2007

RODMRP - resilient on demand multicast routing protocol

Dhammika Pathirana

RODMRP - Resilient On Demand Multicast Routing Protocol

by

Dhammika Pathirana

A Thesis

Submitted to the Faculty

of the

ROCHESTER INSTITUTE OF TECHNOLOGY

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

February 2007

APPROVED BY:

_____

Professor James Minseok Kwon, Thesis Adviser

_____

Professor Matt Cao, Reader

_____

Professor Hans-Peter Bischof, Observer

Abstract

ODMRP (On-Demand Multicast Routing Protocol) [6] [8] [2] is a popular multicast protocol for wireless ad hoc networks. The strengths of ODMRP are simplicity, high packet delivery ratio, and non-dependency on a specific unicast protocol. ODMRP floods a route request over the entire network to select a set of forwarding nodes for packet delivery. However, a single forwarding path is vulnerable to node failures, which are common due to the dynamic nature of mobile ad hoc networks. Furthermore, a set of misbehaving or malicious nodes can create network partitions and mount Denial-of-Service (DoS) attacks. This thesis proposes a ODMRP-based wireless multicast protocol named RODMRP that offers more reliable forwarding paths in face of node and network failures. A subset of the nodes that are not on forwarding paths rebroadcast received packets to nodes in their neighborhoods to overcome perceived node failures. This rebroadcasting creates redundant forwarding paths to circumvent failed areas in the network. Each node makes this forwarding decision probabilistically. Our simulation results indicate that RODMRP improves packet delivery ratio with minimal overheads, while retaining the original strengths of ODMRP.

Acknowledgments

I would like to extend my sincere gratitude to my adviser, Prof. Kwon for his continued guidance and supervision. Without his support, this work would not have seen the light of day. In addition, he also encouraged me to submit this work to the 21st IEEE conference on Advanced Information Networking and Applications (AINA 2007).

I would also like to thank other members of my committee: Prof. Cao and Prof. Bischof for their supervision. I am indebted to RIT CS Department for the graduate scholarship and to the RIT International Student Services for the international student scholarship, which enabled me to continue work on this thesis. In addition, I would also like to thank my employer Aspera Inc. for providing hardware and other resources to complete my thesis. A special thank goes to all my friends at RIT for making my college experience an enjoyable and a memorable one. Without them I would not have survived gloomy Rochester winters.

Last but not least, I would like to thank my parents and siblings for their encouragement and understanding, specially during the worst of times.

Forward

A shortened version of this thesis has been submitted to, and have been accepted for publication at the Second IEEE International Symposium on Pervasive Computing and Ad Hoc Communications (PCAC-07), held in conjunction with the IEEE 21st International Conference on Advanced Information Networking and Applications (AINA07).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "'Begin at the beginning,' the King said gravely, 'and go on till you come to the end: then stop.'" [1]

Despite revolutionary innovations and on going research in ad hoc networks, truly mobile ad hoc communication still remains an elusive goal. On the otherhand, wireless hardware has kept instep with Moore's law [15]. Extensive improvements in mobile hardware, have put more processing and communication capabilities in mobile devices, while shrinking the size and weight of the same devices. Thus, more and more consumer devices (i.e., media players, handheld gaming devices) have built in wireless content sharing capabilities. Furthermore, next generation mobile phones are equipped with wireless networking capabilities in addition to their CDMA, GSM carrier networking functionality.

But firmware and protocols have lagged behind. Dealing with mobility and unpredictable chaotic nature of mobile environments have particularly been a challenge for ad hoc protocol developers. In addition, mobile devices are placed in environments where there are no dedicated networking infrastructure (i.e., subway trains). Therefore, the protocols cannot assume the availability of dedicated networking infrastructure and often

[1] *Alice in Wonderland and Through the Looking Glass* - Lewis Carroll

have to rely on collaborating devices for routing. But hardware, networking and processing capabilities of each device vary to such an extent that communication protocols should not only be simple but also should only rely on most basic networking functionality. Catering to these divergent and conflicting requirements make truly mobile protocol development an intriguing problem.

On the otherhand, mobile devices have become an essential ingredient of every day life. The mobile device sales show staggering double digit growth rates and market analysts forecast nothing but more sales (i.e., in 2006 there were over 1 billion mobile phone sales[3] and in 2007, the portable media players alone will generate 6 billion revenue with 41 million device sales[13]). But content in such mobile devices is restricted to each individual device due to lack of communication capabilities. In addition, collaborative applications (i.e., messaging) have to exclusively rely on proprietary or carrier networking capabilities. In contrast, a simple communication protocol will unfold a new era of disruptive mobile applications, where users will collaborate/share their content (i.e., imagine a subway commuter with a futuristic iPOD with ad hoc networking sharing her song collection with fellow commuters).

A variety of such applications may utilize multicast to disseminate data from a source to a set of receivers in a wireless network. Streaming video and audio applications are considered the most important since they require high-bandwidth, seamless and uninterrupted streaming services, especially for live streaming applications. While currently available wireless multicast protocols (i.e., AODV [1]) offer efficient and reliable data delivery services, abrupt network disconnections or node failures cause service interruption until faulty parts are restored [7] [2]. This service interruption is intolerable for live streaming applications since lost frames cannot be recovered. In wireless ad hoc networks, switching to a backup network links or nodes take longer time because alternate paths are not immediately available and need to be reconfigured. Alternatively, configuring multiple

routing paths can address this issue with a significant amount of extra traffic.

On-Demand Multicast Routing Protocol(ODMRP) [6] [8], which was specifically developed for ad hoc network multicasting, has been one of the more popular and widely researched [18, 10, 11] multicast routing protocols for mobile ad hoc networks. In addition, performance characteristics of ODMRP have been extensively studied [7, 2]. Despite its simplicity and high packet delivery ratios, ODMRP has several drawbacks (i.e., reliance on a single routing path, network flooding of *Join Query* packets etc.), which are discussed in detail in Chapter 3. This research proposes a simple packet forwarding scheme to improve the efficiency of ODMRP. The simulation results indicate that proposed improvements significantly enhance the resiliency of ODMRP, while retaining its original simplicity and elegance.

This thesis is organized as follows, Chapter 2 provides a detailed description of ODMRP and its characteristics, while Section 2.5 documents previous work on ODMRP. Chapter 3 identifies the problems of ODMRP and discusses the motivations for the development of Resilient On-Demand Multicast Routing Protocol(RODMRP). In Chapter 4, an overview of the proposed improvements and rational behind those proposals are explained. In Chapter 5, the actual implementation details of the proposed improvements in ns2 [14] (a widely used network protocol simulator) are documented. Chapter 6 includes the simulation results of the proposed protocol. Finally, this thesis concludes in Chapter 7 and also proposes future enhancements to RODMRP in Section 7.2.

Chapter 2

On-Demand Multicast Routing Protocol (ODMRP)

> "'Would you tell me, please, which way I ought to go from here?'
> 'That depends a good deal on where you want to get to,' said the Cat.
> 'I don't much care where–' said Alice.
> 'Then it doesn't matter which way you go,' said the Cat.
> '–so long as I get somewhere,' Alice added as an explanation.
> 'Oh, you're sure to do that,' said the Cat, 'if you only walk long enough.'" [1]

On Demand Multicast Routing Protocol (ODMRP) [6] [8] [2] is one of the more popular multicast routing protocols for ad hoc networks. This chapter presents a detailed description of ODMRP and documents its characteristics. In addition, this chapter summarizes previous work on ODMRP.

2.1. Background

ODMRP [6] [8] [2] was developed by the Wireless Adaptive Laboratory of University of California, Los Angeles. The unique features of ODMRP are,

[1] *Alice in Wonderland and Through the Looking Glass* - Lewis Carroll

- Forwarding tree concept where the protocol forms a subset of collaborating nodes into a packet forwarding tree.

- Mesh based forwarding tree to avoid drawbacks of traditional tree based multicast protocols (i.e., frequent tree re-configuration, traffic concentration, non shortest path, intermittent connectivity). Unlike clustering based protocols, ODMRP forms multiple potential forwarding paths, which enable multicast receivers to discover better routes.

- Soft state based forwarding concept to avoid explicit messaging for setting up, tearing down and maintaining of multicast groups.

2.2. Protocol overview

2.2.1. Definitions

- *Join Query*

  A control packet sent by the multicast sender to initiate a multicast session. In addition, the multicast source sends *Join Query* packets periodically to refresh the forwarding path.

- *Join Table*

  A control packet sent by the multicast subscribers. *Join Table* packet are forwarded upstream towards the multicast source to reinforce the forwarding path for multicast data.

- *Forwarding nodes*

  Intermediate nodes, which forward multicast data on behalf of the multicast source. Intermediate nodes indicate their willingness to participate in multicast data transmission by forwarding *Join Query* packets, which enable down stream nodes to

select such nodes as forwarding nodes. By forwarding subsequent *Join Table* packets intermediate nodes elect themselves as *Forwarding nodes* and thus form the forwarding path. Forwarding state is maintained by periodic refreshing of routing paths by *Join Query* packets, which enable multicast subscribers to discover better forwarding paths and recover from faulty forwarding paths.

The ODMRP consists of two phases, namely *Join Request* and *Join Reply* phase. To construct a multicast forwarding tree, an ODMRP source periodically floods *Join Query* packets to the entire network to advertise the availability of multicast session. Upon receiving a *Join Query* packet, intermediate nodes record the sender as the upstream parent and a unique identifier of the packet (i.e., sequence number). Then intermediate nodes rebroadcast the packet. Duplicate *Join Query* packets are detected via unique identifier previously observed and suppressed for forwarding. When the *Join Query* reaches a prospective receiver, the receiver selects the best path based on a predefined criteria (i.e., least hop path, least delay path) and sends a *Join Table* packet back to the source. The *Join Table* packet is relayed by intermediate nodes and travels all the way back to the source on the reverse path. The *Join Table* packet reinforces the path established by the *Join Query*. Subsequently, when the source sends data, the intermediate nodes become *Forwarding nodes* in the data delivery tree. ODMRP maintains group membership as a soft-state in which parent-child relationships should be refreshed periodically. Hence, ODMRP does not require explicit join or leave mechanisms. However, periodic flooding of *Join Query* has a trade off. While frequent refreshing can improve the route recovery/discovery and thus result in an increased packet delivery ratio, it can overwhelm the nodes with excessive traffic overhead and waste network bandwidth and node resources. On the otherhand, delayed path refreshing can actually delay route recovery/discovery process and can thus contribute to reduce the packet delivery ratio.

2.2.2. Protocol illustration

Following figures illustrate ODMRP forwarding path setup and multicast data forwarding. Figure 2.1 illustrates *Join Query* packet forwarding, while Figure 2.2 illustrates forwarding path setup by *Join Table* packets. Finally, Figure 2.3 shows multicast data forwarding through previously established forwarding path.

In Figure 2.1 Node A broadcasts a *Join Query* packet as a multicast source, and all the surrounding nodes (B, C, D, E, and F) rebroadcast the received *Join Query* packet. In addition, intermediate nodes retain a unique identifier of the last *Join Query* and the information of their immediate parent towards the multicast source. In the Figure 2.1, these packets open two possible forwarding paths from the source to multicast receiver L. L selects the path from J, and replies a *Join Table* packet back to the source reinforcing the selected path. While the *Join Table* packets travel toward the source, nodes K, J and C update their routing tables as forwarding nodes for multicast sender A (Figure 2.2). Once the *Join Table* packets reach the multicast sender A, it initiates sending of multicast data through the forwarding path (Figure 2.3).

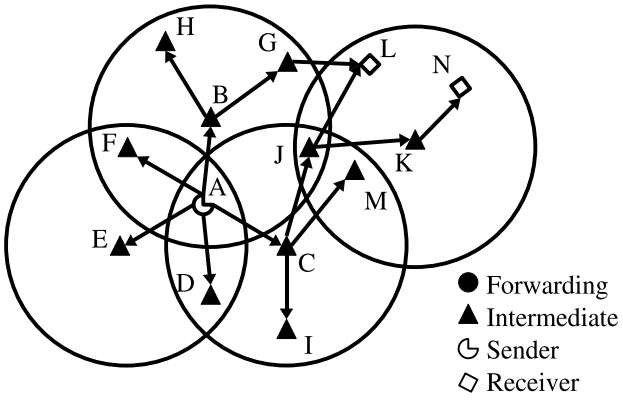

Figure 2.1. ODMRP *Join Query*

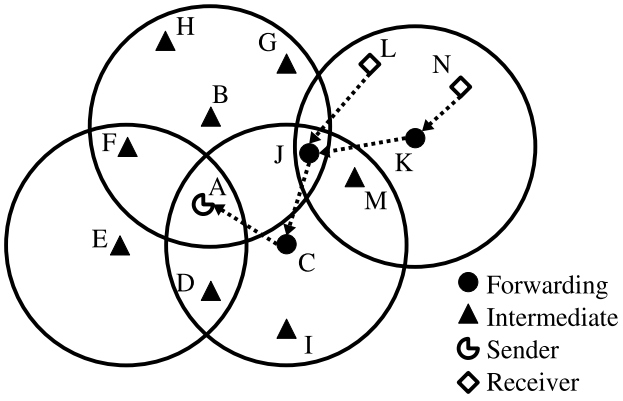

Figure 2.2. ODMRP *Join Table*

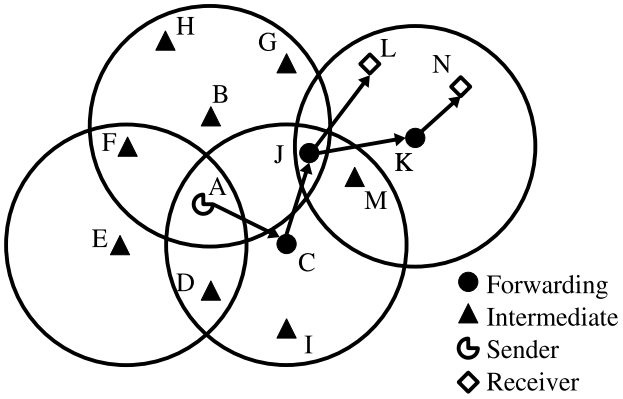

Figure 2.3. ODMRP data forwarding

Type	Reserved	Time to Live	Hop Count
Multicast Group IP Address			
Sequence Number			
Source IP Address			
Previous Hop IP Address			
Minimum Link Expiration Time			

Table 2.1. ODMRP *Join Query* packet format

Type	Reserved	Time to Live	Hop Count
Multicast Group IP Address			
Sequence Number			
Source IP Address			
Next Hop IP Address			

Table 2.2. ODMRP *Join Table* packet format

2.3. Protocol data structures

Following section documents different data structures used in ODMRP [8].

2.3.1. Protocol packet format

ODMRP uses following packet formats for *Join Query* and *Join Table* packets.

- *Join Query* packet (table 2.1) consists of following fields,
  - Type : ODMRP packet type (Join Query)
  - Reserved : Not set
  - Time to Live : Number of maximum hops to travel
  - Hop Count : Number of hops traveled so far
  - Multicast Group IP Address : IP address of the multicast group
  - Sequence Number : Unique identifier assigned by the multicast source
  - Source IP Address : IP address of the multicast source
  - Previous Hop IP Address : IP address of the previous hop

Multicast Group IP Address
Source IP Address
Last Sequence Number
Next Hop Parent
Hop Count
Minimum Link Expiration Time
Forwarding flag

Table 2.3. ODMRP routing table

  - Minimum Link Expiration Time : Link expiration time assigned by multicast source

- *Join Table* packet (table 2.2) consists of following fields,

  - Type : ODMRP packet type (i.e., Join Table)
  - Reserved : Not set
  - Time to Live : Number of maximum hops to travel
  - Hop Count : Hop count observed by the multicast subscriber
  - Multicast Group IP Address : IP address of the multicast group
  - Sequence Number : Unique identifier assigned by the multicast source for previous *Join Query* packet
  - Source IP Address : IP address of the *Join Table* sender
  - Next Hop IP Address : IP address of the next hop parent towards multicast source

2.3.2. Routing table format

Each node maintains following routing table entries for each multicast source (table 2.3).

- Multicast Group IP Address : IP address of the multicast group

- Source IP Address : IP address of the multicast source
- Last Sequence Number : Last observed unique identifier sent by the multicast source
- Next Hop Parent : IP address of the next hop parent towards the multicast source
- Hop Count : Number of hops towards parent through previous hop parent
- Minimum Link Expiration Time : Link expiration time set by previous *Join Query* packet
- Forwarding flag : Packet forwarding flag set for the multicast source based on *Join Table* packets.

2.4. Advantages of ODMRP

- Simplicity

  When compared with other wireless multicasting protocols ODMRP shows remarkable simplicity, which can be attributed to its simple control messaging format.

- High delivery ratios

  Despite its simplicity ODMRP shows high packet delivery ratios [7].

- Low channel and storage overhead

  In ODMRP, both forwarding and non-forwarding nodes maintain similar routing tables. In fact, the only difference between forwarding and non-forwarding nodes is the forwarding flag specifier in their routing table.

- Robustness to host mobility

  The multicast source can easily accommodate the mobility of the nodes by frequent route refreshing.

- Periodic dynamic construction of the forwarding path

  In addition, frequent route refreshing, enables the multicast subscribers to recover from faulty forwarding paths and to discover alternative better forwarding paths soon.

- Maintenance and exploitation of multiple forwarding paths

  Mesh based forwarding path setup enables the multicast subscribers to exploit the availability of multiple forwarding paths. In addition, makes it possible for the multicast receivers to recover paths and discover alternative forwarding paths.

- Non dependence on a specific routing protocol

  Any routing protocol with broadcasting capabilities can easily adopt ODMRP.

- Unicast routing capability

  Unlike most other multicast routing protocols, ODMRP does not require specific multicasting protocols for its implementation. Any networking protocol with broadcasting and unicasting can easily implement ODMRP.

- Low control messaging overhead

  Due to its mesh based forwarding node setup, nodes do not have to maintain forwarding trees. This has enabled the protocol to reduce its control messaging overhead. Route discovery and recovery are entirely done by *Join Query* packets. In addition, *Join Query* packets for route refreshing can be piggy backed with data packets.

- Soft state based multicast subscription and unsubscription

  Forwarding path setup is entirely based on received *Join Table* packets by intermediate nodes. This collaborative setup works well to reduce the control packet overhead. To stop a multicast session, the sender just stops sending *Join Query*

packets, similarly to unsubscribe from an ongoing multicast session, a receiver just stops sending *Join Table* upstream. In addition, forwarding nodes are demoted to non-forwarding nodes if they do not send *Join Table* packets upstream.

Despite above strengths, ODMRP has several drawbacks. Chapter 3 takes an in depth look at the drawbacks of ODMRP.

2.5. Related work

The flooding of Join Query packets in ODMRP [6] wastes network bandwidth, causes congestion, and drains node resources. To decrease the number of redundant packets, efficient flooding with passive clustering was proposed by Teak *et al.* [5, 17, 16]. In passive clustering, nodes maintain soft states by eavesdropping on packet transmissions that indicate successful rebroadcasting.

Enhanced ODMRP with Motion Adaptive Refresh (E-ODMRP) [10] enhanced ODMRP with an adaptive route refresh scheme based on reports from receivers. In particular, the enhancement changes the route refreshing period dynamically to reduce the flooding overhead of Join Query packets. Thus, it improved the efficiency of the protocol. In addition, E-ODMRP proposed a local route recovery scheme based on expanded ring search. However, this approach adds additional control packets (i.e., Receiver Join) and requires additional processing at nodes, which may not be available in low end mobile devices. Furthermore, malicious or misbehaving nodes can drain resources of multicast receivers and forwarding nodes by initiating frequent expanded ring searches.

ODMRP with Multipoint Relay (ODMRP-MPR) [18] presented a multi-point relaying technique to overcome uni-directional links. The multipoint relaying technique selects a set of nodes as multipoint relays for rebroadcasting of Join Query packets. This is also an alternative approach to reduce the flooding overhead of ODMRP. This protocol adds an additional control packet called Hello packet to identify neighbor nodes. Each

node periodically broadcasts a hello packet with a list of its current known neighbors. Upon receiving a Hello packet, a node can identify its two hop neighbors by processing the neighbor list. Although this approach effectively identifies bi-directional links and establishes a reliable forwarding tree, it does not guarantee the delivery of subsequent data transmissions.

Klos and Richard III [4] proposed a reliable group communication protocol based on ODMRP. The rational behind their proposal was to store a subset of forwarded/received packets to improve the reliability of the protocol. The protocol assumes that even though a single node will be able to store a limited number of packets, the group as a whole will be able to store a substantial amount of packets. In addition, this research required a "Reliable Join Query" phase to the protocol, where each node receiving a Join Query packet added a unique identifier (i.e., its IP address) to create a list of all forwarding nodes. This list enabled receivers to identify the whole multicast group members, which could be queried later for missed/delayed packets. Although this technique improves the reliability, it severely limits the scalability of the protocol. Furthermore, this approach may not be effective for real-time data delivery.

Sobeih *et al.* [11] have studied the reliability of ODMRP in Reliable Multicast Protocol for Wireless Mobile Multihop Ad Hoc Networks (ReMHoc). ReMHoc is a receiver initiated NACK based technique to improve the reliability. In addition, ReMHoc is a distributed protocol, where receivers and forwarding nodes maintain packet caches to facilitate lost packet recovery. Upon detecting a lost packet a receiver or a forwarding node initiates packet recovery by sending a recovery request. Upon receiving a recovery request, nodes identify redundant recovery requests and suppress them. Thus, nodes avoid recovery request explosion.

Reliable Multicast of ODMRP (RODMRP) [12] is another proposal to improve the reliability of ODMRP using a "round robin window". Each node maintains a sent and

received packet cache in addition to its neighbor list. Upon receiving a data packet, the receiver identifies any missing packets and indicates the missing packets in its acknowledgment to its parent. The neighboring nodes eavesdrop on these acknowledgments and check their received packet caches for lost packets indicated in the acknowledgment. On detecting lost packets in their packet cache, neighboring nodes forward those packets to the receiver to improve the reliability of the protocol.

Chapter 3

Drawbacks of ODMRP

> ”Alice had been looking over his shoulder with some curiosity. ‘What a funny watch!’ she remarked. ‘It tells the day of the month, and doesn’t tell what o’clock it is!’
>
> ‘Why should it?’ muttered the Hatter. ‘Does your watch tell you what year it is?’
>
> ‘Of course not,’ Alice replied very readily: ‘but that’s because it stays the same year for such a long time together.’” [1]

Despite ODMRP’s advantages, which were presented in Section 2.4, ODMRP has several disadvantages. This chapter discusses drawbacks of ODMRP in detail and presents motivating factors for the development of RODMRP.

3.1. Drawbacks of ODMRP

- Waste network bandwidth

  ODMRP *Join Query* packets are flooded throughout the network. Each node receiving a *Join Query* packet is expected to rebroadcast it. The rational behind this design was that it would improve the forwarding path setup. But on the otherhand, redundant *Join Query* packets not only waste network bandwidth and channel capacity but also increase packet collisions and drain resources of both senders and

[1] *Alice in Wonderland and Through the Looking Glass* - Lewis Carroll

receivers. In addition, this redundant messaging overhead significantly affects the efficiency of the protocol in real world environments with a high density of nodes (i.e., a classroom, a conference etc.).

- Reliance on a single forwarding path for data forwarding

  Once a forwarding path in established through a *Join Query*/*Join Table* cycle, the multicast receivers rely on the established forwarding path for the entire route refresh duration. Although, ODMRP uses mesh based networking for forwarding path setup, it does not allow the multicast receivers to utilize additional forwarding paths. This design flaw essentially converts the mesh based forwarding structure to a tree based one for the entire duration of the refresh cycle.

- Adversaries can impact forwarding behavior

  An adversary or failed node on established forwarding path can easily deny all nodes in its down stream path from receiving data packets. As discussed above, once the forwarding path is established the multicast receivers rely on forwarding nodes for the duration of the refresh cycle. This is illustrated in Figure 3.1 and 3.2.

- Multicast sender has to periodically refresh forwarding path

  In order to maintain the forwarding tree, the multicast sender has to frequently refresh forwarding path by sending *Join Query* packets. Although this enables multicast receivers to discover better forwarding paths and recover from faulty forwarding paths it is not desirable for less faulty networks. In fact, this increases the protocol overhead and directly impacts the efficiency of the protocol. In addition, frequent *Join Query* broadcasts drain resources of the nodes and would discourage other nodes from participating in packet forwarding.

- Does not consider mobility of the nodes

  The protocol does not consider the mobility of the forwarding/receiving nodes. The

design of the protocol assumes that frequent refresh cycles can accommodate node mobility. But as discussed above, frequent forwarding path refresh cycles reduce the efficiency of the protocol. Observing this drawback, the protocol designers have proposed a mobility predication model [8]. But this improvement requires additional hardware (i.e., GPS positioning devices).

- Requires bi-directional wireless capability

  The protocol assumes bi-directional wireless broadcasting capability in both *Join Query* sender and *Join Table* sender. Particularly, multicast subscribers rely on passive acknowledgments from its next hop neighbors. But this may not be a realistic assumption in real world environments as hardware broadcast capabilities vary significantly. In addition, problems such as hidden terminal problem are not taken into consideration.

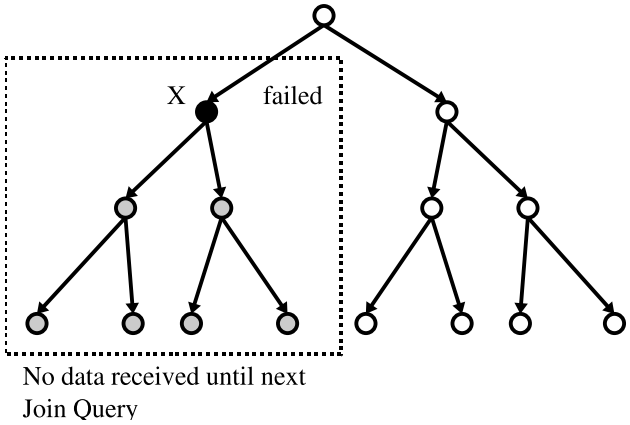

Figure 3.1. ODMRP failed node

3.2. Motivating factors

It is believed that tree based multicasting protocols may not be the best approach for mobile multicasting [7]. In fact, ODMRP shows significant packet delivery ratios with

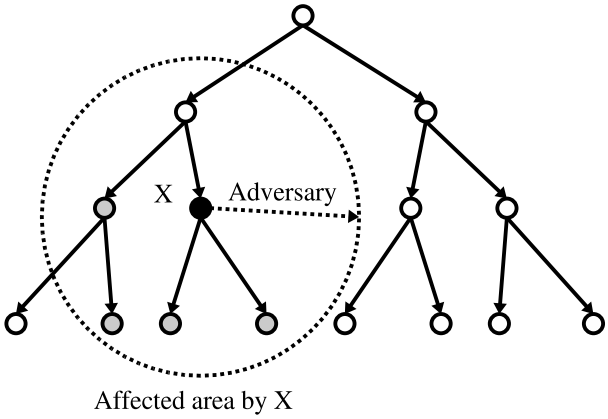

Figure 3.2. ODMRP adversary

a limited control messaging overhead. Apart from that ODMRP's non-reliance on a single networking protocol, its simplicity, robustness and scalability make it specially appealling for real world applications. On the other hand, ODMRP shows several drawbacks, which were discussed in detail. Addressing these drawbacks could further improve the resiliency, efficiency and scalability of ODMRP and would make it a good contender for next generation mobile multicasting protocols.

Chapter 4

Resilient On-Demand Multicast Protocol (RODMRP)

"'Curiouser and curiouser!' cried Alice" [1]

This chapter presents the theoretical background of RODMRP.

4.1. The Framework

As explained in Section 3.1 ODMRP's main route discovery/recovery mechanism rely on *Join Query* packets and frequent route refreshing. Despite its advantages, which were discussed in Section 2.4, this simple route recovery/discovery mechanism can severely impact the efficiency of the protocol. Alternatively, non-forwarding nodes can be exploited to create redundant paths for multicast data transmission. But redundant forwarding itself can become an overhead. RODMRP uses probabilistic forwarding to contain the redundant traffic generated by the non-forwarding nodes. The *Non forwarding* nodes *promiscuously listen* to ongoing *Join Query* and *Join Table* packets and probabilistically elect to establish a redundant forwarding path. Two important questions arise: (1) How

[1] *Alice in Wonderland and Through the Looking Glass* - Lewis Carroll

does a node probabilistically decide on which packets are to be redundantly forwarded? (2) Does the protocol ensure that an additional forwarding node exists with high probability? These questions are addressed in following sections.

4.1.1. Definitions

- *Join Query*

  Refers to ODMRP *Join Query* packet.

- *Join Table*

  Refers to ODMRP *Join Table* packet.

- *Forwarding nodes*

  A *Forwarding node* is an intermediate node in the forwarding path established by the original ODMRP *Join Query*/*Join Table* cycle and other nodes, which are not in forwarding paths are referred to as *Non-forwarding nodes*. The *Non-forwarding nodes* are further categorized as *Active non-forwarding* and *Passive non-forwarding* nodes.

- *Active non-forwarding nodes*

  An *Active non-forwarding* node denotes a non-forwarding node, which is within the broadcast range of a *Forwarding node*. This enables *Active non forwarding* nodes to promiscuously listen to ongoing data transmissions.

- *Passive non-forwarding nodes*

  A *Passive non-forwarding* node denotes a non-forwarding node, which is not within the broadcast range of a *Forwarding node*.

- *Promiscuous listening*

  Active non-forwarding nodes eavesdrop on ongoing data transmissions to observe

data transmission pattern of *Forwarding nodes*. Based on their observations, these nodes develop soft states.

- *Passive forwarding*

  Based on the soft state developed by *Promiscuous listening*, the *Active non-forwarding* nodes forward multicast data to overcome perceived lack of forwarding routes. Thus, they establish redundant routes.

4.1.2. Protocol overview

Despite its simplicity and efficiency, ODMRP shows several drawbacks as discussed in Section 3.1. In addition, the protocol is also vulnerable to more sophisticated attacks, i.e., denial-of-service attacks. While frequent route refreshing could address these issues to some extent, frequent *Join Query* packets would waste network bandwidth, drain network/node resources and would result in high packet collisions. Thus, frequent route refreshing will significantly degrade the efficiency of the protocol. In designing the following improvements, every attempt was made to retain the original characteristics of ODMRP. The rational behind this protocol is to exploit the mesh based structure of the network not only for route setup but also for actual data forwarding. But this would incur additional control messaging for additional routing path setup, which would in tern be against the low control messaging overhead of ODMRP. Instead of additional control packets, RODMRP extends the ODMRP's soft state concept to establish redundant forwarding paths.

In order to form redundant routing paths, RODMRP further categorizes non-forwarding nodes as *Active non-forwarding* and *Passive non-forwarding* nodes (as defined in Section 4.1.1). *Active non-forwarding* nodes are within the range of a *Forwarding* node while *Passive non-forwarding* nodes are not. While *Forwarding* nodes form original routing paths through *Join Query/Join Table* packets, *Active non-forwarding* nodes develop

soft states based on *Promiscuous listening* to ongoing control packets. Based on observed control messaging *Active non-forwarding* nodes compute multicast data forwarding probability as illustrated below. Based on this probability computation, *Active non-forwarding* nodes establish redundant routes by *Passive forwarding* of multicast data. In this manner, a set of *Active non-forwarding* nodes elect to complement data forwarding in order to overcome perceived failure of forwarding nodes.

The forwarding tree construction algorithm in RODMRP is identical to the one found in ODMRP. A source (node A in Figure 4.1) initiates a multicast session by broadcasting a *Join Query* packet. Intermediate nodes relay the *Join Query* packet, and update their *Routing tables* with prospective parents for the multicast source as illustrated in Section 2.2. Receivers reply with a *Join Table* packet back to the source via reverse forwarding paths. *Active non-forwarding* nodes listen to on going *Join Query* and *Join Table* packets from multicast source *A* and initially assign forwarding probability P_A as,

$$P_A = \left(\frac{1}{n} \right) \tag{4.1}$$

where *n* is the number of unique *Join Query* packets observed. Upon receiving subsequent *Join Table* packets, the *Active non-forwarding* nodes increment forwarding probability by a fraction as,

$$P_A = (P_A * m) \tag{4.2}$$

where *m* is the number of observed unique *Join Table* packets. On receiving multicast data, the *Active non-forwarding* nodes evaluate forwarding probability as,

$$P_A >= \frac{k}{f(t)} \tag{4.3}$$

where k is a constant and $f(t)$ is an attribute of the forwarding path, we consider k=1 and $f(t)$ as the least number of hops towards the multicast source along a forwarding path.

As in ODMRP, the multicast source periodically broadcasts *Join Query* packets to refresh current routes or discover new/better forwarding paths. In addition, this allows new nodes to subscribe to the current multicast session. The Receivers periodically send *Join Table* packets upstream towards the source to maintain forwarding routes. When a new route is uncovered, the receiver of the new route redirects its *Join Table* packet to the new parent. To compute the discrepancy more precisely, *Active non-forwarding* nodes reset their broadcasting packet observations whenever they receive a *Join Query* from the source.

4.1.3. RODMRP *Join Query*

RODMRP *Join Query* is identical to the ODMRP *Join Query*.

4.1.4. RODMRP *Join Table*

RODMRP *Join Table* is identical to the ODMRP *Join Table*.

4.1.5. RODMRP Routing table

In RODMRP, active non-forwarding nodes maintain following additional fields in their routing table.

- Number of *Join Query* observations

  The number of unique *Join Query* packets observed from a particular multicast source.

- Number of *Join Table* observations

  The number of unique *Join Table* packets subscribing to a given multicast source.

- Passive forwarding probability

  On expiration of its observation period or on receiving a date packet from a multicast source following its route discovery, the active non-forwarding nodes compute the passive forwarding probability for the particular sender as described in Section 4.1.2.

The active non-forwarding nodes maintain above extended routing information for each multicast source. On receiving a *Join Query* from the multicast source, the active non-forwarding nodes refresh or reset above fields. Otherwise, routing information is purged on a route refresh expiration timer, which usually indicates the termination of the multicast session.

4.1.6. RODMRP Passive forwarding

Upon receiving multicast data from a multicast sender, active non-forwarding nodes compare the passive forwarding probability and least hop path towards the sender as described in Section 4.1.2.

4.1.7. RODMRP Protocol illustration

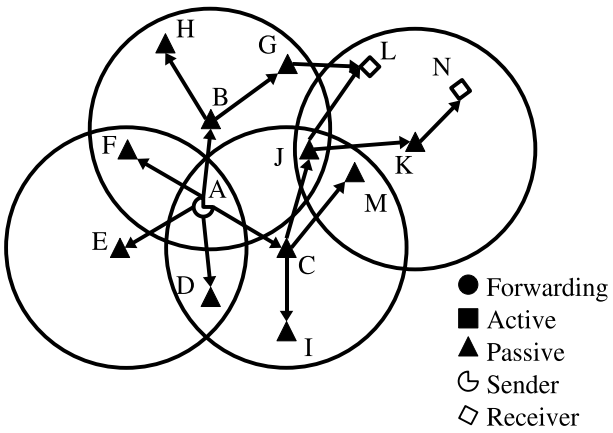

Figure 4.1. RODMRP *Join Query*

An example of the enhanced protocol is illustrated in Figures 4.2 and 4.3. By definition, all nodes are initially *Passive non-forwarding* nodes. Node A broadcasts a *Join Query* packet as the source, and all the surrounding nodes (B, C, D, E, and F) rebroadcast the received *Join Query* packet. In the figure, these rebroadcasting packets open two possible forwarding paths from the source to receivers L and N. L selects the path from J, while N selects the path from K. Each receiver replies with a *Join Table* packet to its parent reinforcing the selected forwarding path. While the *Join Table* packets are forwarded towards the source, following *Active non-forwarding* nodes receive *Join Table* from following nodes,

- B - J
- G - L and J
- M - J, K, L and C

Above *Active non-forwarding* nodes compute their forwarding probability (Table 4.1) based on above observations. Based on the probability computation M elects to forward packets redundantly. Thus, node M creates a redundant forwarding path as illustrated in Figure 4.3, which enables it to overcome the failure of node J.

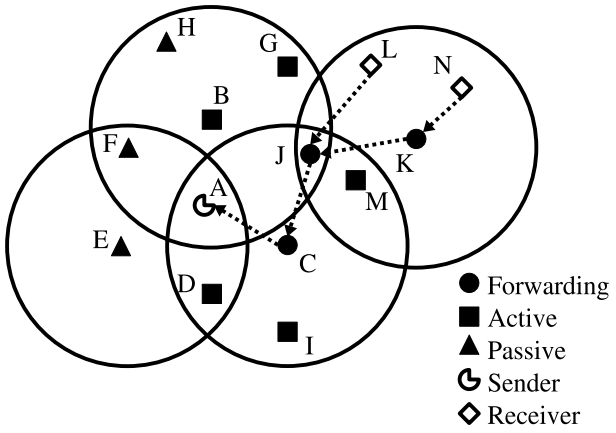

Figure 4.2. RODMRP *Join Table*

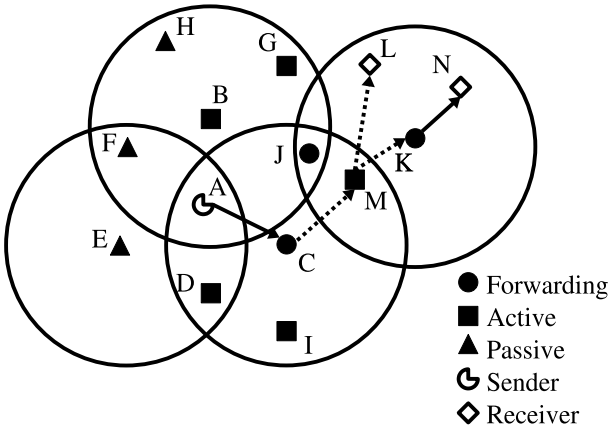

Figure 4.3. RODMRP data forwarding

$Node$	n	m	f	P_A
A (source)	-	-	-	-
B (active non-forwarding)	4	1	2	1/4
C (forwarding)	-	-	-	-
D (active non-forwarding)	4	1	1	1/4
E (passive non-forwarding)	-	-	-	-
F (passive non-forwarding)	-	-	-	-
G (active non-forwarding)	4	2	2	1/2
H (passive non-forwarding)	-	-	-	-
I (active non-forwarding)	2	1	1	1/2
J (forwarding)	-	-	-	-
K (forwarding)	-	-	-	-
L (receiver)	-	-	-	-
M (active non-forwarding)	4	4	1	1
N (receiver)	-	-	-	-

Table 4.1. Computing the packet forwarding probability in RODMRP

4.1.8. Selecting the furthest parent

In order to improve the number of non-forwarding nodes between parent and child nodes, RODMRP selects the furthest forwarding nodes as forwarding nodes. This forwarding node selection packs more non-forwarding nodes, which could later become active non-forwarding nodes and facilitate redundant rebroadcasting. Figure 4.4 illus-

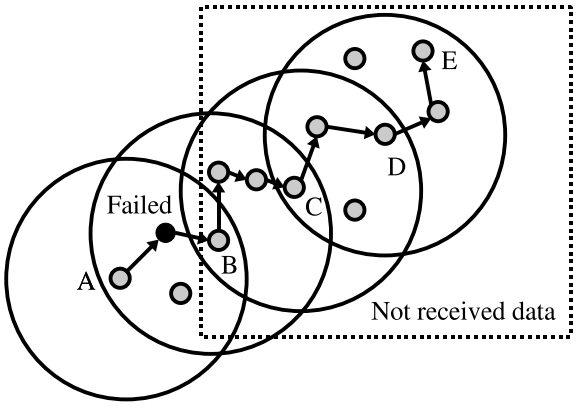

Figure 4.4. RODMRP selecting immediate parent

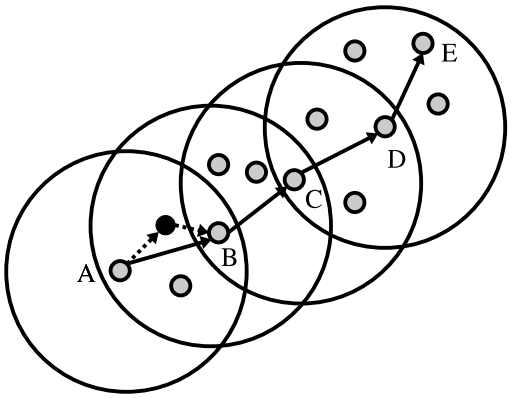

Figure 4.5. RODMRP selecting furthest parent

trates selection of the immediate node as a parent node. As illustrated, this node selection method does not include additional nodes in between the parent and child nodes. On the otherhand, Figure 4.5 packs more additional nodes. To facilitate this, we use the least number of hop path to select the immediate parent node. Ideally, a location aware algorithm would be more suitable. But location awareness requires additional hardware (i.e., GPS devices) or additional processing (i.e., signal strength). We believe least hop path is a simple but effective method to select forwarding parent nodes.

Chapter 5

Implementation: RODMRP implementation in ns2

> "'Where do you come from?' said the Red Queen. 'And where are you going? Look up, speak nicely, and don't twiddle your fingers all the time.'
>
> Alice attended to all these directions, and explained, as well as she could, that she had lost her way." [1]

This chapter presents the implementation details of RODMRP in ns2 [14] network simulator.

5.1. ns2

Network Simulator (ns2) [14] is a discrete event simulator for networking research. It is an ongoing open source project, which is being developed with the contributions of various network researchers. The open nature of the project enables researchers to easily access and modify the implementation of protocols. In addition, the open source development model encourages the researchers to develop on previous work done in their research area. Following object-oriented concepts, the low level simulation engine and protocols

[1] *Alice in Wonderland and Through the Looking Glass* - Lewis Carroll

are developed in C++, while tcl/tk based high level scripting layer makes it possible to change simulation scenarios dynamically. Although ns2 has in depth support for wired protocols such as TCP, it does not have built in support for common wireless protocols. The Carnegie Mellon University's Monarch Research Group [9] has developed several wireless protocols, including ODMRP for ns2 platform. Implementation of RODMRP discussed below is based on this reference implementation of ODMRP.

As discussed above ns2 consists of two layers, C++ lower layer implements the core protocols while OTcl upper layer provides an interpreter for Tcl bindings, which makes it possible to change simulation parameters without C++ code changes and recompilation of the simulator. Although this two layer model makes the simulator highly efficient and flexible, it also makes protocol development much more challenging. The protocol developers not only have to implement the protocols in C++, but also have to implement OTcl interfaces for tcl scripting. In addition, as ns2 is an ongoing project it is difficult to find detailed documentation on protocol development.

5.2. Implementation of ODMRP

Following files define the corresponding classes for the implementation of ODMRP. Figure 5.1 illustrates a class diagram of the protocol implementation.

- $defs.h$ - Defines different protocol parameters
- $hdr\_odmrp.h/cc$ - Defines ODMRP packet header (hdr_odmrp class)
- $odmrpagent.h/cc$ - Defines ODMRP node (ODMRPAgent class)
- $msg\_cache.h/cc$ - Defines packet cache (MSGCache class)
- $mem\_table.h/cc$ - Defines ODMRP member table (McastMemberTable class)
- $packet\_buffer.h/cc$ - Defines packet buffer (PacketBuffer class)

- $jq\_table.h/cc$ - Defines *Join Query* look up table (JQTable class)
- $jr\_table.h/cc$ - Defines *Join Table* look up table (JRTable class)
- $join\_query\_timer.h/cc$ - Defines *Join query* timer (JoinQueryTimer class)
- $join\_reply\_timer.h/cc$ - Defines *Join Table* timer (JoinReplyTimer class)

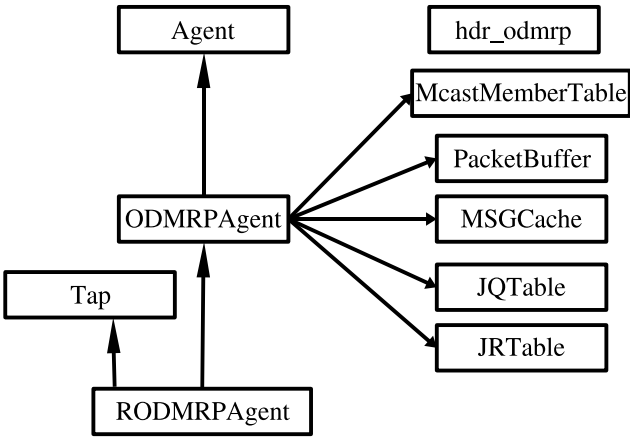

Figure 5.1. Class diagram

5.2.1. ODMRPAgent

In ns2 simulator, endpoints where network packets are consumed or constructed are called *Agents*. Therefore ODMRPAgent class defines the network endpoints for ODMRP protocol. The ODMRPAgent inherits from generic Agent class and ODMRPAgent maintains its routing table as a set of hash tables (maps), which are defined in *JQTable* and *JRTable* classes. *JoinQueryTimer* and *JoinReplyTimer* classes define the route refresh timers and route expiration timers.

5.2.2. *ODMRP packet header*

ODMRP packet header is defined as a wrapper for both *Join Query* and *Join Table* packets. This design avoids addition of multiple packet types for the protocol imple-

mentation. In addition, it follows ns2's programming convention, where each protocol is defined with a single packet header. ODMRP packet header is defined in hdr_odmrp class.

```
class hdr_odmrp {
private:
    int valid_ack_;
    odmrpaddr_t mcastgroup_addr_;
    odmrpaddr_t prev_hop_addr_;

    struct odmrp_join_query join_query_;
    struct odmrp_join_reply join_reply_;
public:
    int valid_;
    static int offset_;
}
```

5.2.3. *Join Query* packet

Join Query packet is implemented as an embedded packet, which could be embedded in ODMRP header. *Join Query* packet is defined as follows,

```
struct odmrp_join_query {
    int  valid_;
    int  seqno_;
    int  hop_count_;
    int  piggyback_;
};
```

5.2.4. *Join Table* packet

Similarly *Join Table* packet is defined as a submessage, which could be embedded in ODMRP header. *Join Table* is defined as,

```
struct odmrp_join_reply {
    int  valid_;
    int  seqno_;
    int  count_;
    struct prev_hop_pairs pairs_[OD_MAX_NUM_PREV_HOP_PAIRS];
};
```

5.2.5. TCL hooks

In order to create instances of ODMRP, tcl hooks are created as follows. This enables creation of ODMRP nodes in tcl scripts.

```
static class ODMRPAgentClass : public TclClass {
public:
  ODMRPAgentClass() : TclClass("Agent/ODMRPAgent") {}
  TclObject* create(int, const char*const*) {
    return (new ODMRPAgent);
  }
} class_ODMRPAgent;
```

5.2.6. TCL commands

Agents implement *command* method that defines different tcl commands, which are supported by the particular agent. ODMRPAgents support following commands:

- *startodmrp* - Initiates execution of ODMRP agent

- *reset* - Resets execution of ODMRP agent
- *print-membership* - Prints current agents membership table
- *mcast-src* - Checks whether current node is a multicast source
- *mcast-sink* - Checks whether current node is a multicast receiver
- *ip-addr* - Returns agent's IP address
- *mac-addr* - Returns agent's MAC address
- *join-group* - Advices agent to subscribe to a multicast group
- *leave-group* - Advices agent to unsubscribe from a multicast group
- *log-target* - Creates a log trace

5.2.7. Timers

JoinQueryTimer and *JoinReplyTimer* classes implement the timers for route refreshing and routing table entry expiration.

5.2.8. Packet handling

ODMRPAgent defines *recv*, *sendJoinQuery* and *sendJoinReply* methods for packet handling. Following code snippet illustrate the implementation of *recv* method for packet receipt.

```
void ODMRPAgent::recv(Packet* packet, Handler*) {
    odmrpaddr_t src, dest;
    u_int16_t sport, dport;
```

```
hdr_ip  *iph =  hdr_ip::access(packet);
hdr_cmn *cmh =  hdr_cmn::access(packet);

assert(cmh->size() >= 0);
assert(logtarget != 0);

dest = iph->daddr(); dport = iph->dport();
src  = iph->saddr(); sport = iph->sport();

if (src == INITIALIZATION_VALUE) {
    iph->saddr() = src = net_id;
}

if (!McastAddress(dest) ||
            packet_cache.lookup(src, cmh->uid())) {
    Packet::free(packet);
    packet = 0;
    return;
} else if (iph->ttl() <= 1) {
    drop(packet, DROP_RTR_TTL);
    packet = 0;
    return;
}

iph->ttl()--;
```

```
    if (iph->sport() != RT_PORT) {
        packet_cache.enterInCache(iph->saddr(), cmh->uid());
    }

    if (src == net_id && cmh->next_hop() ==
                (odmrpaddr_t)MAC_BROADCAST) {
        Packet::free(packet);
        packet = 0;
    } else if (src == net_id) {
        handleDataPacketSend(packet, OD_DATA_DELAY);
    } else if (dest == net_id ||
                dest == (odmrpaddr_t)IP_BROADCAST ||
                cmh->next_hop() ==
                (odmrpaddr_t)MAC_BROADCAST) {
        handlePacketReceipt(packet);
    } else {
        assert(0);
    }

    packet = 0;
}
```

5.2.9. Tracing ODMRPAgents

In order to trace simulation execution, ns2 defines *Trace objects*, which log trace events (i.e., sending/receiving packets). CMUTrace extends this functionality to wireless protocol tracing. Following code snippet implements tracing functionality for ODMRP.

```
void CMUTrace::format_odmrp(Packet *p, int offset) {
    hdr_odmrp *mh = hdr_odmrp::access(p);
    sprintf(pt_->buffer() + offset,
        "-%d- %d [%d %d %d %d] [%d %d %d %d] (%d, %d)",
        mh->mcastgroup_addr(),
        mh->prev_hop_addr(),
        mh->join_query(),
        mh->join_query_seqno(),
        mh->join_query_hopcount(),
        mh->join_query_pb(),
        mh->join_reply(),
        mh->join_reply_seqno(),
        mh->join_reply_count(),
        mh->join_reply_ack(),
        mh->join_reply_pairs_src_addr(0),
        mh->join_reply_pairs_prev_hop_addr(0));
}
```

5.3. Implementation of RODMRP extensions

5.3.1. RODMRPAgent

RODMRPAgent class inherits from ODMRPAgent class. In addition, RODMRPAgent inherits from the Tap class. This enables RODMRPAgent to implement the *Promiscuous listening* behavior of RODMRP protocol. *rodmrpagent.h/cc* files define the implementation details of RODMRPAgent.

5.3.2. TCL hooks

As ODMRPAgent, RODMRPAgent class also defines tcl hooks for instantiation of RODMRPAgent instances.

```
static class RODMRPAgentClass : public TclClass {
public:
    RODMRPAgentClass() : TclClass("Agent/RODMRPAgent") {}
    TclObject* create(int, const char*const*) {
            return (new RODMRPAgent);
    }
} class_RODMRPAgent;
```

5.3.3. Data structures

In addition to the above mentioned data structures, RODMRPAgent maintains following data structures.

- *JR_listener* - A hash map, which maps the number of observed *Join Query* rebroadcasts with the IP address of the multicast source.
- *JQ_listener* - A hash map, which maps the number of observed *Join Table* packets with the IP address of the multicast source.
- *fwd_prob* - A hash map, which maps the computed forwarding probability with the IP address of the multicast source. This map is used in subsequent redundant data forwarding.

5.3.4. Passive forwarding

Promiscuous listening to ongoing *Join Query* and *Join Table* packets enable non-forwarding nodes to compute passive forwarding probability as,

```
if (jq_listener.find(mcast_grp) != jq_listener.end()) {
        double prob = 1.0 / (double)(jq_listener[mcast_grp]);
        if (fwd_prob.find(mcast_grp) == fwd_prob.end())
                fwd_prob[mcast_grp] = prob;
        else
                fwd_prob[mcast_grp] += prob;
}
```

Tapping onto ongoing data transmissions enable *Active non-forwarding* nodes to receive multicast data. On receiving data packets *Active non-forwarding* nodes forward packets as follows,

```
void RODMRPAgent::eval(nsaddr_t mcast_grp) {
    if (jr_listener[mcast_grp] <= data_listener[mcast_grp])
        return;

    double dtrigger = 1 - (double(data_listner[mcast_grp]) /
                          double(jr_listener[mcast_grp]));
    if (fwd_prob.find(mcast_grp) != fwd_prob.end()) {
        if (fwd_prob[mcast_grp] >= dtrigger) {
            if (!pkt_cache[mcast_grp].empty()) {
                handlePacketSend(
                      pkt_cache[mcast_grp].back()->copy(),
                      OD_CP_JITTER);
            }
            fprintf(stderr,
                    "%d\t:eval grp %d, prob %.2f\n",
                    net_id, mcast_grp,
```

```
                    fwd_prob[mcast_grp]);
        }
    }

    data_listener[mcast_grp] = 0;
    list<Packet *>::iterator ite;
    for (ite = pkt_cache[mcast_grp].begin();
         ite != pkt_cache[mcast_grp].end();
         ite++)
        Packet::free(*ite);
    pkt_cache[mcast_grp].clear();
}
```

Chapter 6

Results

> "'Then you should say what you mean,' the March Hare went on.
>
> 'I do,' Alice hastily replied; 'at least–at least I mean what I say–that's the same thing, you know.'
>
> 'Not the same thing a bit!' said the Hatter. 'You might just as well say that "I see what I eat" is the same thing as "I eat what I see"!'" [1]

This chapter presents the simulation results of the RODMRP protocol. Simulations were performed to measure the benefits and drawbacks of the protocol.

6.1. Simulation environment

The simulations were performed in ns2.1b6 [14] network simulator. ns2 is a leading discrete event simulator for network protocol development and analysis.

6.1.1. Scenario

The simulations consist of four test topologies with 25, 50, 75 and 100 mobile nodes in an area of 1200 m x 800 m. Each test scenario simulates a single multicast group for 10 minutes, where a randomly selected multicast source and 25% of nodes participate

[1] *Alice in Wonderland and Through the Looking Glass* - Lewis Carroll

in a multicast session. In addition, random node failures were induced to simulate adversary nodes and forwarding node failures. Each simulation was repeated 5 times with 5 different random seeds for ODMRP and RODMRP. Following results were obtained by normalising simulation results. It is believed that these configurations are close to real wireless networks with high node mobility. Table 6.1 sumarises different simulation parameters.

6.1.2. Movement model

The random waypoint mobility scenario generator in ns2 was used to produce node movement models with maximum speeds of 10, 15, 20, 25, 30, 35 and 40 m/s, and an average pause time of 1 second.

6.1.3. Communication model

The simulation radio model was based on Lucent WaveLAN IEEE 802.11 with a 2Mbps transmission rate and a transmission range of 250 m. The IEEE 802.11 wireless LAN Distributed Coordination Function (DCF) was used as the link layer model. The Carnegie Mellon University's Monarch Research Group's [9] multicast communication scenario generator was used to create multicast communication models.

6.2. Performance metrics

Following metrics were used to evaluate the performance efficiency of the protocol.

- **Packet delivery ratio:** The ratio of packets received against the total number of packets sent.
- **Control packet overhead:** The ratio of control packets against the total number of packets.

Parameter	*Value*
Transmission range	250 m
Simulation time	10 min
Topology size	1200x800 m
Node failure model	Random failure
Number of mobile nodes	25, 50, 75 and 100
Number of sources	1
Number of receivers	25% of total nodes
Traffic type	Constat bit rate
Packet size	512 bytes
Pause time	1 s
Maximum speed	10, 15, 20, 25, 30, 35, 40 m/s
Mobility model	Random waypoint

Table 6.1. A summary of simulation parameters

- **Mobility and packet delivery ratio:** The average packet delivery ratio for different mobility scenarios.

- **Packet forwarding overhead per node:** The ratio of total data packet forwardings against total data packet initiations.

6.3. Simulation results

6.3.1. Packet delivery ratio

Figure 6.1 presents a comparison of the packet delivery ratio of RODMRP and ODMRP when the number of nodes changes from 25 to 100. The horizontal axis denotes the number of nodes and the vertical axis indicates packet delivery ratio in percentage. In the experiment, RODMRP delivered more packets than ODMRP in the range of 2% to 5% in case of failure of ODMRP forwarding nodes. The figure also shows that the delivery ratio for both RODMRP and ODMRP increases as the number of nodes increases up to 50. The more nodes participate, the more active non-forwarding nodes are available. As the number of nodes exceeds 50, the delivery ratio stabilizes since newly added *Ac-*

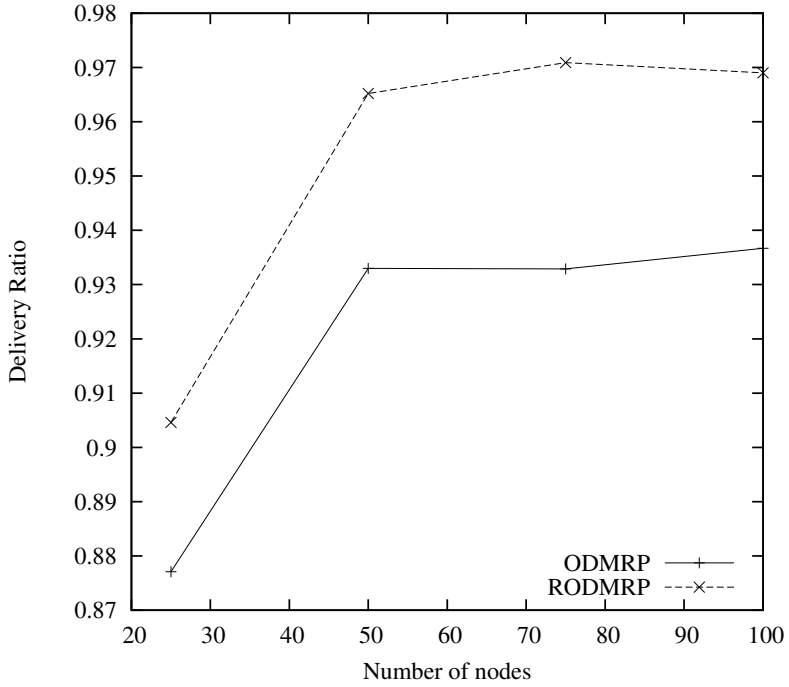

Figure 6.1. Packet delivery ratio

tive non-forwarding nodes provide redundant paths. Overall, these results show that the probabilistic rebroadcasting mechanism in RODMRP increases the probability of packet delivery, especially when the network fails.

6.3.2. Control packet overhead

One of the concerns about the RODMRP performance is the additional data overheads incurred by rebroadcasting. Figure 6.2 analyzes the control packet overhead for different node densities (from 25 to 100). The graph shows that RODMRP introduces virtually the same amount of control packets (or little additional data overheads) for all the number of nodes ranges. For example, when 80 nodes run the protocols, RODMRP incurs less than 1% additional control data overheads to achieve nearly the 97% packet delivery ratio. For most of the network sizes, RODMRP requires control overheads less than 1% to attain packet delivery ratio of over 90%.

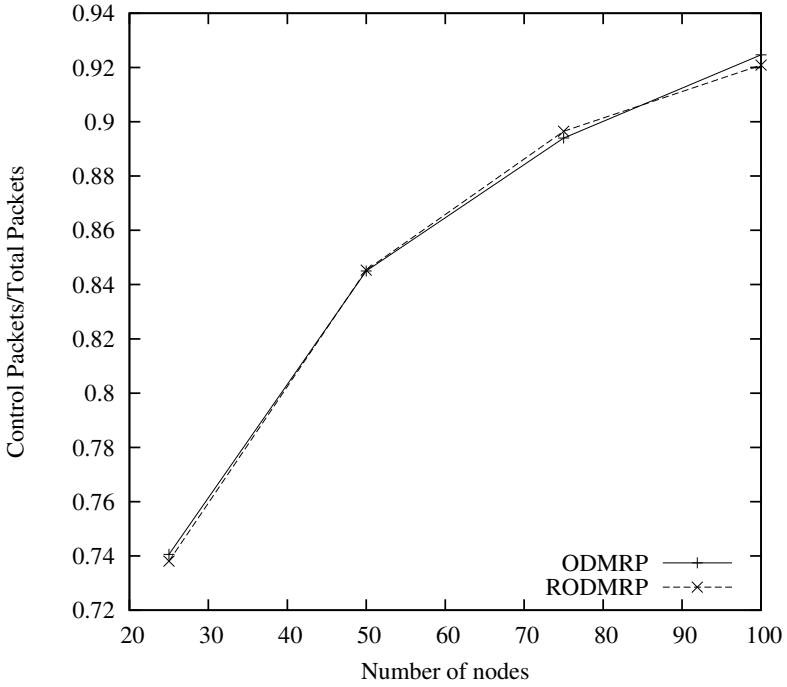

Figure 6.2. Control packet overhead

6.3.3. Mobility and packet delivery ratio

RODMRP can reliably deliver more packets to destinations even when nodes are highly mobile. Figure 6.3 shows average packet delivery ratios of RODMRP in comparison to average packet delivery ratios of ODMRP. When the speed of a mobile node is low (10-15 m/s), the delivery ratio of RODMRP improves approximately 2.5% against the delivery ratio of ODMRP. As mobility becomes higher, the delivery ratio of RODMRP shows a significant increase compared to ODMRP with a maximum of 4.5% Both protocols show similar curves in the figure as RODMRP is designed based on ODMRP.

6.3.4. Packet forwarding overhead per node

Figure 6.4 illustrates the packet forwarding overhead per node. As the number of nodes increases in ODMRP, packet forwarding overhead per node decreases since no additional forwarding nodes are needed. In contrast, perceived high density of multicast receivers

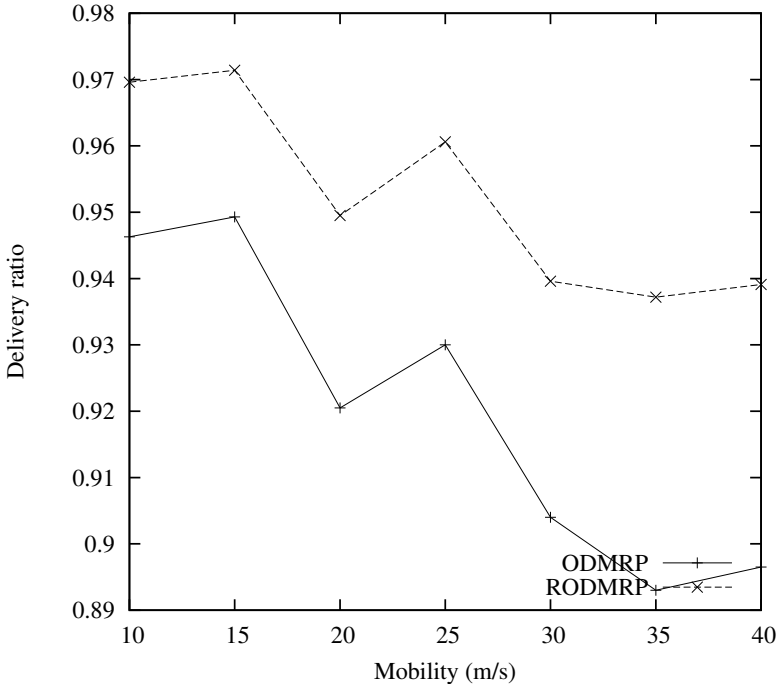

Figure 6.3. Mobility and packet delivery ratio

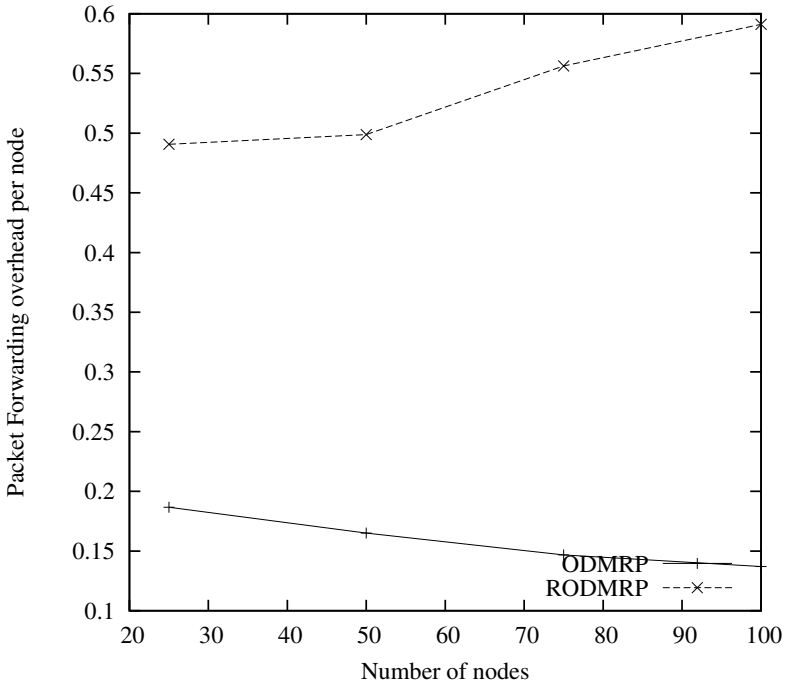

Figure 6.4. Packet forwarding overhead per node

encourages RODMRP nodes to engage in *Passive forwarding* and to form additional redundant forwarding paths. Thus, this results in contributing to the increased per node forwarding overhead observed in above figure. In the worst case, the packet forwarding overhead of RODMRP amounts to four times the corresponding ODMRP packet forwarding overhead.

Chapter 7

Conclusion

> "'Ah, well! It means much the same thing,' said the Duchess, digging her sharp little chin into Alice's shoulder as she added, 'and the moral of that is–"Take care of the sense, and the sounds will take care of themselves."'" [1]

This chapter presents the conclusion and future enhancements to RODMRP.

7.1. Conclusion

Previous chapters presented the design and development of RODMRP: a multicast protocol for wireless ad hoc networks. RODMRP aims to be resilient to network or node failures and provides uninterrupted multicast service for live streaming data. For a receiver, nodes other than its direct parent redundantly broadcast data so that the receiver can receive data even when its parent node fails. In RODMRP, a non-forwarding node determines redundant broadcasting intelligently and efficiently. In addition, the protocol does not add additional control messaging to achieve this improved efficiency. The simulation results indicate that the protocol mitigates the interruption of data delivery considerably while building an efficient multicast tree.

[1] *Alice in Wonderland and Through the Looking Glass* - Lewis Carroll

7.2. Future work

However, there is still room for improvement. In addition to observing Join Query and Join Table packets, active non-forwarding nodes can also observe passive forwardings of other active non-forwarding nodes. This could result in a significant reduction in the number of passive packet forwardings without any major impact on the above performance characteristics. We are in the process of integrating the above observation to our routing protocol. In addition, we intend to incorporate enhancements for reliable multicasting.

We hope to conduct large-scale experiments in more realistic environments and to compare performance of RODMRP with other multicast protocols. In such experiments, we intend to test RODMRP by running an actual video streaming application.

Bibliography

[1] C. Perkins. Ad Hoc On Demand Distance Vector (AODV) Routing. In *Ad Hoc On Demand Distance Vector (AODV) Routing IETF, Internet Draft, draft-ietf-manet-aodv-00.txt*, November 1997. http://citeseer.ist.psu.edu/article/perkins99ad.html.

[2] S. Ho Bae, S. Lee, W. Su, and M. Gerla. The Design, Implementation, and Performance Evaluation of the On-Demand Multicast Routing Protocol in Multihop Wireless Networks. In *IEEE Network Magazine*, volume 14, pages 70–77, January 2000.

[3] IDC Press Release. Record-Setting Fourth Quarter Shipments Propel Worldwide Mobile Phones Past OneBillion Unit Mark. Retrieved January 27, 2007, from http://www.idc.com /getdoc.jsp?containerId=pr2007_01_17_133455.

[4] L. Klos and G. Richard III. Reliable Group Communication in an Ad Hoc Network. In *Proc. of the IEEE International Conference on Local Computer Networks (LCN)*, 2002.

[5] T. J. Kwon and M. Gerla. Efficient Flooding with Passive Clustering (PC) in Ad Hoc Networks. In *SIGCOMM Computer Communication Review*, volume 32, pages 44–56, August-September 2002.

[6] S. Lee, M. Gerla, and C. Chain. On-Demand Multicast Routing Protocol - (ODMRP). In *Proc. of the IEEE Wireless Communication and Networking Conference (WCNC)*, September 1999.

[7] S. Lee, W. Su, J. Hsu, M. Gerla, and R. Bargrodia. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *Proc. of the IEEE INFOCOM*, March 2000.

[8] S. J. Lee, W. Su, and M. Gerla. On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Network IETF Internet Draft. July 2000.

[9] Monarch Research Group. Rice Monarch Project Software Distributions. Retrieved March 01, 2006, from http://www.monarch.cs.cmu.edu/software.html.

[10] S. Y. Oh, J.-S. Park, and M. Gerla. E-ODMRP: Enhanced ODMRP with Motion Adaptive Refresh. In *Proc. of the International Symposium on Wireless Communication Systems (ISWCS)*, September 2005.

[11] A. Sobeih, H. Baraka, and A. Fahmy. On the Reliability of ODMRP in Mobile Ad Hoc Networks. In *Proc. of IEEE International Performance, Computing, and Communications Conference (IEEE IPCCC)*, April 2004.

[12] K. Tang and M. Gerla. Reliable Multicast of the On-Demand Multicast Routing Protocol. In *Proc. of the SCI*, July 2001.

[13] The Consumer Electronics Association (CEA). CEA Forecasts Consumer Electronics Revenue Will Surpass $155 Billion in 2007. Retrieved January 06, 2007, from http://www.ce.org/Press/CurrentNews/press_release_detail.asp?id=11220.

[14] The VINT project. The network simulator (ns-2). Retrieved December 12, 2005, from http://www.isi.edu/nsnam/ns.

[15] Wikipedia. Moore's Law. Retrieved January 01, 2007, from http://en.wikipedia.org/wiki /Moore's_law.

[16] Y. Yi, M. Gerla, and T. J. Kwon. Efficient Flooding in Ad Hoc Networks: A Comparative Performance Study. In *Proc. of the IEEE International Conference on Communications (ICC)*, 2003.

[17] Y. Yi, M. Gerla, and T. J. Kwon. Efficient Flooding with Passive Clustering (PC) in Ad Hoc Networks. In *Proc. of the Second Annual Mediterranean Ad Hoc Networking Workshop (MedHocNet)*, 2003.

[18] Y. Zhao, Y. Xiang, L. Xu, and M. Shi. On-Demand Multicast Route Protocol with Multipoint Relay(ODMRP-MPR) in Mobile Ad-hoc Wireless Network. In *Proc. of the International Conference on Communications Technology (ICCT)*, April 2003.