this maintenance and the multicast tree is automatically erased by deleting the associated routing information from the multicast routing table. In addition, this implementation supports core immigration. Each multicast source assumes itself is the core and begins to broadcast its own CA message when a multicast tree is erased. When receiving multiple CA messages addressing same multicast group, nodes compare core IP address and choose the greatest one. The source which has the biggest IP address thus becomes the core competition winner and continue to send its CA message periodically. The other sources stop sending their CA message and join the multicast tree as a normal group member.

The flow chart in Annex A.2.2 explains in more detail how MRDC operates in this testbed.

Multicast Forwarding Part

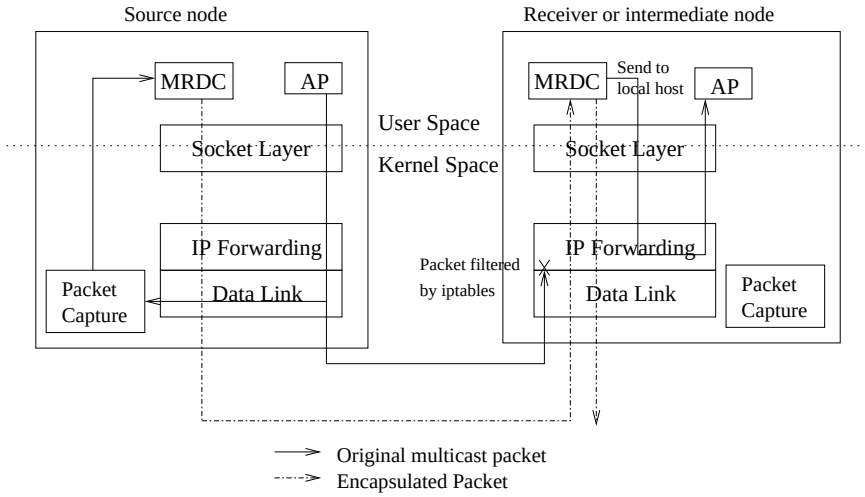


Figure 6.9: Multicast packet forwarding in MRDC implementation

We met several difficulties during MRDC implementation: i) detecting multicast datagram. MRDC cannot get multicast datagram directly since it runs in user space. ii) forwarding of a multicast datagram. In the table-driven unicast case, the routing agent can run in user space and modify the routing table in kernel. The kernel level IP forwarding module forwards unicast datagrams according to the routing

table. MRDC cannot use the same approach because multicasting in MANETs and multicasting in fixed Internet is different. The routing table's multicast tree states consist of local interfaces instead of neighbor identities, and the verification for incoming data is done on incoming interface rather than on the sender. However, one MANET node can use the same interface talking to any neighbor on the same wireless channel. Hence, the incoming physical interface verification done by the IP kernel is no longer applicable. iii) detecting duplications. Duplication cannot be avoided since nodes use the same wireless channel to communicate.

In Multicast Forwarding Part (MFP), we introduce a combination of packet capture, packet encapsulation/decapsulation and packet filtering, to solve the above problems. Figure 6.9 illustrates this procedure. A multicast datagram sent by an application is received by packet capture at the data link layer. Packet capture passes the packet to MRDC. MRDC encapsulates the captured packet into a MRDC data packet and then broadcasts it to neighbors through upd_socket. Nodes relay MRDC data packet to the group receivers according to the MRDC's multicast routing table. At last, MRDC on the receiver side extracts the datagram from the MRDC data packet and sends it to the local application through raw_socket. To detect duplication during forwarding, MRDC uses the information stored in MRDC data packet header. On the other hand, if a group receiver is within the coverage region of a group source, it receives multicast datagrams directly from source, that generates duplication at application side. In order to avoid this phenomenon, packet filtering (iptables installed in Linux) is applied to each node. We will explain in details how they work.

The basic technique used to import multicast traffic packets into user space is the same as Unix tcpdump[98]. Since all machines use Linux, we use the libpcap facility. It listens to traffic at the data link layer and sniffs packets which are in accordance with a pre-defined rule set. For example, in this implementation, we want to capture all multicast datagrams sent by a node itself. If the ip address of node A is 192.168.25.197, the following rule is set:
(tcp or udp) and ip multicast and src host 192.168.25.197
This rule captures all tcp and upd multicast packets sent by node A. Then libpcap facility passes the captured packets to MFP as a raw packet. MFP checks the destination address in the ip header of captured packet by consulting multicast routing table. If the destination address belongs to a new multicast group, MFP sets the group source state in group table, caches the packet and commands the routing part to create a multicast tree. Otherwise, if the destination address belongs to an existing group, MFP refreshes the group source state and encapsulates the captured multicast datagram into an MRDC data packet and broadcasts it through udp_socket. An MRDC data packet header, as show in Figure 6.10, has two fields, Type and Reference. The type field distinguishes a MRDC data packet from other MRDC control message. Data packets are processed by MFP. The reference field stores the sequence number, which is assigned by the source, in order to detect packet duplication.

When receiving an MRDC data packet, MFP scans the IP header and UDP

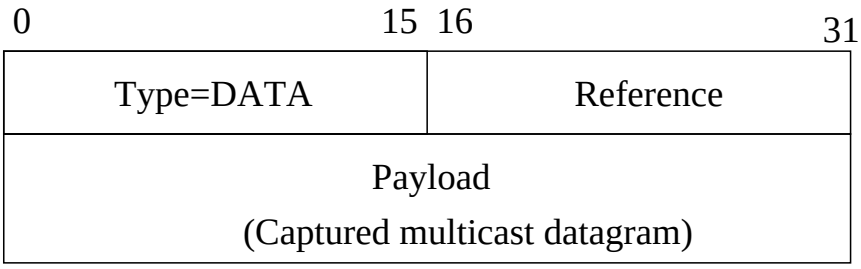


Figure 6.10: MRDC packet structure

header of the payload packet to obtain source address and source port number. Then MFP matches this information combined with the reference number in MRDC data header against the duplication table, for duplication detection. If it concerns a packet received before, MFR drops it. Otherwise, the duplication table stores the packet information and MFP broadcasts the MRDC data packet to its neighbors if it is a multicast tree member according to MRTable. At the same time if the group table indicates that the local host is a receiver of the corresponding multicast group, MFP decapsulates the MRDC data packet and transfers the multicast datagram through raw_socket.

An important point that we should take into account is the broadcast characteristic of wireless link. An application may receive the same multicast datagram which has been already received if the node is within the coverage range of the source. Therefore it is necessary to include the packet duplication avoidance mechanism in the implementation. To filter packets from source,we use the Netfilter/iptables facility [99]. It sits in between the kernel IP stack and network device drivers and manipulates every packet in or out of this host according to pre-defined rules. Rules can be set or changed at any time through a command interface. For example, if the wireless interface is eth0, the following rule is set:
iptables -A INPUT -d 224.0.0.0/16 -p udp -i eth0
This rule drops all udp multicast packets coming from eth0.

IGMP Module

The IGMP Module periodically sends IGMP membership query message through igmp_socket and listens at igmp_socket. It sets up the group receiver state in the group table upon the reception of a membership report message. On the contrary, it unsets the group receiver state when capturing an IGMP leave group message.

On the other hand, group table periodically unsets the states which have not been updated for a fixed amount of time.

Tree Monitoring Module

The node whose IP address coincides with the predefined monitor address is the topology monitor. It consults the multicast routing table to check whether a multicast tree exists for the pre-defined group. If it is the case, the monitor broadcasts a message to the rest of the network. This message is comprised of the monitored group, the monitor address, the sequence number and last hop fields. When this message propagates through the network, a reverse path to the monitor is constructed. All members of the corresponding multicast tree send the information including the IP address of their upstream and downstreams to the monitor through this reverse path. This procedure is executed periodically.

Timers

We selected five seconds for the multicast tree refresh interval. IGMP queries membership every eight seconds and the membership timeout was set to eighteen seconds. Tree monitoring collects tree information every second.

6.3.2 Validation of MRDC Implementation

In this section, we detail how we validated the MRDC implementation in a real ad hoc testbed.

Testbed configuration and Implementation Platform

Some issues, such as expensive hardware and the difficulty to organize mobile tests, discourage researchers from the construction of an ad hoc testbed. Our ad hoc testbed comprises portable personal computers (portable PCs) and PDAs. On one hand, portable PCs are stable and powerful. We use them to generate video stream and show multicast tree. On the other hand, PDAs are cheaper and lighter than portable PCs. This network configuration reduces hardware cost and at the same time facilitates mobility testing.

The Ad hoc network nodes in our testbed are Intel Pentium III based Dell C600 laptops and Intel StrongARM based Compaq iPAQ H3850s equipped with IEEE802.11 wireless network cards.

MRDC was developed on Linux kernel version 2.4.18 as included in Red Hat 7.3. All tools and software packages that we used in our development originate from the software bundle incorporated within Red Hat Linux version 7.3. The PDAs used the Familiar Linux v0.7 package with kernel version 2.4.19-rmk6-pxa-hh13 as their operating system. It was necessary to install packet capture and packet filtering modules on the PDAs since these packages are not available in the installation package bundle. These two modules were used in multicast forwarding and performance evaluation. We used arm-linux-gcc from tool-chain to make cross platform compilation on the Red Hat 7.3 platform.

We created a six node testbed for our multicast experiments. We studied the bandwidth utilization of MRDC in a stationary network scenario and verified the correctness of MRDC in topology and membership dynamic scenarios. During the evaluation, all WaveLan devices operated on 2.4 GHz bandwidth and communicated at the capacity of 2 Mb/s with transmission power of 1mW. The WaveLan devices were operated in an ad hoc mode. An IP address is distributed to each node before the tests. These IP addresses satisfy $@A < @B < @C < @D < @E < @F$.

Stationary Network Scenario and Results

The experimental network setting is shown in Figure 6.11. This topology is similar to [96]. Our network consisted of six nodes among which three are portable PCs (nodes A, B and C) and the other three are PDAs (nodes D, E and F). All nodes can hear each other in the MAC layer. A virtual wall is constructed in the network layer via iptables. For example, a filter is set in node A to drop all packets coming from nodes D, E and F. A topology monitoring program developed by Hitachi ran in Monitor to display the multicast tree based on the informations collected by the tree monitoring module. In this experiment, a file is multicast from node A to the receivers E and F. Figure 6.12 illustrates two multicast tree structures which were shown by topology monitor during the experiment. That is because MRDC chooses the first discovered path. If node F receives a new CA message from node E first, a one branch multicast tree, tree_1, is constructed. Otherwise, a two branches tree, tree_2, is formed. Table 6.8 shows the measurement results. The total throughput is far below the full WaveLan data rate of 2 M/s. There are three reasons. The first, is network layer multi-hop forwarding while nodes were physically placed together. Multicast source and forwarder share the same wireless channel. The second reason is that we did not prevent the original multicast traffic to be injected into the wireless channel. The third one is that two alternative multicast tree contain different number of interior nodes. Tree_1 has three interior nodes while Tree_2 contains four interior nodes. In the former tree, the bandwidth is divided by four (one original traffic and three forward traffic) and in the latter tree, the bandwidth is divided by five. In this MRDC implementation, the MRDC overhead comes mostly from multicast packet encapsulation while routing messages such as CA, RAR and RAA messages can be ignored. IGMP control overhead can be ignored. However, tree monitoring overhead is high because we chose a small tree information collection interval. This small interval permit to observe tree changes in the next experiments.

In this implementation, MRDC operates in user space. If we succeed in moving MRDC to the kernel, costly kernel-to-user crossing for store-and-forward packets can be avoided, the overhead caused by encapsulation can be greatly reduced and original traffic will not be injected into network anymore. We believe consequently that the data throughput can be significantly improved.

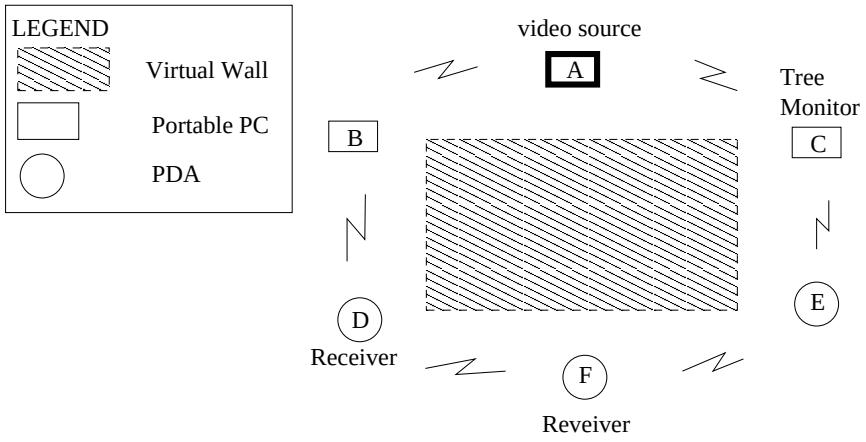
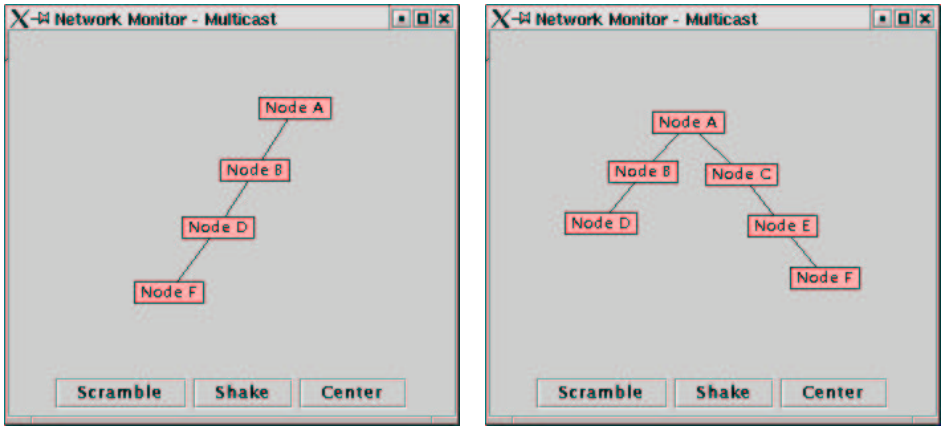


Figure 6.11: Stationary Network



(a) Tree 1 (b) Tree 2

Figure 6.12: Tree Structure in Stationary Network

Table 6.8: MRDC in stationary network with one multicast source

	Value	% of total
MRDC control packet O/H	0.26 kb/s	0.07%
MRDC data packet header	15.75 kb/s	4.11%
Total MRDC O/H	16.01 kb/s	4.18%
IGMP O/H	0.04 kb/s	0.01%
Tree Monitoring O/H	1.29 kb/s	0.34%
Avg. # of multicast tree branch	1.4	N/A
Effective data throughput	365.36 kb/s	95.47%
Total throughput	382.7 kb/s	100%

Dynamic Network Scenario and Results

In this test, we removed the virtual wall constructed by iptables. All nodes were initially within the coverage region of the others as shown in Figure 6.14. Vic [100] addressed to a multicast address ran on node A, D, E and F to form a video conference group. A web-cam is connected to node A to serve as a multicast source and send a video stream to the conference group through vic (Figure 6.13). Because vic sends multicast packets at regular intervals to announce membership to other vics, although there is only one video source at the application level, each vic is a multicast source from the point view of MRDC. Thus this conference group is a multiple source scenario for MRDC. We configured IP address of wireless nodes to satisfy $@A < @B < @C < @D < @E < @F$. We ran vic first on node D and then on nodes A, E and F. This running sequence resulted in node D becoming core. Figure 6.15(a) demonstrates the tree structure.

This test contains two parts: topology dynamic part and membership dynamic part to test correctness and efficiency of the MRDC implementation. In the topology dynamic part, we moved node F outside of the coverage range of node A and D but still within the coverage of node B. During the movement, node F firstly receives video stream directly from node A. Then multicast tree structure changes (see Figure 6.15(b)) and node F get video stream through the relay of node B. In membership dynamic part, we kept node F outside of the coverage range of A and D but within the coverage of node B and stopped vic on node D. Tree monitoring showed that the tree structure changed, and after a short transient time, F became core and node D disappeared. Then, we re-ran vic on node D. This node joined the tree as a leaf node as shown in Figure 6.15(c).

During this test, the video transmission rate was around 300kb/s. The replayed video was fluent in both portable PCs and PDAs and the error rate shown in vic was smaller than 10% most of time.

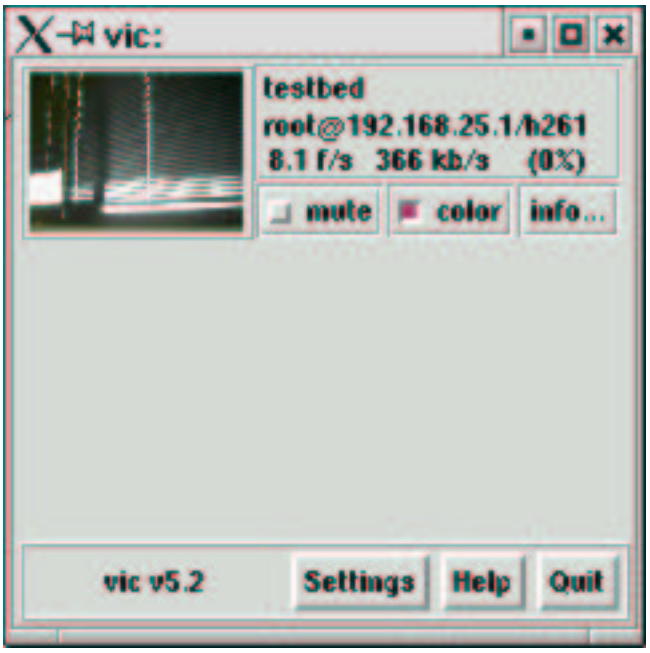


Figure 6.13: VIC operating on the ad hoc testbed

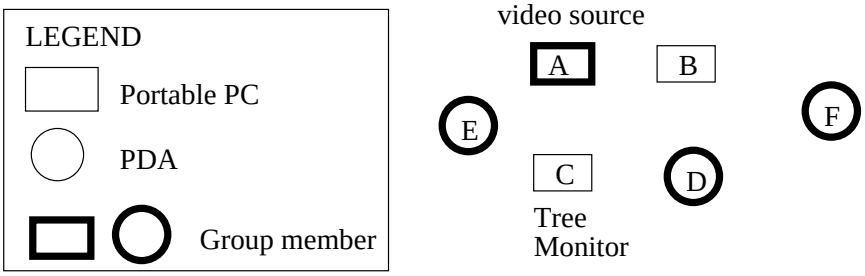
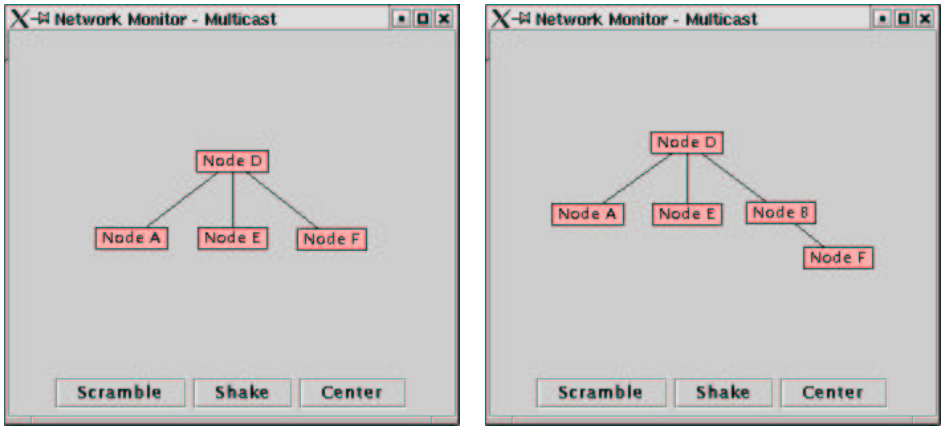
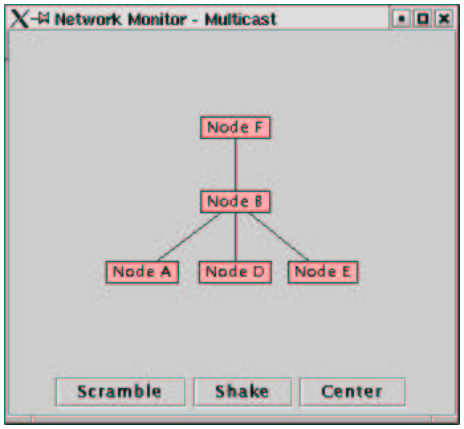


Figure 6.14: Dynamic Network



(a) Initial Multicast Tree Structure (b) Multicast Tree Structure after movement



(c) Final Multicast Tree Structure

Figure 6.15: Tree Structures in Dynamic Network

6.4 Conclusion

We presented our experience on implementation of DDR, a unicast routing protocol, and MRDC in an ad hoc testbed which consisted of portable PCs and PDAs. The hybrid network configuration reduces the hardware cost of the testbed and facilitates mobility tests since PDAs are smaller and cheaper than portable PCs.

Routing programs were run in user space to reduce the difficulty of implementation, installation across different hardware and software environments. In order to realize packet forwarding in ad hoc networks, these programs either manipulate kernel ip forwarding table (the unicast routing case) or catch packet from the data link layer and then transfer these packet in their way (the multicast routing case). Therefore, the applications can always use standard socket layer interface to communicate.

The main parts of MRDC's control plan, including tree construction and maintenance, have been successfully implemented in the user space of Linux operating system, while only the broadcast mode of MRDC's forwarding plan is employed in the forwarding module of the implementation. We also designed a mechanism in the forwarding module to solve the problems of how to forward multicast packet and realize on traffic demand fashion when the program runs in user space. Besides these, the IGMP module and the tree monitoring module have been designed and integrated in the MRDC implementation to form a complete solution for multicast application support and topology monitoring. We evaluated the bandwidth utilization of this implementation in a stationary network scenario and showed that if we do not consider the encapsulation overhead, MRDC creates a moderate control overhead for multicast traffic delivery. Then, we used a MBone traffic - vic to test MRDC with node movement and membership changes. The results prove that MRDC correctly deals with topology dynamic and membership dynamic.

We implemented and tested the implementation of DDR and MRDC separately. Current IP forwarding table does not store some particular information required in ad hoc network routing. For example the information of last update time, which is necessary for stale routing information detection, is not supported. This issue compels DDR and MRDC to possess their own routing tables and prevents them from sharing their unicast routing information. We should study an efficient way to facilitate their cooperation in order to reduce routing and storage overhead.

As the study of unicast routing protocol based on DDR progresses, new functionalities will be added to the implementation to support packet routing across zones and improve route selection. On the other side, the success of MRDC implementation in user space encourages us to bring these functionalities, or the forwarding module as the first step, into kernel and test the scalability of MRDC. This will constitute an ad hoc testbed which can support both point-to-point and many-to-many communications, which will allow us to study protocols of other layers and new ad hoc applications.

Chapter 7

Conclusions and future work

Multicasting for mobile ad hoc networks is crucial studied during last years. It is a method of sending packets to more than one destination node at a time. Using this method, the sender only needs to send every datagram once and compared with broadcast, only relevant routers and hosts take part in the transmission and reception of multicast datagrams. Due to its ability of delivering point/multipoint to multipoint packets in an efficient and scalable way, multicasting is seen as a suitable method to support some potential applications of MANETs which are characterized as group-oriented. However the properties of MANETs such as wireless interface, frequent topology change, limited bandwidth, etc. make the design of multicasting protocol for such a type of networks a challenger task. There are many open issues in multicasting for mobile ad hoc networks, for instance: group management, best-effort multicast routing, reliable multicasting, multicast transport and so on. The basic functionality is to deliver multicast packets to their destinations, or multicast routing.

Multicast routing protocols for MANET have twin design goals of high delivery success rate and low overhead. The overhead of routing a multicast packet to its receivers consists of control overhead and forwarding overhead. We have two methods to improve packet delivery success rate, either update routing information more frequently to keep tracking topology changes or introduce a certain degree of redundancy to overcome transmission failures. The former method increases the control overhead and later adds extra forwarding overhead. Thus, it is difficult to maximize delivery success rate and minimize overhead simultaneously. Some degree of trade-off between them is always required. However, when looking for this trade-off, the application requirements should be also taken into account. Applications are either sensitive or insensitive to packet loss. For those loss sensitive applications, overhead should concede to delivery success rate, while for loss insensitive applications, reducing overhead becomes more important. In brief, multicast routing protocols for MANET should optimize delivery success rate and overhead with the respect of application's properties.

For this goal, we designed a multicast routing protocol for small and medium

size MANETs, named Multicast Routing protocol with Dynamic core (MRDC). This protocol contains two plans: control plan and forwarding plan. In the control plan, we focused on studying an optimal way of constructing and maintaining delivery structure which should consume less bandwidth for routing and future packet forwarding. In terms of providing transmission efficiency, MRDC employs a tree to connect group members. The root of a multicast tree, which is called core in MRDC, is initially the first source of a multicast session. Then the core can move from one source to another according to network and traffic conditions. On consequence the multicast tree is source-based for single source group and group-shared for multiple sources group. This tree is periodically refreshed to adapt to current topology so that MRDC could maintain its efficiency. In point view of reducing control overhead, multicast trees are constructed when traffic begins and destroyed once transmissions finish. That is what we call **on traffic demand**. If node's mobility makes tree fragment, a local recovery procedure is executed to repair the tree. However, this procedure just attempts to maintain tree physically connected, logical faults such as logical fragmentation and containing longer path in tree are left to periodical tree refreshing. In this way, the transmission efficiency of tree structure is maintained in most cases and the cost for tree repairing is reduce. The simulation results show that when all protocols use the broadcast-like method to transmit multicast packets, MRDC outperforms ODMRP, a mesh-based multicast routing protocol, and ADMR source-based tree, in packet delivery success rate, end-to-end transmission delay and overhead (both forwarding overhead and control overhead). Furthermore, MRDC provides a stable performance in most cases as network load and nodes' mobility change.

The forwarding plan of MRDC addresses the problems of forwarding multicast packets with the respect of network situation and application requirement. Two transmission modes are defined in forwarding plan if the underlying MAC protocol is IEEE802.11-like. One transmission sends multicast packets with CSMA/CA mechanism without guarantee, just like IEEE802.11 sending broadcast packets. This mode called broadcast transmission mode. The other mode sends a multicast packet as a set of unicast packet with four-way RTS/CTS/DATA/ACK exchange. Thus this mode is called unicast transmission mode. The broadcast mode creates less forwarding overhead and transmission delay but transmission might fail due to collision or wireless interference. The unicast mode has the inverse effect. It gives certain degree of multicast transmission reliability with the cost of extra forwarding overhead and transmission delay. A mechanism, called adaptive forwarding mechanism, is studied to well choose transmission mode according to network situation. This mechanism selects unicast mode in low load networks and broadcast mode in high load networks. Transmission modes can also be smartly selected to support different type of applications. Broadcast mode is suitable for applications which can tolerate packet loss. Unicast mode can be used to support packet loss sensitive applications. If applications have no specific requirement, adaptive forwarding mechanism can be activated to optimize packet delivery success rate and forwarding overhead. The simulation results prove that unicast transmission mode

can improve the packet delivery success rate of MRDC from 1% to 7% in low load networks compared with broadcast transmission mode. Well selected parameters allow adaptive forwarding mechanism choose a suitable transmission mode so that MRDC could provide a better delivery success rate in both high and low load networks. MRDC can also employ these transmission modes to support different requirements of applications. If applications can tolerate packet loss, MRDC adopts broadcast mode to minimize overhead. On the other hand, for loss-sensitive applications, MRDC activates adaptive forwarding mechanism to provide the best delivery success rate.

MRDC is originally designed for small and medium size networks. As the number of nodes increases in the network, the mechanims of flooding core advertisment in the entire network gradually becomes inconvenient. At the same time, the latency of multicast tree construction and refresh auguements also due to heavy control overhead and large distance. However, the research work of DDR demonstrates a potential solution. After cluster nodes into zones, MRDC can regard each zone as a logical network partition and maintain multicast tree in each partition where there is at least two multicast members. Then we can study a mechanism to maintain the connectivits adn deliver packets among these sub-trees distributed in different zones. In this way, both control overhead and tree refresh latency can be greatly reduce. Another future work on MRDC is the metric used in multicast tree construction. In current version, MRDC uses the first discovered route to construct multicast tree with the assumption that these routes lead to shortest transmission delay. However, some other metric can also be employed for special goal. For example, MRDC can take the transmission power and node's battery level into account in order to provide power-efficient multicasting. It can also select most stable routes to further reduce the cost of tree maintenance.

Some applications do need a guarantee of multicast delivery (hundred percent delivery success rate). In order to satisfy this requirement and reducing retransmission overhead, ARMPIS is proposed. This protocol extends receiver-assistant and router-assistant retransmission to distribute retransmission responsibility. In receiver-assistant retransmission scheme, routers firstly query receivers in its sub network the request packets. ARMPIS defines "sub network " as neighborhood. In router-assistant retransmission scheme, routers store multicast packets for retransmission. Considering memory limitation and frequent topology changes, ARMPIS makes router randomly cache multicast packets. In this way retransmission responsibility is distributed to intermediate nodes, which reduce source's retransmission charge and also total retransmission overhead. The simulation results demonstrate that this protocol can provide a 100 % delivery success rate in most cases.

The simulation results of ARMPIS demonstrate the necessity of developing a better mechanism to distribute multicast packet storage and schedule retransmission among neighbors if we want to reduce more retransmission overhea. In fact the current mechanism cannot avoid storage duplication among neighborhood. When one node does local query, more than one neighbor may have the request packet in their cache and retransmit the packet to the node. These packets may collide at

the node that makes the node believe that local query fails and it should passe the request to the next node. A better mechanism reduces this kind of collision and improve local recovery succes rate.

We developed an ad hoc testbed by implementation of DDR, a unicast routing protocol and MRDC so that the testbed can support both one-to-one communications and many-to-many communications. This testbed will allow us to analyze the performance of MRDC in real network. It can also be used to study protocols and new MANET applications.

The implemenation of MRDC continues in two directions in order to reduce more overhead. It can be integrate into applications with standard interface. This is the architecture used by MBone. This direction makes the smallest modification of current implementation with usage limitation since programmers should modify current applications or develop new applicaitons accoroding to these interfaces. The other direction is to modify MRDC as a loadable module and operate in linux kernel space. It needs more kernel developing skills but will be more easy to use for application programmers. Another issue on testbed appears when two protocols meet in the same machine. The implementations of unicast routing protocole and multicast routing protocol were done separately. When these two protocoles operate in the same machine, we should consider how to establish cooperation between them in order to provide a more efficient routing.

We hope with our contributions, multicasting protocoles can support more efficient group-oriented applications in mobile ad hoc networks.

Chapter 8

Résumé Détaillé en Français

8.1 Introduction

Les avancées dans le domaine de l'informatique personnelle et des technologies sans fil ouvrent des possibilités passionnantes pour le futur de la gestion des réseaux mobiles. Les réseaux mobiles "ad-hoc" (MANET) sont créés par un ensemble de terminaux sans fil qui communiquent entre eux. Les nœuds d'un réseau ad hoc forment dynamiquement un réseau à façon sans utilisation de quelconque infrastructure existante ou administration centralisée. Ses capacités à fournir rapidement et flexiblement des moyens de communication font des réseaux ad hoc un choix idéal pour certaines applications personnelles, publiques ou d'entreprise. Beaucoup de ces applications sont caractérisées par un degré étroit de collaboration. Le multicast peut s'avérer être une manière efficace de fournir les services nécessaires pour ce genre d'application. En raison de la limitation de la couverture radio de l'interface sans fil, le relayage par sauts multiples peut être nécessaire pour qu'un nœud puisse échanger des données avec les autres à travers le réseau. En conséquence, les défis supplémentaires tels que le changement fréquent de topologie et les ressources limitées de réseau sont à relever dans la conception de protocole multicast.

Face à ces défis, les chercheurs préfèrent développer des protocoles de multicast selon différentes conditions prédéfinies d'utilisation et laisser les applications choisir un protocole particulier selon les environnements. Les exemples typiques de tels protocoles sont On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks (ODMRP) et Adaptive Demand-driven Multicast Routing (ADMR).

ODMRP est développé pour supporter efficacement les communications de multipoint à multipoint dans les réseaux en grande mobilité et faible charge. Par contre, ADMR est un bon choix pour les applications de point à multipoint dans les réseaux à faible mobilité et lourde charge. En ce qui concerne notre approche, notre perspective est que les conditions de réseau pourraient changer d'une façon imprévisible. Les nœuds du réseau peuvent être stables pendant un certain moment et puis devenir très mobiles. Quant à la charge du réseau, elle peut aussi varier

très fortement d'un moment à un autre quand il y a des communications qui se terminent ou des communications qui apparaissent.

Notre motivation de recherche est donc de développer des protocoles de multicast qui s'adaptent mieux aux complexités des réseaux mobiles ad hoc sans fil. Ces protocoles doivent optimiser le taux de livraison de paquets pour répondre aux exigences des applications tout en assurant une bonne utilisation des ressources du réseau et notamment la bande de passante.

Cette thèse contient sept chapitres. Le premier chapitre donne une introduction et présente la motivation des recherches en détail. Nos recherches concernent deux sujets de multicast : protocole de routage multicast et protocole fiable de multicast. Les chapitres 2, 3 et 4 sont dédie au sujet de protocole de routage multicast. Le chapitre 2 analyse les défis qui se posent lors de la conception d'un protocole de multicast dans l'environnement des réseaux ad hoc sans fils et les techniques et stratégies adoptées pars les protocoles existants. Le chapitre 3 explique notre protocole de routage multicast : Multicast Routing protocol with Dynamic Core (MRDC). Et sa performance est analysée dans le chapitre 4 avec un simulateur de réseaux sous différents scénarios de trafic et mouvement. Le cinquième chapitre adresse la question de comment offrir efficacement une diffusion fiable dans les réseaux ad hoc sans fil et introduit notre solution : Reliable Multicast Protocol with Intermediate node Support (ARMPIS). Le sixième chapitre présente nos travaux sur l'implémentation des protocoles de routage de unicast et multicast dans un banc de test. Ce banc de test nous permet de valider notre protocole, évaluer la performance dans un vrai réseau mais aussi d'étudier de nouvelles applications des réseaux ad hoc sans fil. Le dernier chapitre, le septième chapitre conclut nos recherches et indique les travaux à l'avenir.

8.2 Protocole de routage de multicast

Un protocole de routage de multicast réalise la fonctionnalité d'acheminer efficacement les paquets multicast de leur source à leurs destinations. Plusieurs protocoles sont déjà proposés pour les réseaux ad hoc sans fil. Cependant, ces protocoles ont un champ d'application limité. Notre perspective est que un protocole de routage de multicast doit donner une bonne performance dans le plupart des cas puisque les conditions des réseaux sont imprévisibles et pourront changer aléatoirement et fortement. Pour ce faire, nous avons étudié comment délivrer le plus possible de paquets tout en réduisant les surcoûts liés au contrôle et à la transmission dans les différentes situations des réseaux ad hoc sans fil et nous proposons un nouveau protocole de routage de multicast

Synthèse des protocoles courants de routage multicast dans les réseaux ad hoc sans fil

Avant de s'engager dans ce travail, nous devons d'abord analyser les caractères des réseaux ad hoc sans fil et les défis supplémentaires introduits dans la conception d'un protocole de multicast. Ensuite nous devons examiner les techniques

qui sont employées par des protocoles courants de routage multicast. Ces travaux constituent le chapitre 2.

Les réseaux ad hoc sans fil sont constitués de nœuds mobiles équipés d'interface sans fil communicant sans l'aide d'une quelconque d'infrastructure. Ces sont donc les nœuds mobiles qui assurent le relayage pour établir les communications à travers les réseaux. Les nœuds peuvent bouger comme ils veulent et dès qu'ils le veulent. Cela rend les changements de topologie fréquents et imprévisibles. L'interface sans fil est utilisée pour relayer des paquets. Il impose les contraintes suivantes dans la conception d'un protocole de multicast : un taux élevé de perte de paquets, la limitation de bande de passante. En revanche, l'interface radio offre une capacité naturelle de diffusion (broadcast). Nous devons prendre également en compte les contraintes dues à la limitation des ressources des nœuds mobiles telles que capacité de processus, mémoire et batterie. En résumé, les nouveaux défis dans les réseaux ad hoc sans fil sont:

- le changement de topologie fréquent et imprévisible,
- le taux élevé de perte de paquets,
- la limitation de bande de passante,
- la capacité de broadcast, et
- les contraintes de ressources telles que capacité de processus, mémoire et batterie.

Face à ces défis les chercheurs conçoivent leurs protocoles de routage multicast en utilisant différentes techniques et stratégies selon les conditions destinées. Ces techniques et stratégies peuvent être classifiés en quatre catégories: la condition d'initiation (Initiate); la structure de routage (Structure); le moyen de construction et maintenance (ReConfiguration) et la mode de transmission des paquets (Forwarding).

Le premier choix (initiation) est le moment démarrage d'un protocole. Tous les chercheurs ont déclaré que leur protocole de routage multicast est activé dynamiquement ("on-demand") ([20], [21], [22], [23], [24], [25], ...). Cependant, nous pouvons observer que ces protocoles sont activés par deux événements différents qui impactent la performance des protocoles.

Une catégorie des protocoles découvre et maintient les chemins entre les membres du groupe (ces chemins forment une structure de routage) quand il y a au moins un membre présent dans le réseau. Ainsi, lors qu'une source commence à envoyer des paquets, il suffit de découvrir le chemin vers la structure de routage puisque tous les récepteurs sont déjà connectés dans cette structure. Le délai de découverte des récepteurs est donc minimisé. Cette stratégie est appelée "on group demand".

Une autre stratégie, nommée "on traffic demand", est aussi adoptée dans les réseaux ad hoc sans fil. Elle active les fonctions du protocole tels que construction

de la structure de routage lorsqu'une source commence à envoyer les paquets et termine (détruit la structure de routage) une fois qu'il n'y a plus trafic dans le réseau afin de réduire le surcoût de maintenance des chemins entre les membres. Néanmoins, l'inconvénient est le délai introduit pour attendre que les chemins vers tous les récepteurs soient bien établis.

Le deuxième défi concerne la manière de connecter les membres d'un groupe (structure de routage). Parce qu'il offre un coût réduit de transmission, l'arbre est une structure naturelle qui est bien utilisée dans les réseaux traditionnels.

Deux types d'arbre sont envisagés, dans les deux cas, les arbres contiennent un seul chemin entre chaque source et destination:

1. "group-shared": un seul arbre est construit par groupe et les sources partagent cet arbre pour réduire le coût de construction des arbres;

2. "source-based": le protocole construit pour chaque source un arbre dont la tête est la source pour réduire le coût de transmission.

Un des problèmes avec la structure d'arbre est sa fragilité face aux mouvements des nœuds et/ou la dégradation des canaux radio. Le protocole doit réparer l'arbre afin de maintenir les transmissions même si l'une des branches est rompues.

Au regard du surcoût de réparation dans les réseaux de forte mobilité, certains chercheurs proposent un autre type de structure (maillée en treillis ou "mesh") qui contient des routes redondantes entre les sources et les destinations pour plus de robustesse. Par rapport aux arbres o nous pouvons directement utiliser la structure pour livrer les paquets, on doit bien réfléchir à comment réduire le coût de transmission avec la structure maillée. Nous allons discuter ce point dans le quatrième défi.

Le troisième défi concerne la cohabitation des protocoles multicast avec les protocoles unicast. Le protocole de routage multicast peut utiliser un protocole de routage unicast existant pour simplifier le design. En effet le protocole unicast peut être utilisé pour construire la structure de routage multicast ou bien pour assurer la livraison des paquets de données entre les nœuds du réseau. Une autre stratégie consiste à inclure cette fonctionnalité dans le protocole pour mieux matriser les performances et aussi par soucis d'indépendance (la standardisation des protocoles unicast n'étant pas encore établie). Nous appelons le premier choix comme "unicast dépendant" et le dernier comme "indépendant".

Le quatrième défi est lié à comment envoyer les paquets de multicast dans la structure de routage. Les protocoles qui construisent un arbre peuvent envoyer les paquets en respectant la structure pour avoir une transmission contrôlée et à faible coût. Pour les autres protocoles : nous pouvons inonder les paquets dans les structures maillées et offrir une transmission redondante afin d'améliorer le taux de réussite. Face à ces deux grandes stratégies, Il y a aussi des exceptions telles que ADMR qui inonde les paquets dans les arbres pour rendre la transmission plus fiable et MCEDAR qui extrait un arbre de la structure maillée pour réduire le coût de transmission.

La table 8.1 classifie les protocoles courants de routage multicast selon ces quatre critères.

Protocols	Initiate	Structure	(Re)Configuration	Forwarding
ABAM	Traffic	Source Tree	Independent	On tree
ADMR	Traffic	Source Tree	Independent	Flooding
AMRIS	Group	Group Tree	Independent	On tree
AMRoute	Group	Group Tree	Full-dependent	On tree
CAMP	Group	Mesh	Demi-dependent	Flooding
DDM	Traffic	Source Tree	Full-dependent	On tree
FGMP-RA	Group	Mesh	Independent	Flooding
FGMP-SA	Traffic	Mesh	Independent	Flooding
LAM	Group	Group Tree	Demi-dependent	On tree
MCEDAR	Group	Mesh	Independent	On forwarding tree
MAODV	Group	Group Tree	Independent	On tree
NSMP	Traffic	Mesh	Independent	Flooding
MRDC	Traffic	Hybrid Tree	Independent	Adaptive
ODMRP	Traffic	Mesh	Independent	Flooding

Table 8.1: Classification des protocoles de routage multicast

Dans cette table, nous pouvons trouver le protocole ADMR qui construit un arbre pour chaque source afin d'améliorer la transmission dans les réseaux en faible mobilité. ODMRP par contre adopte une structure maillée qui est partagée par toutes les sources d'un groupe pour réduire le surcoût de maintenance dans les réseaux en grande mobilité. Tous les deux inondent les paquets de multicast dans leur structure pour obtenir une redondance. Notre proposition, MRDC, prend un arbre hybride comme la structure de routage et transfère les paquets de manière adaptative pour fournir un routage multicast de type "best effort". Nous présentons MRDC en détail dans les prochaines sections.

8.3 Multicast routing protocol with dynamic core (MRDC)

Nous donnons les caractéristiques principales de MRDC (Multicast Routing protocol with Dynamic Core). Ce protocole essaie de trouver comment optimiser le taux de succès de la livraison de paquets tout en réduisant le coût de signalisation du protocole. Il donne des compromis entre les surcoûts liés au routage et les surcoûts de transmission mais également une optimisation entre le taux de succès de la livraison et les surcoûts en regardant des exigences des applications et les conditions du réseau. Il contient deux plans : le plan de contrôle et le plan de transmission.

Le plan de contrôle construit un arbre initialisé par la demande de trafic sans utiliser un quelconque protocole de routage unicast. La tête de l'arbre est la première

source du groupe. Cette stratégie offre un arbre “source-based” pour supporter les communications de point à multipoint. En cas de multipoint a multipoint, l’arbre sera partagé par les sources de même groupe. L’arbre est donc optimal pour la première source et n’est pas optimal pour les autres sources. Toute fois nous montrons que cette stratégie aboutit à un bon compromis entre les coûts liés à la transmission et les coûts liés aux messages de contrôle. D’ailleur, la racine d’un arbre peut bouger d’une source à une autre pendant la session afin d’adapter aux conditions du réseau. Le plan de transmission (“forwarding”) offre trois méthodes pour transférer les paquets de multicast vers leurs destinations. La méthode choisie s’adapte selon la situation du réseau et l’exigence des applications.

8.3.1 Le plan de contrôle (“Control plane”)

La construction de l’arbre débute quand le premier nœud (source) commence à envoyer les paquets de multicast. Cette source devient le “core” et inonde un message “Core Advertissement” (CA) dans le réseau entier. Lorsqu’un nœud reçoit ce CA message, il crée une table de routage multicast en mettant l’identification du core et un état “inactif”. Il sauvegarde aussi dans sa table de routage unicast l’identification du nœud d’o il reçoit ce message pour établir un chemin de retour (Reverse path) vers le core. Le nœud remet le message CA et ne réagit plus au même CA message venant d’autres nœuds.

Les membres du groupe multicast (récepteurs) doivent accomplir quelques tches supplémentaires. Un membre du groupe envoie un message “Route Active Request” (RAR) quand il reçoit le CA message. Ce message est envoyé par le chemin de retour (Reverse path) vers le core. En conséquence, ce chemin est choisi comme une branche potentielle de l’arbre. Les nœuds en ce chemin stockent dans le champ des nœuds en aval (downstream nodes) l’identification du nœud d’o il reçoit ce RAR message. Le premier membre de l’arbre répond par un message de “Route Active Acknowledge” (RAA) afin d’activer les nœuds en tant que branche potentielle de l’arbre. Ces nœuds activés vont participer à la transmission des paquets de multicast. (Il est à noter que dans la phase de construction initiale de l’arbre, c’est le core qui est le seul membre pouvant envoyer le message RAA) Pour ce faire, chaque nœud met à jour le statut de la table de routage multicast à “actif” quand il reçoit le RAA message. Il aussi sauvegarde dans le champ du nœud en amont l’identification du nœud d’o il reçoit ce RAA message et envoie une copie à chaque nœud dans le champ des nœuds en aval. Ainsi un arbre multicast est construit et la transmission peut commencer.

On voit que le processus de construction est en trois phases la première est initiée par la source, la deuxième est la réponse des membres du groupe et la troisième est la validation de la structure. Ces trois phases assurent que l’arbre construit est valide indépendamment des problèmes éventuels de couverture radio.

Pendant la transmission, la topologie du réseau peut changer à cause du mouvement, de l’apparition ou la disparition des nœuds. Ce changement produit deux types d’impact sur la structure d’arbre. Premièrement, il peut casser un bord

d'arbre et le protocole de routage multicast doit penser à le réparer puisque l'arbre contient des chemins uniques entre le core et les autres membres du groupe. Le changement de la topologie peut d'un autre côté donner un meilleur chemin entre le core et un membre du groupe que celui qui est entrain d'être utilisé. Dans ce cas-là, le protocole de routage multicast doit songer remplacer ce chemin. MRDC emploie deux procédures pour maintenir la validité et l'efficacité des arbres à l'issue du changement de topologie : la procédure de maintenance proactive et la procédure de maintenance réactive.

La procédure de maintenance proactive est aussi appelée régénération d'arbre ("periodical tree refresh"). Il détruit l'arbre de multicast et construit un nouvel arbre en exécutant le processus de la construction d'arbre périodiquement. L'intervalle entre deux régénérations de l'arbre est noté comme la période de régénération d'arbre (PERIOD_REF). Ainsi, les défauts de structure qui ont été accumulés dans l'arbre pendant la dernière période seront éliminés et les meilleurs chemins peuvent être inclus dans l'arbre.

D'autre part, les membres de l'arbre multicast observent en permanent les liaisons avec ses membres en amont et en aval. Dès qu'un membre détecte qu'une liaison est cassée, il lance la procédure de maintenance réactive. Cette procédure essaie de réparer le problème localement et évite de reconfigurer globalement l'arbre. Elle est donc également appelée réparation locale ("local recovery") dans MRDC. Cette procédure procède de la manière suivante: le nœud qui se situe en amont de la liaison cassée diffuse (en broadcast) un message "Join Invitation" (JI) qui est adressé au nœud en aval. La propagation de ce message est limitée à n sauts. Quand la destination reçoit ce message, elle répond par un message "Recovery request" par le chemin découvert par le JI message. Puis, le nœud en amont envoie un message "Recovery reply" pour activer les nœuds sur ce chemin après la réception du message "Recovery request". Ainsi, l'arbre est réparé sans toucher à la transmission en cours dans l'autre partie de l'arbre.

Une grande contribution de MRDC par rapport aux autre protocole qui aussi utilisent la conception "core" est la possibilité de changer la racine passivement ou activement pendant une session de multicast. La structure d'arbre partagé permet de la racine connaître tous les autre sources du groupe dans le réseau. La racine peut donc choisir une des sources pour prendre sa role quand il termine de générer paquets multicast ou pour les raision d'optimisation tels que réduire la congestion de la racine ou utiliser équitablement l'énergie des nœuds. Après avoir reçu la dessignation, la nouvelle racine s'occupera de lancer la procedure de régénération périodique d'arbre d'es la période prochaine.

Nous pouvons calculer le surcoût de contrôle des protocoles MRDC, ODMRP et ADMR pour maintenir la structure de transmission d'un groupe multicast qui a *m* membres et *s* sources dans un réseau de *n* nœuds par les formules ci-dessous :

$$\text{MRDC:} \qquad \frac{n + 2 * (m - 1 + x)}{PERIOD\_REF} + OH_{local\_recovery} \tag{8.1}$$

$$\text{ODMRP:} \qquad \frac{n+(m-1+x)}{PERIOD\_REF} * s \tag{8.2}$$

$$\text{ADMR:} \qquad (\frac{n+(m-1+x)}{PERIOD\_REF} + OH_{local\_recovery}) * s \tag{8.3}$$

Dans les formules, le paramètre *x* est le nombre de nœuds qui ne sont pas membres du groupe (ce sont les membres de la structure). Ce paramètre est dépendant de la topologie de réseau quand les protocoles construisent ou régénèrent la structure.

Ces formules montrent que le surcoût de contrôle de MRDC n'est pas une fonction de nombre de sources dans un groupe parce que l'arbre est partagé par ces sources. Par contre, ceux de deux autres sont affectés par le nombre de sources. Un autre résultat de ces formules est que nous pouvons réduire le surcoût de contrôle en augmentant l'intervalle entre deux reconstructions de la structure. Toutefois cette stratégie a des limites car elle peut conduire à un arbre totalement inutilisable. Un compromis doit être trouvé entre la définition de cette période et la fiabilité de la structure. Nous allons étudier ce compromis (définition d'une valeur de période "optimale") par des simulations.

8.3.2 Le plan de transmission ("forwarding plane")

Le plan de forwarding répond à la question comment nous pouvons envoyer efficacement les paquets de multicast dans les arbres. La méthode la plus simple est de passer les paquets de multicast directement à la couche MAC en supposant qu'il peut transférer ces paquets efficacement. Néanmoins, la question n'est pas simple dans les réseaux Ad Hoc sans fil puisque les protocoles MAC utilisés tel que IEEE 802.11 utilisent le mécanisme similaire à CSMA/CA pour les paquets de multicast. Ce mécanisme attend que le medium soit libre pendant une certaine période et puis envoie le paquet sans besoin de confirmation de l 'autre côté. Il a un problème bien connu: le problème de terminal caché ("the hidden terminal problem"). Les collisions de paquet peuvent se produire pendant les transmissions de différents paquets qui sont normalement la conséquence d'une forte charge du réseau, mais aussi se produire lors de la livraison d'un même paquet vers différents récepteurs. La Figure 8.1 donne un exemple. Dans cet exemple, les nœuds B et C ne s'entendent pas l'un et l'autre. Quand nœud C envoie un paquet généré par la source S vers nœud E afin de livrer le paquet au récepteur R2, le nœud B peut facilement croire que le médium est libre et commence à envoyer le paquet au nœud D pour le recepteur R3. En conséquence, les copies du même paquet entrent en collision au nœud E et ce nœud n'arrive pas à recevoir correctement le paquet vient du nœud C. Ce type de collision entraine une perte significative de livraison des paquets de multicast même dans les réseaux stables avec trafic faible.

Pour éviter ce genre de problème durant la livraison des paquets de multicast, nous pouvons profiter de mécanismes proposés pour transférer les paquets unicast. Par exemple, IEEE 802.11 adopte un processus de transmission en quatre phases (RTS/CTS/DATA/ACK) avec un mécanisme d'accord (RTS/CTS) pour résoudre

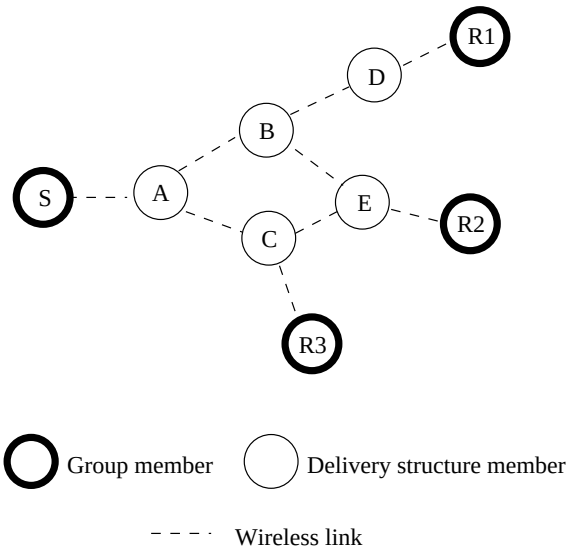


Figure 8.1: La transmission de paquet multicast dans un réseau ad hoc d'IEEE 802.11

le problème de terminal caché quand il envoie un paquet d'unicast. Au niveau de routage, MRDC peut donc dupliquer un paquet de multicast, envoyer aux membres de l'arbre en amont et en aval et demander au MAC IEEE802.11 leurs transferts avec l'option de RTS/CTS. Nous appelons ce mécanisme comme mode unicast ("unicast mode") et celui qui envoie les paquets avec CSMA/CA en mode diffusion ("broadcast mode") dans MRDC. Par rapport au mode broadcast, le mode d'unicast améliore le taux de réussite de livraison de paquets mais en utilisant plus de messages et avec plus de délai.

Au regard des avantages et inconvénients des deux modes, MRDC dispose un troisième mode, le mode adaptatif. Ce mode choisit un mode de transmission selon la charge de medium. Il utilise le mode broadcast si le medium est chargé et le mode unicast en cas contraire. Les nœuds calculent deux métriques: la longueur moyenne de file d'attente (Average queue length (AQL)) et le taux d'occupation du medium (Medium occupation rate (MOR)), pour juger du niveau de charge de médium.

AQL est le nombre moyen des paquets qui attendent d'être envoyés par la couche MAC. C'est une métrique qui est utilisée d'une manière standard pour contrôler les congestions dans les réseaux (ce qui est conforme à notre but de contrôler les congestions créées par le mode unicast). AQL n'est toutefois pas une métrique suffisante car Il réagit lentement face à l'augmentation du trafic. Heureusement, certaines implémentations de IEEE 802.11 offrent l'accès à des compteurs tels que le nombre de paquets ou byte reçus par un nœud. Ces compteurs permettent de calculer la métrique MOR qui est définie comme le nombre de bytes qu'un nœud reçoit pendant la dernière période. Surveillant l'évolution

de cette métrique, les nœuds peuvent détecter rapidement l'augmentation de trafic dans le medium et changer au mode de broadcast rapidement. Si l'une de ces deux métriques dépasse son seuil, le nœud croit que le médium est chargé et MRDC doit utiliser mode de broadcast pour réduire le surcoût de transmission. Dans un autre cas, le mode adaptatif de MRDC prend le mode d'unicast afin d'améliorer le taux de réussite. Nous allons choisir les seuils optimaux par des simulations.

Le surcoût de livraison d'un paquet multicast à ses destinations avec le mode broadcast dépend du nombre de nœuds interne de l'arbre. Celui avec le mode unicast dépend du nombre de branches dans l'arbre c'est-à-dire le nombre total de nœuds moins un. Celui du mode adaptif est compris entre ces deux nombres. Nous pouvons calculer le surcoût de ces trois modes en utilisant le résultat des simulations.

8.4 L'analyse des performances de MRDC

Pour analyser le comportement de notre protocole "multicast routing protocol with dynamic core" (MRDC), nous avons construit une ensemble de simulations basées sur le simulateur de réseau ns2 [9]. Ces simulations ont trois buts:

- obtenir les paramètres optimaux de MRDC tels que la période de régénération d'arbre (PERIOD_REF) et les seuils de la sélection de mode;
- obtenir les paramètres d'arbre dans les différentes conditions pour estimer les surcoûts et la performance et
- comparer la performance de MRDC avec les deux autres protocoles (ODMRP et ADMR) sous différents scénarios de trafic et mouvement.

Les expérimentations simulent les comportements d'un réseau qui contient 50 nœuds pendant 900 secondes de temps de simulation. Le protocole de la couche MAC est IEEE 802.11. Plusieurs scénarios de mouvement et trafic sont créés. Les scénarios de mouvement définissent les positions des nœuds et leur mouvement pendant la simulation. Les nœuds bougent selon l'algorithme "random waypoint" avec une vitesse choisie aléatoirement entre zéro et une valeur maximum sans pause. Parce que les résultats de simulations sont sensibles aux scénarios de mouvement, nous avons généré dix différents scénarios de mouvement pour une vitesse maximum. Chaque résultat est donc la valeur moyenne de ces scénarios. Nous donnons en annexe de la thèse les courbes qui représentent les intervalles de confiance pour toutes les valeurs obtenues par simulation.

Les scénarios de trafic définissent les groupes de multicast et les trafics. Certains nœuds sont choisis en tant que participants d'un groupe. Par défaut, tous les membres sont les récepteurs de ce groupe. Les sources CBR (constant bit rate) envoient 4 paquets par seconde au groupe et chaque paquet contient 512 bytes de trafic. Le nombre de groupes est à la moitié du nombre de sources. En

conséquence, le nombre de groupes augmente quand le nombre de sources augmente.

8.4.1 Sélection des paramètres clés de MRDC

Nous choisissons d'abord la période de régénération d'arbre (PERIOD_REF) en testant deux métriques, le taux de réussite de livraison et le surcoût de contrôle sous différents scénarios de mouvement. C'est à dire un groupe qui contient 10 membres et 2 sources est considéré et en même temps le scénario de trafic fait varier la vitesse maximum de 0 m/s à 20 m/s. Nous observons qu'il y a un compromis entre le taux de réussite et le surcoût de contrôle quand la période de régénération d'arbre augmente. Le taux de réussite est le rapport du nombre de paquets de multicast correctement livrés aux récepteurs divisé par le nombre de paquets de multicast supposés être reçus. Le surcoût de contrôle est représenté par le taux de messages de contrôle transmis par MRDC par seconde. Quand la période est infinie, les arbres ne seront pas régénérés une fois qu'ils sont établis. Le résultat est que MRDC pour une période infinie génère moins de surcoût de contrôle mais le taux de réussite descend plus vite que pour les autres valeurs de période quand le mouvement des nœuds augmente. Les simulations démontrent que la période de 5 seconds donne le meilleur compromis entre ces deux métriques. Elle délivre plus de 95 pourcents de paquets multicast aux récepteurs avec moins 5% de surcoût de contrôle. Cette valeur de 5 secondes pour le paramètre PERIOD_REF est utilisée dans les prochaines simulations.

Ensuite, nous lançons des simulations pour connaître la valeur optimale des seuils de sélection de mode de transmission. Nous varions les seuils et observons les résultats de deux métriques avec un scénario de trafic qui définit 3 groupes, 10 membres et 2 sources par groupe et les scenarii de mouvement dont la vitesse maximale est fixée à 5 m/s. Les deux métriques sont le taux succès de livraison et le délai moyen de transmission de bout en bout. Les résultats montrent un maximum de taux de réussite à 93% quand INTR, le seuil de MOR, est égal à 0.7 et QLEN, le seuil de AQL, est égal à 16. Cependant, le délai de transmission augmente de façon spectaculaire à presque 0.05s. Par contre une autre combinaison des seuils (INTR=0.6 et QLEN=8) paraît optimale, elle donne 92.5% de taux de réussite et 0.035s de délai. Ces seuils sont utilisés dans la comparaison des protocoles pour le mode adaptif de MRDC.

8.4.2 La performance liée à l'arbre de MRDC

Le but des ces simulations est de tester l'efficacité et la robustesse de l'arbre de MRDC mais aussi connaître les paramètres d'arbre de multicast construit par MRDC sous différentes conditions de mouvement et trafic. Ces paramètres nous permettront d'estimer les surcoûts de contrôle et forwarding. Ils peuvent aussi donner la performance de protocole dans des conditions idéales où les protocoles sous-jacents peuvent envoyer les paquets de multicast efficacement. Les quatre

métriques suivantes sont calculées dans cette section.

Le nombre moyen de routeurs multicast Les routeurs multicast sont des nœuds qui transmettent des paquets de multicast. Ils sont aussi des nœuds internes d'un arbre qui ont des nœuds descendants. Cette métrique compte le nombre moyen de nœuds dans un arbre de multicast pendant une simulation. Il est le surcoût de transmission quand MRDC délivre un paquet en mode broadcast.

Le nombre moyen de routeusr multicast non-membres Les routeurs multicast non-membre sont des routeurs multicast et en même temps ils ne sont pas membres du groupe pour le quel ils servent de nœuds intermédiaires. Cette métrique est le paramètre x de la formule de surcoût de contrôle et de la formule de surcoût de transmission en mode unicast.

Le nombre de réparations d'arbre Cette métrique compte le nombre de réparations locales lancées par MRDC pour réparer un arbre. Il nous permet d'évaluer la variation des surcoûts de contrôle de MRDC dans différents scénarios.

La durée de fragmentation d'arbre Un membre peut devenir inaccessible vis-à-vis des autres membres à cause de la partition du réseau. Pour distinguer ce cas avec le défaut de MRDC, nous définissons qu'un arbre est appelé fragmenté s'il ne contient pas tous les membres du groupe dans une même partition du réseau. Cette métrique est utilisée pour calculer le taux de réussite dans le cas idéal.

Un groupe de multicast est simulé dans ces expérimentations. Après que les sources commencent leur transmission, le simulateur calcule à chaque demi seconde le nombre total de nœuds de l'arbre et de nœuds intérieurs d'arbre et si l'arbre couvre tous les membres accessibles du groupe. Il y a au total 1696 de tels rapports de mesure pendant une simulation. MRDC de son côté compte combien fois les réparations sont lancées pendant cette simulation.

Nous faisons d'abord varier la vitesse maximum de mouvement de 0 m/s à 20 m/s pour examiner la robustesse du protocole contre des changements de topologie. Un groupe de multicast contenant 20 membres est simulé. La charge de réseau est très légère (1 source) pour exclure autant que possible l'influence des paquets du trafic sur la transmission de message de contrôle. Les résultats sont donnés dans la table 8.2.

Les résultats prouvent que l'arbre de multicast de MRDC est sensible au changement de topologie. Quand les nœuds bougent plus vite, MRDC lance plus fréquemment la réparation locale et le pourcentage de fragmentations d'arbre augmente aussi de son côté. En conséquence, le surcoût de contrôle va augmenter et le taux de réussite va baisser. Les résultats aussi montrent que le nombre de routeurs diminue avec l'augmentation de la vitesse. Le nombre de routeurs non-membres suit la même tendance. Cela signifie que le surcoût de contrôle lié à la régénération d'arbre et le surcoût de livraison diminuent vis-à-vis de l'accroissement de la vitesse des nœuds.

Maximum speed (m/s)	Average non-member routers	Average routers	# of tree repair times	Tree broken percentage
0	8.24	14.81	0	0.5%
1	7.23	12.61	51	0.5%
2	7.05	12.51	91	1.1%
5	7.1	12.72	205	2.6%
10	7.1	13.06	344	4.4%
15	6.48	12.44	466	5.5%
20	6.64	12.84	596	7.2%

Table 8.2: Performance of MRDC multicast tree as a function of Maximum mobility speed

Le nombre total de paquets à transmettre diminue donc avec l'augmentation de la vitesse des nœuds. Cette propriété est un facteur important quand IEEE 802.11 est utilisé comme couche MAC de transmission. Avec la réduction de nombre de paquets transmis pendant la livraison, il y aura moins de collisions à la couche MAC. Cela pourra réduire le taux de pertes dans les scénarios de forte mobilité.

Nous avons aussi simulé la performance d'arbre de MRDC en variant le nombre de sources et le nombre de membres. Les résultats ont montré que l'arbre de MRDC est scalable face à l'élargissement du groupe. Par exemple, moins de 17 transmissions est suffisant pour délivrer un paquet de multicast à 39 récepteurs.

Cependant, MRDC reste fortement sensible à la croissance de trafic dans le réseau. C'est la conséquence de la stratégie de régénération d'arbre utilisée par MRDC. Pendant la régénération d'arbre les nœuds réinitialisent leur table de routage de multcast quand ils reçoivent un nouveau message CA et l'arbre est correctement reconstruit après que tous les membres ont reçu le message RAA. Cette stratégie réduit le surcoût de transmission mais d'un autre côté accroît la robustesse de l'arbre dépend fortement de la bonne transmission des messages de contrôle. Nous devons étudier si cette stratégie donne un bon compromis entre le surcoût de transmission et la dépendance de transmission des paquets de contrôle.

A partir de ces résultats, nous pouvons aussi comparer le surcoût de deux modes de transmission. Quand le nombre de membres est égal à 10, le mode unicast a besoin de (10+6.9-1)/9=1.76 transmissions pour délivrer un paquet à un récepteur, mais le mode broadcast a besoin seulement de 9.33/9=1.03 transmissions pour le même but. Nous allons comparer ces chiffres obtenus par calcul pour MRDC avec les résultats obtenus par simulation dans le paragraphe suivant.

8.4.3 La comparaison des protocoles de routage de multicast

Dans cette section, nous comparons la performance de MRDC avec deux autres protocoles de routage de multicast ODMRP - un protocole de routage de multicast basé sur une structure maillée, et ADMR - un protocole de routage de multi-

cast basé en arbre dont la tête est source. Nous testons également les trois modes de transmission de MRDC dans ces expériences. Ces modes sont notés comme MRDC-broadcast, MRDC-unicast et MRDC-adaptive dans les figures et discussions suivantes. Quatre métriques sont utilisées : le taux de réussite, le délai de transmission bout en bout, le surcoût de contrôle et du surcoût transmission.

Nous analysons d'abord la performance de ces protocoles dans les différents scénarios de mobilité. Un scénario de trafic qui définit deux groupes de multicast et 10 membres, 2 sources par groupe est choisi. La vitesse maximale de nœud varie de 0 m/s à 20 m/s. Les résultats montrent que MRDC donne le meilleur taux de réussite de livraison par rapport aux autres deux protocoles. Les résultats de la section d'analyse d'arbre ont montré que la fragmentation d'arbre augmente quand la vitesse de nœuds s'accroît. Il peut atteindre 7% dans les réseaux à forte mobilité. Cependant, ces fragmentations touchent seulement une partie du groupe et n'empêchent pas MRDC de livrer les paquets aux autres membres. En réduisant le problème des terminaux cachés au maximum, MRDC-unicast peut maintenir son taux de réussite à 96%. Nous remarquons que l'écart de délai de transmission entre les différents modes de transmission de MRDC est grand. MRDC-broadcast, avec 25ms en moyenne, est le meilleur protocole en termes de délai de transmission mais son taux de réussite est environ 2 points (6 points dans les réseaux stables) moindre que celui de MRDC-unicast et MRDC-adaptive. MRDC-unicast et MRDC-adaptive en revanche créent plus de 50ms de délai ce qui les rend les plus mauvais de ce point de vue. Les applications doivent donc bien choisir le mode de transmission selon leurs exigences.

Quant au surcoût de protocole, MRDC génère moins de surcoût de contrôle malgré sa structure d'arbre. Autrement dit, le coût de la réparation d'arbre de MRDC dans les réseaux à forte mobilité est beaucoup moins élevé que celui de la création des routes redondantes d'ODMRP. Sans surprise, ADMR, un protocole de type "source-base tree", génère le plus de surcoût bien que sa période de régénération d'arbre est beaucoup plus grande que celle de MRDC (25seconde pour ADMR par rapport à 5 seconde de MRDC). MRDC-broadcast génère le moins de surcoût de transmission parmi tous les protocoles, y compris ADMR. En théorie, les arbres basés sur les sources sont plus efficaces que les arbres partagés parce que dans ces derniers, les paquets venant de diverses sources doivent passer par la racine. Néanmoins, la stratégie de construction et maintenance d'arbre d'ADMR limite cet avantage. Il préfère une longue période de régénération d'arbre et complète cette action de maintenance de l'arbre avec la procédure de réparation locale et les chemins redondants pour réduire le coût global. Cette stratégie a trois mauvais impacts sur la performance de surcoût de transmission. D'abord, les arbres contiennent plus qu'un chemin vers les récepteurs à cause des chemins redondants. Deuxièmement, les arbres ne peuvent pas remplacer les chemins utilisés avec les plus courts chemins assez vite après le changement de topologie. Troisièmement, les nouveaux chemins trouvés par la procédure de réparation locale ne sont pas le plus court dans les réseaux et peuvent rester longtemps dans les arbres. En conséquence, le surcoût de transmission d'ADMR est pire que celle de MRDC.

Ces simulations prouvent que MRDC est le protocole le plus efficace. Il donne le meilleur taux de réussite et le meilleur surcoût de contrôle. MRDC-broadcast génère le moins de délais de transmission et le moindre surcoût de transmission. Cependant, les différents taux de réussite et de délai de transmission remarquent que MRDC doit bien choisir un mode de transmission convenable afin de mieux servir les applications.

Par la suite, nous étudions la performance de ces protocoles avec différents scénarios de trafic. La vitesse maximale de nœuds est 5 m/s et chaque groupe contient 10 membres. Nous augmentons le nombre de sources de 2 à 8 en même temps le nombre de groupes de 1 à 4 en suivant la règle que le nombre de groupes est la moitié du nombre de sources. Ce choix donne un équilibre entre les protocoles basés sur les sources et les protocoles en groupe partagé.

Les résultats montrent que MRDC-adaptive est le meilleur choix quand la charge des réseaux varie. Ce protocole donne presque toujours le meilleur taux de réussite dans les simulations. MRDC-unicast surpasse MRDC-adaptive dans les réseaux peu chargé (c-a-d le nombre de sources est inférieur à 4) en termes de taux de réussite. MRDC-adaptive donne aussi un délai optimal pendant la croissance de trafic dans les réseaux en le maintenant à moins de 40 ms. Quant au surcoût de protocole, les résultats montrent une fois de plus que MRDC est le meilleur protocole au terme de surcoût de contrôle et MRDC-broadcast génère le moins de surcoût de transmission.

8.4.4 Conclusion sur les protocole de routage de multicast pour réseaux Ad Hoc

Dans cette section, nous avons discuté les caractéristiques des protocoles de routage multicast pour les réseaux ad hoc sans fils. C'est un sujet bien étudié par de nombreux chercheurs qui ont exploré plusieurs techniques et stratégies. Le point faible commun de ces travaux est la limitation de l'utilisation de ces protocoles à certaines conditions de réseau bien précises (en termes de charges et de mobilité). Le résultat est un manque de flexibilité vis-à-vis des applications qui voient leurs performances se dégrader si les conditions de fonctionnement du réseau ne sont pas remplies. Pour résoudre ce problème, nous proposons un nouveau protocole de routage de multicast (MRDC) qui s'adapte aux changements imprévisibles des réseaux ad hoc sans fil.

Conformément à nos objectifs de recherche, les résultats de simulations ont montré que MRDC avec ses trois modes de transmission a la meilleure performance par rapport à ODMRP et ADMR et donc peut mieux servir les applications dans les situations diverses de réseaux. La table 8.4.4 donne un exemple de sélection de mode de transmission selon la charge du réseau et la demande de l'application.

Networks / Applications	Low load	High load
Loss sensitive	Unicast	Adaptive
Delay sensitive	Unicast/Broadcast	Broadcast
Others	Adaptive	Adaptive

8.5 Protocole fiable de multicast

Les protocoles de routages multicast envisagés jusqu'à présent (y compris MRDC) ne peuvent pas éviter la perte de paquets. Ils offrent un service de type "best effort". Certaines applications exigent cependant la livraison fiable de paquets (cent pour cent de réussite). Cette fiabilité peut être obtenue par des protocoles de plus haut niveau tels que protocoles de transport ou protocoles applicatifs. Ces protocoles sont efficaces pour les réseaux filaires mais ne présentent pas des caractéristiques suffisantes pour s'adapter aux réseaux sans fils. Nous envisageons donc d'étudier des mécanismes de protocoles multicast au niveau réseau pour donner plus de performances à la livraison fiable de paquets dans les liens point à multipoint (ou multipoint à multipoint) des réseaux Ad Hoc. Le chapitre 5 présente nos recherches en protocole fiable de multicast pour supporter ce genre d'application.

Le mécanisme "automatic repeat request" (ARQ) est utilisé couramment dans les réseaux Internet et sans fils. Ce mécanisme fait que la source ou tout autre type de nœud retransmette les paquets perdus afin de rendre la livraison fiable. Selon les différents types de nœud intervenant dans le processus de retransmission, ce mécanisme est classifié dans les trois schémas suivants.

Sender-originated Il n'y a que la source qui retransmet les paquets perdus.

Receiver-assisted Les récepteurs aident à la retransmission

Router-assisted Les routeurs intervient dans la retransmission.

Les efforts actuels de recherche ont essayé d'adapter ces trois schémas de retransmission à l'environnement des réseaux ad hoc sans fil. Nous pouvons citer par exemple le protocole "Adaptive Protocol for Reliable Multicast in MANET" (APRM) [76] qui utilise le schéma de sender-originated; " Anonymous Gossip"[82] qui est basé sur le schéma de receiver-assisted et "Family ACK Tree" (FAT)[77] qui adopte le schéma de router-assisted.

Les recherches ont montré que APRM et FAT ont eu mauvais compromis entre fiabilité et passage à l'échelle dans les réseaux fortement dynamiques. "Anonymous Gossip" donne un meilleur compromis mais ne peut pas garantir cent pour cent de livraison. Au regard de ces limitations, nous étudions comment offrir efficacement ce genre de service dans les réseaux ad hoc sans fil où la topologie peut changer imprévisiblement, le taux de perte de paquet est important et les ressources sont imitées.

Nos recherches commencent par une analyse de la charge de retransmission afin de bien choisir un schéma d'ARQ adapté, ensuite nous proposons un nouveau protocole fiable de multicast et enfin nous analysons sa performance au moyen de simulations réseau.

8.5.1 L'analyse de la charge de retransmission des sources des schémas d'ARQ

Nous analysions la charge de retransmission des sources des trois schémas de retransmission dans un arbre binaire de deux niveaux. La figure illustre comment la charge de retransmission de ces trois schémas change avec la probabilité de perte. Evidemment la charge de retransmission du schéma sender-originated devient bientôt inacceptable dans l'arbre binaire à deux niveaux. Même pour de petites probabilités de perte, la charge de retransmission est plutôt haute. L'amélioration réalisée par l'aide des récepteurs est seulement marginale. Par rapport à ceci, le schéma de router-assisted donne les meilleures performances. Ce schéma mène une charge acceptable même pour les probabilités plus hautes que 50% de pertes. C'est capital pour les réseaux ad hoc sans fil où les pertes de paquets sont importantes mais les ressources de nœuds sont limitées

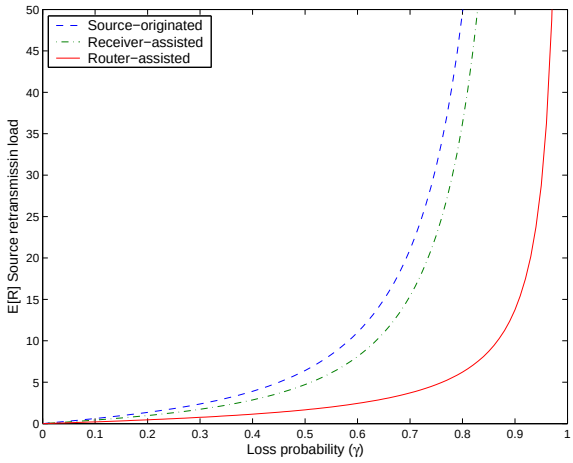


Figure 8.2: Retransmission load when varying γ

8.5.2 Active reliable multicast protocol with intermediate node support (ARMPIS)

Le schéma de retransmission router-assisted est le meilleur choix pour les réseaux ad hoc sans fil. Ce schéma demande aux routeurs de multicast de conserver temporairement les paquets qu'ils envoient afin de répondre aux demandes de retransmission en cas d'échec de livraison. Néanmoins, certaines caractéristiques de réseaux ad hoc sans fil nous empêche de réaliser ce mécanisme directement dans

ces derniers. D'abord, les nœuds ont une capacité de mémoire limitée qui ne permet pas de stocker un grand nombre de paquets. Deuxièmement, un nœud peut ne plus être routeur après le changement de topologie. Les paquets stockés par ces nœuds ainsi que les connaissances des nœuds en aval de la transmission seront perdues.

FAT utilise un arbre de famille ("family tree" afin de résourdre le problème de mobilité de nœuds. Ce protocole transforme l'arbre de multicast en un arbre de famille où les nœuds connaissent leur père, leurs grands-pères et leurs enfants. Si un nœud n'est plus le routeur, il envoie tous les paquets qui ne sont pas encore stabilisés à son père. Et les nœuds qui ont perdu leur père doivent établir les chemins vers leurs grands-pères afin d'obtenir les paquets retransmis. Ce protocole a un surcoût important dans les réseaux à forte mobilité car il doit exécuter fréquemment la maintenance de l'arbre de famille et les transmissions de paquets non-stabilisés.

Nous proposons un protocole fiable de multicast, Active Reliable Multicast Protocol with Intermediate node support (ARMPIS), qui active des noeuds intermédiaires pour aider les retransmissions pour réduire le coût d'assurer la livraison fiable même dans les réseaux à forte mobilité. Dans ce protocole, les nœuds stockent les paquets entendus d'une façon probabiliste. Quand un routeur reçoit une demande de retransmission, il cherche premièrement le paquet demandé dans son cache. S'il ne le trouve pas, il envoie une requête pour interroger ses voisins. Si aucun voisin n'a le paquet demandé, le routeur transfère la demande au prochain saut en direction de la source. Cette procédure est nommée récupération locale ("local recovery"). Ainsi, ARMPIS distribue la charge de stockage et retransmission de routeur de multicast aux voisins en profitant du mode broadcast de transmission des paquets.

La figure 8.3 donne un exemple de fonctionnement d'ARMPIS. Dans cet exemple, un arbre de multicast est construit pour diffuser les paquets du nœud 0 aux nœuds 1, 2, 3 et 4. Nous supposons que le nœud 1 échoue à recevoir correctement un paquet du nœud 28 et il essaie de récupérer ce paquet. Ce nœud d'abord émet (en broadcast) une requête à ces voisins et leur demande de lui envoyer le paquet (s'ils ont ce paquet dans leur mémoire). Nous pouvons en conséquence trouver trois nœuds candidats pour la retransmission. Ce sont le nœud 4 qui est un récepteur du groupe, le nœud 28 et aussi le nœud 12 qui est le voisin des nœuds 1 et 28 et qui pourrait entendre le paquet et sauvegarder le paquet dans son cache par certaine probabilité.

Maintenant nous allons voir un autre exemple d'échec de transmission entre le nœud 11 et le nœud 2. Malheureusement, quand le nœud 2 découvre cette perte, l'arbre a déjà été modifié à la suite d'un changement de la topologie. Le nœud 11 n'est plus un routeur de multicast et le nœud 27 n'est plus un voisin d'arbre de transmission de multicast Néanmoins, en exécutant la procédure de récupération locale, ce récepteur peut toujours espérer que les nœuds 11ou 27 lui transfère le paquet de leur cache. De ce fait, les paquets stockés par des anciens routeurs peuvent toujours servir les retransmissions même après avoir pris le rôle de nœuds

simples.

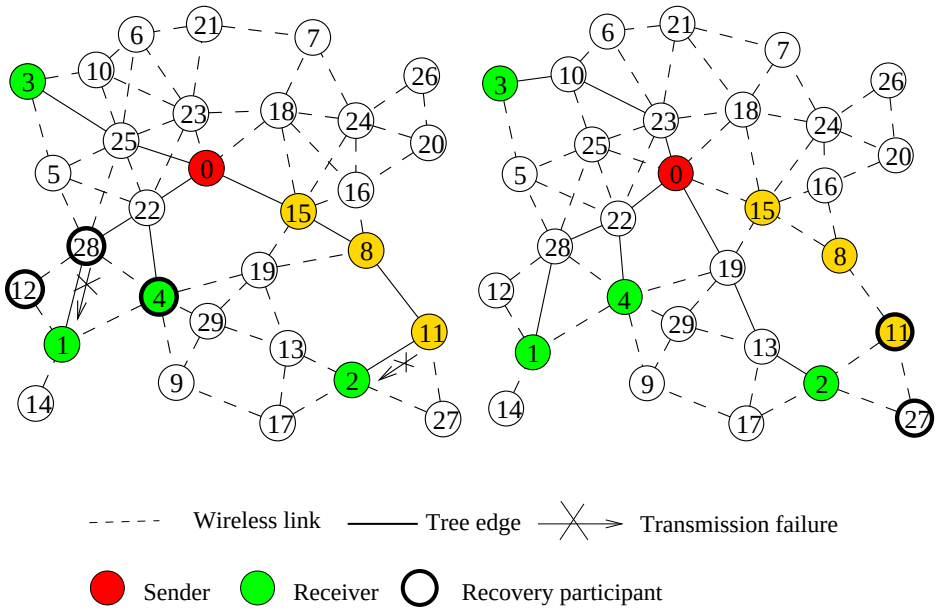


Figure 8.3: ARMPIS retransmisison

8.5.3 L'analyse de performance de ARMPIS

Nous évaluons la performance de notre protocole fiable de multicast en termes de garantie de livraison et consommation de bande de passante par simulation avec ns2. L'environnement de simulation est identique à celui de la simulation de protocole de routage de multicast. Une petite modification est menée sur les scénarios de trafic. Au lieu d'envoyer les paquets jusqu'à fin de simulation, les sources envoient au total 3200 paquets et le reste du temps permettra que le protocole fiable de multicast finisse ses retransmissions. Une espace de mémoire est alloué dans chaque nœud pour permettre le protocole fiable de multicast à stocker 64 packets de 512 bytes. Nous avons fait des simulations pour connaître la probabilité optimale de cache et choisissons finalement 0.3 parce que cette valeur donne le meilleur compromis entre taux de réussite et le surcoût de retransmission. Pour des fins de comparaison, nous avons développé un protocole fiable de multicast basé sur le schéma retransmission de sender-originated qui ressemble au protocole APRM. Deux métriques sont employées dans cette analyse:

Taux de réussite de livraison Donner cent pour cent de taux de réussite de livraison est le but de protocole fiable. Mais dans l'environnement des réseaux ad hoc sans fil, il est impossible de toujours tenir cette promesse dans tous les

scénarios. Nous étudions donc quel protocole peut donner un résultat le plus proche de 100%.

Le surcoût de retransmission des sources Cette métrique compte le nombre de paquets retransmis par la source pendant une simulation. Il montre l'efficacité du protocole fiable de multicast.

Nous d'abord choisissons un scénario de 4 sources (4packets/s) et 10 membres comme scénario de trafic et nous varions la vitesse maximale de 0 à 20 m/s pour observer la performance des protocoles. Les résultats montrent que ARMPIS est fiable en donnant un presque 100% de taux de réussite de livraison. Il est aussi efficace par rapport à APRM puisqu'il génère trois fois moins de charge de retransmission. Ensuite, nous changeons le nombre de sources de 2 à 8 et fixons la vitesse maximale de nœuds à 5 m/s pour voir le comportement des protocoles fiables quand la charge de réseau augmente. Le nombre de membres par groupe est toujours 10. Ces simulations démontrent que ARMPIS peut bien maintenir son taux de réussite de livraison proche de 100% même dans les scénarios où celui d'APRM se dégrade au-dessous de 99%. Le surcoût de retransmission d'ARMPIS dans ces simulations est toujours trois fois moindre que dans l'APRM.

8.5.4 Conclusion sur les protocoles fiables de multicast

Cette section pressente nos recherches en protocole fiable de multicast pour les réseaux ad hoc sans fil notamment comment nous pouvons utiliser le mécanisme de ARQ efficacement. Selon les différents types de nœuds qui sont impliqués dans les retransmissions, nous pouvons distinguer trois catégories de retransmissions, "sender-originated"," receiver-assisted" et "router-assisted". Nos analyses ont montré que le schéma "router-assisted" est le meilleur choix pour les réseaux ad hoc sans fil où le taux de perte est significatif et les ressources tels que bande passante et énergie sont limitées.

Notre proposition de protocole fiable de multicast: ARMPIS est une version adaptive du schéma "router-assisted" pour les réseaux ad hoc sans fil. Il distribue les charges de stockage et retransmission des paquets non seulement aux routeurs de multicast mais aussi leurs voisins. Les nœuds sauvegardent les paquets multicast qu'ils entendent d'une manière probabiliste. Les récepteurs ou routeurs de multicast cherchent d'abord les paquets perdus dans leur voisinage avant d'envoyer la requête vers la source. Ainsi ce protocole résout deux problèmes lorsqu'on impose le schéma "router-assisted" dans les réseaux ad hoc sans fil : la limitation de la capacité de mémoire et le changement fréquent et imprévisible de rôle des nœuds. Les résultats de simulations ont montré que ce protocole est fiable avec presque 100% de taux de réussite de livraison même dans les réseaux à forte mobilité et forte charge. Il est aussi efficace par rapport à APRM, un protocole de type "sender-originated", en générant 3 fois moins du surcoût de retransmission de source.

8.6 Banc de test de réseau ad hoc

Cette dissertation inclut également nos travaux de la construction et de la validation d'un banc de test de réseau ad hoc. L'analyse de la performance par les outils de simulateur est un choix idéal pendant la période initiale de la conception d'un algorithme. Cependant, les simulateurs ont beaucoup de limitations telles que le manque de simulation réaliste du comportement de la couche physique. Ces limitations rendent essentiel les tests en conditions réelles au moyen d'un banc de test. Ce banc de test nous permettra d'analyser les performances des protocoles dans un vrai réseau. Il peut également être employé pour étudier des protocoles et de nouvelles applications de réseaux ad hoc sans fil.

Nous avons implémenté Distributed Dynamic Routing (DDR) [89], un protocole de routage unicast et MRDC de sorte que le banc de test puisse supporter des communications en point à point et aussi des communications multipoint.

8.6.1 Implémentation et validation d'un protocole de routage d'unicast, DDR

Introduction de DDR

DDR est un algorithme de structuration ("clustering") distribué avec des critères déterministes, qui traite le problème de la gestion de topologie dans les réseaux ad-hoc sans fil. L'idée principale de cet algorithme est que les nœuds sauvegardent l'information de leurs voisins dans la table de voisinage (Neighboorhood table). Ensuite, ils choissent un voisin, appelé le voisin préféré (Preferered Node), qui a un degré maximum de connectivité dans le voisinage (c.-à-d. le degré est le critère de l'algorithme d'élection du voisin préféré) et établissent une liaison logique avec lui. Ceci est fait en utilisant simplement un processus d'envoi périodique de messages balises ("beaconning"). Il est montré qu'en reliant chaque nœud à son voisin préféré en respectant la topologie du réseau conduit toujours une forêt [89]. Dans cet algorithme, chaque arbre de la forêt forme une zone, et chaque zone est maintenue proactivement. Des zones sont reliées entre elles par l'intermédiaire des nœuds qui ne sont pas dans la même zone mais sont dans la couverture de transmission directe de l'un et l'autre. Par conséquent, le réseau est divisé en un ensemble de zones non-recouvertes. En effet, cet algorithme combine deux notions: forêt et zone. La forêt réduit le surcoût de transmission des messages en choisissant un sous-ensemble de l'ensemble de noeuds voisins pour transférer les messages, et des zones sont employées pour réduire les délais provoqués par le processus de routage et pour attendre un meilleur passage à l'échelle. Les nœuds sauvegardent les liaisons logiques qui appartiennent à la même zone dans leur table d'intra zone (intra-ZT) et dans leur inter-zone table (inter-ZT). Ces tables sont aussi mises à jour par l'information portée dans les beacons. Les nœuds peuvent utiliser l'information enregistrée dans leur table d'intra-ZT pour acheminer les paquets dont la source et la destination sont dans la même zone. Quant aux paquets envoyés vers les

nœuds en dehors de la zone, le protocole de routage doit découvrir les chemins d'ne manière réactive. C'est pour cela DDR est aussi classifié comme un protocole hybride.

Implémentation de DDR

La figure 8.4 montre l'architecture d'implémentation de DDR. Cette architecture profite du support de transmission des paquets IPv4 qui est développé dans le noyau du système d'exploitation Linux afin de mettre en application le protocole de routage. Un module de transmission d'IP fournit l'opération essentielle d'acheminer les paquets vers la couche réseau. Ce module analyse l'en-tête du paquet IP (l'adresse de destination, TTL et parfois l'adresse de source), et puis envoie le paquet à l'interface correspondante ou rejette le paquet selon la table de routage (IP forwarding table). Linux nous permet de modifier la table de routage dans le noyau par un socket de type netlink. En conséquence, le protocole de routage peut être implémenté comme un "démon" dans l'espace d'utilisateur sans besoin de modifier le noyau. Cette version d'implémentation ouvre deux interfaces pour la communication. L'interface I1 est une socket de type UDP pour l'envoi et la réception des beacons et l'interface I2 est employée pour modifier la table de routage d'IP (IP forwarding table) par une socket de type netlink. En se basant sur l'information fournie par les beacons, DDR établit la table de voisinage et choisit le nœud préféré (Preferred Node). Puis, DDR construit la table d'intra-zone (intra ZT) et la table d'inter-zone (ZT inter) pour définir l'arbre de recouvrement ("spanning tree"). Dans cette version d'implémentation, DDR possède sa propre table de routage. L'information de routage stocké dans cette table prévient directement de la table de sa table d'intra-zone et du processus de découverte réactive dans l'étape prochaine d'implémentation. Selon sa table de routage, DDR met à jour la table de routage d'IP dans le noyau de Linux. De cette façon nous avons mis en application le cheminement proactif de paquet d'unicast en employant la table d'intra-zone de DDR.

Validation de DDR

Nous validons l'implémentation de DDR d'abord dans une configuration de réseaux stable qui est représenté par la figure 8.5. Les quatre portable PCs s'alignent et forment une chaîne. Nous examinons la table de routage d'IP de noyau dans chaque portable PC et les listons dans la table 8.3. Ils prouvent que DDR établit bien les routes entre ces portable PCs.

Puis nous changeons la position du portable PC radio4 et le mettons sous la couverture radio des autres portable PCs. Ainsi, une topologie en étoile est formée comme montré par la figure 8.6.

En conséquence, la table de routage d'IP du noyau Linux dans chaque portable PC est modifiée selon les tables 8.4 qui correspondent bien à la définition de DDR. Ce test valide donc notre implémentation.

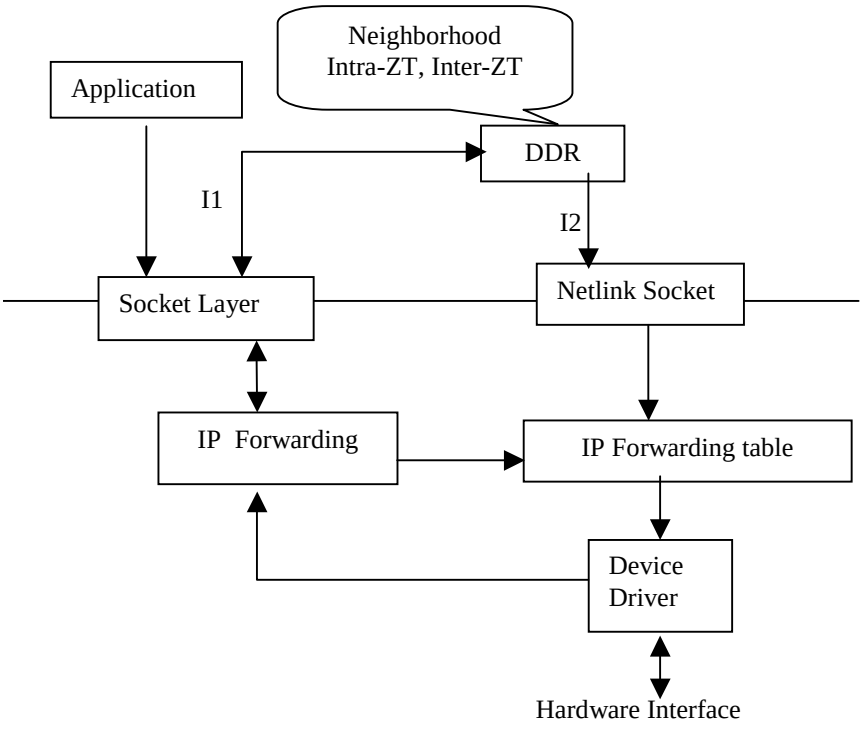


Figure 8.4: DDR implementation structure

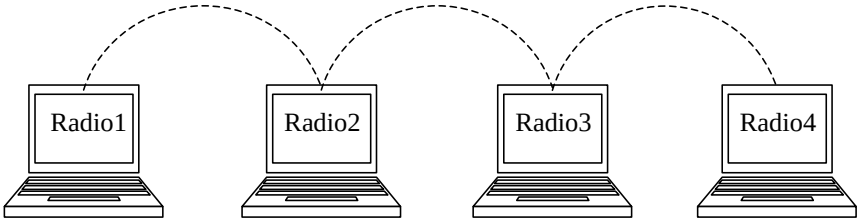


Figure 8.5: DDR validation: line scenario

Table 8.3: IP forwarding tables

Dest.	GW.
Radio2	Radio2
Radio3	Radio2
Radio4	Radio2

(a) Radio1

Dest.	GW.
Radio1	Radio1
Radio3	Radio3
Radio4	Radio3

(b) Radio2

Dest.	GW.
Radio1	Radio2
Radio2	Radio2
Radio4	Radio4

(c) Radio3

Dest.	GW.
Radio1	Radio3
Radio2	Radio3
Radio3	Radio3

(d) Radio4

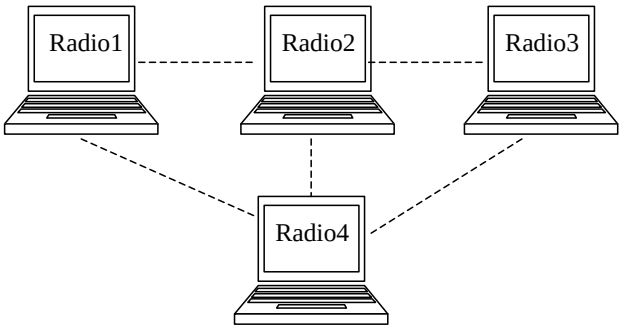


Figure 8.6: DDR validation: star scenario

Table 8.4: IP forwarding tables

Dest.	GW.
Radio2	Radio4
Radio3	Radio4
Radio4	Radio4

(a) Radio1

Dest.	GW.
Radio1	Radio4
Radio3	Radio4
Radio4	**Radio4**

(b) Radio2

Dest.	GW.
Radio1	Radio4
Radio2	Radio4
Radio4	**Radio4**

(c) Radio3

Dest.	GW.
Radio1	**Radio1**
Radio2	**Radio2**
Radio3	**Radio3**

(d) Radio4

8.6.2 Implémentation et validation de MRDC

Cependant, l'implémentation de protocole de routage de multicast ne peut pas employer le même d'architecture que celui d'unicast. D'abord, tous le noyaux Linux ne supportent le relais de paquets de multicast. Deuxièmement, les noyaux qui supportent le relais de paquets de multicast, ne permettent pas le relais à partir sur la même interface. C'est-à-dire le noyau n'expédiera pas un paquet à l'interface où il a reçu le paquet pour éviter des boucles de transmission. Mais dans MANET, l'expédition de multicast se produit sur la même interface sans fil. Pour surmonter ces limitations, une structure illustrée par la figure 8.7 est employée pour mettre en application le protocole de routage de multicast, MRDC. Dans cette architecture, quand une application envoie un paquet de multicast, le noyau de Linux, au lieu de l'envoyer directement à l'interface sans fil, donne ce paquet à l'agent de routage (MRDC dans cet exemple) fonctionnant dans l'espace d'utilisateur. Alors MRDC encapsule le paquet et le transmet de proche en proche (saut par saut) d'une manière appropriée (broadcast ou unicast) jusqu'à l'agent de MRDC de destination. Enfin, MRDC de destination décapsule le paquet et le livrent à l'application locale. Ce procédé permet au protocole de routage de multicast de transférer des paquets sur l'interface radio identique dans l'espace d'utilisateur tout en évitant la modification de noyau. Il facilite l'implémentation et l'installation sous des systèmes d'exploitation hétérogènes.

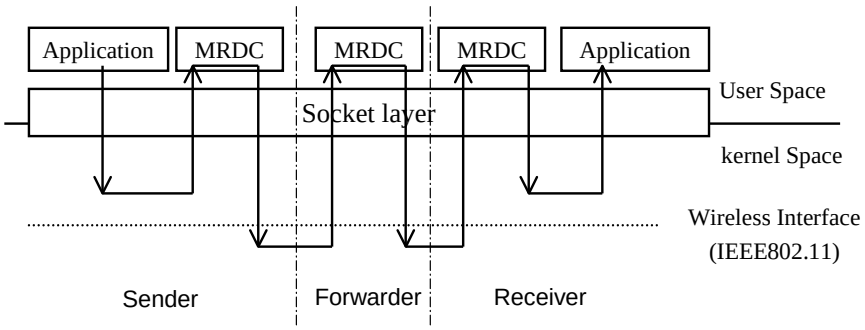


Figure 8.7: User space multicast routing architecture

Nous avons testé notre implémentation de MRDC dans un scénario stable et aussi un scénario dynamique. Le but de ces tests est de vérifier si MRDC peut bien construire l'arbre de multicast avec des branches en multiples sauts et si MRDC réagit bien quand la topologie ou le groupe changent mais aussi pour avoir la première connaissance de sa performance. Dans ces tests, "Videoconference tool" (vic) est choisie comme l'application de multicast. Vic peut distribuer un flux vidéo à un groupe. Il envoie aussi périodiquement un message au reste du groupe dans le but de maintenance de l'appartenance au groupe (procédure de type "keep alive"). Du point vue de protocole de routage, chaque vic est par conséquence à la fois source et récepteur dans le groupe. Cette application nous permet donc de simuler la communication multipoint à multipoint sans besoin de mobiliser plusieurs caméras.

Le scénario stable est représenté par la figure 8.8. Les 6 nœuds (3 portable PCs et 3 PDAs) formaient un anneau autour d'un mur virtuel construit par iptables. Nous avons d'abord lancé l'application de multicast sur le nœud A qui était connecté à une webcam qui distribue le flux vidéo à ce groupe. Ensuite, nous avons lancé l'application de multicast sur les nœuds D et F qui rejouent la vidéo qu'ils reçoivent sur leur écran. Nous observons l'évolution de la structure d'arbre par un outil de moniteur d'arbre. Un arbre de multicast qui connectait les nœuds A, D, F a bien été construit. Deux formes d'arbre apparaissaient alternativement dans le moniteur à cause de deux chemins de même longueur du nœud A vers le nœud F. Nous avons aussi analysé l'utilisation de bande passante dans ce scénario. Les résultats montrent que la partie contrôle y compris celui de IGMP occupe moins 5% de bande passante pour supporter un trafic de 365 kb/s. MRDC est donc efficace en ce terme.

Dans le scénario dynamique, nous avons simulé le changement de topologie et également le changement de groupe. Nous avons d'abord mis les nœuds dans la même couverture de radio. Nous activons les applications de multicast sur les noeuds A, D et F. Un arbre de multicast est construit et c'est le nœud A qui joue le rôle de la racine. Alors nous déplaçons le nœud F hors de la couverture du nœud A mais toujours dans la couverture du nœud B. L'arbre est reconstruit avec

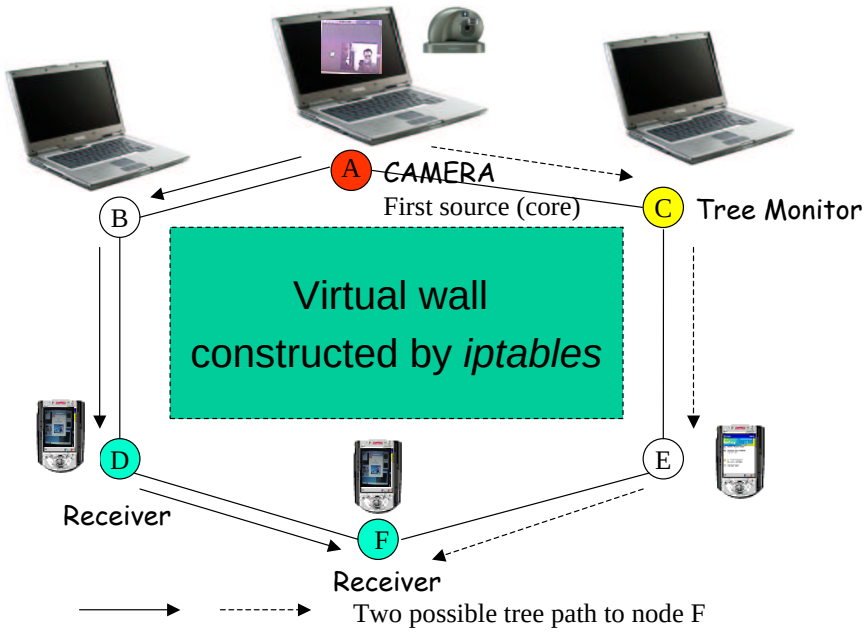


Figure 8.8: MRDC validation in static scenario

un nœud B qui fait le relais. Ensuite, nous terminons l'application sur le nœud A. Avec la disparition du noyau, l'arbre est dissout. Alors après un moment court, un nouvel arbre est formé et sa racine est le noeud F. Nous réactivons l'application de multicast sur le nœud A. Ce noeud rejoint l'arbre et distribue la vidéo au noeud D et F à travers l'arbre.

Pendant la plupart du temps de ces tests, la perte de données est inférieure à 10% et la qualité de vidéo est bonne dans les deux nœuds. Ces tests prouvent que MRDC fonctionne dans les réseaux réels. Ils valident aussi notre architecture d'implémentation.

8.7 Conclusion et travaux futurs

Récemment, les protocoles multicast dans les réseaux ad hoc sans fil reçoivent une grande considération par les chercheurs. Le multicast est une méthode efficace pour supporter les communications de point à multipoint ou multipoint à multipoint qui sont demandées par les applications de réseau ad hoc sans fil ayant le caractère de collaboration étroit dans un groupe. Cette thèse s'intéresse à deux sujets de recherche liés entre eux: le protocole de routage de multicast et le protocole fiable de multicast. Elle présente aussi nos travaux liés à l'implémentation des protocoles de routage dans un banc de test de réseau ad hoc sans fil.

Après avoir analysé les caractères spécifiques des réseaux ad hoc sans fil et l'avantage et limitation de différentes techniques et stratégies adoptées par les pro-

tocoles courants de routage multicast, nous avons proposé le protocole multicast MRDC. MRDC fournit une livraison efficace du trafic aux applications et s'adapte aux conditions de réseaux pour répondre au mieux aux exigences des applications. Ce protocole construit sur demande de trafic un arbre de multicast dont la racine est la première source. S'il y a d'autres sources qui participent au groupe, elles rejoignent l'arbre comme des membres normaux. En conséquence, l'arbre est optimisé pour la première source et il est partagé par les autres sources de ce groupe. MRDC contient trois modes de transmission et choisit un propre mode selon la demande des applications et les conditions courantes des réseaux dans le but d'offrir un bon compromis entre les exigences des applications et le coût de livraison du trafic. Les résultats des simulations ont montré que ce protocole est le plus efficace en termes du surcoût de contrôle et du surcoût de transmission en donnant le meilleur taux de réussite de livraison dans les réseaux suivant tous types de mobilité et trafic.

Quant aux applications qui demandent une garantie de livraison du trafic, nous avons étudié un protocole fiable de multicast en étendant le schéma de retransmission assité par routeur (router-assisted retransmission) puisque nos recherches ont montré que ce schéma est le meilleur par rapport aux deux autres schémas (source-originated and receiver-assisted) dans les réseaux ad hoc où le taux de perte de paquets est important. Ce protocole est nommé Active Reliable Multicast Protocol with Intermediate node support (ARMPIS). Il distribue de manière probabiliste la charge de stockage et de retransmission aux routeurs et à leurs voisins en bénéficiant la capacité de broadcast des nœuds sans fil. ARMPIS prend en compte les problèmes tels que la limitation de mémoire de chaque nœud et le changement de routeur après changement de topologie. Les résultats obtenus par les simulations ont montré que ARMPIS est un protocole fiable et efficace. Il maintient un taux de réussite presque à 100% dans les conditions de réseaux chargés ou à forte mobilité avec une charge de retransmission de sources trois fois moindre qu'un protocole fiable de multicast basé sur schéma de retransmission initié par la source.

Nous avons aussi implémenté les protocoles de routage dans un banc de test de réseau ad hoc sans fil afin d'évaluer leur performance et également de concevoir le prototype des nouvelles applications. Les implémentations des protocoles fonctionnent dans l'espace d'utilisateur de Linux et sont validées dans différents scénarios.

Il nous reste beaucoup de travail à faire pour améliorer la performance des protocoles. Par exemple, MRDC est conçu pour les réseaux de petite taille. Quand le nombre de nœuds augmente, le mécanisme d'inondation les messages de contrôle "core advertisement" CA ne sera plus approprié, ils généreront trop de surcoût de messages de contrôle. En même temps, le délai de la construction et régénération d'arbre va à sa tour augmenter. Ceux-ci gênent les transmissions et réduit l'efficacité du protocole. Un moyen de résoudre ce problème est d'utiliser des mécanismes de contrôle de topologie tels que ceux proposés par DDR. En groupant les nœuds en zone et localisant les zones qui contiennent les récepteurs, nous pourrons contrôler la propagation de message aux zones concernées et faciliter la construction d'arbre

en utilisant les chemins maintenus proactivement dans les zones. Pour l'instant, MRDC choisit le premier chemin découvert pour construire l'arbre. Nous pouvons penser prendre d'autres métriques telles que l'énergie de transmission pour économiser l'énergie dépensée dans le réseau et réduire l'interférence. La stabilité de lien peut également être adoptée comme métrique de construction d'arbre afin de diminuer le nombre de réparations d'arbre.

Les résultats des simulations nous indiquent une direction pour améliorer davantage la performance du protocole fiable de multicast. Nous avons observé que de temps en temps une requête peut provoquer plusieurs retransmissions de différents voisins pour le même paquet. Ces retransmissions provoquent des collisions au nœud qui lance le requête et en conséquence lui fait croire que la retransmission a échoué localement. Nous pouvons donc encore réduire le surcoût de retransmission par un meilleur mécanisme de la distribution de stockage et de retransmission parmi les voisins.

Nous avons implémenté et testé le protocole de routage multicast et celui de routage unicast séparément. Quand nous voulons mettre ces deux protocoles ensemble, nous devons penser comment ces deux protocoles peuvent partager leurs informations de routage afin de réduire le surcoût total de contrôle. La structure de l'implémentation de MRDC doit aussi être modifiée pour rendre MRDC plus performant en évitant les échanges coûteux de paquets qui traversent la couche socket. Nous envisageons deux solutions potentielles. MRDC tourne toujours dans l'espace d'utilisateur et il reçoit les paquets directement des applications par une interface standardisée. C'est l'idée de routage à la couche applicative. Ainsi, les applications peuvent facilement négocier leur demande de service avec protocole de routage et les modifications qui doivent porter sur la version courante d'implémentation seront aussi minimisées. Un autre choix est de porter MRDC dans le noyau de Linux afin de réduire davantage la traversée de la couche socket pendant les transmissions.

Nous espérons qu'avec nos contributions, les protocoles de multicast peuvent servir de support efficace pour les applications collaboratives en groupe dans les réseaux ad hoc sans fils.

Appendix A

A.1 Simultation results with confidence intervals

A.1.1 Simultation results of protocol comparison

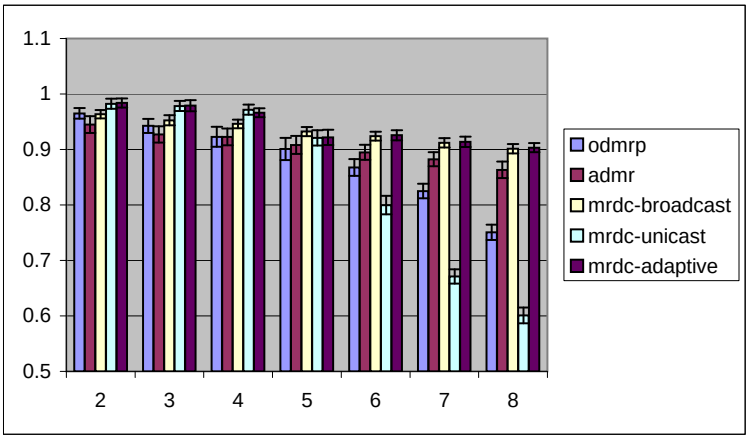


Figure A.1: Packet delivery ratio v.s. Number of source

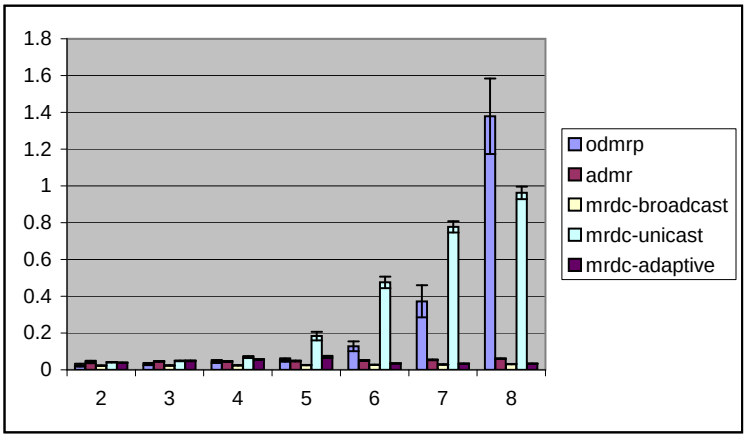


Figure A.2: End to End delay v.s. Number of sources

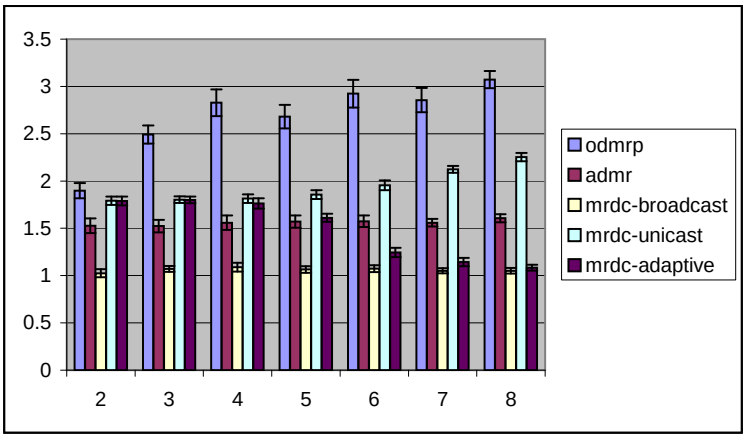


Figure A.3: Number of data packets transmitted per data packet delivered v.s. Number of sources

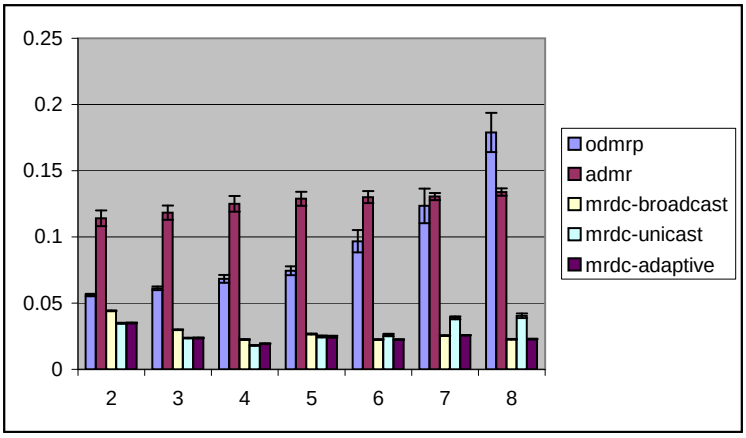


Figure A.4: Number of control bytes transmitted per data byte delivered v.s. Number of sources

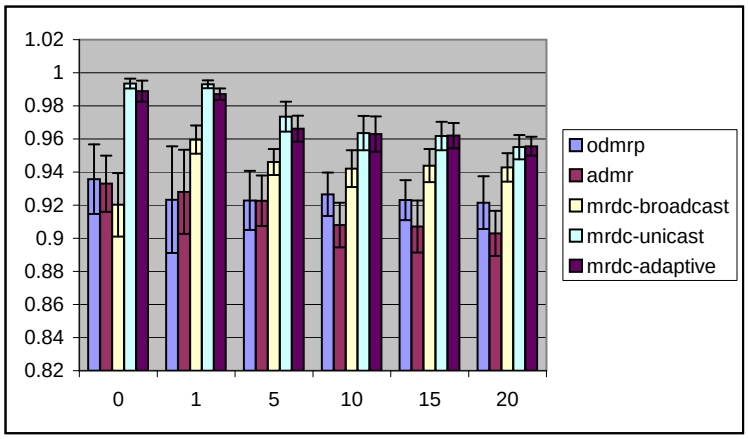


Figure A.5: Packet delivery ratio v.s. Maximum movement speed

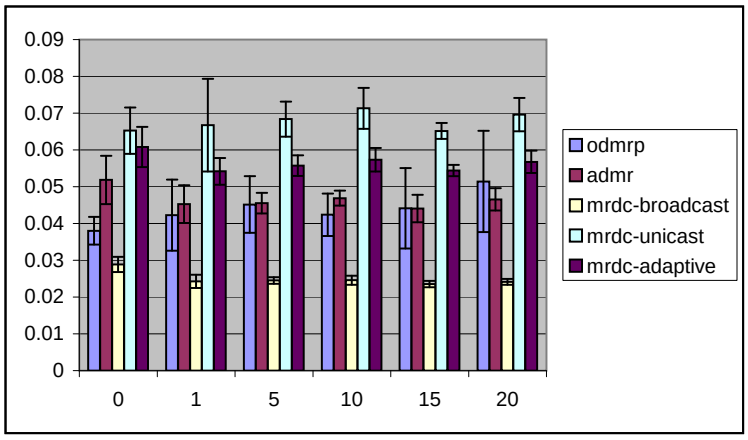


Figure A.6: End to End delay v.s. Maximum movement speed

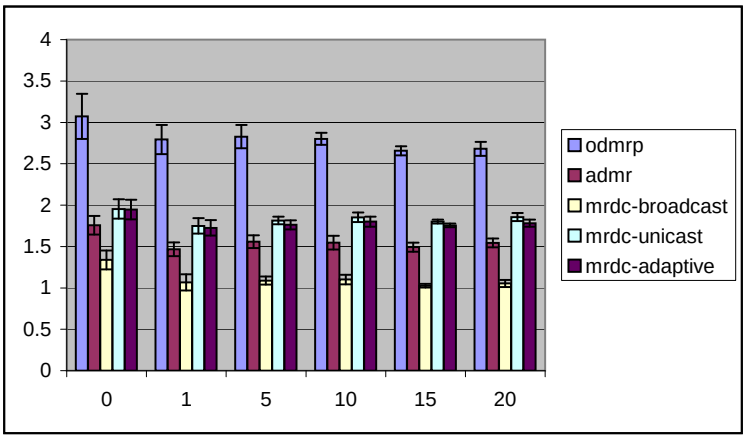


Figure A.7: Number of data packets transmitted per data packet delivered v.s. Number of sources

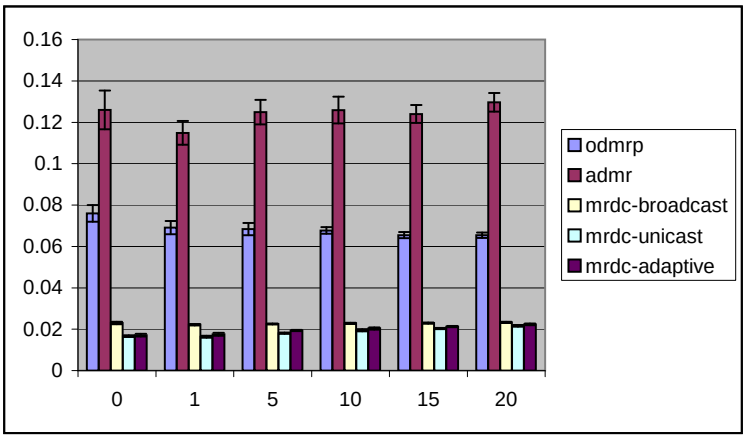


Figure A.8: Number of control bytes transmitted per data byte delivered v.s. Number of sources

A.1.2 Simultation results of reliable multicasting

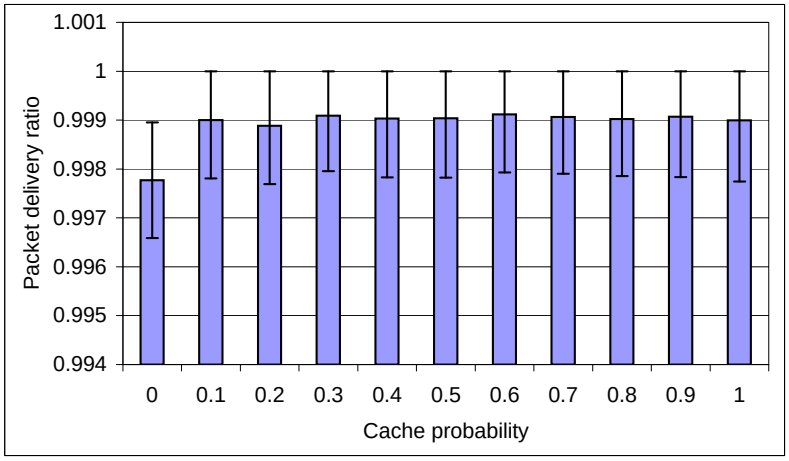


Figure A.9: Packet delivery ratio v.s. cache probability

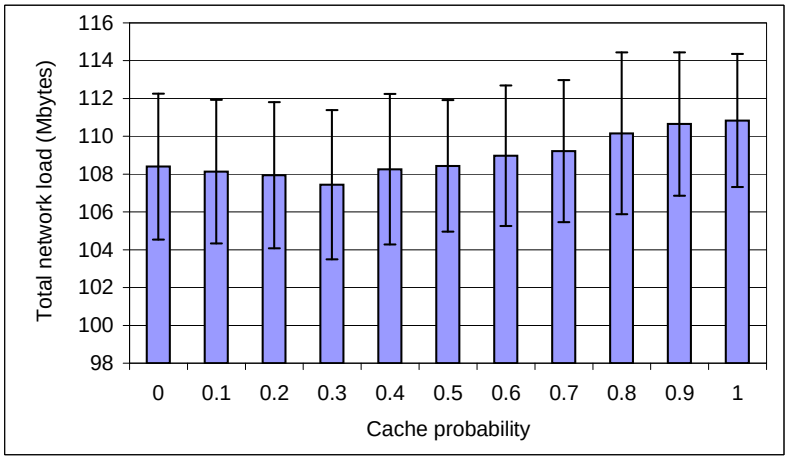


Figure A.10: Total network load v.s. cache probability

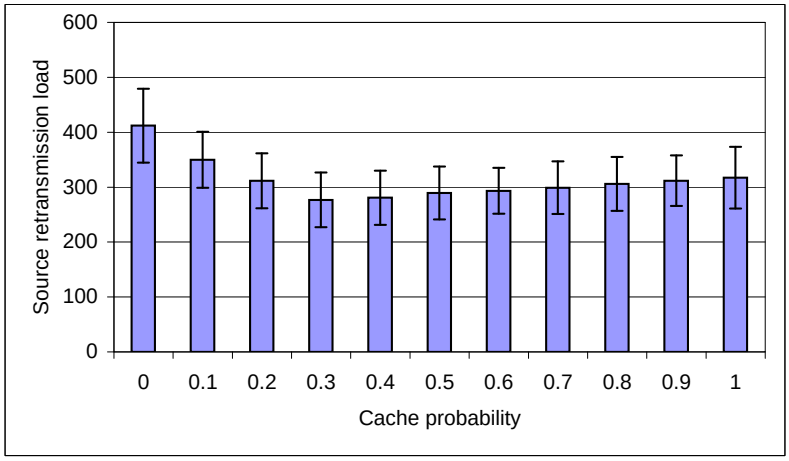


Figure A.11: Source retransmission load v.s. cache probability

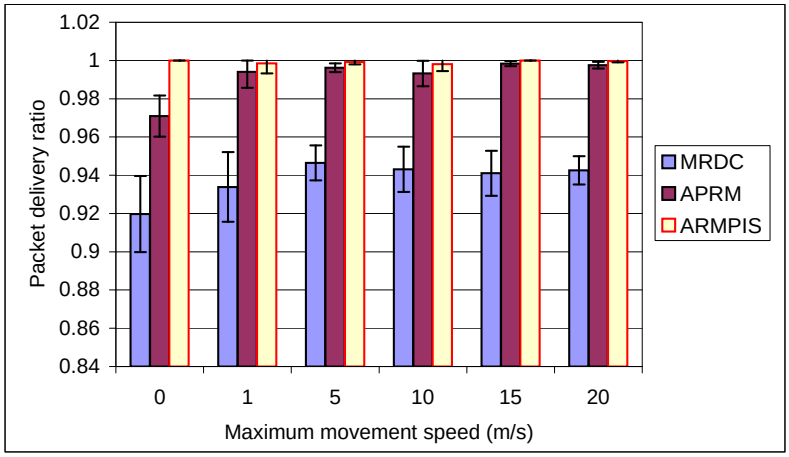


Figure A.12: Packet delivery ratio v.s. Maximum speed

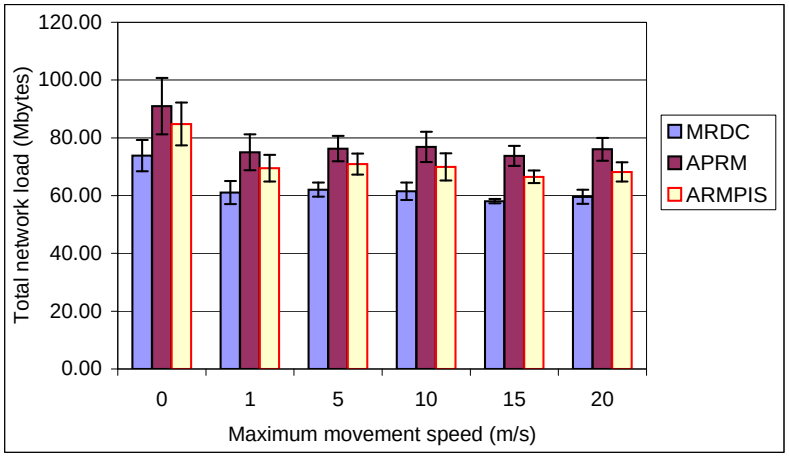


Figure A.13: Total network load v.s. Maximum speed

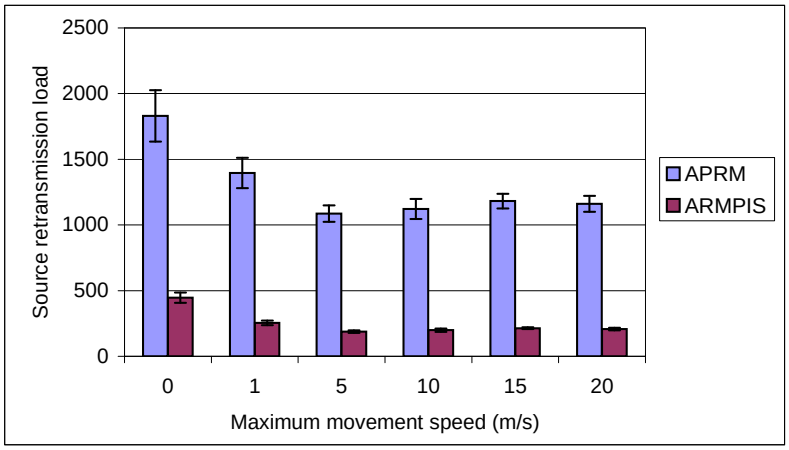


Figure A.14: Source retransmission load v.s. Maximum speed

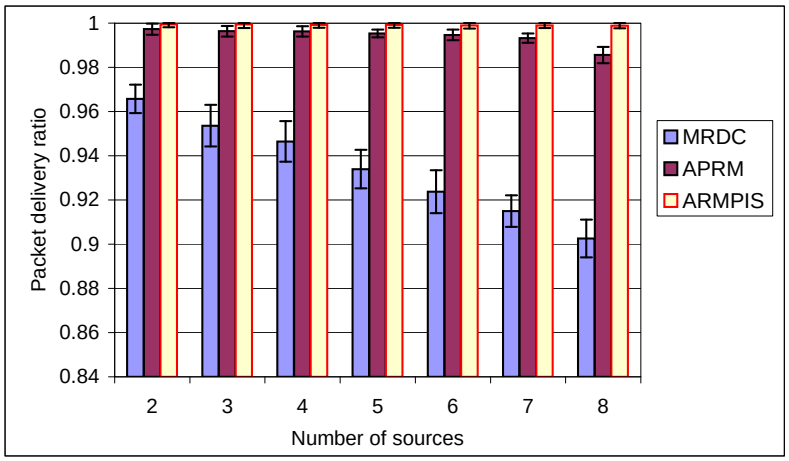


Figure A.15: Packet delivery ratio v.s. Number of sources

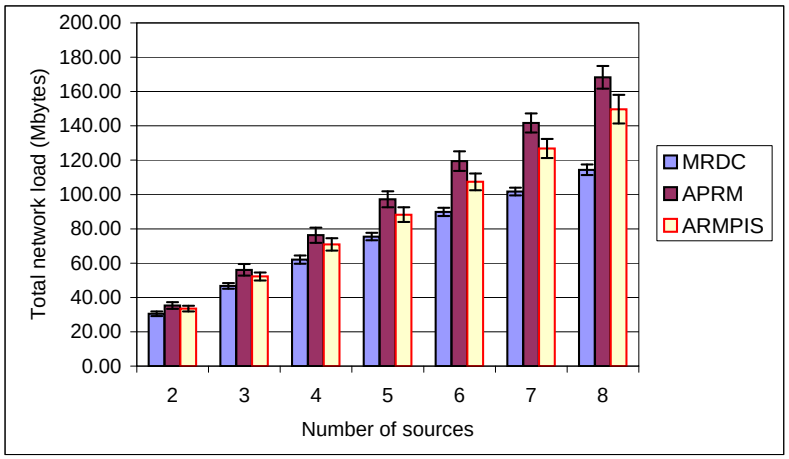


Figure A.16: Total network load v.s. Number of sources

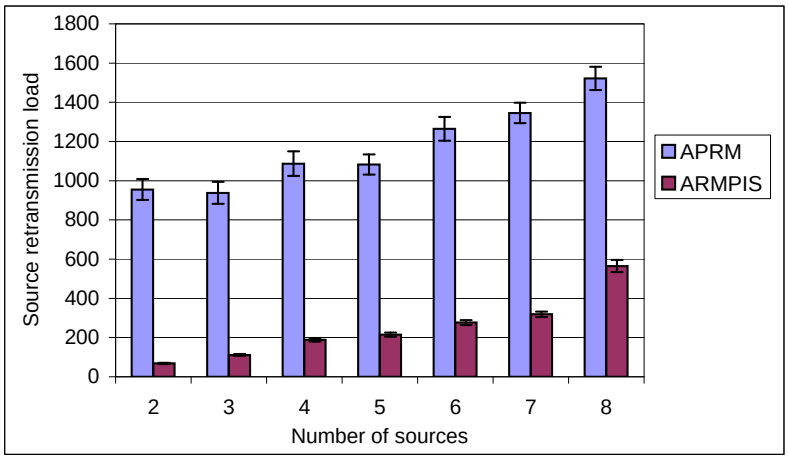


Figure A.17: Source retransmission load v.s. Number of sources

A.2 Flow Charts of Ad Hoc Testbed

A.2.1 Flow charts of DDR implementation

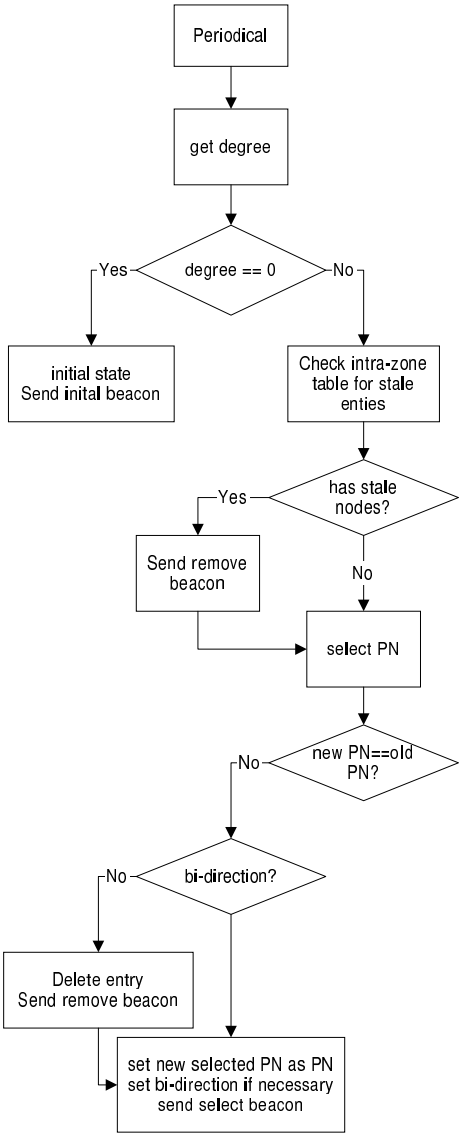


Figure A.18: Flow charts of DDR module

A.2.2 Flow charts of MRDC implementation

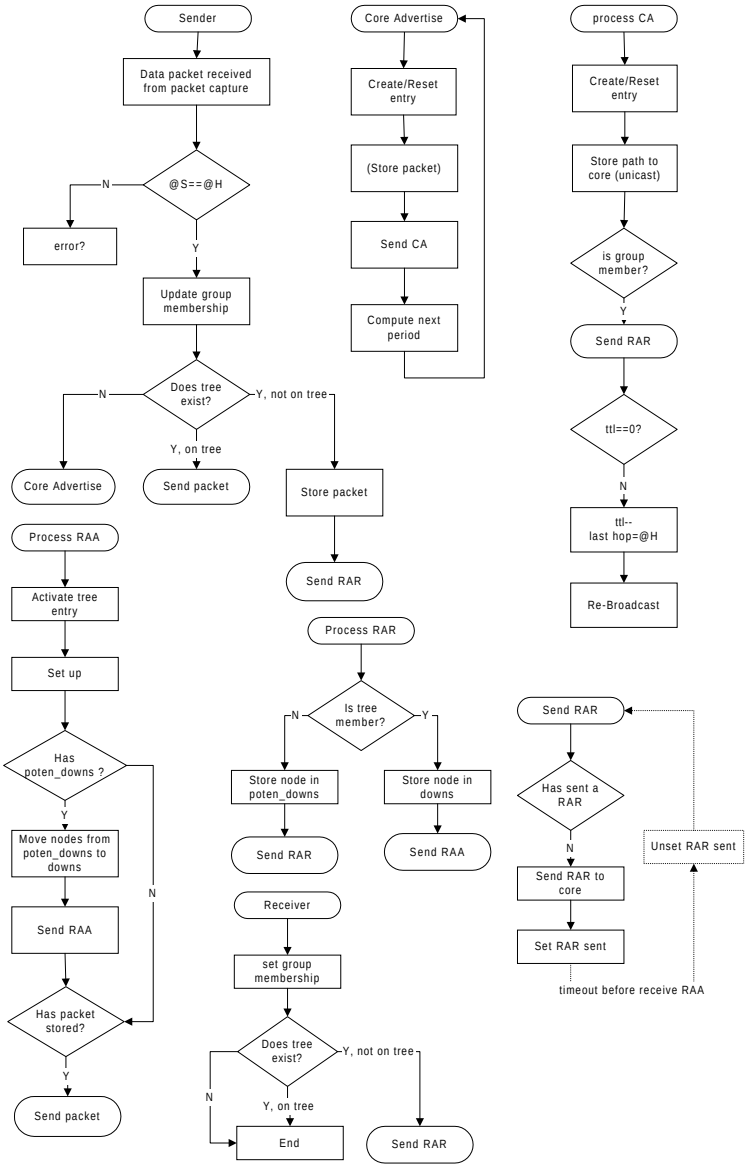


Figure A.19: Flow charts of MRDC module

Bibliography

[1] Mobile ad hoc network (manet). http://www.ietf.org/html.charters/manet-charter.html. Work in Progress.

[2] Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55, April 1999.

[3] Masato Hayashi and Christian Bonnet. A study on characteristics of ad hoc network applications. In *CSN 2002*, Benalmádena, Málaga, Spain, September 2002.

[4] S. Deering. Host extensions for ip multicasting, networkworking group. Request for Comments 1112, August 1989.

[5] S. Deering. Host extensions for ip multicasting, networkworking group. Request for Comments 1112, Auguest 1989.

[6] W. Fenner. Internet group management protocol version 2. Request for Comments 2236, November 1997.

[7] B. Cain, S. Deering, I. Kouvelas, and A. Thyagarajan. Internet group management protocol version 3. Internet draft, draft-ietf-idmr-igmp-v3-09.txt, January 2002.

[8] Wanjiun Liao and De-Nian Yang. Receiver-initiated group membership protocol (rgmp): A new group management protocol for ip multicasting. In *ICNP '99*, pages 51 – 58, October 1999.

[9] Kevin Fall and Kannan Varadhan, editors. *ns notes and documentation*. The VINT project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, February 2000. Avaible from http://www-mash.cs.berkely.edu/ns.

[10] Cordeiro de Morais Cordeiro, Hrishikesh Gossain, and Dharma P. Agrawal. Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Network*, 17:52–59, Jan.-Feb. 2003.

[11] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[12] J. Moy. Multicast routing extensions for ospf. *Communications of the ACM*, 37(8):61–66, 114, Aug. 1994.

[13] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. The pim architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2):153–162, April 1996.

[14] A. Adams, J. Nicholas, and W. Siadak. Protocol independent multicast dense mode (pim-dm): Protocol specification (revised). Internet draft, draft-ietf-pim-dm-new-v2-01.txt, February 2002.

[15] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees (cbt) - an architecture for scalable inter-domain multicast routing. In *ACM SIGCOMM'93*, pages 85–95, Oct. 1993.

[16] Satish Kumar, Pavlin Radoslavov, David Thaler, Cengiz Alaettinoglu, Deborah Estrin, and Mark Handley. The MASC/BGMP architecture for inter-domain multicast routing. In *SIGCOMM*, pages 93–104, Vancouver, Canada, September 1998.

[17] B. Fenner, M. Handley, H. Holbrook, and J. Kouvelas. Protocol independent multicast sense mode (pim-sm): Protocol specification (revised). Internet draft, draft-ietf-pim-sm-new-v2-01.txt.

[18] Ching-Chuan Chiang, Mario Gerla, and Lixia Zhang. Forwarding group multicast protocol (fgmp) for multihop, mobile wireless networks. *Cluster Computing*, 1(2):187–196, 1998.

[19] Sung-Ju Lee, William Su, Julian Hsu, Mario Gerla, and Rajive Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *INFOCOM 2000*, volume 2, pages 565–574, Tel Aviv, Israel, March 2000.

[20] Hasnaa Moustafa and Houda Labiod. Source routing-based multicast protocol for mobile ad hoc networks. In *International Conference on Telecommunication Systems Modeling and Analysis*, October 2002.

[21] Xiaofeng Zhang and Lillykutty Jacob. Multicast zone routing protocol in mobile ad hocwireless networks. In *28th Annual IEEE International Conference on Local Computer Networks*, pages 150–159, October 2003.

[22] C. W. Wu, Y. C. Tay, and C. K. Toh. Amris: A multicast protocol for ad hoc wireless networks. In *IEEE MILCOM'99*, volume 1, pages 25–29, Atlantic City, NJ, November 1999.

[23] E. M. Royer and C. E. Perkin. Multicast ad hoc on-demand distance vector (maodv) routing. Internet-Draft, Jul. 2000.

[24] S. J. Lee, M. Gerla, and C. C. Chiang. On-demand multicast routing protocol. In *IEEE WCNC'99*, pages 1298–1304, New Orleans, LA, Sep. 1999.

[25] Meejeong Lee and Ye Kyung Kim. Patchodmrp: an ad-hoc multicast routing protocol. In *International Conference on Information Networking*, pages 537–543, Beppu City, Oita, Japan, January 2001.

[26] J. Jubin and J. D. Tornow. The darpa packet radio network protocol. In *IEEE*, 1987.

[27] E. Bommaia, M. Liu, A. McAuley, and R. Talpade. Amroute: Ad hoc multicast routing protocol. Internet-Draft, Aug. 1998.

[28] J. J. Garcia-Luna-Aceves and E. L. Madruga. A multicast routing protocol for ad-hoc networks. In *IEEE INFOCOM'99*, pages 784–792, New York, NY, Mar. 1999.

[29] J.J. Garcia-Luna-Aceves and E.L. Madruga. The core assisted mesh protocol. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, 17(8):1380–1394, August 1999.

[30] Lusheng Ji and M. Scott Corson. Differential destination multicast-a manet multicast routing protocol for small groups. In *INFOCOM 2001*, volume 2, pages 1192–1201, Anchorage, AK USA, Apr. 2001.

[31] L. Ji and M.S. Corson. A lightweight adaptive multicast algorithm. In *IEEE GLOBECOM'98*, volume 2, pages 1036–1042, Sydney, NSW Australia, Nov. 1998.

[32] C-K. Toh, Guillermo Guichal, and Santithorn Bunchua. Abam: on-demand associativity-based multicast routing for ad hoc mobile networks. In *VTC 2000*, volume 3, pages 987–993, Boston, MA, USA, September 2000.

[33] Chai keong Toh. Associativity-based routing for ad-hoc mobile networks. *Wireless Personal Communication Journal*, 1997.

[34] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99*, pages 90–100, New Orleans, LA, USA, Febrary 1999.

[35] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. Mcedar: multicast core-extraction distributed ad hoc routing. In *IEEE WCNC'99*, volume 3, pages 1313–1317, New Orleans, LA, Sep. 1999.

[36] Raghupathy Sivakumar, Prasun Sinha, and Vaduvur Bharghavan. Cedar: a core-extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communications*, 17(8):1454–1465, Auguest 1999.

[37] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. Cedar: a core-extraction distributed ad hoc routing algorithm. In *INFOCOM '99*, pages 202–209, New York, NY, March 1999.

[38] Jorjeta G. Jetcheva and David B. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *the ACM Symposium on Mobile Ad Hoc Networking and Computing*, pages 33 – 44, Long Beach, CA, USA, October 2001.

[39] Sang Ho Bae, Sung-Ju Lee, W. Su, and M. Gerla. The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks. *IEEE Network*, 14(1):70–77, 2000.

[40] Seungjoon Lee and Chongkwon Kim. Neighbor supporting ad hoc multicast routing protocol. In *MobiHOC 2000*, pages 37–44, Boston, MA USA, Aug. 2000.

[41] Shiyi Wu and Christian Bonnet. Multicast rotuing protocol with dynamic core. In *IST 2001*, pages 274–280, September 2001.

[42] Shiyi Wu and Christian Bonnet. A reliable multicast protocol for ad hoc networks. In *WWC 2004*, pages 78–82, San Francisco, CA, USA, May 2004.

[43] Shiyi Wu and Christian Bonnet. Armpis: An active reliable multicasting protocol for ad hoc networks. In *SCI 2004*, Orlando, FL, USA, July 2004.

[44] Phil Karn. MACA - a new channel access method for packet radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134–140, London, Ontario, Canada, September 1990.

[45] Vaduvur Bharghavan, Alan J. Demers, Scott Shenker, and Lixia Zhang. MACAW: A media access protocol for wireless LAN's. In *SIGCOMM*, pages 212–225, London, UK, August 1994.

[46] Editors of IEEE 802.11,Wireless LAN Medium Access Control (MAC and Physical Layer (PHY) specification, Draft Standard IEEE 802.11, 1997.

[47] Philippe Jacquet, Paul Muhlethaler, Thomas Clausen, Amin Laouiti, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol for ad hoc networks. In *IEEE INMIC 2001*, pages 62–68, December 2001.

[48] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network

routing protocols. In *The 4th International Conference on Mobile Computing and Networking(ACM MOBICOM'98)*, pages 85–97, Dallas, Texas, USA, October 1998.

[49] Hasnaa Moustafa and Houda Labiod. A performance comparison of multicast routing protocols in ad hoc networks. In *Personal, Indoor and Mobile Radio Communications*, volume 1, pages 497–501, September 2003.

[50] Sang Ho Bae, Sung-Ju Lee, and Mario Gerla. Unicast performance analysis of the odmrp in a mobile ad hoc network testbed. In *Ninth International Conference on Computer Communications and Networks*, pages 148–153, Las Vegas, NV, USA, October 2000.

[51] David B. Johnson and David A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks*. Kluwer Academic Publishers, 1996.

[52] Giuseppe Anastasi, Marco Conti, and Enrico Gregori. *Ad Hoc Networking*, chapter IEEE 802.11 Ad Hoc Networks: Protocols, Performance and Open Issues. IEEE Press and John Wiley and Sons, Inc, 2004.

[53] K. Tang and M. Gerla. Random access mac for efficient broadcast support in ad hoc networks. In *IEEE WCNC 2000*, pages 454 – 459, September 2000.

[54] K. Tang and M. Gerla. Mac reliable broadcast in ad hoc networks. In *IEEE MILCOM 2001*, pages 1008 – 1013, October 2001.

[55] Min-Te Sun, Lifei Huang, Anish Arora, and Ten-Hwang Lai. Reliable mac layer multicast in ieee 802.11 wireless networks. In *ICPP'02*, pages 527 – 536, Vancouver, B.C., Canada, August 2002.

[56] S. K. S. Gupta, V. Shankar, and S. Lalwani. Reliable multicast mac protocol for wireless lans. In *IEEE International Conference on Communications*, pages 93–97, Anchorage, Alaska, USA, May 2003.

[57] http://www.monarch.cs.rice.edu/.

[58] Opnet modeler. http://www.opnet.com/products/modeler/home.html.

[59] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. Glomosim: A scalable network simulation environment. Technical Report 990027, UCLA Computer Science Departmen, May 1999.

[60] David Cavin, Yoav Sasson, and André Schiper. On the accuracy of manet simulators. In *the Workshop on Principles of Mobile Computing (POMC)*, Toulouse, France, October 2002.

[61] Mineo Takai, Jay Martin, and Rajive Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. In *MobiHoc 2001*, pages 87–94, Long Beach, CA, USA, October 2001.

[62] Bruce Tuch. Development of wavelan, an ism band wireless lan. *AT&T Technical Journal*, 72(4):27–33, July/August 1993.

[63] Yu-Chee Tseng, Sze-Yao Ni, and En-Yu Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *IEEE Transactions on Computers*, 52(5):545–557, May 2003.

[64] M. S. Corson and J. Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. Request For Comments 2501, January 1999.

[65] Jinyang Li, Charles Blake, Douglas S.J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *7th ACM International Conference on Mobile Computing and Networking*, pages 61–69, Rome, Italy, July 2001.

[66] Armstrong S., Freier A., and Marzullo K. Multicast transport protocol. Request for Comments 1301, February 1992.

[67] Heinrichs B. Amtp: Towards a high performance and configurable multipeer transfer service. In W. Effelsberg O. Spaniol, A. Danthine, editor, *Architecture and Protocols for High-Speed Networks*. Kluwer Academic, 1994.

[68] John C. Lin and Sanjoy Paul. Rmtp: a reliable multicast transport protocol. In *INFOCOM '96*, volume 3, pages 1414–1424, San Francisco, CA, USA, March 1996.

[69] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weigt sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[70] Yavatkar R., Griffioen J., and Sudan M. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia '95*, pages 333–344, 1995.

[71] Sneha K. Kasera, Supratik Bhattacharyya, Mark Keaton, Diane Kiwior, Steve Zabele, Jim Kurose, and Don Towsley. Scalable fair reliable multicast using active services. *IEEE Network*, 14(1):48–57, January/February 2000.

[72] Li-Wei H. Lehman, Stephen J. Garland, and David L. Tennhouse. Active reliable multicast. In *INFOCOM '98*, pages 581–589, San Francisco CA, March 1998.

[73] T. Speakman, D. Farinacci, S. Lin, and A. Tweedly. Pretty good multicast. Internet-Draft.

[74] Don Towsley, James F. Kurose, and Sridhar Pingali. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Selected Areas in Communications*, 15(3):398–406, April 1997.

[75] Elena Pagani and Gian Paolo Rossi. Reliable broadcast in mobile multihop packet networks. In *Mobicom*, pages 34–42, Budapest, Hungary, September 1997.

[76] Sandeep K. S. Gupta and Pradip K. Srimani. An adaptive protocol for reliable multicast in mobile multi-hop radio networks. Technical report, the 2nd IEEE workshop on mobile computing systems and applications, 1999.

[77] Ming-Yu Jiang and Wanjiun Liao. Family ack tree (fat): a new reliable multicast protocol for mobile ad hoc networks. In *IEEE International Conference on Communications, ICC 2002*, volume 5, pages 3393–3397, New york, USA, April 2002.

[78] Wanjiun Liao and Ming-Yu Jiang. Family ack tree (fat): Supporting reliable multicast in mobile ad hoc networks. *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, 52:1675–1685, November 2003.

[79] Ken Tang, Katia Obraczka, Sung-Ju Lee, and Mario Gerla. Reliable adaptive lightweight multicast protocol. In *IEEE International Conference on Communications*, volume 2, pages 1054–0158, Los Angeles, CA, USA, May 2003.

[80] Thiagaraja Gopalsamy, Mukesh Singhal, D. Panda, and P. Sadayappan. A reliable multicast algorithm for mobile ad hoc networks. In *International Conference on Distributed Computing Systems*, pages 563–570, Vienna, Austria, July 2002.

[81] Ahmed SOBEIH, Hoda BARAKA, and Aly FAHMY. Remhoc: a reliable multicast protocol for wireless mobile multihop ad hoc networks. In *Consumer Communications and Networking Conference,*, pages 146–151, 2004.

[82] Ranveer Chandra, Venugopalan Ramasubramanian, and Kenneth P. Birman. Anonymous gossip: Improving multicast reliability in mobile ad-hoc networks. In *The 21st International Conference on Distributed Computing Systems, ICDCS 2001*, pages 275–283, Mesa, AZ, USA, April 2001.

[83] Jun Luo, Patrick Th. Eugster, and Jean-Pierre Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *INFOCOM*, volume 3, pages 2229–2239, San Francisco, CA, USA, March 2003.

[84] Einar Vollset and Paul Ezhilchelvan. A survey of reliable broadcast protocols for mobile ad-hoc networks. Technical Report CS-TR-792, niversity of Newcastle upon Tyne, 2003.

[85] Victor O.K. Li and Zaichen Zhang. Internet multicast routing and transport control protocols. *IEEE*, 90(3):360–391, March 2002.

[86] Sagar Sanghani, Timothy X Brown, Shweta Bhandare, and Sheetalkumar Doshi. Ewant: The emulated wireless ad hoc network testbed. In *IEEE WCNC2003*, volume 3, pages 1844–1849, March 2003.

[87] http://www.linux.org/.

[88] Rich Stevens. *Unix network programming*. Prentice-Hall, 1998.

[89] Navid Nikaein, Houda Labiod, and Christian Bonnet. Ddr-distributed dynamic routing algorithm for mobile ad hoc networks. In *MobiHoc 2000*, pages 19–27, Boston, MA, USA, August 2000.

[90] Shiyi Wu, Yukihiro Takatan, Christian Bonnet, and Masato Hayashi. Implementation and validation of a multicast routing protocol in an ad hoc network testbed. In *Med-Hoc-Net*, Bodrum, Turkey, June 2004.

[91] Guy Pujolle. *Les réseaux*. Eyrolles, August 2002.

[92] David A Rusling. *The Linux Kernel*. Online Book, 1999. URL: http://www.tldp.org/LDP/tlk/tlk-title.html.

[93] David A. Maltz, Josh Broch, and David B. Johnson. Experiences designing and building a multi-hop wireless ad hoc network testbed. Technical Report CMU-CS-99-116, CMU School of Computer Science, March 1999.

[94] Lusheng Ji, Mary Ishibashi, and M. Scott Corson. An approach to mobile ad hoc network protocol kernel design. In *IEEE WCNC'99*, pages 1303–1307, New Orleans, LA, USA, September 1999.

[95] J. Postel. Internet control message protocol. Request for Comments 792, September 1981.

[96] Sang Ho Bae, Sung-Ju Lee, and Mario Gerla. Multicast protocol implementation and validation in an ad hoc network testbed. In *ICC 2001*, volume 10, pages 3196–3200, Helsinki, Finland, June 2001.

[97] Navid Nikaein and Christian Bonnet. Harp - hybrid ad hoc routing protocol. In *IST- International Symposium on Telecommunications,*, 2001.

[98] tcpdump. URL: http://www.tcpdump.org/.

[99] R. Russell. Linux 2.4 packet filtering howto. URL: http://netfilter.samba.org/documentation/.

[100] A video conference application. URL: http://www-nrg.ee.lbl.gov/vic.

List of Publications

Conferences and Workshops

Shiyi WU and Christian BONNET, “Multicast Routing protocol with Dynamic Core (MRDC)”, in Proceeding of IST 2001: International Symposium on Telecommunications, Iran/Tehran 2001.

Shiyi WU and Christian BONNET, “An Alternative Packet Transmission Procedure For Mobile Network Simulation”, in Proceeding of SPECTS2002: 2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems, San Diego, CA, USA, 2002.

Shiyi WU and Christian BONNET, “A Reliable Multicast Protocol For Ad Hoc Networks”, in Proceeding of WWC2004: World Wireless Congress, San Francisco, CA, USA, 2004.

Shiyi WU, Yukihiro TAKATANI, Christian BONNET and Masato HAYASHI, “Implementation and Validation of a Multicast Routing Protocol in an Ad Hoc Network Testbed”, in Proceeding of Med-Hoc-Net 2004 : The Third Annual Mediterranean Ad Hoc Networking Workshop, Bodrum, Turkey, 2004.

Shiyi WU and Christian BONNET, “ARMPIS: An Active Reliable Multicasting Protocol for Ad Hoc Networks”, in Proceeding of SCI 2004 : The 8th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA, 2004.

Poster Papers

Shiyi WU, Svetlana. RADOSAVAC and Christian BONNET, “Adaptive power-aware metric for Mobile Ad-hoc Network”, WTC2003: XVIII-World Telecommunications Congress, Paris, France 2002.

Shiyi WU and Christian BONNET “DDR-based Multicast Protocol with Dynamic Core (DMPDC)”, pfhsn2002 : 7th International Workshop on Protocols For High-Speed Networks, Berlin, Germany, 2002.

Demonstration

Jointly by Eurecom, HSAL, Hitachi Sophia Antipolis Laboratory ”Ad Hoc network”, in Symposium2003: 7th symposium of Eurecom-Hitachi, France/Sophia-Antipolis 2003.

Journal

Shiyi WU and Christian BONNET “Synthesis on Multicasting for Ad-Hoc Networks”, submitted to IEEE Communications, Ad Hoc and Sensor Networks Series