12-20-2009

Reliable Multicast in Mobile Ad Hoc Wireless Networks

Lawrence Klos
University of New Orleans

Reliable Multicast in Mobile Ad Hoc Wireless Networks

A Dissertation

Submitted to the Graduate Faculty of the
University of New Orleans
In partial fulfillment of the
Requirements for the degree of

Doctor of Philosophy
in
Engineering and Applied Sciences

By

Lawrence Klos

B.S. Mathematics, University of Oregon, 1984
Master of Architecture, Harvard University, 1991
M.S. Computer Science, University of New Orleans 1999

December, 2009

Acknowledgements

I would like to thank Dr. Golden G. Richard III for initially accepting me as a research assistant, for his ongoing support and encouragement throughout my academic work at the University of New Orleans, and finally for his patience and encouragement in my dissertation research and projects. I would also like to thank Dr. Bourgeois, Dr. Abdelguerfi, Dr, Tu, and Dr. Jensen for their willingness to be on my committee and their patience.

I would also like to thank my wife Debbie, for her extreme fortitude, patience, understanding and love throughout the time of my dissertation research work. And finally, thanks to Robin, Alexandra and Daniel, for reminding me how magical it truly is that machines can talk to one another.

Table of Contents

List of Figures

Abstract

A mobile wireless ad hoc network (MANET) consists of a group of mobile nodes communicating wirelessly with no fixed infrastructure. Each node acts as source or receiver, and all play a role in path discovery and packet routing. MANETs are growing in popularity due to multiple usage models, ease of deployment and recent advances in hardware with which to implement them. MANETs are a natural environment for multicasting, or group communication, where one source transmits data packets through the network to multiple receivers. Proposed applications for MANET group communication ranges from personal network apps, impromptu small scale business meetings and gatherings, to conference, academic or sports complex presentations for large crowds reflect the wide range of conditions such a protocol must handle. Other applications such as covert military operations, search and rescue, disaster recovery and emergency response operations reflect the "mission critical" nature of many ad hoc applications. Reliable data delivery is important for all categories, but vital for this last one. It is a feature that a MANET group communication protocol must provide.

Routing protocols for MANETs are challenged with establishing and maintaining data routes through the network in the face of mobility, bandwidth constraints and power limitations. Multicast communication presents additional challenges to protocols. In this dissertation we study reliability in multicast MANET routing protocols. Several on-demand multicast protocols are discussed and their performance compared. Then a new reliability protocol, R-ODMRP is presented that runs on top of ODMRP, a well documented "best effort" protocol with high reliability. This protocol is evaluated against ODMRP in a standard network simulator, ns-2.

Next, reliable multicast MANET protocols are discussed and compared. We then present a second new protocol, Reyes, also a reliable on-demand multicast communication protocol. Reyes is implemented in the ns-2 simulator and compared

against the current standards for reliability, flooding and ODMRP. R-ODMRP is used as a comparison point as well. Performance results are comprehensively described for latency, bandwidth and reliable data delivery. The simulations show Reyes to greatly outperform the other protocols in terms of reliability, while also outperforming R-ODMRP in terms of latency and bandwidth overhead.

Keywords:
Ad hoc networks, reliable multicast, mobile networking, routing algorithm, transport protocol.

1 Introduction

1.1 Background and Motivation

1.1.1 Mobile Wireless Ad Hoc Networks

A mobile wireless ad hoc network (MANET) consists of a group of mobile nodes communicating wirelessly without the benefit of any fixed infrastructure. This type of communication is not like cellular, or even wireless LAN networks, which rely on a fixed infrastructure of centralized base stations or wired routers with antennas. In MANETs, nodes that spontaneously move to be within wireless transmit range of each other can begin to communicate by wirelessly transmitting packets back and forth, becoming 'networked' in an ad hoc manner. Each node can act as a data source or receiver at any time. Since mobile nodes have a limited wireless transmit range, data packets sent from a source often will travel hop by hop, forwarded by intermediate nodes within the range of each other along the paths to distant receivers. In MANET communication protocols, some or all nodes in the paths between sources and receivers usually play a role in both data path discovery and the ongoing routing of data packets through the network to receivers.

Initial MANET communication protocols focused on unicast communication, where a single node establishes a packet routing path to another single node, and the two nodes communicate. Later, one-to-many "multicast" communication protocols became a topic of research. In fact, due to the broadcast-type transmission that occurs in the wireless medium, MANETs are a natural environment for multicast applications. Currently proposed MANET multicast applications range from personal network apps, impromptu small scale gatherings and business meetings, to academic, conference and sports complex presentations involving large crowds. This range of applications reflects the wide range of conditions that an ad hoc multicast, or "group" communication protocol must be able to handle. Other proposed applications such as search and rescue team communications, covert military operations, disaster recovery and emergency response operations reflect the "mission critical" nature of many ad hoc group applications.

1.1.2 Reliability Issues in Multicast Routing Protocols

As opposed to wired networks, wireless nodes in an ad hoc network can dynamically join and leave a network at any time, either by choice or not. Since each node is potentially moving all the time, protocols must account for ongoing link breaks, as well as temporary formation of new links. The network topology is reconfigured frequently and constantly, and routing information can become stale quickly. Many MANET mission critical multicast applications operate in "sparse network" scenarios where a small number of nodes form the network, and network partitioning is frequent and potentially long lasting. Many factors can affect reliability in other application scenarios, however, such as physical conditions with a high level of natural interference, networks consisting of nodes with high mobility, nodes sending high traffic loads, or many nodes creating dense networks in small spaces, to name a few. Each of these scenarios exerts a different type of stress on reliable communication protocols, such as broken or ephemeral links, links overloaded with contention, or constantly changing network topologies requiring ongoing frequent protocol topology reconfiguration.

The ability of a reliable group communication protocol to deal with all these forces within a MANET is further complicated by the fact that multiple factors can be constantly in play concurrently across a network. A dense network could have spot conditions of sparseness and partitioning, networks with high mobility could have regions where nodes are stopped. In all networks, nodes may be out of range of all network communication for indeterminate and potentially long periods. For these reasons there is an ongoing need for general reliable multicast protocols that perform well in any network environment.

1.2 Dissertation Contributions

This dissertation focuses on the topic of reliability in MANET group communication protocol design. First we examine various existing "best effort" multicast routing protocols, in terms of the reliability they provide. Experimental protocol comparisons are presented along with a discussion of how overall protocol design

topologies affect reliability. A new reliable multicast routing protocol is introduced, with a performance evaluation comparing it to the top performing "best effort" protocol.

Following this, newer reliable multicast routing protocols are examined, along with a discussion of design issues involved in providing reliability. A second reliable multicast routing protocol is presented, along with a comprehensive performance analysis comparing it to the two existing protocols with documented best performance and the previously mentioned new reliable multicast protocol, under a wide range of network scenarios. Protocol comparisons are performed using a detailed network simulator which provides common ground for evaluating many aspects of the routing strategies, with repeatable results.

1.3 Related Work

1.3.1 MANET Multicast Protocols

Routing protocols designed specifically for mobile ad hoc wireless multicast communication first began to appear around 1999. Many protocols were proposed between 1999 and the early 2000's, and could be generally classified by the mechanisms they used to perform common tasks. First, all protocols usually defined some type of topology in order to construct packet routing paths. Common topologies and protocol classifications were *shared tree* (AMRoute [LTMB99], MAODV [RP99] and AMRIS [WT99]), *source tree* MCEDAR [SSB99], BEMRP [OKS99], MZRP [DSS01], ABAM [TGB00], DDM [JC01], WBM [DMM02b] and PLBM [SMM02]), *mesh* (ODMRP [LGC99], DCMP [DMM02a], FGMP-RA [CGZ98], NSMP [LK00] and CAMP [GM99]) or no topology (flooding and hyperflooding [OTV01]). Another common task protocols had to carry out was protocol initialization. Protocols could be classified according to this task as *source initiated* (MZRP, ABAM, AMRIS, ODMRP, DCMP and NSMP), or *receiver initiated* (BEMRP, DDM, WBM, PLBM, FGMP-RA and CAMP). A third common task was ongoing topology maintenance. Protocol classifications for this task were *hard state*, with routes continuously updated to repair newly broken links (BEMRP, ABAM, WBM, PLBM, AMRIS and CAMP), or *soft state*, with periodically refreshed

routes, with no repair mechanism operating between refreshes (MZRP, DDM, ODMRP, DCMP, FGMP-RA and NSMP). Other more specialized protocol classifications focus on specific protocol features. This group includes protocols that depend on physical locations of individual nodes, *energy efficient* protocols, protocols with *quality of service* guarantees, and protocols dependent on specific applications. There tended to be very few protocols in these categories, as most efforts went to developing more generic communication protocols.

Design of new generic multicast communication protocols slowed in the mid to later 2000's. Protocols developed during this time included AQM [KC05], a quality of service protocol, ExOR [BM05], a source initiated soft state protocol with no topology, PUMA [RG04], a receiver initiated, soft state, shared mesh protocol with core nodes, and SPBM [TFWME04], a physical location based protocol, and OBAMP [DB08], a shared tree, application specific protocol.

1.3.2 MANET Reliable Multicast Protocols

Recent years have also seen the creation of MANET multicast protocols that have focused on reliability as a topic of research. These protocols can generally be classified as *deterministic*, where an attempt is made to guarantee fully reliable data delivery (RMA [GSPS02], RALM [TOLG02], ReACT [ROLTG03] and Scribble [VE04]), or *probabilistic*, where the attempt is to provide a certain probability of reliability (AG [CRB01], RAPID [DFKS06], EraMobile [GO07], RDG [LEH03]). Several of these protocols are actually hybrids.

Another method of categorizing reliable MANET multicast protocols relates to the mechanism used to recognize missing packets. In *Sender Initiated* reliable multicast protocols, the source is responsible for detecting packet losses among receivers. Here, receivers are responsible for sending the ACKs for each packet, ,so if the source doesn't receive an ACK, it knows to initiate a resend (RMA). In *Receiver Initiated* reliable protocols, receiver nodes are responsible for detecting missed packets and notifying the sender or other nodes with some form of NACK message. Several reliable protocols use a

combination of the two techniques (RALM, ReACT), so this categorization is not as useful for reliable multicast protocols.

1.4 Dissertation Organization

Chapter 2 provides a review of previous work in MANET multicast protocol design, and chapter 3 introduces R-ODMRP, a reliable MANET multicast protocol implemented in the ns-2 network simulator. This chapter contains a performance evaluation of R-ODMRP, comparing it with ODMRP, a well known "best effort" protocol with documented top performance, in terms of packet delivery ratio, network bandwidth overhead and forwarding efficiency.

Chapter 4 presents a review of related work in reliable multicast protocol design for MANETS, and presents a discussion of design issues for reliable ad hoc multicast protocols. Chapter 5 presents Reyes, a new reliable ad hoc multicast protocol also implemented in the ns-2 simulator. This chapter includes an in depth performance evaluation, comparing Reyes to flooding, ODMRP and R-ODMRP in a wide variety of scenarios, including sparse, dense, high mobility and high data rate networks.. For each scenario, metrics are presented for reliability, overhead and delivery latency. Finally, chapter 6 discusses conclusions and future work.

2 Review of Related MANET Multicast Protocols

2.1 Introduction

Multicasting is a natural method of communication in ad hoc networks, given the broadcast nature of the wireless medium. Multicast communication in ad hoc networks has been a topic of research for over a decade, with some current protocols providing relatively high packet delivery ratios under various network conditions. Since ad hoc network communication is based on multihop packet transmits, each node must be able to act as a router, packet forwarder, potential source and potential receiver. In this environment battery power, bandwidth congestion and packet collisions are ongoing problems, and a common goal for these protocols is a reduction of the bandwidth required for ongoing communication in order to reduce congestion and achieve the highest packet delivery ratio possible. Bandwidth is consumed both by the overall percentage of network nodes needed to forward data packets, and by a protocol's operational requirements for various types of control packets.

The earliest ad hoc multicast protocols were not designed specifically for MANETs, but were adapted to them by modifying the existing wired internet multicast protocols DSDV [PB94], WRP [MG96], STAR [GS99]. These protocols were classified as *Proactive*, or *Table Driven*, in that current topology data was maintained in the form of tables at every node all the time. This topological information was kept current even when no sources were communicating, so that routes to destination nodes were always available to all nodes. Early studies showed that these protocols, often based on establishment of a routing tree reaching all receivers, did not perform well in mobile ad hoc wireless environments Maintaining connections in the face of ongoing random changes in topology required a large amount of control packets, which consumed much of the available bandwidth in the network. The fact that large amounts of bandwidth were consumed even when no communication was occurring made this early category of protocols impractical.

By the late 1990's and early 2000's ad hoc multicast communication became an area of active research. A new category of protocols, classified as *Reactive* or *On Demand*, were

designed specifically for MANETs. These protocols were designed so that no routing information was needed until the point when a source needed to communicate, removing a large amount of network overhead. When a source needed to communicate, a route discovery process would be initiated, and communication would begin once routes were learned. Flexible multicast group operations (group join and group leave mechanisms) were important factors in dealing with the unpredictable nature of link lifetimes. Topologies for these early On Demand protocols were either *tree based*, *mesh based* or had no defined topology (*Flooding based)*. Network simulation comparison studies [LSHGB00], [BMJHJ98], soon showed the advantages of mesh based over tree based On Demand protocols. While not as efficient in terms of network overhead, the presence of alternate routes to receivers that the mesh topologies provided were an antidote to node mobility, greatly increasing the overall network packet delivery ratio under many conditions.

Most protocols designed recently can be generally categorized based on the algorithms they use to implement common operations. For example, as mentioned previously, protocols often depend on establishing a specific topology for nodes in the network, in order to define routing channels to reduce network bandwidth requirements. Also, the algorithm controlling which node carries responsibilities for initiation of the protocol, for data resend requests, etc.. can be used to classify protocols. Next, the algorithm used to maintain the constructed topology can be used to classify protocols. For example ODMRP [LGC99], a robust multicast protocol with a relatively high packet delivery ratio, is a Mesh Based Source Initiated Soft State protocol.

While these categories can be used to describe the majority of ad hoc multicast protocol approaches today, many hybrid protocols exist. Attempts are made to capture the advantages of multiple categories, or remove the innate disadvantages of one category or the other. Also, there have been a number other approaches to protocol design. Approaches such as packet fragmentation, duplication and forward error correction, location assisted multicasting, and multicast protocols that are dependent on a specific application, protocol feature or underlying unicast protocol have been developed and

published. The bulk of research, however, has focused on the previously mentioned categories with the goal of developing generic protocols that can provide communication services for multiple applications, and that can be easily utilized by common hardware devices currently existing, or soon coming to market.

Reliability as a feature in these multicast protocols usually consisted of attempts to deliver packets initially sent by the source to the greatest number of receivers. In other words, reliability equated to the highest possible "initial delivery" count of receivers per packet. These protocols are often termed "best effort". Later, as ad hoc multicast routing issues became more clearly understood, work on reliable data delivery moved beyond "best effort" operations to various methods of storing and resending missed packets, with attempts to provide fully reliable packet delivery to all receivers. Work on these later protocols is described in chapter four.

2.2 Multicast Protocol Category Descriptions

Proactive or *Table Driven* protocols maintain routes continuously, even when the source has no packets to multicast. Examples of table driven protocols are DSDV [PB94], WRP [MG96], STAR [GS99]. *Reactive*, or *On Demand*, protocols on the other hand, typically invoke a path discovery / path reply mechanism only when a source has data to multicast, rather than continuously maintaining paths throughout the lifetime of a network. Usually the path discovery portion of the mechanism requires a packet to be flooded throughout the network, with the replies often unicast back along the same path. Paths are either stored at each node in the form of previous/next hop, or accumulated and cached at the source. Examples of On Demand protocols are ODMRP [LGC99], AmRoute [LTMB99], CAMP [GM99], AMRIS [WT99] and MAODV [RP99].

The *Soft State* category refers to protocols where full route refreshes occur periodically via the flooding of control packets through the network, while the *Hard State* category refers to protocols where routes are not refreshed, but rather the initially defined topology is maintained and updated in an ongoing manner. This maintenance usually takes the form of control packet transmissions when a link is discovered to be broken. The soft state approach usually requires more control packet overhead, but the result is often a

greater packet delivery ratio. The reverse is usually the case for the hard state approach. Examples of soft state protocols are MZRP [DSS01], DDM [JC01], ODMRP [LGC99], DCMP [DMM02a], FGMP-RA [CGZ98] and NSMP [LK00]. Examples of hard state protocols are (ref nbrs and initials) BEMRP [OKS99], ABAM [TGB00], WBM [DMM02b], PLBM [SMM02], AMRIS [WT99] and CAMP [GM99].

Tree based topologies work to maintain a tree structure, with a single linked list of nodes connecting any given source/receiver pair. The tree topology could be instantiated per individual source, or a single "shared tree" topology shared by several sources, often based on a core node coordinator. Simulation comparisons [LSHGB00] have shown tree topologies are generally more fragile than meshes, with a correspondingly lower packet delivery ratio. Examples of on demand shared tree based protocols are AMRoute [LTMB99], AMRIS [WT99] and MAODV [RP99]. Examples of on demand source tree based protocols are MCEDAR [SSB99], BEMRP [OKS99], MZRP [DSS01], ABAM [TGB00], DDM [JC01], WBM [DMM02b] and PLBM [SMM02].

Mesh based topologies work to provide multiple paths to all receivers from the source, with the goal being a higher packet delivery rate at the cost of a greater amount of network overhead. This topology is a better fit for mobile ad hoc wireless environments where individual links are prone to breaking. When one path breaks in the middle of transmission other paths will still provide data to endpoint receivers. Examples of on demand mesh based protocols are ODMRP [LGC99], DCMP [DMM02a] FGMP-RA [CGZ98], NSMP [LK00], SRMP [ML02] and CAMP [GM99].

The third topological configuration, *No Topology* or *Flooding based* protocols are designed to require no underlying data delivery path topology, completely removing the need for the overhead control packets needed to create and maintain a given topology. Some simple mechanism must be introduced to flooding based protocols in order to prevent broadcast storms, a condition where a given data packet is rebroadcast multiple times unnecessarily by nodes in a given area. This conserves the bandwidth that would have been used for this, allowing the ongoing data transmissions more bandwidth. The tradeoff is that since no data delivery routes are established, a greater amount of

bandwidth is often consumed with duplicate packet sends. Examples of on demand flooding based protocols are basic flooding and Hyperflooding [OTV01].

In a multicast protocol, when formation of the multicast group and ongoing communication, topology updates and missed data requests can only be initiated by the source, the protocol can be categorized as *Source based*. If it can only be initiated by the group receivers, it is categorized as *Receiver based*. Examples of source based protocols are MZRP [DSS01], ABAM [TGB00], AMRIS [WT99], ODMRP [LGC99], DCMP [DMM02a] and NSMP [LK00]. Examples of receiver based protocols are BEMRP [OKS99], DDM [JC01], WBM [DMM02b], PLBM [SMM02], FGMP-RA [CGZ98] and CAMP [GM99].

2.3 Multicast Protocol Related Work

2.3.1 ODMRP

ODMRP [LGC99] is an on demand, mesh-based, source initiated soft state ad hoc multicast protocol. It performs scoped flooding of data packets to all group members by establishing a 'forwarding group' of network nodes between a source and all group members. Route refreshes update the broken links arising from node mobility or resource changes. The route setup and ongoing periodic route refresh operations each consist of two phases: Request and Reply.

Mesh Establishment

When a source has multicast data to send but no knowledge of receivers, it builds a "Join Query" packet, adds its IP address, and broadcasts it. Each downstream node receiving the Join Query will store the source IP address and packet ID, add the IP addresses of the upstream node and originating source to its routing table, add its own IP address into the last hop IP address field, and rebroadcast it downstream. The Join Query packet floods the network, eventually reaching all receivers.

A group member, upon receiving a Join Query, completes the processing described above for the Join Query, then initiates a "Join Reply" packet once the upstream multicast route is selected. The receiver node adds the source and next upstream hop IP address for

the group from its routing table, adds its own IP address into the previous hop field, and broadcasts the Join Reply packet upstream. Each neighbor node receiving this packet checks the next hop IP address. If the next hop IP address matches the neighbor node's own, the node is on the forwarding path between source and receiver, and is part of the forwarding group. The node sets its Forwarding Group flag, looks into its own routing table entries for the group ID and next upstream hop node id, and builds its own Join Reply packet to broadcast upstream if it has not already done so. Once Join Reply packets have propagated back to the source, the mesh of forwarding group nodes is established and packets can be delivered to all receiver nodes.

Ongoing Mesh Maintenance and Data Forwarding

When a node receives a multicast data packet, it first checks to see if the packet is a duplicate, then checks its Forwarding Group flag. If the packet is not a duplicate, and the node's forwarding flag is set, the node is a forwarding group member, and rebroadcasts the packet to its neighbors.

Periodically, the source will refresh routes with another Join Query. All forwarding group members will then be reset according to the new network topology. Nodes no longer on a datapath to receivers due to a topology change will soon have their forwarding flag turned off via a timeout. Group membership is preserved in a soft state at each node. Once a source has no data to multicast, it stops sending periodic Join Query packets. All forwarding nodes will then eventually timeout and revert to non-forwarding status for that source. If a receiver wants to leave the group it stops sending Join Reply packets.

Unicast Functionality

Using the same Join Query/Join Reply protocol with a unicast IP address as the destination, a unicast sender can discover a route to a unicast receiver. Since duplicate Join Query packets are dropped (based on source IP address and data packet sequence number), the route created by unicast operation is a single path.

Data Structures

Following are the standard data structures of ODMRP.

- *Message Cache*: When a node receives a Join Request or data, it stores the source ID, sequence number and group address of the packet in this cache to detect duplicates. This cache is timed out in Round Robin fashion.
- *Member Table*: This table holds the multicast address and source node address identifying combination for each source data stream that the current node is a receiver or forwarder for. An expiration time variable is in the table in order to expire stale entries.
- *Forwarding Group Table*: This table holds the multicast addresses and expiration time for each multicast group for which the current node is a forwarding group member of.
- *Routing Table*: This table holds the multicast and source addresses for all multicast senders the current node is a receiver for, along with the next hop (upstream) address on the path to the source. This next hop address is used as the destination for Join Reply packets from the current node.

Protocol Advantages and Disadvantages

An advantage of ODMRP is that it has a very high packet delivery ratio, partially due to its soft state approach, with all routes periodically refreshed through control packets, and partially to its mesh based topology, which increases the chances that a given packet will eventually reach a given receiver. Both the control packet overhead and the multiple data paths per receiver however, contribute to an increased network bandwidth overhead, increasing the possibility of link contention.

2.3.2 Flooding

The standard flooding protocol is a simple and effective approach to multicast communication. When a node receives a packet, if it is receiving it for the first time, it will broadcast the packet. In order to recognize previously received packets each node must keep a cache of recently received packet sequence numbers.

Protocol Advantages and Disadvantages

The downside of this approach is that since each packet is re-broadcast as many times as there are nodes in the network, a large amount of network overhead is consumed for

operation throughout the lifetime of each communication session. However, the upside of this approach is the extremely high packet delivery ratio that flooding can provide. Generic flooding is a current standard to beat for new reliable multicast protocols, because it has one of the highest delivery ratios of protocols compared in current research. This topology is the extreme form of a mesh topology, where every potential data path in the network is enabled, and if one or several links break, receivers will still often receive packets over any remaining existing data paths.

2.3.3 Hyper Flooding

Hyper flooding [OTV01] is designed for high mobility environments where the main goal is reliability. The tradeoff in order to gain the added reliability is a greater amount of network overhead. In Hyperflooding, nodes record neighbors by listening to, and sending, hello messages with neighbor lists. Received data packets are rebroadcast and stored. Another rebroadcast of these stored data packets will occur when a packet is received from a node that is not on the neighbor list, or when a hello message is received from a new node. If one of these two events occurs all packets in the node's data packet cache are retransmitted. The intent of these rebroadcasts is to ensure that new nodes that might possibly not have received the cached packets will now be able to receive them.

Protocol Advantages and Disadvantages

Although node caches are periodically purged to limit this resending overhead, the downside of this approach is a far greater amount of network overhead used for the packet retransmissions, and the large amount of storage required at each node, while the upside is the added reliability gained by the ongoing packet retransmissions.

2.3.4 AMRIS

AMRIS [WT99] is an on demand shared tree source initiated hard state protocol. The central mechanism in the protocol is that each node in the shared tree obtains a multicast session member identifier (MSM-ID) that defines its logical height in the tree. The MSM-ID mechanism provides the protocol with a way to repair broken links locally, and avoid packet routing loops.

Tree Establishment

When a source wishes to initiate communication, it broadcasts a New Session message, which contains it's MSM-ID, a multicast session ID and parameters for routing. All nodes receive this message, and store the data in a table for a timeout period and create their own MSM-ID with a value greater than the received one. A built-in gap in identifier numbers allows space for open id numbers for ongoing local repairs. Each node also keeps a current neighbor status table, listing existing neighbors and their MSM-ID's. This is built from incoming beacon messages all nodes are required to broadcast periodically.

If a receiver node not already in the multicast group wants to join, it sends a JoinReq control packet to one of the neighbor nodes listed in its neighbor status table that has a lower MSM-ID. If the receiver of this control packet is not a member, it will send the packet further upstream. Once a group member node is reached, the node contacting it will initiate a JoinAck control packet, sending it back along the downstream path to the original initiating receiver, to establish the new data path.

Ongoing Tree Maintenance

After formation, ongoing tree maintenance operations are started to repair broken links. Each node sends "beacon" packets once per second. If a broken link cuts off a node from the group, noticed after 3 "beacon" packets are not received, the node will attempt to rejoin by selecting a new neighbor and sending a JoinReq control packet, with the same operation as described above. If the node has no current listing of a possible neighbor node, or it receives a JoinNack back from its neighbor instead of a JoinAck and has no other neighbors to attempt a connection with, it attempts a second operation where it floods a new JoinReq packet with a TTL value attached. It will receive JoinAck packets back from all receivers and will then select a route by sending a JoinConf packet to the targeted receiver.

Protocol Advantages and Disadvantages

AMRIS has one advantage in that its hard state local link repair mechanisms reduce the amount of control overhead needed network wide. Data path loop formations are avoided by the MSM-ID mechanism. The ongoing requirement for beacon packets from all nodes

consumes a large amount of network bandwidth however, and the large amount of time required for the link repair mechanisms to operate increases packet delivery delay, and reduces the delivery ratio. Also the link repair mechanisms can lead to unnecessarily long data paths, contributing to packet delivery delay. The tree topology, being more fragile than a mesh, also reduces the packet delivery ratio.

2.3.5 CAMP

The Core Assisted Mesh Protocol (CAMP) [GM99] is a mesh based, receiver initiated, hard state ad hoc multicast protocol. Rather than using flooding of a control packet to establish routes, with the associated cost in bandwidth overhead, CAMP defines certain mesh nodes as core nodes, and uses a core node based mechanism to establish routes. CAMP requires an underlying unicast protocol for operation. WRP was used for this in the documented study. This unicast protocol must have a mechanism to provide a node with "next node ID on the path to the core node" information.

Mesh Establishment

To begin the protocol, every node establishes a Core to group Address Map (CAM) table, containing the group's core node ID. When a node wants to join the group, it unicasts a JoinRequest packet to the group core node, addressed to the next node on the path to the core node. When this JoinRequest packet is forwarded to a node directly linked to the core node, that node will send an ACK back along the path taken by the JoinRequest, establishing the path. New receiver nodes directly linked to the core node have no need to send JoinRequest packets.

CAMP allows data source nodes to join an established network in "sender only" mode, if they will not receive data from other sources in the group. Nodes directly linked to sender only nodes will not forward packets received from other sources in this case, unless they have current links to other group receiver nodes downstream from the other source.

Once a new source joins the mesh, receivers note the number of hops taken for packet delivery from this source. Receivers periodically send HeartBeat messages that record hopcount to neighbors via the underlying unicast protocol, which are forwarded to mesh node members. When mesh node members receive such HeartBeat messages, they will

potentially enable the data paths if they are shorter, ensuring that shortest path nodes are actively made part of the mesh of forwarding nodes.

Ongoing Mesh Maintenence

With the mesh topology, multiple links to receivers can still provide data delivery when any given link is broken due to node movement. Also, the shortest path discovery mechanism works to build new, more optimal links for receivers with broken links. If the mesh is partitioned, each partition works to define its own core node. Partitions are repaired by each core node sending CoreExplicitJoin packets to cores in other partitions. When a core in one partition receives such a packet from another core, it will reply with an ACK packet and repair the partition.

Protocol Advantages and Disadvantages

CAMP relies on a core node for topology construction and repair, and so has no flooded topology construction packets. Due to this its control overhead is less than many mesh based protocols. One downside to the core node approach however, is that core nodes become single points of failure, and when they fail it significantly impacts packet delivery ratio. The requirement for a secondary unicast routing protocol is another downside to the protocol, making it less generalized.

2.3.6 MAODV

MAODV [RP99] grew out of AODV, a unicast ad hoc communication protocol. MAODV utilizes flooding for its periodic topology construction control packets. MAODV is a shared tree, receiver based hard state protocol.

Tree Establishment

MAODV operates by means of a group leader node, which is usually the first node to join the group. This group leader periodically broadcasts group hello packets, with a continuously updated sequence number. Nodes that want to join the group after it has been established do so by either unicasting a route request packet to the group leader, if they have its address, or by broadcasting a route request packet if not. This packet includes the receiver's last known group sequence number. If broadcast, the route request packet is rebroadcast by non member nodes until it reaches a member node. Member nodes with a

group sequence number greater than or equal to the new receivers sequence number will unicast a route reply packet back along the same data path. This reply packet contains the distance of the replying node from the group leader, and the current sequence number of the multicast group. If the new receiver attempting to join receives several replies, it selects the one with the greatest sequence number first, and the shortest path second, and unicasts a multicast activation packet back to the sender of the route reply. As this multicast activation packet traverses the data path, nodes on the path are activated to become new members of the data forwarding tree.

Ongoing Tree Maintenance

Ongoing maintenance works by means of the same route request, route reply, multicast activiation series of packets. When a receiver is partitioned from the group, its route request will contain the last sequence number it was aware of, and its last known hopcount from the group leader. This route request will be answered by nodes with a group sequence number greater than the requesting node, and a lesser hop distance to the group leader. This guarantees that a partitioned node's route reply will not be answered by nodes downstream from it that are also partitioned. Receiver nodes at the far end of the datapath that wish to leave the group will send a prune packet upstream. This packet propagates upstream through nodes that are only on the datapath in order to supply the leaving node with packets, and all such nodes will halt the forwarding of packets.

If the network is in a partitioned state, it is possible that each partition will have its own leader. Any node receiving group hello messages from more than one group leader will start a group leader election mechanism, which will reduce the number of group leaders to one, and heal the partitions where new links are formed.

Protocol Advantages and Disadvantages

Since MAODV has a shared tree topology, one advantage it has is relatively low data forwarding overhead. This shared tree topology, however, leads to fragile data paths with relatively lower packet delivery ratios compared to some other protocols. Also, the group leader represents a single point of failure, and failure or partitioning of this node has negative consequences for packet delivery ratios of all existing sessions.

2.3.7 FGMP-RA

The Forwarding Group Multicast Protocol, based on Receiver Advertising (FGMP-RA) [CGZ98] is an on demand mesh based, receiver initiated soft state protocol.

Mesh Establishment

The protocol begins by all receivers flooding a join request control packet, which is forwarded to the sources. As the sources receive the join request packets, they update their internal member tables with ID's of all receivers in the group. After receiving all join request packets, each source creates a forwarding table, containing next hop information to all receivers. The forwarding tables are then sent back to all receivers along reverse shortest paths, activating data paths along the way to each receiver. As each intermediate node receives the data packet with the forwarding table it will recognize that it is on the path from source to receiver, builds its own forwarding table, and send it downstream.

Ongoing Mesh Maintenance

Maintenance is achieved via the soft state approach, by receivers periodically flooding join request packets through the network. When a receiver's link to the source is broken it will send the source a join request packet. The source will then send a forwarding table packet back along the same path, establishing a new route to the receiver with a broken link.

Protocol Advantages and Disadvantages

Given FGMP-RA's mesh topology, it has a higher packet delivery ratio when compared to tree based protocols, since in general there are more data paths to receivers in the network. The soft state approach increases the control overhead compared to a hard state approach, however.

2.4 Multicast Protocol Performance Comparisons

2.4.1 Protocol Performance Modeling

In the pre-2000 timeline of mobile ad hoc multicast protocol development, research papers describing new protocols either did not evaluate their performance, evaluated performance by means of mathematical modeling (using formulas to determine, for

example, relative communication complexity or time complexity of various protocol operations such as membership join/leave operations or initial topology setup, where the variables were the average number of network nodes, theoretical number of links, theoretical number of receivers affected by a topological change, theoretical hop count of the farthest route length, etc…), evaluated it by using limited home-grown network simulators, or simply discussed the pros and cons of the categories the protocol mechanisms fell into. With these methods of evaluation it was impossible to truly evaluate protocols against each other in order to definitively state the superiority of one over another in terms of most common metrics such as reliable data delivery or latency.

As protocol development grew more sophisticated, researchers turned to standardized network simulators to more fully evaluate the performance of their protocols, and to provide the common ground necessary to compare performance between different protocols. Initially some relatively simple simulators were developed by individual research groups. Eventually, the research community as a whole settled on what are now a handful of well known and commonly used network simulators. Standardization on a few widely used simulators currently allows researchers to replicate the work and results of others, as well as allowing individuals to implement their protocol designs and fairly evaluate what works and what doesn't. The network simulators commonly used for ad hoc multicast research currently are ns-2, GloMoSim and QualNet.

- Ns-2 [FV02] is a discrete event network simulator. It was originally developed as ns, a wired network simulator, at Lawrence Berkeley National Laboratory. It was extended to version 2 as part of the VINT project with USC/ISI, Xerox PARC, LBNL and UC Berkeley to support mobile wireless environments. The CMU Monarch group's extensions allow ns-2 to simulate Mobile Ad Hoc networks [C99] as well. Ns-2 is the most commonly used simulator for implementing and studying ad hoc communication protocols. The source code is split between C for its core engine, and OTcl for simulation execution code.
- GloMoSim [ZBG98] is a discrete event wireless network simulator developed at UCLA's wireless networking lab. PARSEC [BM98], a C based parallel

simulation language, is the language used for its implementation. GloMoSim can run in sequential or parallel mode, and is the second most commonly used simulator.

- QualNet [SN] is another discrete event simulation environment. It is the commercial successor to GloMoSim. It is not as commonly used as ns-2 or GloMoSim, due to the cost of the simulator.

OpNet is another high quality detailed simulator used in research, but as it is a more expensive commercial product, it is not as commonly used as those listed above. In current research, Ns-2 is the simulator most commonly used by the community, because it is high quality, freely available and well documented. It is a robust tool with several protocols already implemented for ad hoc networking both at the MAC level and at the routing level, making protocol comparisons easier to implement. At least one research project [OTV01] implemented a protocol on two of these three simulators (ns-2 and GloMoSim), and discussed comparison of results. They found a high degree of correlation between the two, with minor differences being explained by implementation details such as the node mobility model, or the specific MAC layer protocol used. In general, the simulator parameters commonly set for protocol modeling are: Total simulated time, number of nodes in the network, number of receivers in the network, mobility model, average node speed, field size, wireless channel transmission range, wireless channel capacity, dddata packet size, data packet rate and MAC protocol.

2.4.2 Evaluation Metrics

There are many metrics by which a protocol's performance can be measured. The goal of this work is, of course, to maximize reliable data delivery, which is usually measured in terms of *packet delivery ratio*, which is the number of packets received by all receivers over the total number possible to receive (i.e., the number of packets sent by the source multiplied by number of receivers).

Other metrics that are critical for reliable ad hoc performance comparisons are the *data and control overhead* a protocol requires to operate, and average *packet delivery latency*.

Data and control overhead is a critical factor because mobile ad hoc networks generally have highly constrained bandwidth, which data packets and control packets actively consume. Data and control overhead is often measured as the ratio of data and control packets or bytes transmitted per data packet or byte delivered. It is the count of every single transmission of each packet type for every node over the network. This includes packets that are eventually dropped prior to final delivery, along with all packet original sends and retransmissions by all intermediate nodes. This metric reflects the efficiency of channel access. It is important because the maximum bandwidth only exists when nodes are close enough to each other to establish a link, and due to the inherent broadcast nature of the wireless medium, when links are established there is often contention occurring between nodes. For mobile ad hoc reliability, packet delivery latency is also a critical factor, since reliability protocols often will either utilize a NACK request mechanism that takes some amount of time to fulfill, thus delaying data delivery, or will implement a unique data delivery mechanism that could introduce latencies at every hop along each data path. It is usually the case that these three parameters are closely linked and optimizing one will come at the expense of the other two.

Following is a listing of metrics less commonly used by researchers to study protocol performance:

- *Node Storage, Processing Power, Battery Life Requirements* – the amount of storage, processing power or battery life receiver nodes must have for the protocol to operate effectively. These metrics are seldom used, since advances in technology are continually changing the thresholds of acceptability.
- *Packet Reliable Delivery Ratio* – Of the packets delivered to network receivers, this metric represents the fraction that were delivered to all receivers.
- *Reliable Goodput* – this is the throughput obtained for reliably delivered packets only. (i.e., throughput measurement of only the packets that all receivers have received).
- *Normalized Overhead* – a metric using the total number of packets sent by each node as measured at the MAC layer. Normalized Overhead is the ratio of total

packets sent at the MAC layer to total data packets delivered to all members. It measures the total number of packets actually transmitted to successfully deliver one data packet to all members.

- *Multiple variations for measuring control overhead* – percent of control packets, percent of control bytes, percent of control and data bytes transmitted vs data bytes delivered, and other variations.

2.4.3 Simulations and Results

As mentioned, many ad hoc multicast protocols were designed and evaluated by other means before researchers turned to using network simulators for fair, repeatable performance evaluations and comparisons. An initial study, documented in 1999 by J.J. Garcia-Luna-Aceves and E.L. Madruga, titled "*The Core-Assisted Mesh Protocol*"[GM99], compared ODMRP to CAMP, with results showing that CAMP data delivery had lower latency than ODMRP, with a smaller percentage of control packets used, and roughly equivalent reliability. The results were later viewed as somewhat flawed, since the authors used a very simple simulator they had developed a few years earlier that assumed perfect communication channels, did not take into account radio propagation or differing data packet size and used an old MAC protocol, FAMA, instead of the emerging standard of 802.11 for wireless networks. Also, the study simulated movement for only a small portion of the network nodes, with the protocol-critical nodes (i.e. source and core nodes) stationary, and all nodes in the network were modeled as multicast receivers, rather than only a fraction of them.

The first generally accepted protocol comparison study was presented at InfoCOM in 2000 by S. J. Lee, W. Su, J. Hsu, M. Gerla and R. Bagrodia, titled *"A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols*" [LSHGB00]. This study evaluated 5 current protocols, AMRoute, ODMRP, AMRIS, CAMP and flooding, using the GloMoSim network simulator, a well known and commonly accepted detailed simulator. All of the protocols simulated have been described previously with the exception of AMRoute, a shared tree based protocol with a logical "core" node per group, that relies on an underlying unicast protocol for group member links. AMRoute does not

guard against temporary routing loops, and so has limited performance. All the protocols rely on "best effort" delivery, where there is no means to make up for a data packet that is not received from the initial source send. In this study, different scenario tests varied node speed, source count, multicast group member size and traffic load in order to evaluate the relative strengths and weaknesses of the protocols under various network conditions within the common framework of the simulator. All simulations used a network of 50 nodes in a 1000 by 1000 meter area for 600 seconds. Radio propagation was 250 meters, channel bandwidth was 2 Mbps, data packet size was 512 bytes and the MAC protocol used was IEEE 802.11. The metrics used for evaluation were:

- Packet Delivery Ratio
- Number of Data Packets Transmitted per Data Packet Received
- Number of Control Bytes Transmitted per Data Byte Delivered.
- Number of Control and Data (Total) Packets Transmitted per Data Packet Delivered

These metrics were studied for four different scenarios, first, increasing average node speed, next, increasing number of senders, third, increasing multicast group size and finally increasing network data traffic load. The increasing node speed scenario had 5 senders sending 2 pkts/sec with 20 group members in the 50 nodes. Nodes moved from 0 km/h to 72 km/h. The increasing senders scenario has senders varying from 1 to 20, with 1 m/sec constant node speed, a consistent 10 pkts/sec traffic rate from all senders combined, and 20 group members. The increasing group size scenario had 5 senders, mobility at 1 m/sec, network traffic at 10 pkts/sec, and group size varying from 5 to 40 members out of the 50 nodes. The increasing data traffic scenario had 5 senders with a consistent group member count of 20 nodes, 0 m/sec node speed, and data traffic varying from 1 to 50 pkts/sec.

Increasing Node Speed Scenario:

Figure 2.1 shows results for packet delivery ratio, or reliability, as a function of node speed, which varied from 0 up to 72 km/hr, or 20 meters per second. The three mesh

based protocols, flooding, ODMRP and CAMP, had far better reliability than the two tree based protocols, as node speed was increased across the simulations. Between the mesh based protocols, flooding enabled every possible mesh link all the time, and ended up with the highest reliability of the three, given the data path redundancy. CAMP's reliability suffered from the fact that protocol operations often required nodes to send packets to specific remote router (core) nodes located on the edges of the network, that had fewer redundant paths to them than nodes centered in the mesh, so the packets had a higher likelihood of encountering a link break with no redundant path to allow eventual delivery.

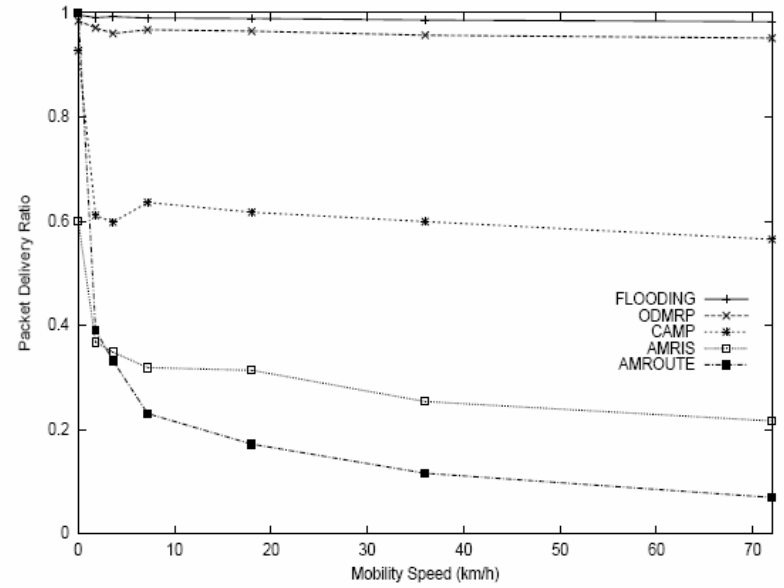


Figure 2.1: Packet Delivery Ratio as a function of Mobility Speed [LSHGB00**]**

Between the tree based protocols, AMRIS always relied on a single data path between group nodes, so if any given link in the set between a source and receiver breaks, encounters packet collision, or congestion, the receiver will not receive the data packet. Since nodes send beacons every second in AMRIS, and neighbors are not considered to be out of range until 3 beacons are missed, a link break is not accounted for by tree reconfiguration until 3 seconds have passed, during which time many packets can be lost. Interestingly, unlike the other protocols, at 0 mobility, AMRIS showed only a 60%

packet delivery ratio. With every node sending a beacon packet every second, beacon and data packet collisions occurred frequently due to link contention, severely impacting data delivery for a network of stationary nodes. AMRoute suffered severely from both the existence of loops, and the formation of trees with excessively high hop counts (double the number of hop counts in the other protocols). Data path loops formed during tree reconstruction when some nodes were forwarding per stale tree paths while others forwarded per new tree paths.

The other metrics charted for the mobility scenario reflected the network bandwidth costs of each protocol's reliability. The *Number of Data Packets Transmitted per Data Packet Received* in Figure 2.2 showed a very high number for AMRoute, due to the existence of data loops in the protocol operations. Flooding, ODMRP and CAMP were all in a similar mid-range, while AMRIS was at the low end of the five protocols. Clearly the tree structure of AMRIS conserves network bandwidth, and while flooding, ODMRP and CAMP all rely on a mesh topology, the more redundant paths enabled, the higher the number for this metric, with flooding being the highest since every path is enabled. ODMRP typically enables more paths than CAMP, and so had a higher metric.

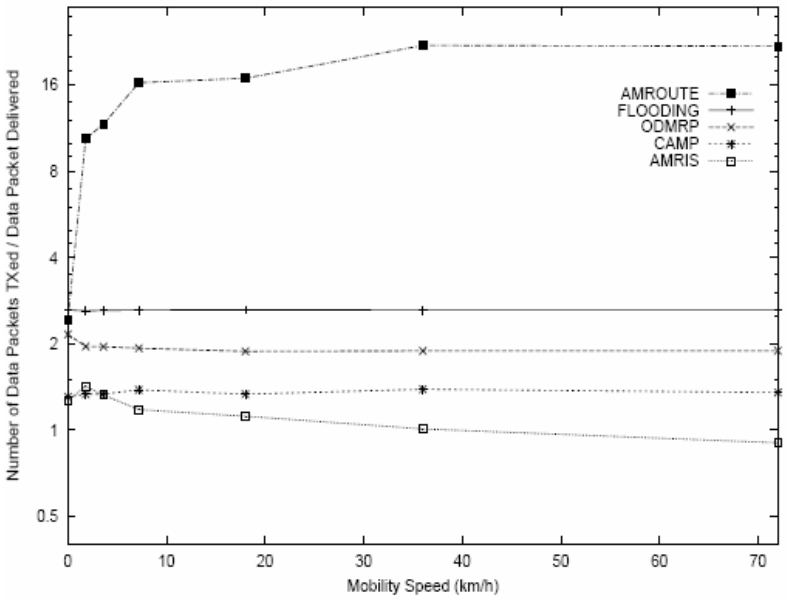


Figure 2.2: Number of Data Packets Transmitted per Data Packet Delivered [LSHGB00**]**

The *Number of Control Bytes Transmitted per Data Byte Delivered* in Figure 2.3 showed a very high number for AMRoute, as expected, since packets can be caught in routing loops or unusually long data paths. ODMRP held consistent across all node speeds, since mobility has no effect on the periodicity of its JoinQuery/JoinReply control packets. CAMP started off low, crossing to be higher than ODMRP at about 20 km/h and continually rising, while AMRIS was consistently low, and flooding was even lower. Control bytes for flooding consisted of the packet header only, and did not change with mobility. AMRIS overhead was low due to the high number of data packets dropped from the network. CAMP's overhead increased over time since its underlying unicast protocol, WRP, sent more update packets when triggered more often by higher mobility.

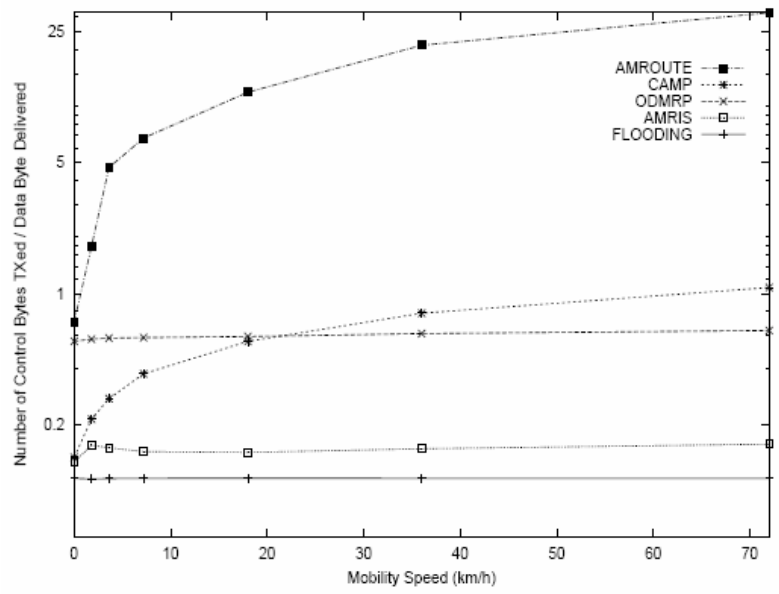


Figure 2.3: Number of Control Bytes Transmitted per Data Byte Delivered [LSHGB00**]**

Finally, the *Number of Total Packets Transmitted per Data Packet Delivered* in Figure 2.4 also showed a very high number for AMRoute as expected, but the number for CAMP and AMRIS were lower than the others. AMRIS was lower due to its tree topology, and CAMP was lower due to its reliance on fewer redundant paths. ODMRP

and flooding were almost equal. This result is somewhat surprising, since though ODMRP generates redundant paths, it should have fewer redundant paths than flooding, with a correspondingly greater efficiency in bandwidth utilization, while still delivering packets to the same number of receivers. Likely the receiver count did not place a sufficient stress on communication to where this became apparent.

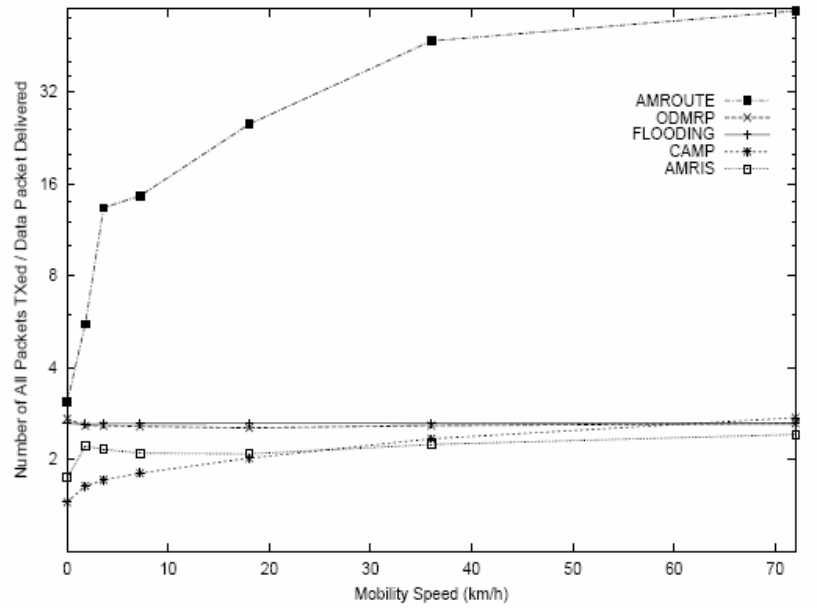


Figure 2.4: Number of Total Packets Transmitted per Data Packet Delivered [LSHGB00**]**

Increasing Number of Senders Scenario:

The second scenario holds node mobility constant at 1 m/s, but has the number of multicast senders increasing from 1 source to 20. Packet delivery ratio in Figure 2.5 shows similar relative results to the mobility scenario, but one difference is that though flooding shows slightly better reliability than ODMRP initially, at 10 senders and above ODMRP is a few percentage points better than flooding. This is where the overly redundant data paths of flooding work against its reliability, and flooding begins to suffer due to congestion and link contention. CAMP also shows a rising metric, though less than ODMRP, due to the increasing number of core nodes required.

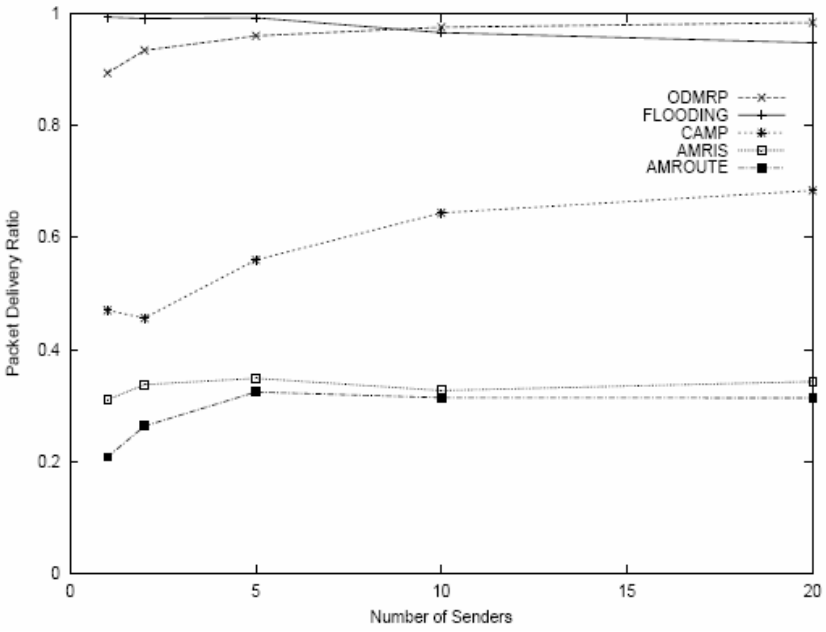


Figure 2.5: Packet Delivery Ratio as a function of Number of Senders [LSHGB00**]**

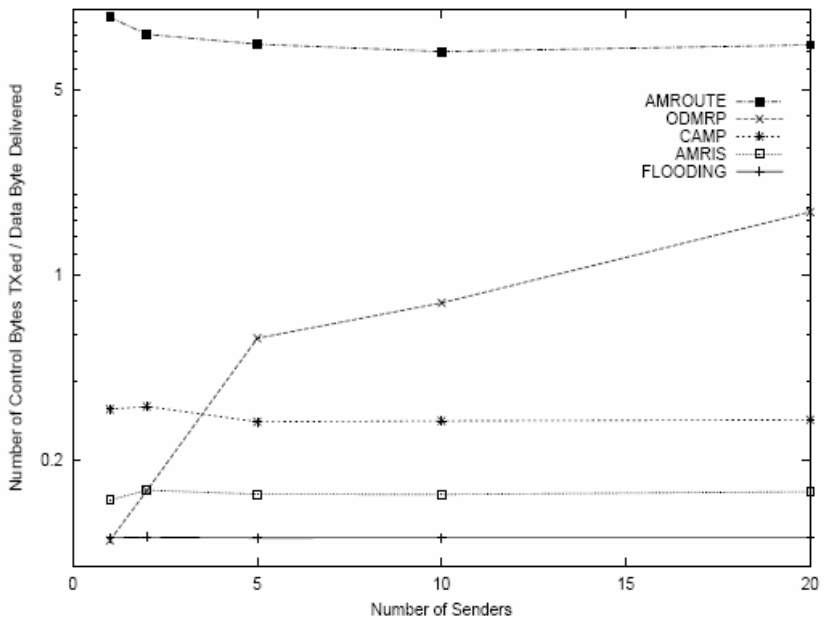


Figure 2.6: Number of Control Bytes Transmitted per Data Byte Delivered [LSHGB00**]**

For *Number of Control Bytes Transmitted per Data Byte Delivered,* results in Figure 2.6 are also similar to mobility results except ODMRP's: instead of holding flat,

ODMRP's metric rises steadily, while CAMP's metric remains consistent. CAMP creates a shared mesh, while ODMRP constructs per-source meshes, resulting in a greater amount of control overhead as the source count increases.

Increasing Multicast Group Size Scenario:

As the multicast group size increased from 5 to 40 members, flooding and ODMRP had similar reliability results to previous scenarios, as shown in Figure 2.7. CAMP, however, showed a great increase as the group size increased. In CAMP, as the delivery mesh grew to include more members, a greater number of redundant paths were formed, improving packet delivery metrics. AMRIS showed a lesser improvement, since its tree structure naturally inhibits redundant paths. AMRoute reliability declined due to the costs of data loops and unnecessarily long data paths.

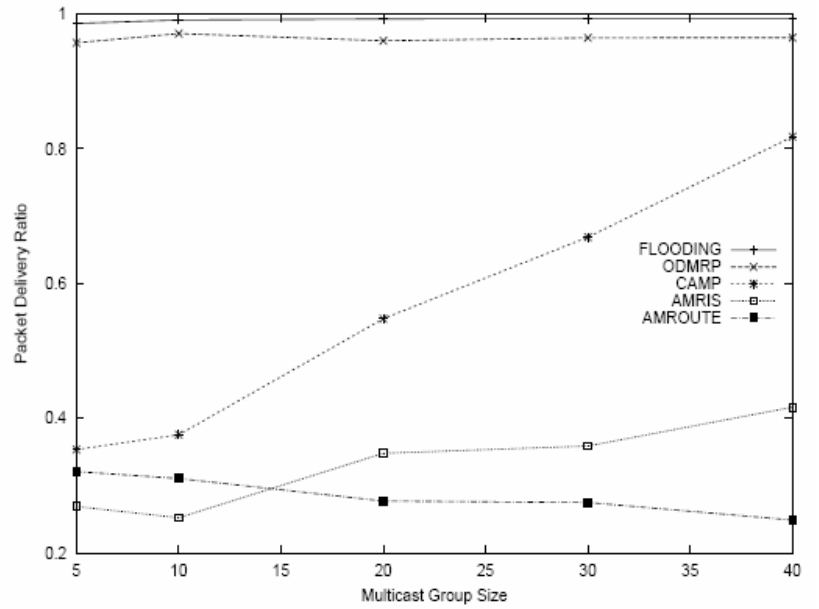


Figure 2.7: Packet Delivery Ratio as a function of Multicast Group Size [LSHGB00**]**

Increasing Network Traffic Load Scenario:

For the increasing network traffic load simulation, after a certain point ODMRP reliability crosses over to be better than flooding, remaining slightly above flooding when more than ten packets per second were sent, as shown in Figure 2.8. CAMP, also a core

based mesh topology protocol, did have significantly better reliability than the two other tree based protocols.

In terms of the other three metrics, all protocols were within range of each other with the exception of AMRoute. Since it has no guard against temporary data packet routing loops, its numbers were much higher than the other protocols for the three metrics.

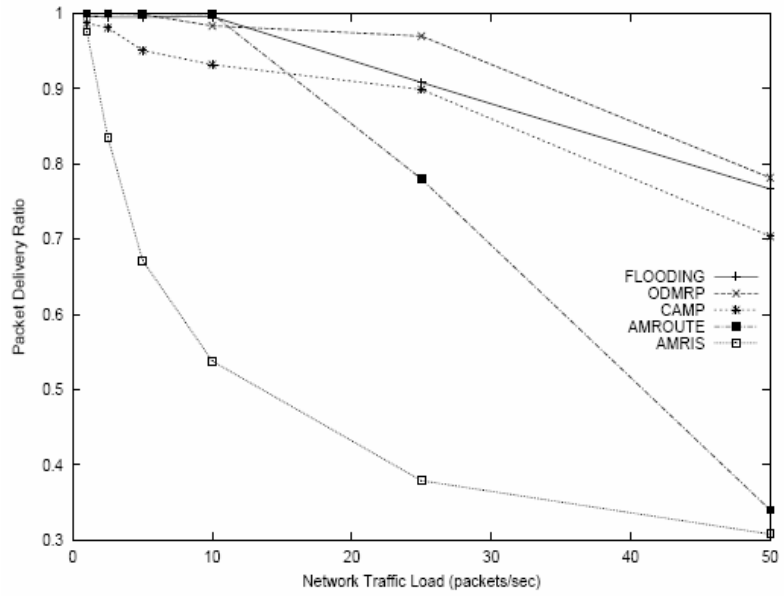


Figure 2.8: Packet Delivery Ratio vs. Traffic Load with no Mobility [LSHGB00**]**

Another definitive study was presented at the 21st International Conference on Distributed Computing Systems in 2001 by K. Obraczka, G. Tsudik and K. Viswanath titled "*Pushing the Limits of Multicast in Ad Hoc Networks*" [OTV01]. It compared flooding with ODMRP and MAODV (all described previously), using both the GloMoSim and ns-2 simulators. Part two of this study also investigated two types of modifications to the basic flooding protocol, *scoped flooding* and *hyper flooding*, that were designed to take advantage of flooding's strengths or minimize its weaknesses, in terms of reliability.

Part One: Flooding vs ODMRP vs MAODV

One result of the first part of this study was the confirmation that, with a balanced ratio between senders and receivers, the redundant packet delivery provided by a mesh

topology made for better reliability. Also, the mesh topology caused reliability to be less impacted by increased mobility. At higher mobility, the tree based MAODV had more frequent link changes requiring tree reconfigurations. This generated a greater amount of control traffic, and higher packet loss from link contention. In the scenarios tested, flooding had higher reliability than ODMRP and MAODV, due to its extreme amount of redundant packet delivery, with the gap growing as mobility grew. This is natural, since no topology needs to be maintained for flooding and every data path is constantly enabled, so increased mobility does not affect its reliability as much. Generally, flooding had the highest control overhead (control overhead for flooding being restricted to the IP header portion of each packet) given that many more packets are sent in the network, and MAODV the least, except for corner conditions that could be due to protocol parameter settings, a natural result since flooding has the most redundant packet delivery and MAODV the least, due to its tree structure. The network traffic load scenario tested showed flooding with the highest reliability and MAODV with the lowest, except for corner cases, however the simulated packets per second load did not reach the threshold where the reliability of flooding could drop due to link contention from overly redundant delivery in any of the scenarios. The latency scenario showed flooding to have the lowest latency of all protocols, again, a natural result, because since each data path is enabled the shortest path is always available. MAODV has the highest, since it consistently has the fewest data paths enabled.

Part Two: Scoped Flooding and Hyper Flooding

The second part of the study compared basic flooding with two schemes that attempt to reduce the amount of redundant packet delivery with the goal of reducing network overhead, classified as *scoped flooding*, and two that provide an even greater amount of redundant delivery with the goal of increasing reliability, classified as *hyper flooding.*

The first scoped flooding scheme was based on received power. In this scheme packet transmission power is fixed for all nodes. A node receiving a packet compares the power detected in receiving a packet to a certain fixed threshold, rebroadcasting only if power is

less than the threshold value. The higher the received power, the closer the transmitting node must be to the receiver, and the greater the overlapped area would be if the receiver node were to rebroadcast the packet, with less chance of increasing network reliability. The second scoped flooding scheme was based on neighbor discovery. In this scheme each node periodically sends a hello message containing a list of known neighbors. A node receiving a hello message will update its neighbor list with the received list and the sender. When a node receives a broadcast data packet, it compares the attached neighbor list with its own neighbor list. If the receiving node's neighbor list is a subset of the transmitting node's neighbor list, the receiver does not rebroadcast the packet.

The first hyper flooding scheme was based on neighbor discovery. Here, each node also periodically sends hello messages. When a neighbor gets a hello message, it adds the node to its list of neighbors. When a node receives a packet for the first time it rebroadcasts the packet in a flooding manner. When the node receives the packet a second time it will queue it in a packet cache. Then, if the node receives either a data packet from a node not in its neighbor list, or a hello message from a new node, a rebroadcast of all packets stored in the cache is triggered. Nodes periodically purge their cache to prevent excessive reflooding. The second hyper flooding scheme was based on received power. Here, when a node receives a packet for the second time it checks the power of the received packet. If the power is above a threshold, the packet is discarded, since it is likely there is a large amount of overlap in coverage region. If the power is below a threshold, the packet is rebroadcast with likely greater coverage, higher reliability and higher overhead.

Simulations were run comparing basic flooding to the two scoped flooding schemes and the two hyper flooding schemes. Results for 20 senders, 20 packets per second, were typical, showing that both hyper flooding schemes had higher reliability compared to basic flooding across all node speeds, and both scoped flooding schemes had lower reliability compared to basic flooding across all node speeds. This result showed that the increased delivery redundancy of hyper flooding did in fact help reliability at the cost of greater network overhead, and the reduced redundancy of scoped flooding hurt it, with

the benefit of lower network overhead, though the scoped flooding schemes lowered the network overhead by only a small amount. For both hyper flooding and scoped flooding, the neighbor discovery scheme showed higher reliability than the received power scheme. This illustrates the benefits of actions taken based on known neighbors, even at the cost of greater overhead in network wide learning the neighbors, versus actions taken based on theoretical broadcast coverage. Delay results showed basic flooding to have the least delay, which was expected. Basic flooding had more data paths enabled than scoped flooding, and while hyper flooding had higher reliability, the added packets received were due to later resends, increasing delay on those packets, specifically, and average packet reception overall. Scoped flooding operated with fewer data paths than basic flooding, leading to a higher average delay in reception since packets reached receivers along non-optimal paths.

2.5 Multicast Routing Strategy Discussion

From the S. J. Lee study [LSHGB00], discounting the flawed AMRoute protocol, it was clear that for a variety of different network conditions, the basic flooding protocol had the highest reliability for the greatest number of scenarios, the exceptions being the specific scenarios of high sender count, and high traffic load. Since flooding generates an extreme amount of redundant data delivery, these two conditions where the problems of link contention are magnified illustrated the weakness of flooding. Much more bandwidth was consumed by redundant data delivery in flooding than in either mesh based or tree based protocols, and network scenarios where limits of bandwidth overhead are reached will show the weaknesses of flooding.

Mesh based protocols had the next best reliability, and used the next largest amount of network overhead, and tree based protocols came in third in both categories. The downsides of tree based protocols were more fragile topologies, with links more easily broken due to mobility, and this was made clear in the scenario tests. The benefits of redundant, alternate routes in mesh based protocols, allowing for greater data delivery even when some intermediate links were broken, were also made clear.

Looking specifically at comparisons between the two mesh based protocols ODMRP and CAMP, it became clear that ODMRP has better reliability across a range of network scenarios, while also consuming more network bandwidth. ODMRP was fully distributed with a greater number of redundant data paths, while CAMP relied on individual core nodes. These core nodes could easily be located near the edges of ad hoc networks with fewer redundant paths to them, and so for CAMP, these core nodes became single points of failure. With the smaller number of redundant data paths, if a core node could not receive a packet, reliable delivery to multiple nodes was affected. Also, CAMP's preferred underlying unicast protocol, WRP, required a period of time to handle link breaks, impacting reliability, since delivery was best effort. Another implicit advantage for ODMRP was that previous comparison studies of ad hoc unicast routing protocols [BMJHJ98] showed that "soft state" protocols ultimately required less control overhead to maintain routing state, specifically in the face of increasing mobility, than "hard state" protocols. It is possible that if CAMP adopted a soft state approach similar to ODMRP its performance under increasing mobility would improve.

Results of the Obraczka study generally confirmed the overall Lee findings that, when comparing best effort flooding based, mesh based and tree based protocols, the more data paths enabled, the higher the overall reliability of the protocol and the higher the overall bandwidth consumed. Scenarios where the issue of link contention was magnified were not focused on, so flooding always won out in terms of reliability. Also confirmed in this study was the fact that the more data paths enabled, the lower the delay for protocol packet delivery.

In fact, confirmation of the benefits of more enabled data paths held true even for the variations of flooding tested in the second part of the paper. Hyper flooding, which augmented flooding with packet resends at a later point in time, had better reliability than flooding with more consumed bandwidth, and scoped flooding, which restricted specific data paths from basic flooding had worse reliability and less overhead than flooding. Both variations increased average delivery delay, hyper flooding since packet resends occurred

later in time, and scoped flooding because packet delivery paths were not the shortest possible that basic flooding would have used.

With these conclusions, a good approach to building a reliable ad hoc multicast protocol could consist of taking an existing ‘best effort’ protocol that had the best performance in terms of reliability without the down sides of flooding, and augmenting it in ways to improve reliability to be better than “best effort”, while impacting packet delivery latency and bandwidth overhead utilization to the minimum degree possible. This then, became the goal of my protocol R-ODMRP, described in the next section.

3 R-ODMRP: A Reliable Enhancement to ODMRP

This R-ODMRP [KR05] project represents an attempt to design such a protocol as was described in the previous section. The idea was to take mesh based ODMRP and add mechanisms to increase reliability. These reliability mechanisms consist of store and resend capabilities added to group receiver nodes in the network. Given the discussion in the previous section, the issue was to develop request / retransmit mechanisms that would cause a minimum of contention with ongoing packet delivery, and increase packet reception across all receivers in the network. In this section, first an overview of R-ODMRP will be presented, \ followed by an in depth look at its reliability mechanisms. Finally, a protocol evaluation will be presented.

3.1 Overview

R-ODMRP was designed to provide each source with a means to work with the two parameters of reliability and overhead cost, moving the reliability ratio up or down dynamically, over a single multicast session, if desired.

In R-ODMRP the responsibility for data storage and retransmit is assigned to all receivers of the multicast group, with the source of each data stream coordinating responsibilities. All group members are divided up by the source into sets of local neighborhoods. The source sets the number of nodes per neighborhood, with the option of determining the node's storage overhead. With each neighborhood member storing a portion of the data packets, each local neighborhood stores a distributed "sliding window" of all transmitted data packets. Nodes Nacking missing packets will be answered by nearby neighbors sending replies.

3.1.1 Packet Storage

In R-ODMRP, when a source initially sends out a Join Query, it becomes a Reliable Join Query (RJQuery) packet. The RJQuery packet has a timeout value attached. Once the RJQuery packet is sent, each node receiving it (whether a receiver node or not) will decrement this timer value by a preconfigured "two hop time" before sending the

RJQuery downstream. After the RJQuery timer expires at each node, each receiver node will send a Reliable Join Reply (RJReply) back upstream. If a node with an expiring timer is not a receiver, it will send an RJReply only if it receives other RJReplies from downstream.

Each RJReply contains a 2D table, known as the Network Datapath table. When a node (receiver node or not) receives RJReplies from downstream nodes, it stores their Network Datapath table as a block in its own table sorted relative to other received blocks with the topmost block having the longest datapath. On timer expiration, just before the table is sent upstream in an RJReply, each table entry is shifted such that entry (x, y) becomes entry (x, y + 1), emptying the leftmost column, column 0. The node stores an entry for itself in entry (0,0) containing its id, branch count (the number of RJReplies received from downstream), and receiver status, and then forwards the table upstream in its own RJReply.

The end result of the RJQuery/RJReply phase is that the source obtains a full positional listing of all receivers and forwarding group members in the network. RJQuery/Reply operations occur periodically, but at a lower frequency than the standard Join Query/Reply operation.

The source will then set a number for the "nodes per neighborhood" count, and, with the Network Datapath table as input, partition all receiver nodes into local neighborhoods using its "Source Neighborhood" Algorithm. The source then assigns data packet storage responsibilities such that the set of nodes within any given neighborhood will store the full set of data packets in sliding window fashion.

On the next multicast data packet after a Reliable Join Query, the source piggybacks a table defining the range of packet sequence numbers each receiver in each neighborhood is responsible for storing. Each receiver then begins storing its share of data packets. This recovery scheme does not depend on which node stores the packets, only that they are stored somewhere in each neighborhood.

As nodes leave the group, their storage responsibilities are reassigned on new RJQuery/Reply rounds. However, as more and more nodes join over time, more

neighborhoods are created and duplicate storage responsibilities will be assigned. The individual neighborhoods storing the duplicate packets will become smaller and smaller, relative to the overall network. Additionally, the source can reassign neighborhood size and data packet storage responsibilities on any RJQuery/RJReply round, dynamically adjusting reliability versus overhead over the course of a multicast.

3.1.2 Packet Retransmission

The second responsibility, data packet retransmission, will be initiated by a receiver node noticing a gap in data packets. It will broadcast a Resend Request packet to its local neighborhood, with a local time-to-live scope, listing all packets needed by sequence number. The requestor will give its ID for unicast replies. Upon receiving the packet, neighbor nodes will check their storage for the requested sequence numbers and unicast found data packets back along a single path. If the requesting node receives an incomplete reply or no reply at all, it will retain all needed packet sequence numbers, sending them out in its next Resend Request.

3.1.3 Data Structures

These structures are in packets sent to other nodes:

- *Network Datapath Table*: This table holds the current node's network positional information (node id, branch count and receiver status), accumulated from nodes on all downstream datapaths. This table is inserted into an RJReply packet, just before sending. The bandwidth requirements for a single node entry in the 2D Network Datapath Table in this implementation is 2 ½ bytes, based on a 15 bit node id, a 4 bit branch count (holding a maximum of 15 branches from a node), and a 1 bit boolean receiver status. A single 64k data packet will hold data describing approximately 26,200 nodes. As yet, it is unclear what the maximum feasible size of ad hoc networks will be, but one line of thinking holds them to be smaller than this, such as an informal gathering of conference attendees, or a lone group of rescuers.

- *Data Packet Gap List*: This list contains sequence numbers of all data packets that have not been received by the local node. The bandwidth requirements used in this implementation are 2 bytes per packet id number.
- *Storage Responsibility Table:* This table holds data packet storage responsibilities for all receiver nodes in the network. It is multicast out by the source to all receivers after the source neighborhood algorithm completes. The bandwidth requirements for a single receiver node entry in the 2D Storage Responsibility Table in this implementation is (2 + 1/n) bytes, 2 bytes per node and 1 byte for the neighborhood, with n being the 'nodes per neighborhood' count. For a 'nodes per neighborhood' count of 3, a Storage Responsibility Table handling the node count of 26,200 described above will fit into a 64k data packet for the worst case, where every network node is a receiver.

These structures are needed for a node's internal processing:

- *ResendRequestReply Cache*: This table holds data packets a node is responsible for storing. Additionally, it holds snooped data packets carried in resend replies forwarded by a node. To identify each data packet, the group address, id of the source node, address of the request originator and previous hop forwarder, originators sequence number of the request, and the id of the replier are all stored. Entries are aged out in Round Robin fashion. All replies are stored so that if any nearby receiver node sends a request for the packet in the future, it can be answered locally. The sending of resend requests for different nodes are staggered by a random time, in order to make this likely to happen. If a local group of nodes all miss a data packet but get the next one in sequence, one node will send out a request for a data packet while others wait. By the time others begin to initiate a request for the same packet, they will likely find it stored in their cache already and stop the send process.
- *Data Packet Sequencer*: This list holds recently received data packet sequence numbers along with their received time, for a given source. When a sequence number is received causing a gap, and over two seconds has elapsed since its reception, the missing number is listed as a gap in the received sequence. The data packet sequencer list is added to

from the tail, and the head is periodically trimmed. Trimming occurs either periodically when received data packets are all in sequence, or when gaps have been identified and loaded to the gaps list.

3.2 Neighborhood Creation

3.2.1 Overview of Neighborhood Building

As the group of receivers grows in size, neighborhood partitions and node data storage responsibilities are dynamically reallocated by the source, allowing partitioned neighborhoods to be composed of a diminishing percentage of network receiver nodes that are more closely grouped. As the number of receiver nodes and neighborhoods grow in an ad hoc network, Resend Requests and replies will travel fewer hops, reducing overall network traffic. Scalability is built in to the data storage and retransmit process.

3.2.2 Neighborhood Building Parameters

A set of parameters govern R-ODMRP Neighborhood Building operations. Some variables are simply inputs to the Neighborhood Building algorithm, while others are configured by the source. They are described below:

- *Size of a node's data packet storage buffer (NodSiz):*It is assumed that all nodes in the network will be homogeneous, and all will have the same fixed amount of storage space to devote to reliable communication.
- *Amount of network data produced per sec (AmtSec):* Fixed based on a single source generating a set number of fixed size packets per second.
- *Number of seconds of data to store (NbrSec):* Set to a value greater than the average time a node going out of range stays disconnected from the network.
- *Total storage capacity for a neighborhood(TotStg):*

  Set by the formula: TotStg = AmtSec * NbrSec
- *Number of nodes per neighborhood (NbrNod):*

  Set by the formula: NbrNod = TotStg / NodSiz

Following are R-ODMRP timing parameters:

- *Source timeout after RJQuery, before processing RJReplies (SrcTimOut):* Set based on the maximum simulation time for the first RJQuery to travel to the farthest receiver, and the RJReply to return to the source.
- *Time for a packet to travel one hop and back (TwoHopTim):*Used to set timers for Resend Requests, and to determine the amount to subtract from the SrcTimOut remainder at each downstream node.

3.2.3 Neighborhood Building Algorithm

The algorithm takes as input the Network Datapath Table and uses the NbrNod variable to partition the network into neighborhoods. Then it builds a table assigning each neighborhood node packet storage responsibilities, and a maximum hop count between nodes for each neighborhood. It inserts this table into the next JQuery packet before broadcasting it. The algorithm works as follows:

1. First, a pass is done through the Network Datapath Table, identifying the number of receiver nodes in each row (data path), summing to find the total number of receivers. The total number of receivers divided by NbrNod will give the number of neighborhoods the receivers will be partitioned into, as well as a remainder. The source keeps track of both the NbrNbrhds variable and the RmdrNbrhds variable.
2. Next, construction of the Storage Responsibility Table begins. Starting with the receiver at the far right end of the top row in the array, processing moves left, the hopcount is tracked, and receivers are added until a neighborhood is completed or a branch node is reached. Once a full neighborhood of receivers is identified, the source loads a row in its Storage Responsibility Table and records the maximum spanning hop count.
3. Upon reaching a branch node, if a full neighborhood is not yet built, the algorithm loads the branch node position on a stack, and steps down to the end of the next row. It continues to build the current neighborhood from the furthest node from the source forward, tracking hop count. Similarly, if a branch node is reached in this row,

processing steps down another row, but never moving outside a block. Once processing again reaches the node originally stepped down to on a given row, the stack node is popped and processing continues with it.

4. Once the first block is complete, the algorithm moves on to the next. The algorithm continues on in this manner until all receiver nodes within each block are either partitioned into a neighborhood or the count of remaining receiver nodes within each block is less than NbrNod.
5. Remaining receivers in all blocks are closest to the source. They are partitioned in the following way:
   - Unpartitioned nodes are sorted from bottom to top, with associated hopcount to source retained.
   - Selection of nodes for a new neighborhood begins at the bottom, and works sequentially to the top.
   - Hopcount per neighborhood is the sum of the two greatest numbers from either of the following two sets of hopcounts:
     - The hopcount to the source from each receiver in the nbrhd
     - The hopcount between any two receivers in one block in the neighborhood

   This algorithm will result in a table of partitioned neighborhoods, each with a spanning hopcount. Storage responsibilities are assigned by assigning a data packet sequence number range to each column in the table. This table is then put into the next JQuery packet and broadcast out to all receivers. Each receiver, upon receiving this packet, will learn its storage responsibilities and begin storing packets in a circular buffer.

3.2.4 Example of Neighborhood Building

Figure 3.1 shows a diagram of an example ad hoc network. The bold outlined node is the source, dotted outlined nodes are forwarding nodes and solid outlined nodes are receivers.

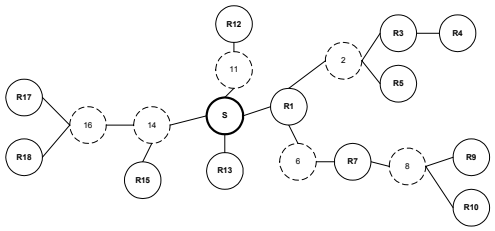


Figure 3.1: Example Ad Hoc Network.

Figure 3.2 shows example node network datapath tables, sent from the listed nodes to those upstream. Eventually the source will receive four RJReply Network Datapath Tables, sort them by block, and build a Network Datapath Table representing the composition of the overall ad hoc network.

R3	R4

2	R3	R4
	R5	

R1	6	R7	8	R9
				R10
	2	R3	R4	
		R5		

Figure 3.2: Example Network Datapath Tables (R3, 2 and R1)

Each entry in this Network Datapath Table is a structure with three elements: NodeID: the node's individual id, Branch_Count: the number of downstream branches (table rows) extending from the node, and Receiver_Status: a Boolean indicating receiver/forwarder status. The Network Datapath Table constructed by the source is shown in Figure 3.3. This table has four blocks, built from four RJReplies.

S	F	R1	R	6	F	R7	R	8	F	R9	R
	4		2		1		1		2		0
										R10	R
											0
				2	F	R3	R	R4	R		
					2		1		0		
						R5	R				
							0				
		14	F	16	F	R17	R				
			2		2		0				
						R18	R				
							0				
				R15	R						
					0						
		11	F	R12	R						
			1		0						
		R13	R								
			0								

Key:

Node Id	Rcvr/Fwdr
	Branch Ct

Figure 3.3: The Source's full Network Datapath table.

The source then begins the task of partitioning this table into neighborhoods. If the node count per neighborhood (NbrNod) is three, for example, the partitioning would happen in the following manner:

- R9 is selected for the first neighborhood. Node 8, a branch node, is placed on the stack, and R10 is added. Node 8 is popped, and R7 completes the neighborhood. The neighborhood's max hop count is set to 2.
- R1 is reached and added to neighborhood 2. R1 is seen as a branch, placed on the stack, and nodes R4 and R3 are added to neighborhood 2. The max hop count is set to 3, and R5 is a block remainder.
- Next, R17 is selected as the first node of neighborhood 3. 16 is placed on the stack and R18 is added to the neighborhood. Then node 14 is placed on the stack and R15 is added to the neighborhood, which is set to have a max hop count of 3.
- Now the algorithm shifts to phase two. The resorted array of remainder nodes is shown in Figure 3.4. R1 has already been partitioned into a neighborhood, so a flag is set for the entry to indicate this.

S	F	**R1**	R	2	F	**R5**	R
		11	F	**R12**	R		
		R13	R				

Figure 3.4: Network Datapath Table Remainders

- Starting with shortest hopcount to the source first, this array is traversed bottom up. First, the algorithm selects R13 for neighborhood 4. Next, R12 is selected for the neighborhood, and finally R5 is selected. The max hopcount is 5.
- The algorithm completes with the Storage Responsibility Table shown in Figure 3.5. For every 100 data packets sent from the source, nodes in the first column will store packets 1-33, nodes in the second will store packets 34-66 and nodes in the third will store packets 67-100.

Pkts 1-33	Pkts 34-66	Pkts 67-100	Nbrhd Hopct
R8	R10	R7	2
R1	R4	R3	3
R17	R18	R15	3
R13	R12	R5	5

Figure 3.5: Node Packet Storage Responsibility Table.

3.3 Protocol Performance Evaluation

R-ODMRP was implemented in the ns-2 network simulator [FV02], developed by the University of California, Berkeley, and the VINT project, with Carnegie Mellon's Monarch Project mobile and wireless ns-2 extensions [C99] incorporated. The ns-2 simulator is commonly used in networking research. [FV02] provides a full description of the software layers and the IEEE 802.11 MAC protocol used in these simulations. The USC/ISI ns-2 implementation of ODMRP [UC01] was also used.

3.3.1 Simulation Details

The ODMRP and R-ODMRP simulations all executed with identical randomly generated baselines of network traffic and node movement files to more accurately compare performance. This baseline consisted of five node movement scenarios and six traffic pattern scenarios. All scenarios established fifty mobile nodes with a single node as multicast source within a 1000m x 1000m area. The radio propagation range for each node was 250 meters, and the channel capacity was 2 Mbits/sec. Each simulation executed for 600 seconds of simulated time. Once all nodes joined the group the multicast source began transmission of 512 byte packets with a constant bit rate of 3 packets per second. The traffic pattern scenarios had 25, 30, 35, 40, 45 and 49 receiver nodes respectively.

30 simulation runs were executed each for ODMRP and R-ODMRP. A total of 60 simulations were performed. This baseline was chosen because simulations [LSHGB00] have shown that ODMRP performs best in conditions of relatively good network connectivity and low network traffic load and speed, and any protocol with the goal of increasing its reliability would have to outperform standard ODMRP under these

conditions. The reliability technique proposed in this paper likely has its greatest advantages in sparse networks with frequent longer partitions, however.

For ODMRP and R-ODMRP, parameters were set to 3 seconds for the Join Query flood interval and 9 seconds for the forwarding state timout, the values used by ODMRP's creators in their simulation studies. R-ODMRP sets a flag in every fourth Join Query packet, turning it into a Reliable Join Query packet. The node count per neighborhood for R-ODMRP was set at 3, and all nodes were preset to store a maximum of 500 data packets, in Round Robin fashion.

3.3.2 Initial Simulation Experiments

Beginning experiments lead to some modifications to the basic protocol of R-ODMRP that produced better end results. Originally, the time-to-live hopcount for a resend request packet was set to the maximum distance between nodes within a given neighborhood, but this produced relatively poor results. Data packets that would have been correctly delivered under ODMRP were dropped due to network traffic contention with the Resend Requests, causing the R-ODMRP portion of the protocol to work that much harder to try to fill the gaps, leading to further network contention. In the end, for these simulations of high network connectivity, a TTL of 1 gave best results for Resend Request packets. A consequence of this was that data packets that were undelivered to a group of receiver nodes tended to "bubble" across nodes over many cycles, increasing latency for those packets.

3.3.3 Simulation Results

Initial results for *Total Data Packets vs. Delivered Data Packets* (Packet Delivery Ratio) were encouraging. Figure 3.6 shows that when ODMRP ran alone, the Packet Delivery Ratio varied between 92.8% and 93.8% for the thirty simulations, given the same number of network nodes and an increasing percentage of receivers.

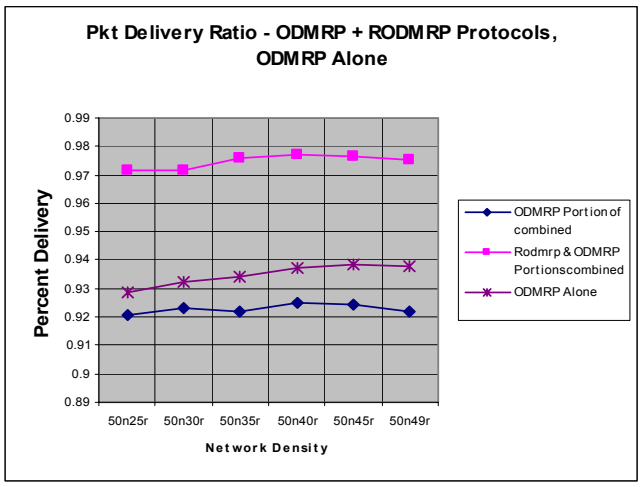


Figure 3.6: Packet Delivery Ratio

When R-ODMRP ran, the ODMRP portion operated between 1% and 1 ½% worse than its standalone counterpart, due to the added network contention, but the reliability portion increased Packet Delivery Ratio by approximately 4% overall, to between 97.1% and 97.7%.

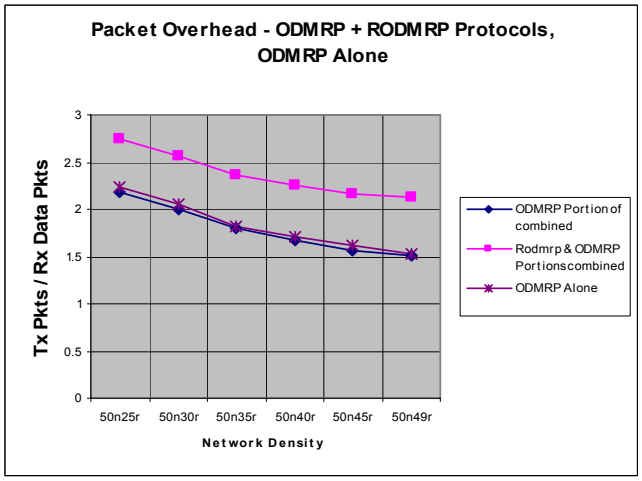


Figure 3.7: Packet Overhead Ratio of Data+Control Pkts per Delivered Data Pkt.

Other metrics showed the tradeoff for this increased reliability, however. The *Ratio of Data and Control Packets vs. Delivered Data Packet* (Control Overhead), shown in figure 3.7 reflects a consistent and unavoidable increase for R-ODMRP. The higher

number represents greater channel contention, working against the basic goal of reliable data delivery. The number here for R-ODMRP must be greater than that for ODMRP, since R-ODMRP uses additional control packets. On the positive side, the increase shown for R-ODMRP scales similarly to that of ODMRP, rising a similar percentage as the number of receivers in the 50 node network declines.

Data Packets Forwarded vs. Data Packets Delivered (Forwarding Overhead), shown in Figure 3.8 also shows an unavoidable increase for R-ODMRP. The differential in this metric also represents greater channel contention, working against the basic goal of reliable data delivery. An increase here must exist, given the store and retransmit mechanism, but the differential between ODMRP and R-ODMRP increases with an increase in receiver count. The mechanism used for Resend Request/Reply will be modified to increase scalability of this portion of R-ODMRP.

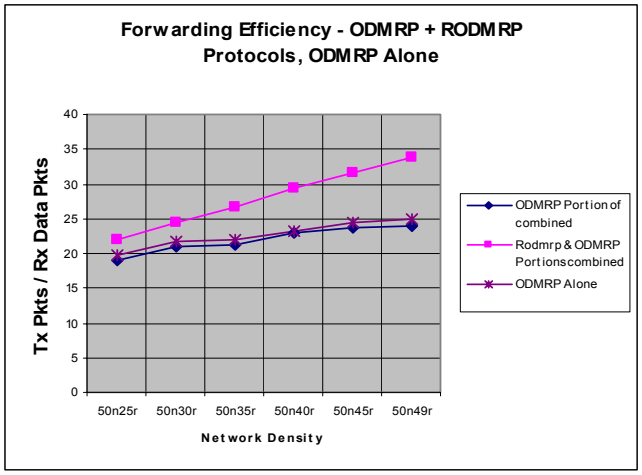


Figure 3.8: Forwarding Efficiency

The data delivery latency of the two protocols shows the greatest differential, however. While the average latency of ODMRP, and the ODMRP portion of R-ODMRP averaged about 10ms across all receiver counts, the extra packets delivered by the Resend Request/Reply portion tended to have a latency of seconds, due to several factors. One is the fact that two seconds elapse after a gap is noticed and a Resend Reply packet is sent. Another is that a random delay before sending was added to allow snooping of other

node's Replies before sending a request. A third is the mechanism used to trigger requests, which causes data to "bubble" across nodes.

The competing metrics involved in enhancing reliability for ODMRP have been clarified as a result of this work. Four central factors balance against each other: Packet Delivery Ratio ("Reliability"), Ratio of Data and Control Packets per Delivered Data Packet ("Control Overhead"), Forwarding Efficiency ("Forwarding Overhead") and Data Packet Delivery Latency to all Receivers ("Latency"). Comparing the basic ad hoc multicast protocol of ODMRP to R-ODMRP, overall latency tends to be lower, reliability is based on the basic protocol's best-effort delivery technique, and network traffic overhead is lower. When the store and retransmit reliability components are added to ODMRP, reliability increases, overall latency increases and network traffic overhead increases, due to the control and forwarding mechanisms. A successful reliability component will, under various network conditions, always increase reliability (by a varying amount, depending on the scenario and the strength of the reliability component), increase overhead by an 'acceptable' amount (acceptable meaning low enough so that the extra overhead causes minimal additional network contention resulting in minimal additional dropped data packets), and increase data packet latency as little as possible. Of the three competing factors, the two overhead metrics are more tightly linked to increased reliability, and latency is the least linked metric.

In most multicast ad hoc protocols, reliable packet delivery falls off sharply as network node density becomes more sparse, with fewer links between nodes. It is expected that the sparser the network, the more successful a store and retransmit reliability component such as R-ODMRP will be in achieving its goals. In sparse networks increased network traffic overhead required by the reliability component will have a lesser negative effect, since contention is less of an issue. It is expected that latency will be affected to a greater degree, since packets that would have been undelivered will be delivered much later, when a link is finally obtained, but latency will be due to the unavailability of a link rather than the mechanisms of the reliability component.

3.3.4 Protocol Results by Phase

Statistics were gathered for the normalized packet counts for each phase of the ODMRP portion and the reliability portion of R-ODMRP. Figure 3.9 reflects the normalized packet counts for all phases of ODMRP. Here it can be seen that the number of forwarded JQuery packets holds flat across the 6 scenarios, while the forwarded data packet count rises gradually. This makes sense, because as more receivers are added, data packets will at times be forwarded to further endpoints, given the same network. The JReply packet count shows a sharper increase, however. This portion of ODMRP would be the first to investigate in order to raise ODMRP's overall efficiency.

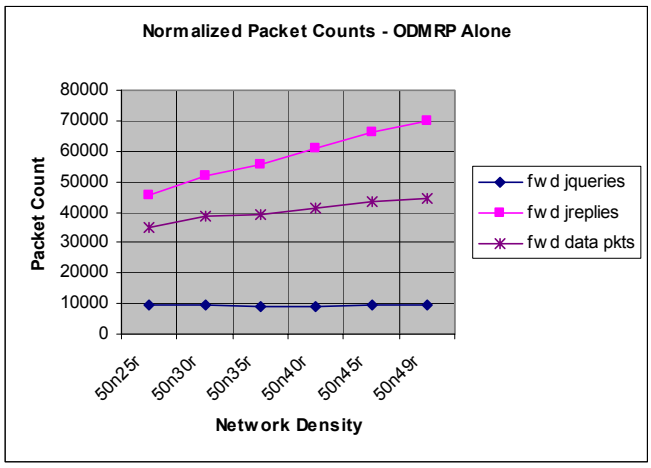


Figure 3.9: Normalized Packet counts for ODMRP.

Figure 3.10 shows the corresponding normalized packet counts for the phases of R-ODMRP added in over the baseline series of runs. Here it can be seen that the number of RJQueries holds flat. This is expected, since the ODMRP protocol is reused for this component. The count of RJReplies rises very gradually, almost holding flat, as the number of senders is increased. This count reflects the new timeout mechanism for gathering all downstream RJReplies before initiating one upstream. This metric shows that ODMRP's network contention due to JReply traffic can be reduced by adopting the R-ODMRP mechanism. This would increase ODMRP's scalability and efficiency by reducing control overhead. The R-ODMRP counts for Resend Requests and Resend

Replies rise at a similar steep pace relative to the other protocol components, however. The Resend Request/Reply mechanism would be the first to look at in terms of increasing the efficiency of the overall R-ODMRP protocol. A technique to unicast out a Resend Request should help reduce this packet count. This will have the secondary effect of reducing the Resend Reply count.

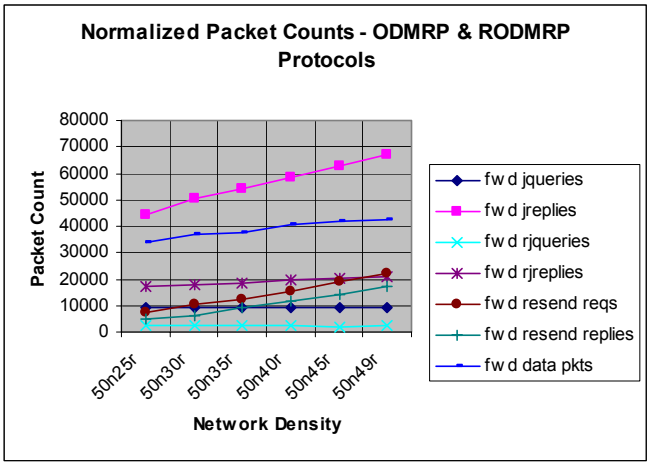


Figure 3.10: Normalized Packet counts for ODMRP, R-ODMRP

3.4 Conclusions for R-ODMRP

This section described R-ODMRP, a reliability protocol added to ODMRP. R-ODMRP consists of reliability mechanisms that store and retransmit sequenced data packets between receiver nodes, with overall coordination by the source. R-ODMRP has been implemented in ns-2 and run against a baseline of a light density network with increasing receiver count, ideal conditions for the base ODMRP protocol, the current standard for reliability among 'best effort' protocols. . Results show that R-ODMRP does outperform ODMRP under these conditions in terms of reliability, at an acceptable cost of an increase in routing efficiency and forwarding efficiency bandwidth overhead. The data delivery latency metric can be improved with fine tuning on the Resend Request /Reply protocol phases.

4 Review of Related Reliable Multicast Protocols

4.1 Introduction

By 2001 a large amount of research had yielded a substantial number of "best effort" unicast and multicast MANET communication protocols. Between 2001 and 2003 some research papers were published that focused specifically on the issue of multicast reliability in MANETs (e.g. AG [CRB01], RDG [LEH03], RALM [TOLG02], ReACT [ROLTG03],).These protocols were implemented either by means of taking a known "best effort" protocol that was well documented with high initial reliability, and adding reliability mechanisms to increase the packet delivery ratio, or by creating standalone mechanisms that could be added to other "best effort" multicast protocols to enhance their reliability.

From 2002, papers on multicast reliability began to appear describing new protocols specifically designed for high reliability, with RMA [GSPS02], Scribble [VE04], RAPID [DFKS06] and EraMobile [GO07]. As with the more general multicast MANET protocols, these protocols can be categorized by the central mechanism used to provide reliability.

4.2 Reliable Multicast Protocol Category Descriptions

Recent reliable multicast protocols can generally be classified as *deterministic* protocols (RMA [GSPS02], RALM [TOLG02], ReACT [ROLTG03], Scribble [VE04]), which attempt to guarantee fully reliable data delivery, or *probabilistic* protocols (RDG [LEH03], AG [CRB01], EraMobile [GO07], RAPID [DFKS06]) which try to guarantee a certain probability of reliability.

Deterministic protocols usually attempt to guarantee fully reliable data delivery by requiring individual group member nodes to detect their own missing packets and make their needs known by negative or positive acknowledgements (N/ACKS). These ack messages are transmitted to the source or to another group member, who then has the responsibility of retransmitting the missing packets.

Probabilistic protocols do not attempt to guarantee fully reliable data delivery, but rather are designed to provide a certain data delivery probability. They are currently being investigated as a means to deal with what appears to be the essentially non-deterministic nature of MANETs. The hope is that in relaxing the requirement for full reliability, operational overhead can be reduced and the scalability and inherent reliability of protocols enhanced. The fact that no reliable packet delivery guarantees can be provided by probabilistic protocols is a clear downside to this approach however, as is the potential for larger latencies associated with the packet dissemination and recovery mechanisms.

4.3 Reliable Multicast Protocol Related Work

4.3.1 RMA

RMA [GSPS02] is a sender initiated reliable ad hoc multicast protocol that assumes sources know the ids of all receivers in the network. This source knowledge is enforced through the lifetime of a network by JOIN and LEAVE messages, transmitted by receivers. The topology utilized by RMA is not strictly tree or mesh based, but rather based on an undirected graph, created dynamically, that can take the form of a tree or mesh.

RMA operates with two phases: initial multicast then retransmission. During the multicast phase a source transmits MKNOWN messages, which are unicast to receivers on routes known by the source (learned through JOIN or ACK messages), and MUNKNOWN messages that are broadcast to aggregated receivers on unknown routes. Sources choose best routes to a destination based on link lifetime, rather than shortest hop. Source route choices are also influenced by routes aggregating more receivers rather than fewer. After waiting a period of time, if the source is not able to gather acknowledgements (MACKs) from all group members, it begins the retransmission phase, sending an MUNKNOWN message with a RETRANSMIT flag. This is repeated until the source gets ACKs from all receivers for all packets. If a receiver senses a return path is not valid, it will send a broadcast MACK (BMACK) back to the source. All network nodes will refresh their routing tables with reception of JOIN, MACK or BMACK messages. JOIN

messages are broadcast by all receiving nodes. Network connectivity is maintained with neighbor HELLO messages, to publicize neighborhood changes. No state information is propagated network wide, instead, each node keeps only next hop routing information for other next hop nodes. When a next hop node moves out of range of a networked node, the networked node removes routes with that node listed as next hop.

A downside of this approach is the necessity for all receiver ACKs to travel back to the source. This makes scalability of the protocol problematic, due to ACK implosions. This ongoing requirement for multiple receivers to interact with the source directly conflicts with network stresses imposed by dense networks, high data rate and high mobility scenarios, where high degrees of link contention and link breaks occur.

4.3.2 RALM

RALM [TOLG02] is a reliability protocol that achieves a higher data delivery ratio by enforcing a congestion control mechanism similar to TCP. It is a precursor to ReACT, developed by many of the same researchers. Reliable data delivery is guaranteed to one group member at a time, in round-robin fashion. RALM begins with an assumption that the source knows the full group membership. The source selects one receiver at a time, in round robin fashion, to transmit a window of data to. The selected receiver will unicast a reply to the last packet with either a positive ACK, showing the window of data was successfully received, or a negative NACK, requesting retransmission of missing data, on a per packet basis. This per packet retransmission forces the source to slow its data rate until the specified receiver has achieved full reliable reception for the window of data. The selection of a single receiver negates the possibility of N/ACK implosions at the source, since no other receivers other than the selected one may reply. This feedback from the receiver is also used to adjust the source window size. The window initially increases exponentially until a “slow start threshold” is reached, then linearly. If losses begin to occur the window is halved. The downside of this approach is that decreased throughput is the tradeoff for increased reliability. Another downside is that even with congestion control, dropped packets are resent from the source rather than locally, creating greater network overhead from both the receiver NACK and the source

retransmit packets. High mobility has a good potential to unnecessarily shrink the sending window. Instead of network congestion causing a receiver to miss packets, broken links due to mobility would be the cause, and the congestion control mechanism would be unable to differentiate. This mechanism is different from TCP in that one source window of data is used for all network receivers, and only the last packet of the window requires a receiver generated ACK.

4.3.3 ReACT

ReACT [ROLTG03] combines receiver based, local recovery with RALM's source based congestion control mechanism. For the congestion control, the source in ReACT initially probes the network to deduce a packet send rate, then multicasts data at the configured rate until hearing a NACK from any receiver experiencing congestion. As in RALM, on source reception of a NACK, loss recovery is begun, during which the source collects more NACKs from other receivers. In loss recovery, the source cycles through its list of NACKing receivers, selecting one at a time to communicate with and discover missing packets from, replying with one packet at a time, multicast, since other receivers may also be missing it. As the source does this, the selected receiver unicasts individual packet ACKs back specifying sequence numbers of a packet still missing. The goal of this single packet transmission is to slow the source data rate until congestion is relieved. There is still no ACK implosion issue, since the only node sending ACKs is the individual receiver working with the source at any given time. If the source receives no reply from a given receiver after three send attempts, it removes the receiver from its list. When the source's list of receivers needing resends is empty, the source reverts to its original data send rate.

ReACT adds a local recovery mechanism in an attempt to alleviate unnecessary slowing of source sends. The mechanism works by each receiver node maintaining a current 'reliability' metric for itself, calculated by a sliding window formula using the numbers of packets it received and packets it should have received, and sending this metric, a congestion indicator and its node id downstream in the ip portion of the data packet header. Downstream nodes store constantly updated tables of this information

along with reception timestamps, used to timeout old entries. When a node initiates local recovery, it selects a recent 'upstream' node id from its table with the highest reliability metric and no congestion indicator, and unicasts a request to it. If all upstream nodes show congestion, the node reverts to the congestion control mechanism. If the number of packets requested is below a set threshold, the node receiving the request checks its store for the requested packets, forwarding them back if found, forwarding a rejection if not. A requesting node receiving a rejection will attempt local recovery two more times with different nodes before reverting to source based recovery. If the number of packets needed by a requestor is over a set threshold, the request is forwarded to the source for source based recovery. Receiver nodes only store a small number of data packets, 20 in the simulations published.

4.3.4 Scribble

Scribble [VE04] provides a deterministic guarantee that at least K receiver nodes will receive a given packet. In Scribble, data packet dissemination responsibility begins with the source but is later passed to other nodes on a per-packet basis, deemed to be in better positions to disseminate than the currently responsible node, with lower overhead. Data dissemination does not require a routing protocol or any knowledge of network topology. If a node becomes responsible for delivery of packet m, it must transmit m every x seconds, with x a specified protocol parameter. A counter mechanism controls transfer of dissemination responsibility.

Dissemination Responsibility: A node N_n receiving m for the first time sets its new logical clock $L_n(m)$ to 0. If the node becomes responsible for transmission of m it increments $L_n(m)$ by 1, stamping m with the value of $L_n(m)$ prior to sending it. If another node N_j not responsible for m receives this packet, it becomes responsible for sending it if its timestamp $L_j(m)$ is less than or equal to that in the packet and if N_j has not received m in the past (beta – sigma) seconds. If these occur, N_j sets $L_j(m)$ to equal the timestamp in the received packet m, and selects a random delay time, after which if a second packet m is not received, N_j then becomes responsible for sending packet m. If a responsible node

N_j receives a packet m with time greater than or equal to its $L_j(m)$, it relinquishes responsibility for m and cancels pending transmissions of it.

Realization: Each node responsible for transmitting m will append a list of *signatures* of nodes known to have received it to the packet header. A node receiving m that does not see its own signature in the header and decides not to transmit is still responsible for transmitting a small ack packet one hop back to the sender. A node N will *realize* m when its header contains K signatures of other nodes. On *realization* or on receiving m later, the node will transmit a *realization* packet containing the id of m. Other nodes receiving the *realization* packet will also realize m, without necessarily sending out their *realization* packets for m.

Termination: Network conditions that are extremely "adversarial" (in the author's words) will require extreme measures to ensure reliability to K receivers. The extreme measure implemented in Scribble allows all receiver nodes in the network to eventually become responsible for transmission of m after a period of time in which it appears packet m is not being realized. Here, each node N_n has a parameter Theta such that if the logical clock in the newly received packet m.l is greater than Theta (where it is clear the packet has spent over the threshold period of time in an unrealized state), then node N_n becomes responsible for transmitting m as well. In this way, eventually all nodes in the network will be added to the set of nodes transmitting m until realization is eventually reached. The more nodes in this state, the higher the transmission overhead, but the authors state that in adversarial network conditions this may be the only remedy to the provision of reliability. Also, the authors suggest that a high value be selected for Theta, since potentially prolonged partitions are common in ad hoc networks, and can easily be mis-read as "adversarial" network behavior when they are not.

4.3.5 Anonymous Gossip

Anonymous Gossip [CRB01] is a reliability mechanism that operates on top of any multicast protocol. It does not require group membership information. It allows for randomly selected pairs of group nodes to exchange information on received and lost packets by operating in two phases. First, a source node multicasts data packets using the

underlying best effort multicast protocol. Secondly the recovery phase operates. In this phase, a group member missing packets will periodically send a gossip request, containing data on lost packets, sequence number of next expected packet and source and group address, along with a list of received packets, to a pseudo randomly selected neighbor node. If the neighbor node is a group member, there is a certain probability that it unicasts requested data packets back to the initiator with its own request for its missing packets known to be received by the remote requestor. If the neighbor node is not a group member or if the probability of replying is not enforced, the receiver randomly selects a neighbor to forward the message to. The multicast protocol in use must provide nodes with hopcounts to their nearest neighbors, to accomplish this. Requests are answered by nearer neighbors with a higher probability than farther neighbors to reduce network overhead. The interaction with farther neighbors, though at lower probability, attempts to solve the issue where an entire region of the network has failed to receive a given packet. An unavoidable downside of this approach is that the increased network traffic created by the protocol negatively impacts the data loss the protocol attempts to correct. Also, given that requests/replies occur between random nodes, this is a probabilistic technique.

4.3.6 RDG

RDG (Route Driven Gossip) [LEH03] is built on top of a MANET unicast routing protocol, such as AODV or DSR. It uses a pure gossip based mechanism both for initial multicast packet transmission and for missing packet recovery. Unlike RMA, RALM and ReACT, the source is not required to have full group membership knowledge, only partial knowledge. In RDG, nodes can join groups through JOIN sessions, where a node announces itself, and other group members probabilistically reply, so the joining node has a partial view of the group on joining. Each group member then periodically gossips, sending a packet containing new received packet id's, missing packet id's, new group members and leaving group members to a subset of nodes in its partial group member list via the routes reported by the underlying unicast routing protocol. In this way the gossip messages are "route driven" rather than being "view driven". No overall source oriented group membership view driven mechanism exists. A group member receiving such a

gossip message examines the lists, sending back packets it has received that are on the gossiper's missing list, requesting packets it is missing that are on the gossiper's newly received list. This eliminates the need for source nodes to participate in missed packet retransmits, putting the burden for retransmission on all group members. The receiving group member will also remove obsolete members from and add new members to its partial group member view. A node will leave the group by announcing its departure through such a gossip message. RDG operates entirely probabilistically, with no mechanism for complete delivery of all packets to all group members.

4.3.7 RAPID

The Reliable Probabilistic Dissemination protocol (RAPID) [DFKS06] utilizes probabilistic packet forwarding with deterministic recovery mechanisms to achieve high reliability. Design of the protocol is based on the conclusions that: 1) a node's forwarding probability should be inversely proportional to the number of nodes in its one-hop neighborhood at a given time, and 2) the forwarding probability should be kept to a relatively low value, based on formal analysis of an optimal number that is low but ensures "good" reliability.

The designers state two assumptions in their paper that would seem unrealistic regarding ad hoc wireless networks: 1) ad hoc networks are continuously connected, and 2) in wireless networks most message losses are caused by packet collisions. Assumption 1 is unrealistic since in very sparse and even moderate density networks, network partitioning is an ongoing issue to deal with. Assumption 2 is also false due to the network links lost to partitioning.

This initial probabilistic packet dissemination scheme works by nodes, on receiving a packet, probabilistically determining if they will broadcast it or not. This mechanism is then deterministically assisted with timer based corrections, to either cause a node about to forward a packet to not forward after a timeout if it receives the packet from a second node during the timeout, or to cause a node not forwarding to decide to forward after a timeout if it does not receiver the packet from a second node during the timeout.

A deterministic mechanism is also used for packet recovery. Every node is required to gossip with all nearest neighbors via broadcast, sending received packet headers, and allowing nearest nodes to request missed packets. This mechanism is deterministic in that all 1 hop neighbors are included, and all could possibly reply with requests, though the same two timer based corrections are in operation for packet resends. In packet resends, a node deciding to resend a requested packet will not if it overhears a neighbor's resend, and a node deciding not to resend will change its actions if it doesn't overhear a neighbor node resending. No two hop neighborhood information is used in RAPID. Nodes purge received messages via timeouts, in order to avoid unbounded required memory problems during protocol operations.

4.3.8 EraMobile

EraMobile [GO07] is a gossip based multicast protocol that does not require the maintenance of any tree or mesh type structure, nor a global or partial view of the network, nor any neighbor node or group member information, or even a formal routing protocol for data dissemination. EraMobile disseminates data packets purely through a gossip mechanism, yet has the goal of providing fully reliable data delivery. Once the source has sent a data packet, dissemination occurs purely through nodes periodically gossiping with nearest neighbors only. The gossip packets contain the digest of the sender's stored data packets. A nearest neighbor receiving the gossip packets can reply with a packet requesting data it is missing, and on receiving this, the gossiping node will send the packets back. With this technique data delivery latency is greatly increased, but the authors state that the protocol is not intended for low latency applications, and in addition to the benefits of reliability, control overhead is very low. The number of packets requested, as well as the number of requested packets sent, is limited in any given round in order to not congest the network with the resent packets. Due to this it could take several rounds for a requestor to send out all its requests, spreading the control and network bandwidth overhead over time. A receiver queues received packets in FIFO order, delivering them to an upper layer when no gap exists, or when missing packets are declared lost.

EraMobile has a data dissemination unit that distributes packets through gossip messages with no multicast or unicast protocol. Each gossip message carries the digest of the sender's data buffer contents. The rate of gossip broadcasts is controlled by a parameter. A receiver, on getting a gossip message, examines the contents then sends back a request packet with a listing of packet sequence numbers needed, with the list count limited by a parameter, to keep the packet small. Single packets are then forwarded to the requestor. Sequence numbers are aged out of gossip message inclusion over time.

EraMobile has a buffer management unit that maintains protocol buffers at a node, performing data delivery in FIFO order. It also has an adaptivity unit that listens for gossip and request messages to determine the number of one hop neighbors. In low density areas nodes send out gossip messages more frequently to take advantage of more ephemeral links, while in high density areas the rate is decreased in order to not waste limited bandwidth.

Given the basic probabilistic mechanism EraMobile uses to make decisions on packet forwarding at each link, a large increase in latency is unavoidable, and must be accepted. The tradeoff is the low amount of control overhead, also a direct result of the probabilistic mechanism for packet forwarding.

4.4 Reliable Multicast Performance Comparisons

4.4.1 Reliability Protocol Performance Modeling

Without exception, the 2001 and later research papers presenting multicast MANET protocols that focused on reliability as the primary feature include a performance evaluation section where the authors implement their protocol in a MANET network simulator to study its reliability and other performance properties. The three network simulators mentioned in section 2.4.1 are the most commonly used, though two of the reliable multicast protocols discussed here use other network simulators. These two relatively new simulators are RELSIM and JIST/SWANS.

- RELSIM [TG01] – A distributed mobile ad hoc network simulator developed in 2001 to test reliability properties of routing algorithms in MANETs. The

simulator is developed in Java with CORBA providing the infrastructure, and uses multithreading to simulate actions of individual nodes. This simulator was developed by the creators of the RMA protocol.

- JIST/SWANS [CU] – this simulator was developed in 2004 by Cornell University. JIST (Java in Simulation Time) is a discrete event simulation engine that runs over a standard Java virtual machine, while SWANS (Scalable Wireless Ad hoc Network Simulator) is built on top of the JIST platform, designed specifically for large scale network simulations.

Typically the new reliable protocols are compared to one or, at most two, other 'best effort' protocols, usually selected because they are well documented and known to have high reliability. The most commonly selected protocols used for comparison are flooding, ODMRP and MAODV.

- Flooding – this basic protocol is commonly known to have one of the highest broadcast reliability metrics of any best effort protocol. This fact first came to light in the research comparisons described in section 2.4.3, and since then has been commonly used as a standard of comparison for MANET multicast reliability protocols.
- ODMRP – this multicast mesh based MANET protocol has been well documented, simulated in multiple studies, and shown to have one of the best reliability metrics of existing multicast 'best effort' protocols. Furthermore, mesh based topologies have been shown to provide higher reliability in general, given the redundant packet delivery inherent in the topology.
- MAODV – this multicast tree based MANET protocol is also well documented and simulated in multiple studies. While it is generally accepted that tree based multicast protocols do not provide as high reliability as mesh based protocols in many scenarios given the relatively fragile nature of the tree topology, there are specific scenarios where multicast tree based protocols can have higher reliability.

While several multicast MANET protocols featuring reliability have been developed and documented since 2001, to date there has not yet been a paper presentation that has compared any against each other, like the two comparisons described in section 2.4.3 have done for 'best effort' multicast MANET protocols. I believe this is partially because the reliable multicast protocols developed to date tend to be more complex and so new as to be not yet available to public access, and because efforts are still ongoing in developing new techniques for reliability, so no one has yet taken a step back and taken the time to implement any of these protocols strictly for purposes of comparison.

Below is a listing of the reliable multicast MANET protocols discussed in the next section, showing the simulators used for implementation and comparisons, along with the protocols selected for comparison. Also listed is a note about whether the protocol was designed as a standalone multicast communication protocol, or as a reliability mechanism to be used on top of another MANET protocol.

- RMA – implemented in the RELSIM simulator
  - Designed as a standalone multicast protocol
  - Compares performance to MAODV
  - Unique feature: "link lifetime" metric used for routing.

- RALM – implemented in the QualNet simulator
  - Can run on top of other multicast protocols, ODMRP, AODV.
  - Uses source constriction, source based resend.
  - Compares performance to UDP and SRM (both on top of ODMRP and AODV)

- ReACT – implemented in the QualNet simulator
  - Enhancement of RALM, extends it with a 'local recovery' feature, still running on top of ODMRP and AODV.
  - Compares performance to RALM only

- Scribble – implemented in the GloMoSim network simulator
  - Designed as a standalone multicast protocol
  - Compares performance to ODMRP

- AG – implemented in the GloMoSim network simulator
  - Runs on top of any multicast protocol
  - Compares performance to MAODV

- RDG – implemented in the ns-2 network simulator
  - Runs on top of the DSR unicast protocol
  - Compares performance to AG

- RAPID – implemented in the JIST/SWANS network simulator
  - Designed as a standalone multicast protocol
  - Compares performance to flooding and GOSSIP3 (though only a few charts compare performance to flooding)

- EraMobile – implemented in the ns-2 network simulator
  - Designed as a standalone epidemic-based multicast protocol
  - Compares performance to flooding for reliability and MAODV for overhead consumption

4.4.2 Reliability Evaluation Metrics

The same 3 core evaluation metrics listed in section 2.4.2 used to evaluate multicast MANET protocol performance are used to evaluate reliable multicast MANET protocol performance. The three are:

1. packet delivery ratio – the number of packets received by all receivers over the total number possible to receive (a.k.a reliability).
2. data and control overhead – the amount of data bytes and control bytes sent over the network over data bytes delivered, in order for the protocol to operate.
3. data delivery latency – the amount of time in between a source sending a packet and a receiver receiving it, averaged over all receivers, averaged over all packets sent.

In general, ongoing research reflects that these three metrics are interdependent, and tightly linked in protocol operations. Usually, when a protocol is designed to enforce stricter reliability, either overhead consumed, or latency, or both will suffer. Though there have been several methods designed to strengthen reliability to date, all have come with some cost associated with these other two metrics.

4.4.3 Reliability Simulations and Results

4.4.3.1 RMA

RMA was implemented in the RELSIM simulator, and compared to MAODV, a multicast version of the AODV protocol. Three metrics were used to compare performance between the two protocols, packet delivery ratio, data overhead (total count of data packets in the system vs the number of data packets delivered) and control overhead (count of control bytes vs the count of data packets delivered). Simulations were performed with an environment of 50 nodes in a 1000m x 1000m area, each with 200m broadcast range, with a random waypoint motion model. No mention was made as to the length of time the simulations ran, or the number of senders and receivers among the 50 nodes. Fixed resting times of 10 seconds and 50 seconds were used. Since both numbers are relatively high for resting times, only the 10 second resting time charts will be discussed here. Even a 10 second resting time tends to make for a fairly static network. In Figure 4.1 shown below, RMA has a packet delivery ratio near 100%. This is compared to MAODV which starts, at low speeds, at over 95%, but drops steadily as node speeds rise. Finally at 50 m/sec MAODV has ~63% reliability at 10 seconds of motion time per 10 seconds of resting time, and ~55% reliability at 10 seconds of motion time per 50 seconds of resting time. Results were also shown where the hopcount metric is used for RMA path selection instead of Link Lifetime metric, but results are similar, so only Link Lifetime results will be discussed here.

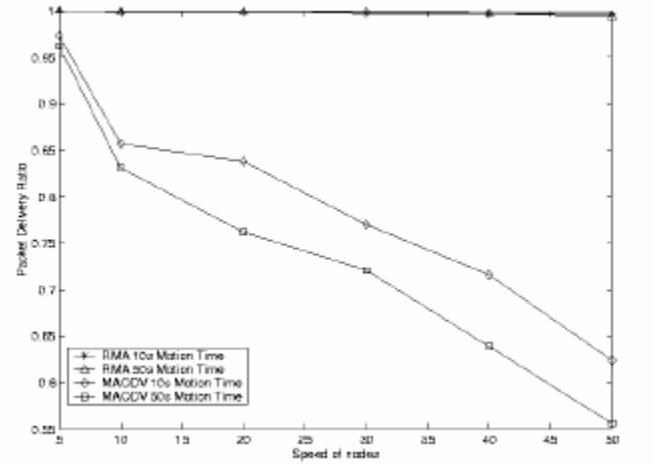


Figure 4.1: Packet delivery ratio vs Speed at 10s rest time using Lifetime metric

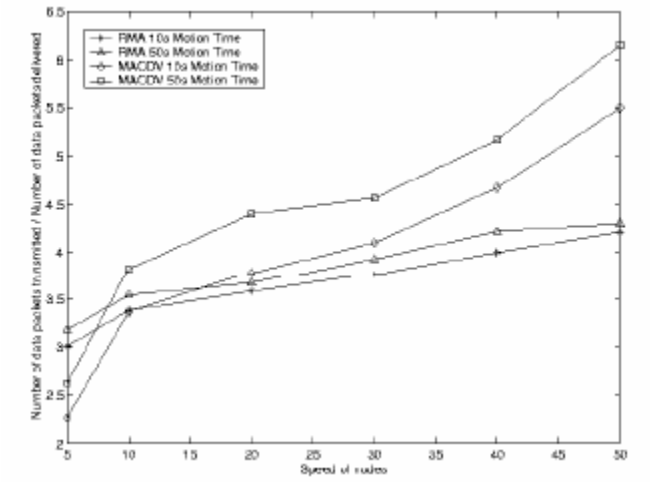


Figure 4.2: Data Overhead vs Speed at 10 sec rest time using lifetime metric

Data Overhead results shown in Figure 4.2 also showed advantages for RMA over MAODV. As node speeds rose, RMA overhead ramped at a low but steady rate, while MAODV overhead rose at a much faster rate.

Figure 4.3 shows that control overhead also rising slowly for RMA as node speed rises, while it rises at a much faster pace for MAODV with higher node speeds.

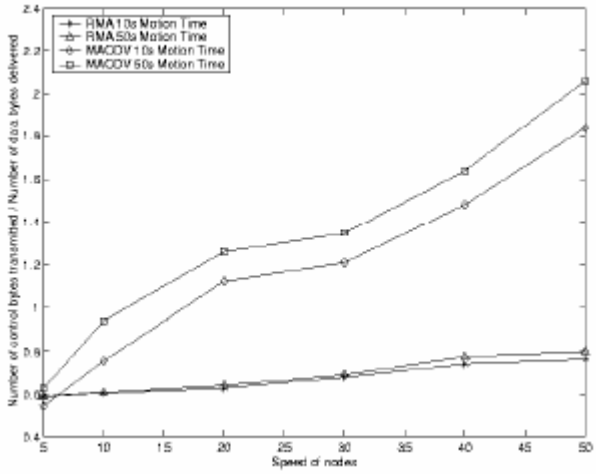


Figure 4.3: Control Overhead vs Speed at 10 sec rest time using lifetime metric

Several factors account for the differences in performance of the two protocols, and the overall results shown for RMA. The major distinction between the protocols is that RMA is designed for reliability, relying on missed packet retransmissions to increase

packet delivery ratio. MAODV is best effort, and packets missed from initial delivery cannot be made up for.

Next, as mentioned previously, 10 seconds pause time is significant for overall node motion definition. For the 10 second mobility time, this means half the scenario time all nodes are essentially static. Intuitively, since RMA relies on retransmissions for reliability, a very stable network provides breathing space that could account for a tremendous boost to overall reliability. Since MAODV is best effort, it can only benefit partially from this periodic stability.

The authors mention that they combine data and control information together, with only a few standalone control packets, namely MACKs and BMACKs. Further, these standalone packets are much smaller than the standalone control packets in MAODV, namely RREQ's, RREP's and MACT's. Since higher mobility requires more control packets for both protocols to account for a greater number of broken links, this difference becomes more significant with rising node speeds. The new "link lifetime" metric does not seem to perform appreciably better in the side by side performance comparisons with scenarios where the hopcount metric is used instead.

RMA's routing path creation algorithm would appear to construct graphs similar to the meshes created in ODMRP, since broadcast is used to reach all receivers with unknown routes. This broadcast mechanism has been shown to provide high reliability in scenarios such as this one, 50 nodes in a 1000 x 1000 meter area. It is likely that if the node count were increased to form dense scenarios, the negative effects of broadcasting, both for source retransmitted data packets and for receiver sent BMACK acknowledgements, would result in decreased reliability.

Another factor that the performance scenarios were not able to draw out was the known issue of the necessity of receiver node interactions with the source resulting in decreased reliability. In this case, all nodes are required to send ACK packets back to the source, an interaction that severely impacts scalability, resulting in decreased reliability as scenarios include either much larger networks or more dense networks. The scenario details also did not include how many packets per second were transmitted. The

broadcasting would have little to no impact at very low data rates, but would have a potentially large negative impact at high data rates.

4.4.3.2 RALM and ReACT

ReACT is an extension of RALM, so performance of both will be discussed in this section. Initially, RALM was created as a transport layer protocol running on top of any multicast protocol. RALM's contribution is twofold. On the one hand, it provides a congestion control mechanism to increase reliability by reducing the source's packet send rate when packet loss occurs in the network, and on the other hand it provides a source based packet retransmit mechanism for receiver nodes that have missed reception of packets during the initial send process.

RALM was compared to UDP (unreliable packet send) and SRM [FJMLZ97], all running on top of ODMRP. The UDP comparison was used as a basic multicast protocol with no reliability guarantees, and SRM was used as a basic error control oriented reliable multicast protocol. The QualNet simulator was used for the comparisons, in scenarios with 50 nodes in a 1500 x 1500 meter area (a density of ~22 nodes in a 1000 x 1000 meter area). 512 byte packets were sent over channels with 2Mb/second capacity by nodes with a maximum radio range of 375 meters. MAC 802.11b DCF was the MAC protocol used, and random waypoint was used for the mobility model, where mobility was added. Packet delivery ratio, control overhead and packet delivery latency were examined for the scenarios.

Three scenarios were tested: varying network traffic rate, varying the number of sources and varying the node mobility speed. First, in the varying traffic rate scenario, nodes were stationary. There were 5 senders and 10 receivers. Packets were sent at rates varying from 2 packets/second to 10 packets/second. Figure 4.4 below shows RALM achieving full reliability, and outperforming UDP and SRM in terms of control overhead and latency, shown in Figures 4.5 and 4.6.

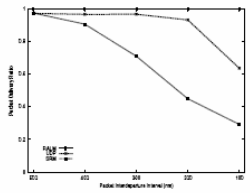


Figure 4.4: Reliability

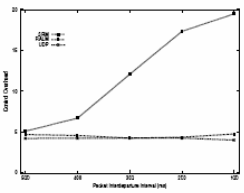


Figure 4.5: Overhead

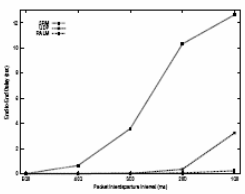


Figure 4.6: Latency

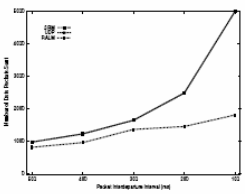


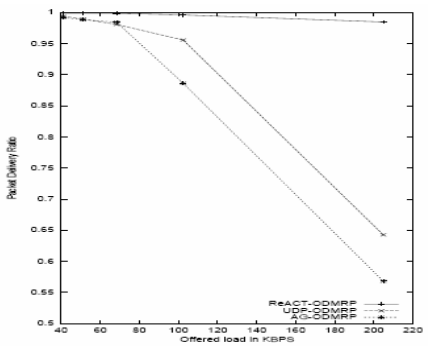
Figure 4.7: Source data rate

While RALM appears to perform well in terms of reliability, overhead and latency, it must be kept in mind that the central feature of the protocol by necessity results in a slowing of the source data rate. RALM assumes packet loss to be the result of congestion, which is not necessarily the case for many scenarios, including sparse networks or networks with high node mobility. Figure 4.7 shows the true source data rate for RALM in the test point to the far right, where the unconstrained source data rate at a packet interdeparture rate of 100ms was 10 packets/second, RALM's source data rate was restricted to ~ 2/5 of this, or about 4 packets/second. At this rate, UDP results are very similar to RALM's across all three metrics. Results here are questionable, however. In the traffic rate scenarios all nodes were kept stationary, a condition favorable to RALM since non-congestion factors for missed packets (i.e. missing, broken or ephemeral links) are greatly reduced, and congestion is in fact generated over time as the source packet send rates increase.

A second scenario of increasing number of sources is also presented, where both the multicast source count and the receiver count increase from 10 to 40. Here the source send rate is held constant at 2 packets/second, for an unknown length of time. Again, nodes are kept stationary throughout all test points in this scenario, so this scenario also exhibits a condition favorable to RALM, since congestion is increased over time with non-congestion factors for missed packets minimized. In this scenario packet delivery ratio only is shown, with no results for overhead or latency, or the reduction of source data rate. In the delivery ratio chart, RALM outperforms UDP and SRM to a greater degree as source count increases.

A third scenario of increasing mobility is presented, where node speed is varied from 0 meters/second to 50 meters/second, using the random waypoint model with 5 multicast sources and 10 receivers. No indication is made as to the node pause time between directional movements, or overall simulation time. While nodes are allowed to move in this scenario, results are shown for packet delivery ratio only, not for overhead or latency. Here RALM slightly outperforms UDP and SRM, since all three protocols have fairly high reliability over all the test points. The cost of RALM's high reliability, in terms of the constriction of the source data rate, is not shown either.

ReACT's implementation of source constriction changes some details of RALM's but has basically the same effect, however ReACT introduces a relatively weak local recovery scheme. Its published simulation data reflect the consequences of this. When compared to RALM, ReACT showed a significant decrease in source data rate constriction, and resulting increase in source 'goodput', defined as throughput of packets that were reliably received by all receivers. ReACT was also compared to ODMRP in a 'congestion' scenario, with increasing traffic rate. In this scenario nodes were again kept stationary, and only packet delivery ratio, goodput, overhead and packet dropped counts were measured. ReACT has a much higher reliability metric for increasing data traffic, but a lower 'goodput' metric than ODMRP. In other words, of the packets the source sent in both protocols, a greater percentage were reliably delivered in ReACT, but in ODMRP since the source had no transmission rate constriction and sent far more data packets, the result was a greater number of packets reliably delivered to all receivers, as shown in Figures 4.8, 4.9 and 4.11.



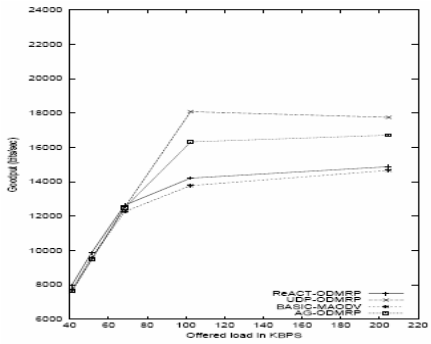
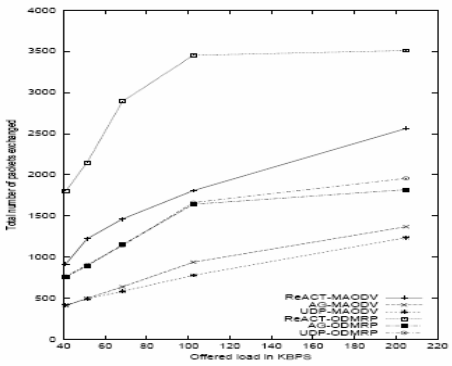


Figure 4.8: Congestion Pkt Dlvry Ratio Figure 4.9: Congestion Goodput



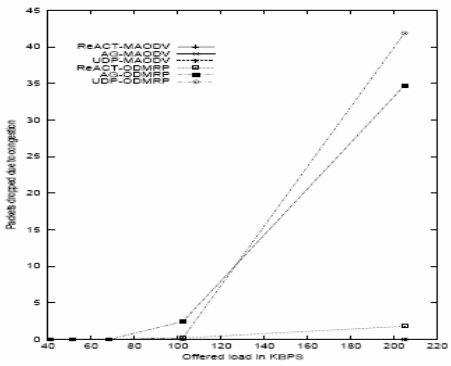
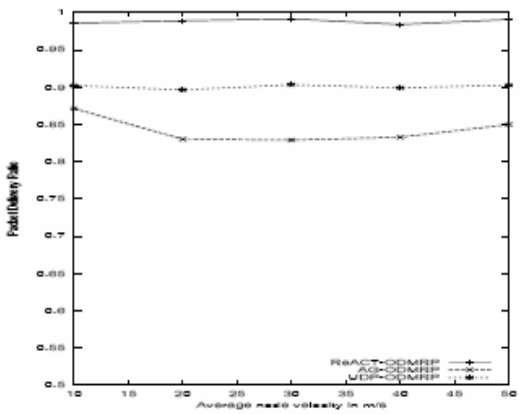


Figure 4.10: Transmission Overhead **Figure 4.11: Congestion Control**

Figure 4.10 shows the total number of packets exchanged. Here, it is apparent that ReACT's source congestion comes at a cost of much greater packet sent overhead through the network, both for local recovery and source based constriction.

ReACT was also tested in a mobility scenario, with results described below. A random waypoint mobility model was used, with node speed varying from 20m/sec to 100m/sec, with a 10 second pause time. No mention was made as to the packet rate. Here, the packet delivery ratio for ReACT is significantly higher than for UDP over ODMRP, but source rate constriction due to broken links is reflected in the 'goodput' metric, where UDP on ODMRP has a significantly higher packet delivery ratio than ReACT. No mention was made as to overhead or latency metrics either.



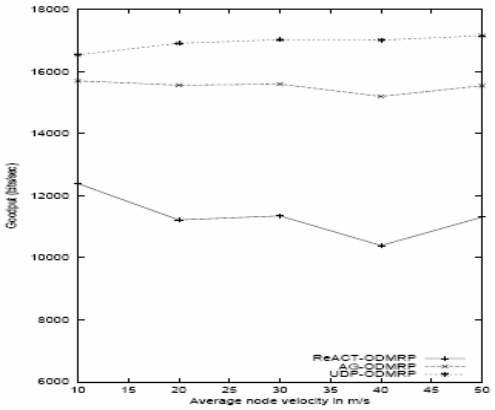


Figure 4.12: Mobility Pkt Dlvry Ratio **Figure 4.13: Mobility Goodput**

The authors did not test very sparse network or dense network scenarios at all. The network stresses related to sparse networks could interact with ReACT's source based congestion control to severely restrict the amount of data sent, since multiple ongoing network partitions are common occurrences, with many nodes missing many packets.

ReACT's local recovery scheme always looks for a node to recover packets from an upstream neighbor. It does not account for the fact that in mobile networks all nodes move in random ways: chances are good that one node upstream from another may move to a downstream position, or both may move sideways to totally new relationships with new nodes. In reality, at any instant, given a pair of nodes, its quite possible the downstream node has a recent history of greater reliability than the upstream node.

4.4.3.3 Scribble

Scribble was compared to the best effort multicast protocol ODMRP using GloMoSim for k = n (i.e. all nodes in the network, deterministically reliable multicast). The simulation environment was 50 nodes, 1 sender and 50 receivers, in a 1000 x 1000 meter area, and ran for 3000 seconds. Node movement used the Random Waypoint mobility model, with a pause time of 0 seconds, discarding the first 1000 seconds worth of data. Data rate was set to 1 512 byte packet / second for the first 500 seconds, a very light load. Two scenarios were executed, one with increasing mobility (from 1 to 35 meters/second) and one with increasing wireless radio range (from 150 to 350 meters). Three metrics were presented, packet delivery ratio, latency and transmission overhead.

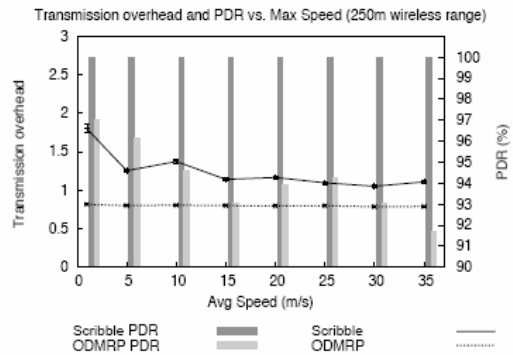


Figure 4.14: Mobility Overhead And Pkt Delivery Ratio

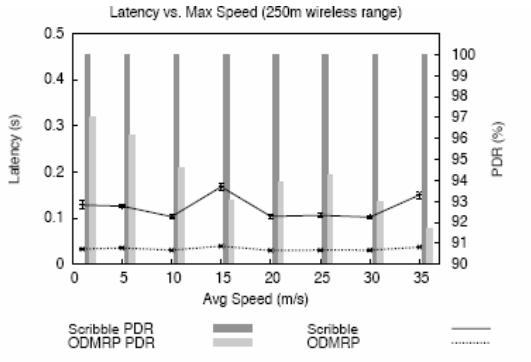


Figure 4.15: Mobility Latency

In the increasing node mobility scenario, Figure 4.14 shows Scribble to have fully reliable packet delivery at this low data rate, while ODMRP varies from ~97% to ~92%, dropping as mobility increases. Scribble's overhead is nearly double that of ODMRP's at low speeds, dropping to be just above ODMRP's at the higher speeds. The author states that this is due to the fact that at lower speeds nodes remain in a partitioned state longer, requiring a greater amount of overhead to eventually deliver all packets to all receivers. Figure 4.15 shows that Scribble's latency is higher than ODMRP's. The authors state that this is due to the longer time Scribble takes to deliver to previously unreachable nodes that ODMRP fails to deliver to altogether.

In the decreasing radio transmission range scenario, decreasing radio range leads directly to a rise in overhead and latency for Scribble, though it retains full packet delivery ratio. ODMRP's packet delivery ratio drops to ~50% at the lowest radio range. Its latency and overhead metrics are deceiving, because they are collected only from packets that were delivered to receivers, likely the majority of which were located much closer to the source in the scenario. At high radio range, transmission overhead drops to equal ODMRP's, likely because of less partitioning, and all packets being successfully received as a result of the first send for nodes within range.

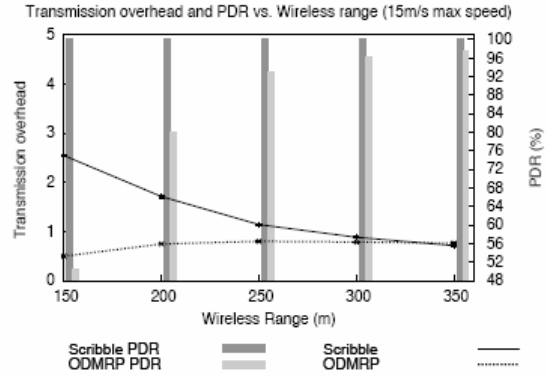


Figure 4.16: Radio Range Overhead And Pkt Delivery Ratio

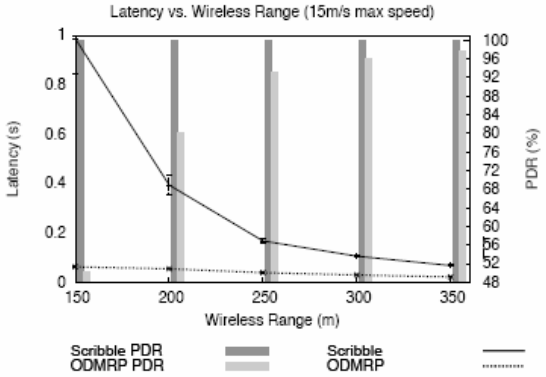


Figure 4.17: Radio Range Latency

It would be interesting to see metrics as the data traffic load was increased, and as true network density increased. In the simulations, radio range increases approximated increasing network density, but true network density may have different effects on

reliability, latency and overhead. Also, the 'push' model utilized by Scribble likely consumes more overhead than other receiver based 'pull' models, especially, as shown, for low radio range (sparse) and low mobility networks. Also, extremely sparse networks would likely cause the overhead and latency to greatly increase, since 'pushing' data under those circumstances would be resource intensive. Finally, Scribble assumes that some level of 'group membership' is known, even if only the number of group receivers.

4.4.3.4 AG

Anonymous Gossip was implemented in the GloMoSim network simulator on top of MAODV, and compared to basic MAODV. A 200 x 200 meter area was used, and the random waypoint mobility model was selected, with pause times uniformly distributed between 0 and 80 seconds. Simulations were run for 600 seconds. The MAC layer protocol used was 802.11b with 2Mb/sec channels. Each receiver could send at most one gossip request per second, requesting at most 10 messages at a time. Each receiver could store up to 200 missed packet id's, and up to 100 received packets. The source sent 64 byte packets at the rate of 5 pkts/sec, starting at 120 seconds (allowing MAODV time for topology formation) and ending at 560 seconds.

There were three scenarios run. One varied transmission range, one varied node max speed and one varied the number of nodes in the network. Figure 4.18 shows packet delivery ratios for transmission range variations from 45 to 85 meters where nodes move at a maximum speed of 2 meters/second, a relatively slow rate. Figure 4.19 shows packet delivery ratios for node speed variations, where nodes move at speeds ranging from 1 to 10 meters/second, still quite slow. No metrics were reported either for control overhead or for latency. From the packet delivery ratios reported, it is clear that, while Anonymous Gossip has higher reliability than MAODV, its reliability is weaker than deterministic protocols. Since it operates with a probabilistic mechanism, and since the gossip routes are guided by the topology established by MAODV, reliability is unpredictable, and cannot be guaranteed. Being probabilistic, the expectation is that control overhead would be less than a deterministic protocol while latency would be greater.

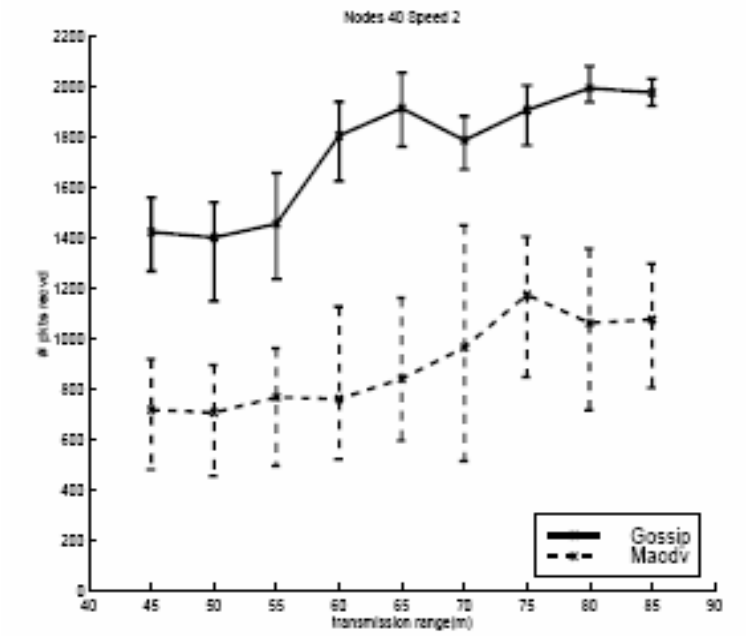


Figure 4.18: Pkt Dlvry Ratio with transmit range variation

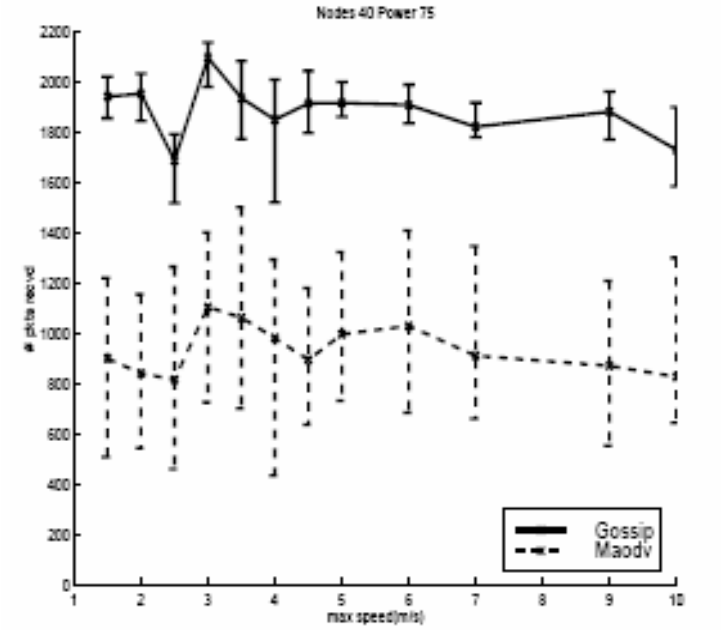


Figure 4.19: Pkt Dlvry Ratio with max speed variation

4.4.3.5 RDG

RDG was implemented on the ns-2 network simulator, using a 1000 x 1000 meter area populated with 100 to 200 nodes with 250 meter transmit ranges. A random

waypoint mobility model was used with a maximum speed varying from 2 to 20 meters/second, and average pause time of 40 seconds. 64 byte packets were sent at the rate of 5 packets per second, for 280 seconds. RDG was compared to AG in a very limited simulation. In this simulation, the large transmit range, relatively long pause time and relatively low node speed all call into question the protocol's true performance characteristics, especially under specific network scenario stresses such as high node speeds, dense networks or high traffic rates. For the scenario presented, at a speed of up to 10 meters/second, RDG achieved a slightly higher packet delivery ratio than AG. No results were presented for control overhead or delivery latency. A second simulation where node speeds were increased to 20 meters/second showed RDG with a packet delivery ratio of ~88% at the 20 m/s data point.

4.4.3.6 RAPID

Initial packet delivery latency is an issue for RAPID, since each node along each data path uses the timeout delay mechanism to decide whether or not to forward the packet. Of course the gossip mechanism adds latency on top of this for packets that are initially missed and must be resent. Sparse networks are another problematic area for RAPID, with results showing a 69% delivery ratio for a sparse simulation. In sparse networks, the frequent and ongoing network partitions disrupt both the initial probabilistic packet sends and the gossip request mechanism. This issue is magnified with the gossip mechanism limited to a single round in the simulations. Both the gossip mechanism and the requirement for nodes to broadcast heartbeat messages to neighbors, as input to all node's neighbor count based probabilistic calculation to send new packets contribute to the protocol's control overhead consumption of network bandwidth.

RAPID was compared to GOSSIP-3, a probabilistic dissemination protocol with no means to recover missing packets, in simulations using the JiST/SWANS simulator. The Random Waypoint mobility model was used for node movement, with a pause time of 0 seconds and the first 1000 seconds of simulation data discarded. Number of broadcasting nodes ranged from 1 to 200, sending ten 512 byte packets followed by a cooldown period before simulation termination. Gossip rounds were limited to one per packet.

The results are difficult to evaluate, since each simulation had from 1 to 200 senders sending a total of 10 messages each, followed by the 'cool down' period for the gossip request mechanism to operate before simulation termination. Ongoing initial data delivery, and the gossip request/delivery mechanism metrics may change substantially if allowed to operate over an extended period.

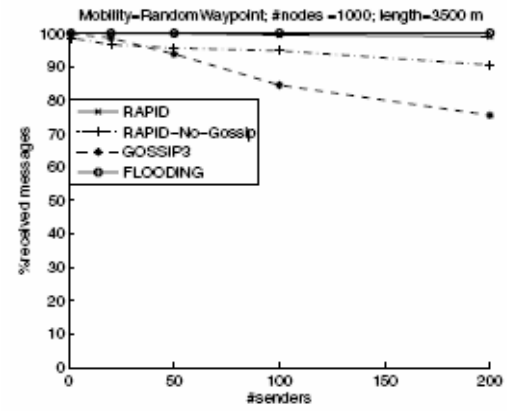


Figure 4.20: Packet Delivery Ratio

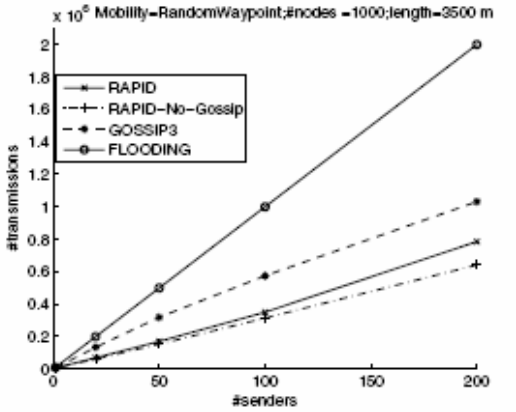


Figure 4.21: Network Load

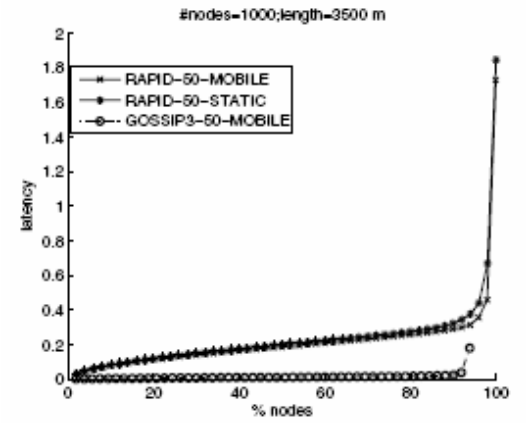


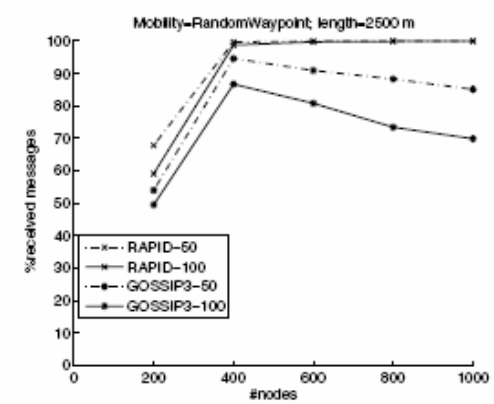
Figure 4.22: Latency

Figures 4.20, 4.21 and 4.22 are for a simulation where a network of 1000 nodes broadcasted to each other with a transmit range of 200 meters in a 3500 x 3500 meter area. This represents a medium density of nodes, the equivalent of about 80 nodes in a 1000 x 1000 meter area, corresponding to about 10 neighbors per node. Packet delivery ratio for RAPID was 99.9%, just under the 100% of flooding, with overhead less than that of

flooding. Figures A and B represent simulations where nodes are static. Figure C showed latency specifically with 50 broadcasting mobile nodes. Here, Gossip-3 is significantly faster than RAPID, but does not deliver to 100% of all nodes, since it has no retransmit mechanism.

There was a simulation varying mobility, but only two speeds were modeled: static nodes, and nodes moving between 1 and 5 meters per second. This was not a good selection that would give meaningful results about the effects of mobility on RAPID packet delivery.

Network density changes were modeled by changing the number of nodes from 1000 to 200 in a 2500 x 2500 meter area, about 4 nodes per neighborhood or about the equivalent of 31 nodes in a 1000 x 1000 meter area, RAPID's reliability goes down to about 69% with 50 broadcasting nodes, compared to GOSSIP-3's packet delivery ratio of 51%. Results for this set of simulations are shown in Figures 4.23 and 4.24. The authors state the decrease in reliability is due to the poor network connectivity. Further, they state that at this node density "no protocol can achieve high delivery ratios", though this is disputable. RAPID's packet delivery and retransmission mechanisms have a true weakness in sparse networks.



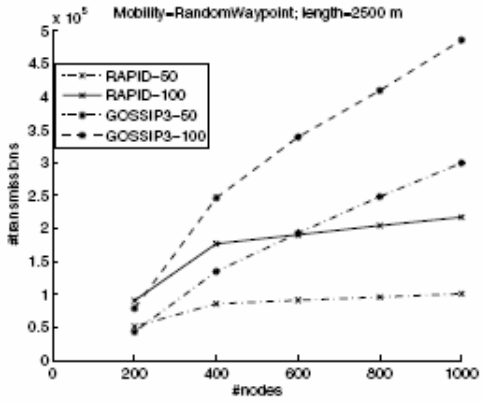


Figure 4.23: Packet Delivery Ratio **Figure 4.24: Network Load varying density**

4.4.3.7 EraMobile

EraMobile was compared to flooding and MAODV. Flooding was selected due to its acceptance as one of the most reliable protocols, and MAODV was chosen due to its low

overhead requirements. Ns-2 was selected as the network simulator, using the IEEE 802.11 MAC protocol with 2Mbps node throughput and a range of 250m. The random waypoint mobility model was used. Total simulation time was set at 1000 seconds, with a single sender starting to send after 20 seconds, sending continuously through 870 seconds. The final 110 seconds were used to disseminate packets through the gossip mechanism. Traffic rate was 2 packets of 512 bytes per second. Area size was not stated.

Three metrics were measured: throughput (packet delivery ratio), reliable throughput (fraction of packets that were received by all receivers) and receive overhead (ratio of bytes belonging to control packets and duplicate data packets received by a node to the bytes of data it delivered, i.e., all bytes received except the bytes of data packets received for the first time, are counted as receive overhead). Not measured in this work was delivery latency.

The authors state that poor latency is a known issue with EraMobile, and a tradeoff for increased reliability. EraMobile is not intended for delay sensitive applications. Several factors tradeoff EraMobile latency in order to enhance the other performance metrics, for example, the basic gossip transmission mechanism delays packet delivery at each node. Further, for missed packets, gossip based requests not only increase latency, but are limited to a fixed maximum number per round in order not to congest the local network. Multiple rounds may be required for a node to receive all packets missed from initial delivery, greatly increasing the latency for eventual delivery.

Mobility Results: the mobility scenarios ranged from 1 to 30 m/s, not a very high speed. The network was made up of 50 nodes in a single group. EraMobile was fully reliable, with flooding following closely in second place. MAODV suffered in terms of reliability as node speed increased, due to the fragile tree topology MAODV implements. EraMobile showed extremely low overhead compared to both flooding and MAODV, since packets are transferred solely through the gossip mechanism, consuming very little network bandwidth. Flooding showed the worst overhead, due to its highly redundant data paths.

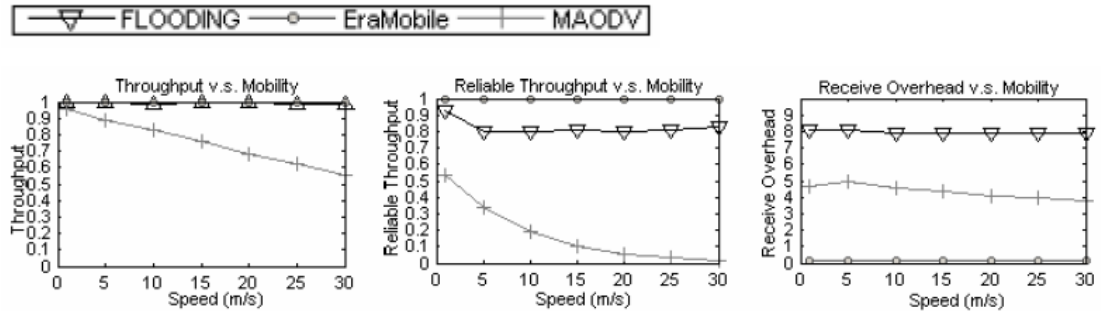


Figure 4.25: Throughput Figure 4.26: Reliable Throughput Figure 4.27: Overhead

Group Size Results: group size scenarios ranged from 10 to 50 nodes. For these scenarios EraMobile shows near fully reliable delivery for almost all scenarios, while flooding and MAODV show a substantial drop as the scenarios become more sparse at 20 node and 10 node networks. Overhead drops for these two protocols at the more sparse networks also, though for EraMobile it remains very low, just a fraction of the other two. As density rises EraMobile overhead increases, but by a negligible amount.

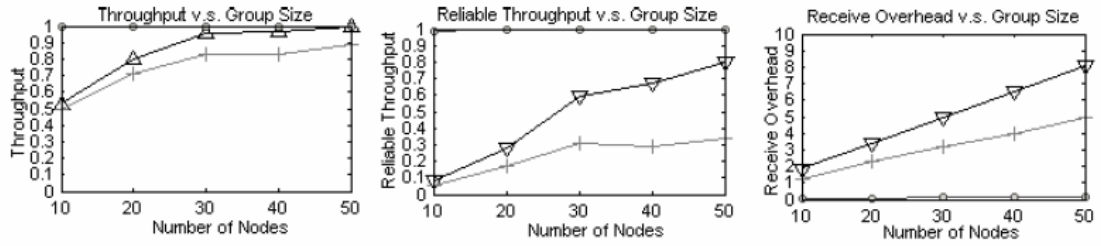
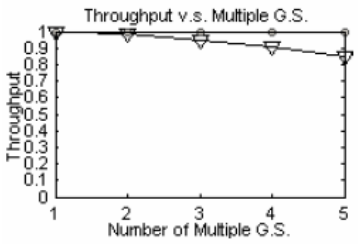
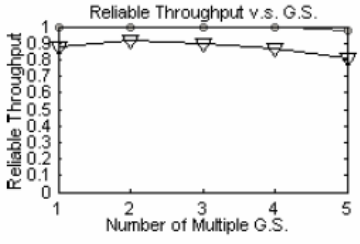


Figure 4.28: Throughput Figure 4.29: Reliable Throughput Figure 4.30: Overhead

Group Number Results: group number scenarios ranged from 1 to 5 groups, with one sender for each group. Node count was fixed at 60, with 5m/s node speed. Due to this increasing number of packets in flight across the network, flooding showed a noticeable drop in throughput as the group count went up. Receive overhead for flooding rose sharply as well, with flooding's nodes relaying many data packets that end up being not delivered. In EraMobile the increasing number of groups had a lesser effect, in that nodes receiving packets from other groups will just drop them, reducing the overall impact to receive overhead, but still showing some effect.





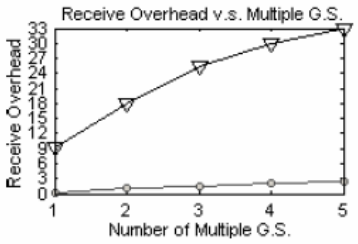


Figure 4.31: Throughput Figure 4.32: Reliable Throughput Figure 4.33: Overhead

4.5 New Reliable Multicast Routing Strategy

4.5.1 Goal

From the performance results described in section 4.4, it is apparent that in general, protocol design approaches tend to sacrifice performance in one key metric in order to strengthen another. The designers of EraMobile, for example, declare that the reliability of their protocol comes with a known cost of a long latency in packet delivery. Though there have not yet been a large number of ad hoc multicast protocols developed that focus on reliability, enough have been developed to both allow for an initial categorization of design approaches, described in the next subsection 4.5.2, and to support initial conclusions about how these design approaches, and the mechanisms making them up, translate into performance tradeoffs between the three key interdependent performance metrics of reliability, network overhead and delivery latency, described in the following subsection 4.5.3. Since the related reliable multicast protocol descriptions and evaluations left many unanswered questions as to true performance across a wide variety of scenarios (with measurements not taken at all, taken from ideal scenarios or taken from scenarios with only one type of stress to communication) some of what follows in the conclusions section 4.5.3 is conjecture. In order to get a complete picture of protocol performance, testing needs to occur across a variety of network stresses.

The overall goal for this dissertation then, is to enhance reliable multicast in ad hoc wireless networks using the following two guidelines:

1. The protocol must balance the requirement for reliability with a desire to minimize both network overhead and packet delivery latency (i.e. not

sacrifice performance in one key metric to strengthen another any more than is necessary).

2. The protocol must perform well under all types of network communication stresses: links overloaded with contention in dense and high traffic rate networks, broken and ephemeral links in high mobility networks and ongoing network partitioning in sparse networks.

An initial approach for development of this protocol is discussed in the final subsection of this chapter. This initial approach forms the basis for the resulting protocol, Reyes, which is fully described in chapter 5.

4.5.2 Categorization of Existing Approaches

In general, existing reliable multicast protocol designs can be broken down into two building blocks. The first building block is the mechanism for system wide initial dissemination of source generated data packets, and the second is the mechanism for individual receivers that missed specific packets during initial dissemination to recover those missed packets. Some reliable protocol designs, such as Scribble, consist only of the first building block, requiring the initial dissemination mechanism to provide reliable delivery to all receivers, but most have a secondary mechanism for packet recovery.

For the first building block there are several categories that the reliable multicast protocols discussed in section 4 can be grouped into. These include:

- Flooding based Packet Dissemination – has some drawbacks, but some advantages as well.
- Topology based Deterministic Global Packet Dissemination – this broad category encompasses the initial dissemination mechanisms for RMA, R-ODMRP and most tree, mesh or dynamic graph based data path creation techniques. Dissemination of source packets occurs along data paths periodically defined by a path creation mechanism utilizing flooding or some other means of network path creation.
- Non-topology based Deterministic Local Packet Dissemination – This category represents unique strategies for disseminating packets that are

designed to guarantee delivery to all receivers. In this category data paths are not predefined according to any topological creation mechanism. Scribble and EraMobile fall into this category, with no specific topology used for data dissemination. Both these protocols have no secondary mechanism for missed packet recovery, relying instead on the initial packet dissemination mechanism to provide reliability.

- Probabilistic Packet Dissemination – RDG and AG fall into this category using probabilistic mechanisms for initial packet dissemination.
- Congestion Control – RALM and ReACT fall into this category, which represents a way of modifying initial packet dissemination to optimize reliability.

For the second building block, missed packet recovery, there are several categories that the reliable multicast protocols discussed in chapter 4 can be grouped into, including:

- Source Based Deterministic Recovery – RMA, RALM and ReACT's strong source mechanism all fall into this category. Though R-ODMRP does not have source based recovery, it utilizes a source based neighborhood building algorithm that carries similar costs.
- Local Deterministic Recovery – R-ODMRP's strong local mechanism and ReACT's weak local mechanism fall into this category, as does RAPID's recovery mechanism.
- Local Probabilistic Recovery –AG and RDG's gossip based mechanisms fall into this category.

4.5.3 Performance of Existing Approaches

4.5.3.1 First Building Block – Packet Dissemination

Flooding based Packet Dissemination – In the basic strategy every network node receiving a new data packet for the first time will broadcast it out. This essentially utilizes every available data path for dissemination of every packet, from both the source and all network nodes, all the time. The basic flooding protocol has no secondary mechanism for

packet recovery, but protocols could easily be developed with flooding as the packet dissemination mechanism followed by a secondary mechanism for packet recovery.

- Reliability – Flooding has been proven to have high reliability in specific scenarios where the conditions are ideal. It has also been proven to have severely degraded reliability in scenarios where those ideal conditions deteriorate, and the central mechanism of using all available data paths all the time works against packet delivery due to growing link contention. These scenarios include increasing network density and increasing data traffic rate.
- Network Bandwidth Overhead – The flooding mechanism itself requires no extra overhead for control data beyond the standard overhead of an IP data packet header. However, the overall network overhead, or utilized portion of available bandwidth required by using all data paths all the time can be large in the previously mentioned scenarios where link contention becomes high. In these scenarios link contention quickly results in greater numbers of packets dropped, and decreasing reliability. In flooding, network overhead is sacrificed for the sake of reliability. This is a known tradeoff with this design approach.
- Packet Delivery Latency – Protocols implementing flooding can achieve low delivery latency in those scenarios where networks are neither too dense nor too sparse, and where the data rate is not high. Since packets are instantly broadcast out on first reception, those delivered from initial sends usually have low latency, comparable to or beating other protocol delivery mechanisms. However, in dense or high traffic networks it is very difficult for a secondary mechanism to enable missed packet recovery, since little to no bandwidth is left over to allow the secondary mechanism to operate. It is likely that if a secondary mechanism were to operate in these circumstances, the latency in missed packet recovery would be very high, given the adverse circumstances within which it would be operating.
- Specific factors for a flooding mechanism:

- State and Control overhead - Flooding mechanisms are often stateless, with no special packet types or initiation mechanisms. When a secondary mechanism for recovery is added, it usually comes with some amount of required node state and control overhead. For reliable protocols there must be at least a minimal state stored at each node, and control bytes of some type to allow the packet reception information from targeted receivers to find its way back to the source.
- Packet Delivery Latency – If flooding involves a secondary mechanism for recovery, then if the source is involved, its data rate is halted or changed periodically. In this case, packet delivery latency as a metric loses meaning when compared to another protocol where the source continually sends data at a constant rate. However, it is a net negative for a protocol to deliver packets with the same reliability but an overall lower data rate when compared to another protocol.
- Centralized Operation – a flooding based protocol can get around the negative impact of required interactions with the source for resend requests by explicitly limiting such interaction to one receiver at a time. With this limitation there is no chance of packet implosions in the region of the source. The negative factor of consumption of network overhead is also averted, since the source must stop sending new data packets when notified of a receiver node's packet requests, so bandwidth can be completely devoted to these requests. The negative factor of a potentially longer latency in packet recovery still holds however, due to either a broken or a long data path between the source and a remote requesting node. Flooding based mechanisms that do not have centralized operations would have an advantage in throughput.

Topology based Deterministic Global Packet Dissemination - This strategy for data path establishment, used by many tree and mesh based protocols, has several advantages that

account for its popularity among ad hoc group communication protocols. Typically, the source will flood some type of control packet that is recognized by all nodes in the network. This packet causes all network nodes to flood it after receiving it as well. There is usually a response mechanism for receivers to identify themselves with control information that is then propagated node by node upstream back to the source, with all nodes along these paths turning on data forwarding flags for some period of time to establish a restricted set of data paths from the source to all network receivers, rather than the "all data paths all the time" approach of basic flooding. This restriction of the number of data paths often works to limit overall network bandwidth overhead, allowing for better packet delivery ratios in scenarios where bandwidth overhead is a limiting factor, such as high density or high traffic rate scenarios. Periodic operation of this basic data path establishment mechanism within a path expiration time is used to refresh routes as node movement makes old data paths obsolete. This flooding based data path creation mechanism is used as the initial packet dissemination mechanism for RMA, R-ODMRP and most tree, mesh or dynamic graph based protocols.

- Reliability – as an initial packet delivery mechanism, this technique has relatively high reliability. Several protocols have been shown to give good results when combining this technique with a secondary packet recovery mechanism. Though the basic mechanism has been proven to have worse reliability than flooding in scenarios where conditions are ideal for flooding, as conditions worsen (growing density or data traffic rate) intuitively, this basic mechanism can outperform basic flooding.
- Network Bandwidth Overhead – Several of the protocols developed with this mechanism to date have a significant amount of overhead required for operation, but much of this overhead may not be truly necessary for correct operation. This method always requires an increased amount of network overhead for the periodic network flood of the special discovery packet, but has reduced overhead when compared to pure flooding. Investigating ways to

reduce this overhead to an absolute minimum is an area that deserves exploration.

- Packet Delivery Latency – Once data paths are established by this mechanism, delivery latency is usually very low, with no operations interfering with the initial source sent packet traversing all data paths to reach all receivers. Depending on the scenario, latency for this mechanism could be better or worse than basic flooding. For example, if the restricted set of data paths were established with this mechanism such that they were not optimal, it is possible that the "all data paths" approach of flooding would naturally utilize a shorter data path to reach some set of receivers, resulting in lower latency. On the other hand, in a very dense or high traffic network it is quite possible that nodes trying to send a newly received packet out would sense other node's send attempts and backoff sending for a random period of time before attempting a resend. This could occur multiple times for a node's initial send of a given packet, resulting in a longer overall latency for flooding.
- Factors specific to R-ODMRP:
    - Centralized Operations – the R-ODMRP protocol's neighborhood building algorithm is a centralized operation. The source broadcasts a discovery packet that causes nodes at increasing distances from the source to set timers with decreasing timeouts. Eventually the farthest node's timer goes off first, causing it to forward a control packet to its upstream neighbor that identifies itself as a receiver. This information is grouped with the upstream neighbor's other downstream neighbor replies to build a table describing the upstream receiver and all downstream receivers relative locations along datapaths. This gathering and grouping occurs at all receiver nodes along all reverse data paths going back to the source. Eventually the source receives all tables and is able to build a full picture of the overall network of receivers. The downside of this approach is that first it relies on data

paths not being broken at any point along the chain due to node movement, or to link contention. Also it relies on overhead packets relayed from node to node upstream that are of ever increasing size. If an ad hoc network were to continuously grow in size, eventually it is likely that these control packets will fail to reach the source, resulting in periods where areas of nodes have no means to request missed packets.

Non-Topology Based Deterministic Local Packet Dissemination – This category is used to group the unique non-flooding based, non-data path predefined packet dissemination strategies designed to guarantee delivery to all receivers. No topological model is utilized to create fixed data paths here, but rather the dissemination mechanism itself is used to create temporary individual links along which data packets are transmitted. Scribble and EraMobile are examples of protocols that fall into this category, relying solely on their dissemination mechanism to provide reliability with no secondary mechanism to handle missed packets.

- Reliability – since protocols falling into this category use unique mechanisms specifically designed for high reliability in ad hoc networks, it is likely this metric will be high for future protocols as it is for the current ones, Scribble and EraMobile, at least in the simulation results presented.
- Network Bandwidth Overhead – protocols falling into this category often utilize mechanisms that operate at the level of interactions between neighboring nodes in order to make a determination of sending a data packet or not. Often this mechanism is optimized for a general "good condition" network scenario (the network of receiver nodes of reasonable density, data traffic at a reasonable rate, and other favorable conditions) and performs well in such environments. Intuitively though, specific "bad condtion" scenarios could generate poor performance in terms of overhead or latency or both. Also, since reliability is often seen as a more important metric to maximize, the cost is often made up for with tradeoffs in overhead and latency.

- Packet Delivery Latency – individual calculations made to determine whether or not to transmit packets often depend on random delays during the send process at a node, during which neighbor sends are listened for. If overheard, one general strategy is to set the overhearing node to not perform the packet send, in order to reduce data paths and network overhead from pure flooding. Higher latency is often a tradeoff in this category, especially when compared to the previous category of transmitted packets traversing data paths that have been previously reduced in number by a path defining protocol mechanism.
- Factors specific to Scribble: comments on the performance metrics of Scribble below are based on the published protocol description and simulation results. One danger with Scribble's approach is that it's protocol mechanisms operate until the desired reliability is reached, so the longer network nodes remain out of range, for example in sparse networks, the more the protocol's overhead and latency metrics will suffer, eventually impacting reliability itself if a sufficient data traffic load is modeled. The published simulations modeled sending 1 packet per second for 500 seconds, followed by a 1500 second "cool down" period, far removed from standard network communication conditions.
    - Sender "push" oriented – Scribble doesn't rely on out of range receivers moving into range advertizing themselves in a "data pull" mechanism, rather it relies on senders continually sending until packets are received by all. Sparse network scenarios could cause a high degree of overhead with nodes continually attempting to deliver packets to receivers that remain out of range.
    - Necessity for sender to have group membership knowledge – this requirement could be seen as unrealistic for many situations. Also, the description for Scribble assumes the source has prior knowledge, not accounting for a realistic situation where the source must build this knowledge through some mechanism involving consumption of

network resources, and a delay in transmissions until the knowledge could be obtained.

  - Reliability – Since the mechanism does not terminate until the reliability metric is reached, reliability is prioritized in Scribble. The published simulations show it to have higher reliability than ODMRP, however, the danger is that under a normal network data load, in the attempt to achieve the desired reliability, latency and overhead will become unreasonably large, causing the desired reliability to be ultimately unattainable.
  - Packet Delivery Latency – Since a given network node adds a random delay to overhear neighbor packet sends before sending its packet, the published simulations show Scribble to have a higher latency than ODMRP. This is a clear downside to the protocol.
  - Network Bandwidth Overhead – Again taking the scenario of sparse networks, if a packet is unrealized, more and more nodes will initiate sending it until all nodes in the network are sending it. A second downside to this approach is that as a packet travels farther from the source, the header portion containing the node signatures becomes larger and larger, consuming more overhead.

- Factors specific to EraMobile: the network scenarios and conditions within which EraMobile was tested were somewhat more adverse than those for Scribble, but still far from realistic. The data rate never varied from 2 packets per second, and after source sends stopped, 110 seconds were provided to allow for full dissemination of data packets in flight. Given the 3 step pure gossip based mechanism used to advertize / request / send packets downstream at each hop, an increasing data rate would likely have a severe impact on packet dissemination and resulting reliability.
  - Packet Delivery Latency - the 3 step mechanism used to advertize new packets to downstream nodes adds a large amount of latency to initial

sends of packets. Since a limit exists to the number of packets advertized in each gossip round, more latency is added to packets transferred as a result of subsequent gossip rounds. This protocol was designed to trade latency off for reliability, and this impact to overall performance is acknowledged by the authors, who state that EraMobile is not intended for low latency applications.

- Network Bandwidth Overhead – EraMobile shows very good results for this metric. Though data packets are transferred as a result of a 3 step process for each link, they only travel one hop at a time, and are only transferred when proven to be needed. This metric showed very good results across the three scenarios tested: increasing mobility, group size and number of groups.
- Reliability – The scenarios tested in the paper play to the strengths of EraMobile. In them, reliability was shown to be higher than MAODV and flooding, though coming at a known cost in large latency. Scenarios targeted to the weaknesses in EraMobile however, would be interesting to study, especially an increasing data rate scenario. Here, a threshold may be crossed, ultimately causing reliability to drop below that of flooding or MAODV, given the amount of time the 3 step process of packet transfer requires at each node.

Probabilistic Packet Dissemination – Generic probabilistic packet forwarding is an alternative strategy to reduce the large number of data paths along which packets travel to reach receivers during flooding, and thereby reduce network overhead. The basic strategy is to provide each node with a certain probability of forwarding received packets downstream. For each packet received, each node then makes a determination based on this probability as to whether or not to forward the packet. Different protocols modify this determination in various ways, usually to increase probability in the presence of fewer neighbors, and decrease probability in the presence of more neighbors, in order to

provide adequate coverage. Reliable protocols using this basic strategy often have a secondary probabilistic or deterministic mechanism for missed packet recovery. RDG and RAPID are examples of this category, using probabilistic mechanisms for initial packet dissemination.

- Reliability – since packet delivery is based on probabilities of nodes sending packets and probabilities of downstream nodes receiving packets, reliability guarantees are weaker than those of deterministic protocols. Typically reliability is a weak point for probabilistic protocols.
- Network Bandwidth Overhead – since probabilistic mechanisms effectively cut the number of active data paths with no associated overhead consumed for transfer of control data, bandwidth is conserved. Bandwidth conservation is a strong point for probabilistic protocols.
- Packet Delivery Latency – probabilistic forwarding by itself does not add latency to packet delivery for nodes receiving packets on the initial source send. Missed packet resend latency is then dependent on the secondary mechanisms used. If probabilistic forwarding is combined with randomized wait times, for example to overhear the packet sends of neighboring nodes, then the latency metric will suffer.
- Factors specific to RDG:
    - Reliability - since RDG operates purely probabilistically for both initial packet sends and missing packet resends, reliability is not as high as for many deterministic reliability protocols. One of the studies in the research described a scenario where node speeds were increased to 20 meters per second, and RDG's reliability metric dropped to 88%, a relatively poor number.
    - Network Bandwidth Overhead – The effect of the probabilistic mechanisms on bandwidth conservation for packet dissemination and missed packet recovery are unknown, since the few scenarios tested in the research did not measure overhead. Conservation of bandwidth is

typically a positive aspect for probabilistic mechanisms, and so could be a strong point in favor of RDG.

  - Packet Delivery Latency – RDG does not include random pause times as part of its mechanisms, so scenarios could possibly show relatively low delivery latency. The authors did not measure this metric as part of their simulations however.

- Factors specific to RAPID: RAPID's probabilistic dissemination mechanism operates similarly to the deterministic dissemination mechanism in Scribble, given the way it is implemented. Regardless of the probabilistic decision of a node to send or not send a packet, the deterministic corrections to that decision will cause nodes not overhearing neighbors sending a packet during the pause time to send it, and will cause nodes overhearing a neighbor sending a packet during the pause time to not send it.
  - Reliability – the scenarios used to test RAPID were not extensive or comprehensive, but they did illustrate a critical weak point in the protocol, that it has low reliability in sparse networks. Modeling of mobility and other factors were not sufficient to establish protocol performance in those scenarios.
  - Network Bandwidth Overhead – given the missing packet request mechanism a node uses, i.e., broadcasting a gossip advertisement packet to nearest neighbors and potentially receiving multiple requests back, overhead could be minimized greatly from the existing protocol.
  - Packet Delivery Latency – long latency is a downside to RAPID, for both initial packet delivery, given both the pause time before broadcast and the three step advertisement / request / reply process for initial sends, and the same pause and three step process for missed packet request fulfillment.

Congestion Control – This category represents a mechanism that is usually overlaid on another packet dissemination protocol. This mechanism is designed to optimize reliability

by modifying the rate of a source's initial packet dissemination as a response to current network conditions. RALM and ReACT have mechanisms that fall into this category.

- Reliability – In the case of RALM and ReACT, reliability is enhanced by congestion control, but at the cost of a constriction in data rate. Some applications would find this artificial rate constriction unacceptable, especially when compared to potentially higher data rates possible with reliability optimized through receiver based mechanisms.
- Network Bandwidth Overhead – source data rate constriction mechanisms must come with a cost in consumed network overhead for the control data passed back to the source. If the rate constriction is acceptable to the networked receivers, the cost in overhead is likely to be small compared to the resulting increase in reliable delivery, since it takes only a minimal amount of control data to cause the source to recognize a signal to slow the data rate.
- Packet Delivery Latency – this metric is a difficult one to measure in a situation where the source's data rate is constricted. Is the latency measured as the actual difference between when a given data packet leaves the source and arrives at a receiver, or the difference between when the packet would have left the source had there been no rate constriction, and arrives at a receiver? If the latter, then latency is a huge negative for source data rate constriction.

4.5.3.2 Second Building Block – Missed Packet Recovery

Source Based Deterministic Recovery – RMA, RALM and ReACT's strong source based recovery mechanisms all fall into this category.

- Reliability – Comparing source based recovery to local recovery, source based recovery will almost always have a higher cost in terms of the related metrics of network overhead and latency, since requests and missed packet replies have farther to travel over the network, consuming more bandwidth and taking longer to fulfill. Also, source based recovery has weaknesses in terms of reliability when applied to specific scenarios. In sparse networks for example, there may be situations where a unified linked path between

requesting receiver and source will never exist. In networks with a high data rate, there may be situations where a linked path without link contention at some point between requesting receiver and source will never exist.

- Network Bandwidth Overhead – one downside of source based recovery is the relatively larger amount of overhead required for each node to interact with the source to recover missed packets as opposed to a networked receiver at some midpoint between the requesting node and the source. Another downside is the potential for N/Ack implosions as the network scales as exhibited in RMA, where the requirement for all networked receiver nodes to interact with the source can eventually congest the network in the region of the source. RALM and ReACT get around this downside by specifying that the source can interact with only one receiver at a time for packet recovery, though packet multicasts are made to the group as a whole. In RALM and ReACT the cost associated with this is a greater latency for all receivers due to the source temporarily suspending new packet transmissions in order to handle resend requests. One advantage of source based recovery is that it removes the requirement for networked receivers to store data packets after consuming them.
- Packet Delivery Latency – as mentioned under the two bullets above, there are several ways in which increased latency could be a direct result of the source based recovery mechanism. The requirement for source interaction, by itself, will directly increase latency, and source interaction that may be a long time in coming, for example in sparse networks, could hugely increase latency for recovered packets.

Local Deterministic Recovery – R-ODMRP's mechanism and ReACT's weak local recovery mechanism fall into this category, as does RAPID's recovery mechanism.

- Reliability – local recovery can improve reliability, but whether or not it does depends on the implementation and the scenario. For example, RAPID and R-

ODMRP's local recovery mechanism requires a broadcast of a packet request from a receiver node to all nearest neighbor nodes. All can then reply, though this is modified by a replier node overhearing packet resends and changing its decision to perform the send. This localized "flooding" of requests and potential replies could quickly congest local areas of the network ultimately degrading reliable delivery of packets sent from the source at the same time.

- Network Bandwidth Overhead – there are scenarios where local recovery could increase overhead when compared to source based recovery, and ReACT exhibits this. Specifically, when a node tries to recover missed packets from local receivers, but no local receivers have obtained the packet to begin with. In this case the node will make several attempts consuming a relatively large amount of network overhead before reverting back to source based recovery and consuming even more overhead before finally obtaining the missing packets.
- Packet Delivery Latency – the same scenario in ReACT where local recovery increases overhead when compared to source based recovery would cause it to increase latency also.

Local Probabilistic Recovery –AG and RDG's gossip based mechanisms fall into this category. The secondary mechanism of probabilistic recovery has similar implications to the primary mechanism of probabilistic dissemination of packets originally sent from the source.

- Reliability – since packet recovery is based on probabilities of nodes receiving and responding to missed packet requests, reliability guarantees are weaker than those of deterministic protocols. Typically reliability is a weak point for probabilistic recovery mechanisms.
- Network Bandwidth Overhead – since probabilistic mechanisms effectively cut the number of active data paths, overhead is conserved. Minimizing the

costs of recovery in terms of overhead is a strong point for probabilistic mechanisms.

- Packet Delivery Latency – probabilistic resend requests could add latency to reception of missed packets if the relaxed guarantees mean that several requests must be made in order for a node to successfully receive missed packets. If a probabilistic resend mechanism is combined with randomized wait times at intermediate nodes then latency will suffer as well.

4.5.4 Design Strategy for a New Protocol

An overall goal for this dissertation is the development of reliable multicast protocol mechanisms that balance the requirement for reliability with a desire to minimize network overhead and packet delivery latency. The protocol must perform well under all types of stresses to network communication, including overloaded links in dense and high traffic networks, broken and ephemeral links in high mobility networks and ongoing network partitioning in sparse networks.

From reviewing the research done to date for reliable multicast MANET protocols, there are at least two promising directions that can be pursued, each with its own advantages and disadvantages that must be answered in the design of the protocol. One is to design a new topology-based deterministic global packet dissemination protocol, the other is to design a new non-topological deterministic packet dissemination protocol. Based on simulation results, these two avenues seem to hold more promise than the others discussed.

Develop a Topology-based Deterministic Global Dissemination Protocol:

This category is an all encompassing one containing all new packet dissemination mechanisms that are designed to operate with a defined topology. This category contains the highest number of existing protocols due to the fact that topology based approaches have many advantages over other categories.

Advantages:

The examples mentioned in the categorization above generally had the natural advantage that once the paths through the network were defined, initial packet delivery latency and overhead was very low, since packets were forwarded along predefined paths with as minimal latency and overhead as possible. A key to minimizing these metrics will be development of a route discovery process that will create optimal routes using a minimal amount of overhead. Non-optimal routes will add to both network overhead and delivery latency, and the route creation mechanism itself could consume a large amount of network bandwidth. Many protocols in this category utilize network flooding of an initial packet with control data in order to guarantee that each receiver node in the network is reached and has a data path constructed for packet delivery. This flooding of a packet is a potential downside of many path creation mechanisms in dense or high traffic networks, and must be examined closely in protocol development.

Disadvantages:

The overhead required by the path creation mechanism could be a big disadvantage depending on the amount of overhead used by passing control data over the network, especially in specific scenarios such as the previously mentioned dense or high traffic situations. Possibly the path creation mechanism could be designed to adapt itself to these specific conditions to answer this disadvantage.

Another disadvantage is that if the path creation mechanism creates sub-optimal data paths, each delivered packet will require a higher cost in terms of network overhead and delivery latency. Also, given that network topologies are dynamic and potentially fluid with rapidly moving nodes, a topology that may be optimal in one moment might be sub-optimal or even completely broken in the next. Mesh networks tend to minimize this disadvantage by taking advantage of multiple paths, but mesh networks likely turn at least partly into trees in sparse networks, for example. Frequent path discovery may be required to keep highly optimal paths, but will come at the cost of extra network overhead. Experimentation with networks of all types in a simulator will be informative as to what works and what doesn't.

Another disadvantage of this approach is that predefined data path delivery mechanisms often require a secondary mechanism for receiver nodes to recover packets missed in the initial delivery attempt. Operation of this secondary mechanism is generally pure overhead in terms of network bandwidth, working against ongoing initial delivery in high traffic networks for example, and could be the cause of a large amount of latency for the packets recovered.

Discussion:

One consideration in developing a robust reliable topology based protocol is to minimize the overhead of the periodic path creation mechanism to the point where it can occur very frequently in order to keep paths optimal. Optimal paths that deliver packets to as many receivers as possible will minimize the work required by any secondary packet recovery mechanism.

Another point to consider is to design a path creation mechanism that will maximize the effectiveness of each data path created in specific scenarios. Fewer paths that each deliver to a greater number of receivers will be preferable in dense networks for example, to minimize bandwidth overhead consumed, in a scenario where bandwidth overhead is in short supply. On the other hand, more data paths delivering to potentially difficult to reach receivers will be preferable in sparse networks.

A secondary packet recovery mechanism will need to be designed very carefully in order to minimize the bandwidth overhead it requires for operation, since network overhead is a potentially large negative factor for topology based protocols. Interactions with the source, and any required broadcasts or multicasts should be minimized for this reason. Broadcasts and multicasts create local congestion that will directly compete with initial packet delivery. Requiring receiver nodes to interact with the source will work against protocol scalability and cause increased latency, as well as competing with initial packet delivery.

Develop a Non-Topology Based Local Deterministic Protocol:

This category essentially contains all new dissemination mechanisms that operate via nodes making non-probabilistic individual decisions on packet forwarding based on conditions local to a node.

Advantages:

The examples in this categorization have the natural advantage that since they do not require any network topology to be defined in order to operate, the relatively large amount of pure overhead required to create a topology is not needed. With no network topology to predefine the set of network links, individual links are established as needed between individual nodes according to some other criteria. Another advantage is that, depending on protocol design, there may be no need for a secondary missed packet recovery mechanism. Scribble's single dissemination mechanism, though it is adaptive, is used throughout the dissemination process with no secondary mechanism as backup in case of missed packets.

Disadvantages:

One disadvantage of the protocols currently falling into this category is that since links are not predefined, time and bandwidth must be spent between each pair of individual nodes in order to make the decision on whether or not to send a packet. Sometimes the time required is high, for example where each node is required to pause for a random timeout to overhear neighbor nodes sending the packet before making a final decision on whether or not to send for itself. This can result in a high latency between the original transimssion by the source and the final reception by nodes at the edges of a network. Sometimes the bandwidth required can be high, for example where the protocol requires a three step [advertise / request / send] process between neighboring nodes in order to transfer packets downstream. Some of these costs can be mitigated by the design of the transmission mechanism however.

One potential issue with protocols in this category is that if the protocol is 'push' oriented such as Scribble, then senders of a given packet are required to have global knowledge of both the overall list of group members, and the overall packet reception

percentage in order to verify the dissemination operation can be completed. In a realistic scenario this knowledge would require a relatively large amount of overhead to build in to the protocol operations, especially if group membership is dynamic.

Discussion:

One consideration in developing a robust reliable non-topology based protocol is to find ways to minimize the time and bandwidth required between individual nodes for the upstream node to make its decision on whether or not to send a given packet, while keeping the natural advantage of saving the bandwidth required by a topology defining mechanism. The key is to create a mechanism allowing for full dissemination across all reachable receivers on a node to node basis with as few packet transmissions from as few receivers as possible, while allowing partitioned nodes moving back within range to learn about and receive missed packets as quickly as possible, with a minimum cost in bandwidth.

The issue of global knowledge must be dealt with in some form, especially for packets initially missed due to receivers being partitioned, either by defining a receiver based pull-type mechanism, or by providing a realistic means for nodes to gain global knowledge of the group receiver list and percent received for a given packet. A push mechanism for this will have a larger amount of bandwidth required in order to provide receiver nodes on one side of the network with knowledge of reception statistics for receiver nodes on the other side of the network, in order to terminate the send process for a given packet. On the other hand, a pull mechanism would require receivers to know what to pull, leading to the three step process of [advertise / request / resend] utilized by the eraMobile protocol, with its associated additional overhead and latency.

New Protocol Design:

Given the advantages and disadvantages of the previously mentioned design categories with the highest potential for success, this author felt intuitively that the topology based deterministic protocol category offers the richest potential for development of a successful reliable multicast protocol. Success here can be measured as "better performance than the current benchmark protocols in terms of reliability, cost in

bandwidth and latency, when evaluated across a wide variety of realistic ad hoc network scenarios". This rich potential lies both in the natural advantages of the design category, and in the possibility of solutions to its inherent disadvantages.

Once this decision was made, collateral from previous development of the R-ODMRP protocol turned out to be invaluable. The R-ODMRP code implemented in the Ns-2 network simulator became a testbed in which new ideas for packet dissemination and reliability mechanisms could be implemented, tested and verified for their overall effects on reliability, latency and required bandwidth. Mechanisms were tested in combination with each other to see their sum effects, discarding what didn't work and combining and enhancing what did. The end result of this work, the Reyes protocol, is described in the next chapter.

5 Reyes: Reliable Multicast with Neighborhood Sets

5.1 Protocol Design Goals

The central design goal for Reyes, the topology based deterministic protocol described here, is to achieve a balance in dealing with all competing stresses and constraints imposed upon reliable group communication while providing the best performance possible in terms of reliability, latency and overhead under all network scenarios. In order to do this, the protocol must do two things: first, it must create optimal data paths that will provide for the best initial packet delivery. Second, it must provide mechanisms for nodes to obtain missing packets with minimal costs in terms of latency, bandwidth and control overhead, for all conditions.

For initial data delivery, some amount of redundant data delivery (for example, mesh based vs. tree based paths) is beneficial, though redundant delivery works against reliability as traffic load or network density is increased. Early experimentation with an initial dissemination mechanism that will work well in different network scenarios such as dense, sparse, high traffic and high mobility networks will be required. The goal is for Reyes to balance the benefits of redundant data delivery with the need to minimize consumed network bandwidth.

For data request mechanisms, the goal for Reyes is to provide a variety of receiver based mechanisms that don‘t require interaction with the source, are fully distributed, and require a minimum amount of bandwidth overhead and recovery latency. The mechanisms should instantly adapt to network conditions surrounding an individual requestor node.

5.2 Initial Design Ideas

Reyes is designed to be a fully distributed protocol, where each group member node shares as much work as possible in contributing to overall reliability. The source takes on no extra tasks, and there is no dynamic constriction of the source data rate. Reyes is

designed for high reliability, low protocol overhead and low delivery latency from the beginning.

The global path creation portion of the protocol was designed in several steps: first, four realistic network scenarios were implemented in ns-2 to simulate dense sparse, high traffic and high mobility networks. These scenarios were used as testbeds for path creation design ideas. Next, the path creation mechanisms for ODMRP and R-ODMRP were exercised in these scenarios with all operations captured at the lowest level of detail in a log file, such as locations of all nodes, and the details on a per node basis of a node attempting to send (i.e. MAC level status), a node sending and a node receiving individual packets. Strategies for improved mechanisms were then developed, implemented and run using the same scenarios, and analyzed for performance at the same level of detail. Ideas proven to work well were kept, and used to build the final path creation portion of the protocol.

The same process was used to design the packet recovery portion of the protocol. First ODMRP [LGC99] and R-ODMRP [KR05] were analyzed at the lowest level of detail in multiple network scenarios, in terms of packet recovery efficiency. Strategies for improved mechanisms were then developed, implemented and analyzed for performance at the same level of detail, with proven ideas used to construct the final packet recovery portion of the protocol.

5.2.1 Fully Distributed Workload

Mechanisms relying on individual receiver node interactions with the data source often create congestion, data rate slowdown or other issues in the area of the source. With Reyes, the source is not responsible for specialized protocol tasks. There are no special tasks or duties the source handles that are not also required of all group member nodes, including data rate constriction. In fact, the source typically handles fewer tasks in providing reliability than a typical group member node.

5.2.2 Minimizing Latency and Control Overhead

A large part of minimizing latency and overhead is the implementation of a robust initial delivery scheme, requiring less work from the protocol later to make up for missed

packets. Previous research has shown the benefits of mesh based mechanisms for reliability [LSHGB00], so Reyes implements a mesh mechanism developed with details designed to provide high data delivery to more nodes with fewer data paths in dense networks, and to enhance sparse network edge delivery, two areas that experimentation proved to have high impacts on reliability.

To handle missed packets, the approach for Reyes is to develop resend mechanisms targeted to respond to the specific causes for the missed packets. These mechanisms were developed to minimize latency and control overhead for specific conditions, in addition to providing reliability. Three such mechanisms have been developed in Reyes, to respond to missed packets due to link contention in dense and high traffic networks, broken or missing links in highly mobile and sparse networks, and full network partitions in sparse networks.

Another idea to minimize latency and control overhead was to implement a means to periodically take a 'snapshot' of the network, where all nodes are logically partitioned into small groups that will be enabled with local 'neighborhood' knowledge and work together to provide for the reliability needs of each other. There were two thoughts behind this idea. The first is that if packet requests are responded to locally, latency and control overhead will be minimized, with only a few hops needed for the request and the reply. Scalability also benefits, since the same costs are incurred regardless of the size of the overall network. Second, if receiver nodes have local 'neighborhood' knowledge both of nearby nodes assigned to store needed packets, and of routing paths to them, then requests and replies can be unicast, minimizing the toll taken by reliability operations on overall network bandwidth.

5.2.3 Global Topology Based Path Creation Mechanism

Previous research ([LSHGB00], [OTV01]) has shown the benefits of mesh based topologies compared to other topologies in terms of reliability for MANETs. Generally, mesh based global path creation mechanisms are source or receiver based:

- the source periodically floods a control or data packet throughout the network guaranteeing that all receivers that can possibly be reached are reached, then

forwarding paths are constructed according to the protocol's path construction mechanism.

- receivers flood a request to join (or re-join) an existing network, either constructing new links or repairing broken links.

Previous work [LSHGB00], [ROLTG03] has also shown the benefits of source based data path construction as well, for reducing the overhead required for path construction. Reyes design began by first building example scenarios for the ns-2 simulator to model four different extreme network conditions for reliable group communication, dense networks, sparse networks, high traffic networks and high mobility networks. These four scenarios were used to test different ideas for initial data path creation. The mechanisms used by ODMRP and R-ODMRP were studied initially, then various experiments were run using different approaches to gain an understanding of the resulting data paths they generated. Reading through the resulting log files of each experiment, which showed the bottom level details of the packet dispatches and receptions on an individual node basis gave a clearer insight into the overall impacts of both the path creation mechanisms and the resulting data path patterns.

In the course of this examination it also became very clear that any categorized scenario had elements of other scenarios embedded within it, and developing a protocol to work well in all scenarios was necessary in order to successfully handle any one scenario. For example, a close examination of a typical sparse network revealed areas of dense accumulations of nodes surrounded by areas of relative sparseness. If a protocol didn't handle dense networks well, it would fail in the denser areas of the sparse network scenarios.

In the course of experimentation it was observed that the resulting data path patterns for ODMRP were different when comparing results in sparse networks with dense networks. In the denser areas of network scenarios, data paths tended to circle the areas of density instead of going through them. This resulted overall in a greater number of data paths being created that skirted around the sides of the area of density, and the nodes

in between these paths in the dense area ultimately receiving a greater number of duplicates for each source generated data packet. The resulting data paths were also longer than optimal, since they circled areas of density instead of bisecting them. Once this was noticed, it was also seen to a lesser degree in sparse networks where spot areas of momentarily greater density would cause a similar effect, and in high traffic networks as well.

Debugging and examining the MAC layer of the protocol's stack during live execution, the reasons for this became clear. A given node trying to send a packet in a momentarily dense and/or high traffic area frequently would test the wireless medium prior to sending the packet with the Collision Sense Multiple Avoidance mechanism, notice a conflict caused by another node sending a packet, backoff the send process for a random timeout, then retry sending. ODMRP exacerbated this congestion / backoff effect by requiring a node receiving a new data packet to actually send two packets out as a result, one a JReq packet intended to travel further downstream, and the other a JReply packet intended for the upstream receiver. With this 'doubling' of network traffic per each node, areas of high density and high traffic caused the protocol to consume large amounts of network bandwidth at the very time when it was most needed in order to establish an optimal data path that ensured delivery to the highest number of nodes over the course of operations until the next route refresh. Each node attempting to send out two packets had the effect of creating multiple backoffs and retries per multiple nodes in the direction of high node density, significantly delaying sends from all nodes in the area Packet forwarding progress in other directions of lesser node density did not have this obstacle of constrained bandwidth to deal with. Since the data path creation mechanism establishes the node sending the packet that is the first one received by downstream nodes as a "forwarding node" in constructing the data paths, the final paths tended to be built around both sides of high density areas, instead of through their center. Figure 5.1 shows a typical small network with a spot area of density that has data paths constructed around it by the ODMRP protocol.

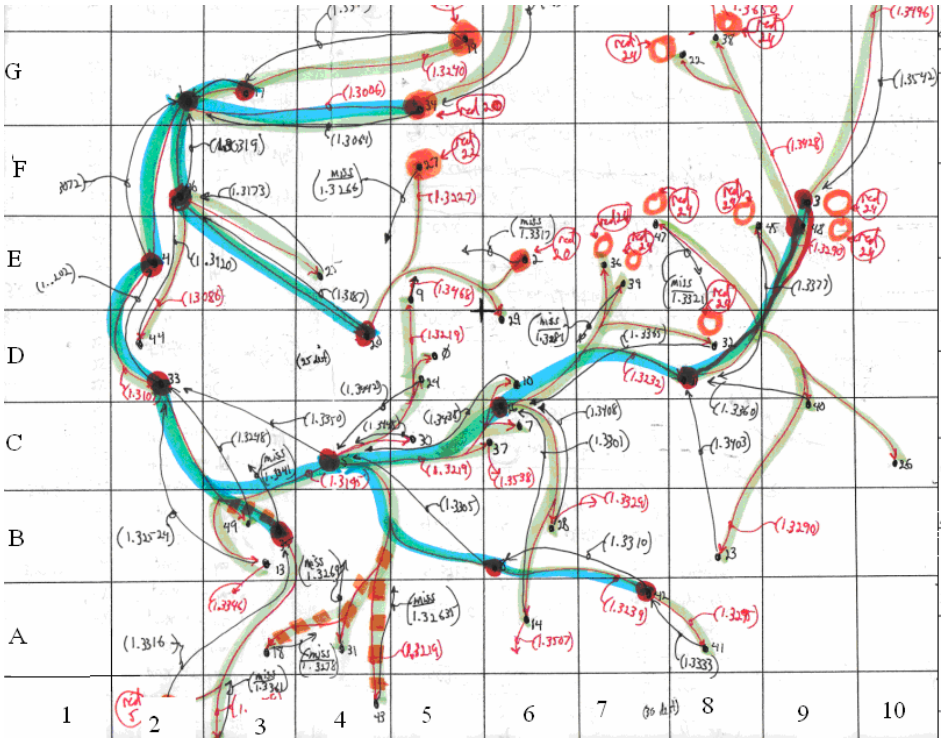


Figure 5.1 – ODMRP Constructed Data Paths Around Area of Spot Density.

In Figure 5.1, it can be seen that an area of spot node density exists at cols 4-5, rows E-F. ODMRP's resulting data paths end up circling this area. Experimentation showed that if it was in fact necessary to send two different sets of information out, one to an upstream node and one to multiple downstream nodes, adding both sets into a resulting slightly larger packet that was then sent out only once essentially solved the problem, allowing the more optimal data paths to be constructed with the global path construction mechanism. Paths constructed tended to be more optimal, bisecting areas of density, resulting in much less link contention, fewer paths constructed, with more receivers linked to each path and receivers generally receiving fewer duplicate packets. The resulting data paths also delivered data with less latency and with less bandwidth consumed. Figure 5.2 shows the same small network with its spot area of density that has more optimal data paths constructed through it by the experimental mechanism.

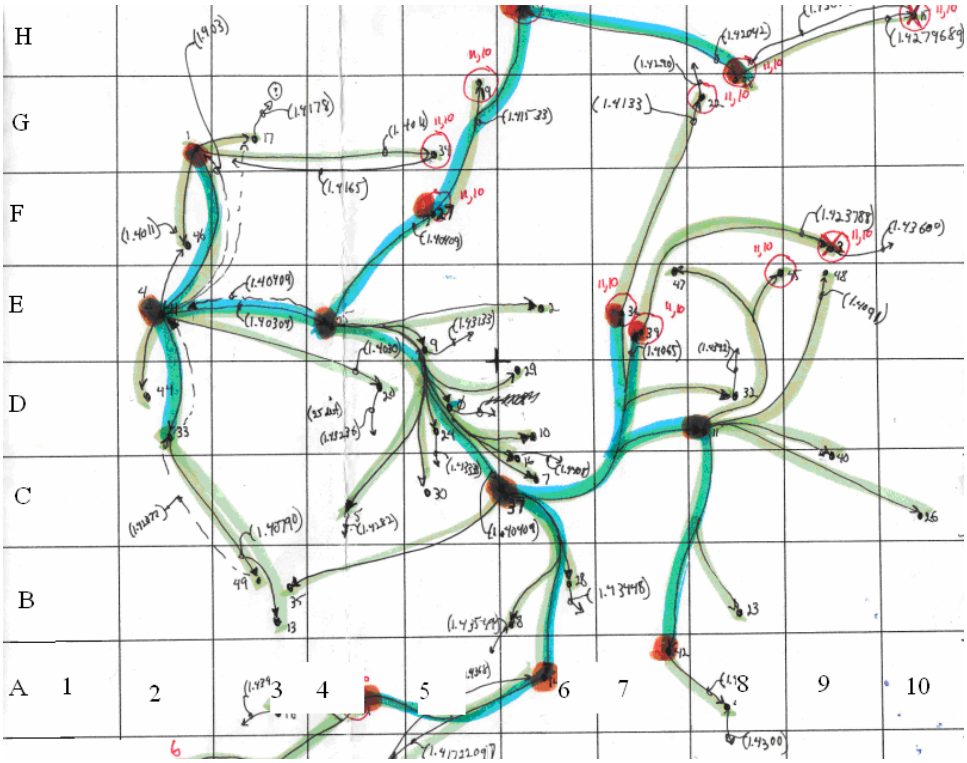


Figure 5.2 – Experimentally Constructed Paths Through Area of Spot Density

In the Figure it can be seen that in the same area of spot node density shown in the previous Figure, the experimental resulting data paths end up bisecting the dense area and creating more optimal data paths for the dense condition.

After implementing this new scheme in a communication protocol and executing 600 second performance runs with direct comparisons between it and ODMRP, the expectation was that since the new scheme required less bandwidth overhead for data delivery to the same number of receiver nodes (due to fewer data paths, fewer hops along the optimized shorter data paths, and fewer forwarding nodes sending fewer packets reaching the same number of receivers with fewer duplicates received) in the specific network conditions where bandwidth overhead was scarce, the overall packet delivery ratio would be better than ODMRP, with fewer packets dropped due to link contention.

The result did not match the expectation however. Reliability was consistently a few percentage points lower with the new scheme than with ODMRP. Another close analysis in the log files of hop by hop packet dissemination across the network using the same

scenario for the same time period studying differences between ODMRP and the new scheme revealed the reason. Since ODMRP tended to skirt areas of density, more data paths tended to be created, and the paths created were more likely to be located away from the centers of the networks, closer to the edges. Since the new scheme tended to bisect areas of density, it tended to create fewer data paths, and the paths created were more likely to be located nearer the centers of networks, instead of along the edges. Initially, after completion of the global path creation mechanism in the new scheme, all reachable network nodes were supplied by data paths, but nodes on the edges of the network moved out of range quicker, since they tended to be supplied by fewer redundant data paths. These nodes on the periphery were in fact the nodes receiving fewer data packets, dropping the overall network reliability consistently by a few percentage points.

So the new scheme minimized latency and bandwidth in areas of spot density within a typical network, but the new problem to solve was then how to deliver data to areas of spot sparseness in a typical network, on and beyond the edges of the connected networked nodes. In areas of density, bandwidth was constrained, but on the edges of networks there was no such constraint, since very little of the available bandwidth was used, due to the optimized data paths. Link contention was not a constraint in this local condition of sparseness. The approach to this new issue then was to find a means to discover which nodes were at the peripheries of the connected network, and turn more of them into forwarding nodes, in order to allow nodes moving beyond the edge of the connected network to still receive the data packets, and hopefully even to allow further out nodes already beyond the edge in a partitioned state to begin receiving data packets again from these newly enabled forwarding nodes.

After some experimentation, looking for a way to identify those nodes located on the edges of networks and about to move beyond reach of the network, an observation was made that they shared a common condition. Almost universally, nodes within the reception area but moving toward the edge of a network in any direction began by receiving a typical number of redundant packets for each data packet, usually between three and five. As they moved to the boundary, redundant packet reception would drop by

single packets, for example a node would receive one original packet with a given sequence number closely followed by three duplicates for a packet or a couple packets in a row, then one original packet followed by two duplicates for a packet or two, then one original packet with one duplicate for a packet or two, then finally only one original packet with no duplicates as the node reached the connected network boundary. When the node moved beyond the connected network boundary it would receive no packets at all. The solution then, was to allow nodes to notice their own drops in reception of duplicate (not original) packets, and when a node identified that it entered a condition where it received an original packet only, with no duplicates, it would turn on its packet forwarding flag. One would think that this would have little effect, since the node would only forward a few packets before finally moving out of range, but what actually happened is that a couple nodes moving out of range in approximately the same direction would near simultaneously begin forwarding received packets, and if one did move out of range it could continue receiving packets forwarded by the other. This mechanism caused the forwarding mesh to be extended out into what was previously a partitioned area, allowing disconnected nodes to begin receiving new packets again. In this way, delivery to all edges of the network was "beefed up", making good use of the relatively larger amount of available network bandwidth in those areas. These new forwarding nodes had the same timeouts to disable their forwarding flags as nodes along data paths created by the global data path creation mechanism.

When these two mechanisms were combined into a new data path creation scheme, the total effect was very positive. Essentially, they worked hand in hand to enable optimized data paths. On the one hand, these data paths minimized the overhead of redundant transmissions resulting from unnecessarily long data paths in specific network conditions where bandwidth was constrained. At the same time, they maximized necessary redundant sends specifically in sparse areas of constrained packet delivery where bandwidth was more freely available. When this new combined scheme was implemented and compared to ODMRP, it greatly outperformed ODMRP in reliability, even without a secondary packet recovery mechanism. It was also compared to basic

flooding with very good results. It performed nearly as well as flooding, with just a slightly reduced reliability ratio, but with far less bandwidth required.

5.2.4 Secondary Missed packet Request Mechanism

To begin with, the concept of the "local neighborhood reply" approach initiated in R-ODMRP seemed very promising and open to exploration, from the results obtained in the R-ODMRP performance study. In the situations where local nodes were capable of supplying the requesting nodes with missing packets, this solution greatly minimized bandwidth overhead and latency for packet recovery. Situations where nodes local to the requestor could not supply the requesting node with missing packets would have to be instantly recognized and efficiently handled though.

In closely examining operations of R-ODMRP, and reading the published performance results of other reliable group communication MANET protocols, a few observations became clear. First, any required interactions between receiver nodes and the source had negative impacts on one or several of the linked metrics of reliability, bandwidth overhead and delivery latency. This can be observed in RALM, where the secondary request mechanism requiring responses from the source causes both a latency in the response, and an increasing bandwidth requirement since the request and the reply must travel from each receiver all the way to the source and back, and a latency in reception of new data packets, since the source must stop transmitting new packets in order to handle the resend request. In R-ODMRP the required interaction with the source occurred in the form of neighborhood control data travelling back to the source in order to supply the neighborhood partitioning algorithm with needed neighborhood information, then the control data with defined partitions travelling back to each receiver throughout the network. This relatively large amount of control data travelling from each receiver to the source node then back out to each receiver consumed a great amount of network resources. Also, the required source interaction for network definition caused other problems, depending on the particular scenario. In sparse or high mobility networks, the paths could easily become severed, with neighborhood control information never reaching some neighborhoods, and being lost for the entire round. In this case

packet recovery would be delayed for entire regions of receiver nodes. This source interaction also crippled network scalability, since as the networks grew larger the negative effects would be more pronounced. One goal for the Reyes secondary recovery mechanism then, is for the bulk of vital neighborhood control information to be constrained to travelling the length of a single neighborhood in both directions, with no need for this control information to be transmitted back to the source, or from the source to individual neighborhoods.

A second observation was that all recovery operations that involved sending a packet were essentially occurring in direct competition with ongoing new packet dissemination, consuming network resources. For this reason, request / reply packets constrained to the width of a neighborhood had less negative impact on ongoing data transmission than did packets travelling to and from the source. Also, unicasts had less negative impact than broadcasts. Experiments with R-ODMRP showed that broadcast requests with multiple receivers replying tended to create a momentary congested area in the immediate location of the requestor, directly causing new data packets to be dropped due to link contention. Some of the reliable group communication MANET protocols that relied on broadcasts showed suspicious performance data that seemed to uphold this observation as well. For example, the broadcast packet requests being answered by all nodes receiving the broadcast in the RAPID protocol simulation results could have been the reason the authors chose to publish only the results for scenarios where the source is initiating new packet sends at the rate of one packet per second, a very low data traffic rate. This rate is far less likely to be negatively affected by local areas of network congestion occurring throughout the network. The goal for Reyes in this area of the secondary packet request mechanism is to first, restrict most packet requests and resends to local neighborhoods only, except for situations where local nodes cannot supply the requestor; second, to send packet requests only in unicast rather than broadcast form; and third, to allow replies to be unicast back from a single receiver to the requestor as well, further conserving network bandwidth. In situations where the requestor cannot find a single receiver node responsible for storage of the requested packets with a good likelihood of fulfilling its

request, it is better that the request is simply not made until more favorable circumstances were available.

A third observation in the course of experimentation was that there was more than one cause for a node to have missed receiving a packet from the source's initial send. A recovery mechanism would work more optimally, in terms of minimization of bandwidth overhead, minimization of latency and chances of successful packet recovery, if targeted to respond directly to the cause of the packet being missed. Some packets were missed due to momentary or ongoing link contention, and the upstream node still had the packet available for a quick resend. Other packets were missed due to an existing link being severed either by node mobility or a sparse condition where links between nodes were ephemeral and easily severed. Still other packets were missed due to receiver nodes being out of range of the network, in a momentary or ongoing partitioned state, and unable to receive packets. The goal for Reyes in this area was to develop a suite of mechanisms that could recognize and instantly respond to the causes of missing packets given the condtions local to the requesting node, designed to minimize bandwidth overhead and latency while maximizing reliability.

5.3 Reyes Protocol Overview

In Reyes the source has no knowledge of group membership. Initially all network receivers have a unique id and a specific data storage responsibility based on id, on protocol startup. The protocol frequently takes a snapshot, grouping all network receivers into local neighborhoods of a small number of receivers each, based on proximity. This grouping is triggered by a single data packet traversing the network. On receiving this data packet, all receiver nodes have knowledge of the id's of other receivers in their neighborhood, their data storage responsibilities and both the hop count and next hop node id to each, as well as which neighbor is currently operating as the node's upsteam hop. Each node can then immediately send targeted unicast data requests that will travel a few hops at most, and are likely to be successful. Simulations have shown this to be successful across a wide variety of scenarios including high traffic, high mobility, dense and sparse networks.

To accomplish this a key initial design idea for Reyes was to create a small packet header added to every data packet. This header allows a node, in a single packet send, to transfer protocol control data to both a single upstream and multiple downstream nodes, while simultaneously transmitting a new data packet downstream, negating the need for dedicated control packets. With a small header packet transmit time is increased only minimally, with little real impact. Most protocol control data in Reyes is transferred only within a small area of the overall network, so scalability is not impacted.

Reyes has three mechanisms a node can use to request data packets, each with a different cost in terms of network overhead and delivery latency. They are triggered by network conditions in the immediate vicinity of a requesting node. They are *Packet Header Requests*, *Resend Requests* and *Beacon Requests*. Resend Requests require neighborhood knowledge, provided by the neighborhood building algorithm, but the other two are independent, and can be triggered at any time. High level functional operations of network nodes are shown in the "Network Node" pseudocode below. The "Source" pseudocode shown below reflects the fact that the source plays no part in these three types of request mechanisms, other than performing the standard actions of a network node when a request is obtained. The Reyes neighborhood building algorithm has three phases: *Network Establishment*, *Neighborhood Formation* and *Neighborhood Confirmation*. For purposes of description, nodes that are part of the group of multicast receivers will be called *receiver* nodes, while non-receiver nodes configured to forward packets will be called *forwarder* nodes.

During Reyes operation, each receiver is responsible for reliably storing a portion of all data packets for a period of time, with an upper limit on required storage of 500 packets. A responsibility number of 0 means a node stores packets with sequence numbers ending in 0 to 33, 1 means storage of packets ending in 34 to 66, and 2 means storage of packets ending in 67 to 99.

==========

Source Psuedocode

```
start discovery_confirm timer
while (packets to send)
   add updated discovery_confirm data to packet
```

```
   send packet
Discovery_Confirm_Timer()
   if (current phase is path discovery)
      update neighbor_confirm seq_number
   else
      update path_discovery seq_number
   Set discovery_confirm timer
=====================================

=====================================
```

Network Node Psuedocode

```
Receive_Packet ( )
   if ((hdr_request) & (im upstream hop))
      check store, send back found requested packets
   if ((hdr_reply))  & (im missing pkt))
      receive packet

   if ((rsndreq pkt) & (im next hop to rsndreq target))
      forward packet
      store requestor, prev hop id in routing table
   if ((rsndreq pkt) & (im rsndreq target))
      check store, send found requested packets
   if ((rsndreq reply) & (im next hop to reply target))
      forward packet
   if ((rsndreq reply pkt) & (im missing pkt))
      receive missed packet

   if ((beacon req pkt) & (im group member))
      send found requested packets
   if ((beacon req pkt) & (im connected group mbr))
      turn on packet forwarding flag if not on
   if ((beacon req pkt) & (im non-receiver))
      broadcast beacon request
   if ((beacon reply pkt) & (im next hop))
      forward reply packets
      turn on packet forwarding flag if not on
   if ((beacon reply pkt) & (im missing pkt))
      receive packet

   if (new data packet)
      update gaps list, received list, reliable storage
      if (gaps found)
         add hdr_request to packet
   if (path discover packet)
      load path reply info to header
      set neighbor_reply timer
   if (path reply packet) & (im upstream hop)
      turn on packet forwarding flag
```

```
   if (neighbor reply packet) & (im upstream hop)
      add info to internal neighbor table
   if (neighbor confirm packet) & (from upstream hop)
      set randomized timer to send rsndreq packet
      zero out old nbr confirm table entries, add new ones
   if (new packet) & (forwarding flag on)
      if neighbor_reply timer expired, add nbr table to pkt
      send packet
=====================================
```

5.4 Reyes Data Structures

There are two standard data store types needed for Reyes operations. One type is contained in the data packet header, and is the means by which Reyes allows each node to communicate protocol control information to both upstream and downstream nodes on an ongoing basis. This 'Reyes header' is added to each data packet sent out, and the use of this mechanism negates the need for most types of dedicated protocol control packets, although dedicated control packets still exist for two types of data resend requests. The Reyes header contains id numbers and sequence numbers that enable nodes to identify the current protocol phase, and obtain needed control information from neighborning nodes.

The second data store type is a local store inside each node, where the node tracks the current state of protocol operations and control information in a series of structures.

Reyes Header Type Structures

All Reyes packet header types add between 13 and 25 bytes to the standard data packet header. This is in addition to the 20 bytes of control data added by the standard IP header used by all ad hoc routing protocols, including flooding. This does not include the two types of request packets, which require a dedicated packet. The Reyes Header portion of network data packets contains the following:

- 'packet type' - numerical field that allows other nodes receiving a packet to easily identify packet type. Packet types align to all protocol phases (described later), header gap request packets, resend request packets, resent data packets, and beacon request packets.

- Packet sender specific information needed by neighboring nodes:
    - Previous hop sender id
    - upstream node id - to identify previous hop on data path
    - recorded hopcount field (holds hopcount for different purposes, depending on phase: hopcount to downstream previous receiver on this path, to upstream next receiver, etc..).
- Standard source originated Data Packets have the following fields in addition to those above:
    - Protocol phase specific information:
        - wait time to initiate reply,
        - node *level* number - used to define neighborhood boundaries, explained later
        - sequence numbers per phase - path discovery, path reply, reply table and path confirm
    - Header gap information:
        - requestor id
        - gap start packet sequence number
        - gap end packet sequence number
    - Reply table packets and path confirm packets also have small associated data tables added to the header for protocol control information. Both these tables contain, with one entry per receiver node:
        - Neighborhood lead id and neighborhood number - to uniquely identify each neighborhood in the network, used to allow nodes to identify their neighborhood, and all nodes in related neighborhoods the current node is on a data path in between.
        - Receiver id – identifies the actual neighbor node sending a reply.

        - Id of next downstream hop to receiver – to provide data path information for routing requests.
        - Receiver's storage responsibility number (explained later).
        - Hopcount to receiver – incremented at each hop, so nodes know all members of their neighborhood and the path hopcounts to each.
    - also, the path confirm data table has an additional field:
        - Id of next upstream hop to receiver – to provide data path information for routing requests.
- Data packets retransmitted in response to a data request have the following fields:
    - Packet type – to id the packet.
    - Resend request data sender id – identifies node originating the data packet reply to the requestor.
    - Hopcount – from requestor to current sender.
    - Resend requestor id – node originating data request.
    - next hop to resend requestor id – reverse path next hop to forward data packet back to requestor.

The dedicated control packet header for resend requests covers both resend requests and beacon requests. Fields for this packet header include:

- 'packet type' field – to id the packet.
- Sender id for this particular packet, and upstream hop node id its being sent to, for resend requests
- Requestor id of node originating this request.
- Storage responsibility table for needed packets (explained later).
- Number of gaps stored in packet, gap list and sequence number of last packet received.
- Beacon packet id number.

Reyes Local Node Internal Store

Local storage required in each node consists of a series of lists, a set of timers and a node protocol state table:

- Reliable Packet Store – storage for a maximum of 500 data packets. The packets for each receiver group member to store are determined by the node's 'storage responsibility number' – a storage responsibility number of 1 requires storage of packets with sequence numbers ending in 1 – 33, 2 requires storage of packets with sequence numbers ending in 34 – 66, and 3 requires storage of packets with sequence numbers ending in 67 – 00.
- Data Store Cache – a list of sequence numbers for packets recently received, used to check for duplicates received.
- Gap List – storage for the list of sequence numbers for packets sent from the source but not yet received.
- Resend Request Routing Table – a table with an entry for each data packet resend request where a node is on the path between requestor and local provider. Each entry contains the id number and next hop to the requestor, and the id number and next hop to the provider.

- Timers and scheduling flags for:
    - Path_Discover-Neighbor_Confirm – timer used to change protocol phases.
    - Load_Reply_Neighbor_Table – timer used to determine when to load and send the reply neighbor table upsteam.
    - Receiver_Beacon – timer to control periodic sending of beacon packets for partitioned nodes.
    - Various purge timers – to purge forwarding group node status, reliable packet store (to verify the 500 packet maximum is not exceeded) and entries in the routing table.

- The internal state table has entries for:
    - Node type identifier – identifies each node as source, receiver, new node or forwarding group member.
    - Node storage responsibility number.
    - Path Discovery phase variables – flag, current sequence number, hopcount from source, reply wait time, timeout time. Nodes per neighborhood count, node level number.
    - Path Reply phase variables – flag, sequence number and forwarding group timeout time.
    - Reply Table phase variables – flag, send and receive reply table sequence numbers, internal temporary neighbor table used for storing and processing received neighbor tables.
    - Path Confirm phase variables – flag, last received time, received and sent confirm sequence numbers, internal path confirm neighbor table used to store and process a received path confirm neighbor table.
    - Most recent data packet received – sequence number and time.
    - Current upstream hop node id.
    - Timeout values for path discovery, path reply,
    - Set of flags, activated by phase so current node knows the protocol phase its in.

5.5 Reyes Neighborhood Set Construction

5.5.1 Network Establishment

The Network Establishment phase is initiated by the source node incrementing the path discovery parameter value in the Reyes packet header to match the current data packet's sequence number before sending the packet. The first packet with this newly incremented value is called a *path discovery* packet. Each node receiving this *path discovery* packet will broadcast it. If a node receiving the packet is a receiver node, it will compare its internal path discovery parameter value to the one in the packet, and recognize the

initiation of a new path discovery phase of the protocol by seeing that the value in the packet is greater than its internal value. The receiver node sets the Reyes packet header path reply sequence number to match the path discovery sequence number and fills in the packet header upstream hop id field with the id of the node it received the *path discovery* packet from, storing this node's id internally as its current upstream hop id. When this receiver node multicasts the *path discovery* packet downstream, its upstream hop receives it and the node identifies itself as the target of a *path reply* packet, since the packet's path reply sequence number is greater than the upstream hop node's internally stored number, and it recognizes its id in the upstream hop parameter. This upsteam hop node turns on its data forwarding flag, sets a timer, multicasting all new data packets. On expiration of the timer, if another *path discovery* packet is not received, the node stops forwarding data packets.

By the time the single *path discovery* packet propagates through the network, all datapaths are established. Figure 5.3 depicts an example with established data paths after the *path discovery* packet has propagated through the network. This example will be used as the context for figures through this paper

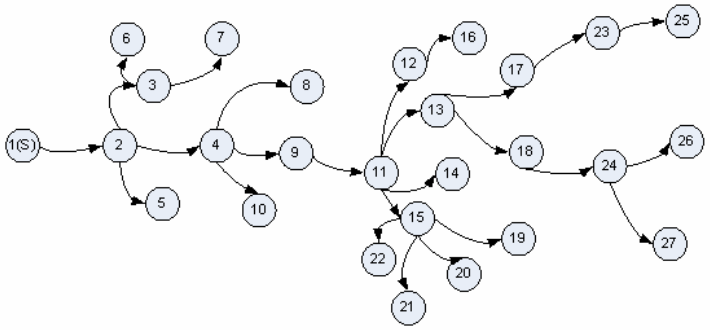


Figure 5.3: Example Established Network

5.5.2 Neighborhood Formation

Neighborhood Formation also begins with the *path discovery* packet. Initially, the source sends *path discovery* packets with 0 for the *level* parameter in the Reyes header. Use of the *level* parameter will be explained here.

Each node one hop from the source sets its internal *level* number to 0, sets a *neighbor reply* timeout value, increments the header's *level* parameter to 1, and forwards the *path*

discovery packet. A node receiving this packet sets its internal *level* number to 1, sets its *neighbor reply* timer with a shorter timeout value, increments the header *level* parameter to 2 and multicasts it. The next nodes in the data path set their *level* number to 2, entering 0 in the header's *level* parameter before sending it. Level 2 nodes have no need to set a *neighbor reply* timeout value, as will be explained.

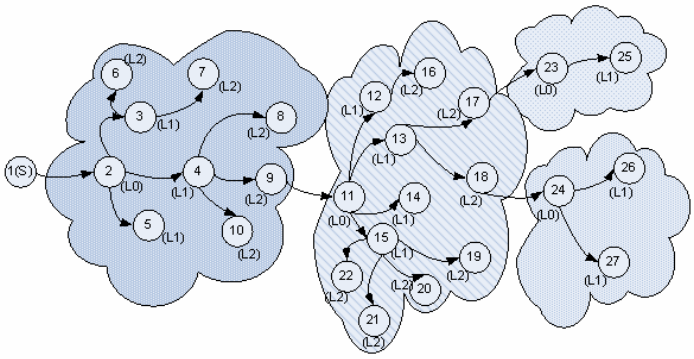


Figure 5.4: Example Network Neighborhood Sets

After learning its level number, a node knows where it resides within its neighborhood set. Figure 5.4 depicts the network shown in Figure 5.3 with node *level* numbers shown in parentheses. Neighborhood sets are partitioned based on *level* 0 nodes. In this example, nodes 2, 11, 23 and 24 are *level* 0 nodes, and define the boundaries between the four neighborhood sets composing the overall network. *Level* 2 data receivers are at the outer boundary of their neighborhood sets, hence on receiving a *path discovery* packet can immediately load their id as both receiver and next hop to receiver, along with their storage responsibility number and their upstream hop's id to the *neighbor reply* portion of the Reyes packet header in the corresponding outgoing *path discovery* packet. When this packet is forwarded downstream, the upstream hop node also receives it and recognizes it as both a *path reply* and a *neighbor reply* packet.

A level 1 node's *neighbor reply* timeout value is set to allow reception of all downstream level 2 node's *neighbor reply* packets. Each received *neighbor reply* packet causes the level 1 node to create a corresponding entry in its internal neighbor reply table, adding the hopcount to the level 2 node. Once the level 1 node's timer expires, it processes all received level 2 *neighbor reply* entries before forming its own *neighbor*

reply packet to be broadcast. Processing consists of the level 1 node first creating an entry for itself in the table, then grouping all neighbor reply table entries into sets of 3, and determining remainders. For each set of 3, the level 1 node lists itself as neighborhood lead and assigns a neighborhood number. This neighborhood lead / neighborhood number pair uniquely identifies all neighborhoods in the overall network. Remainder nodes not grouped (at most two per level 1 node) will be put in the *neighbor reply* portion in the Reyes packet header of the next data packet multicast out, which will be received by the node's upstream level 0 node.

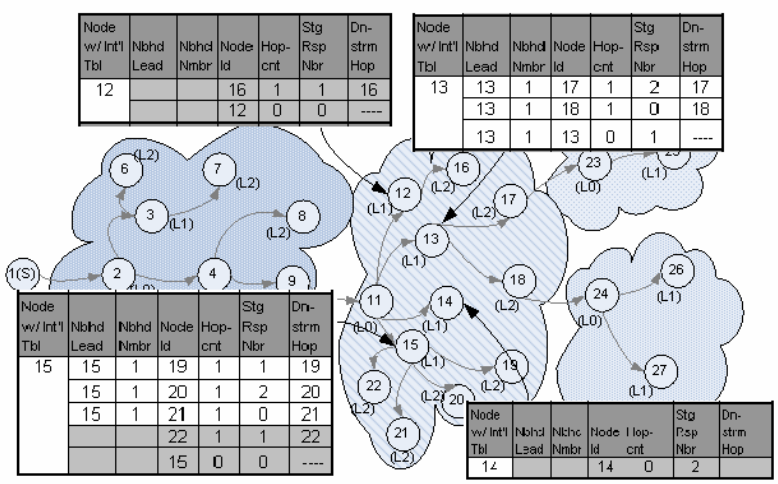


Figure 5.5: Nbr Reply Tables for Level 1 Nodes

Figure 5.5 shows the internal tables of level 1 nodes in the center neighborhood set after processing all received level 2 node's *neighbor reply* data. The rows in gray represent remainder entries to be sent upstream to level 0 receivers on the neighbor reply timeout. Level 0 nodes, with a longer wait until neighbor reply timeout, will receive all downstream level 1 neighbor reply tables, increment hopcounts, then group entries into neighborhoods, with one difference: remainder ungrouped nodes are added to the neighborhoods grouped by the level 0 node rather than forwarded upstream. Figure 5.6 shows the internal table of the level 0 node in the center neighborhood set after processing level 1 *neighbor replies*.

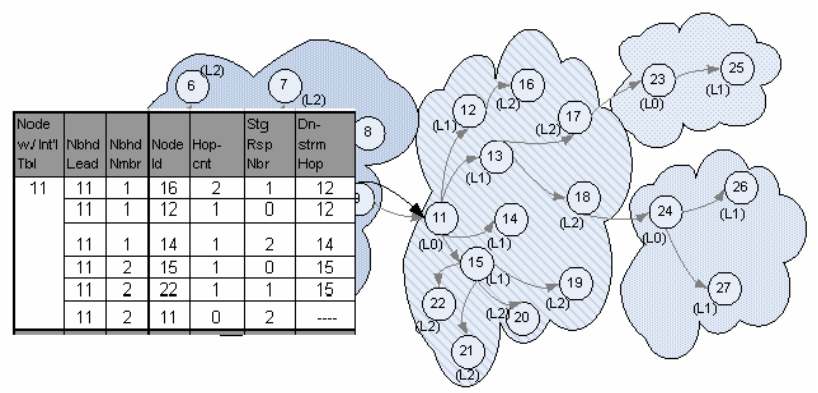


Figure 5.6: Nbr Reply Tables for Level 0 Node

Figure 5.7 shows the packet sequences for *path discovery, path reply, neighbor reply* and *neighbor confirm* packets for a neighborhood of receiver nodes shown in Figure 5.3. After the *neighbor confirm* data packet, no more packets with specialized flags about neighborhood formation are sent until the next *path discovery* data packet. During this time packet request mechanisms are free to operate.

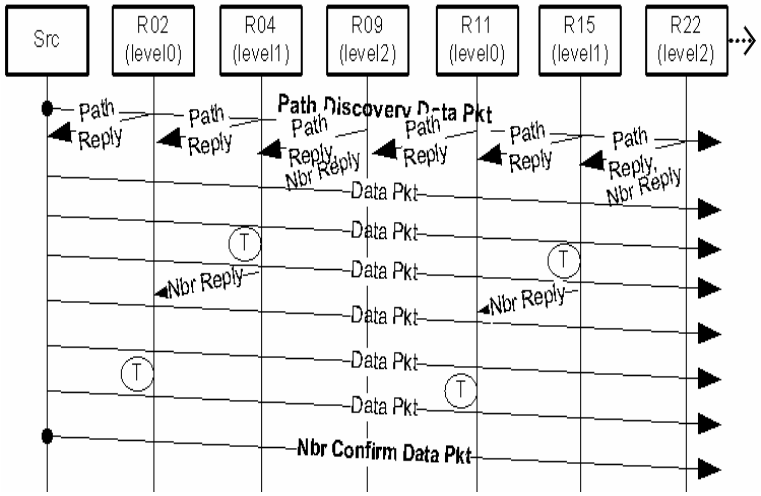


Figure 5.7: Packet Sequences for Path Discovery, Neighborhood Formation

5.5.3 Neighborhood Confirmation

The Neighborhood Confirm phase is initiated by the source incrementing a *neighborhood confirm* sequence number in its packet header and broadcasting the *neighbor confirm* packet out. Each level 0 node receiving this data packet will first add an entry for its upstream hop to its Reply Neighbor Table, then shift its finished Reply

Neighbor Table into its Confirm Neighbor Table. Next, the level 0 node will load its Confirm Neighbor Table into its *neighbor confirm* packet, and multicast it downstream.

Level 1 nodes receiving a *neighbor confirm* packet from their upstream level 0 node must determine the entries in the table that apply to them, to store in their own confirm neighbor table. If an entry identifies the node receiving the packet as both the receiver and the next hop, the node stores the entry with a hopcount of 0 and a signifier that the upstream and downstream hops are not applicable (since it is the node itself). If the entry identifies the node as the next downstream hop but not the receiver, it stores the entry with a signifier that the upstream hop is not applicable, a hopcount of [table entry hopcount – 1] since it is one hop closer to the receiver, and does a lookup based on receiver in its own reply neighbor table to find the next downstream hop.

In both cases, if the level 1 node is either a member of or on the path between members of a unique neighborhood, the node records the level 0 neighborhood lead node id and neighborhood number uniquely identifying the neighborhood, then adds all other entries from the received confirm table for that unique neighborhood to its own confirm table. The node will also load all entries matching that unique neighborhood to its *neighbor confirm* packet, so the downstream level 2 neighbor has a full picture of the neighborhood it is part of.

If a received neighbor confirm table entry for a neighborhood that a node knows it is either a member of, or on a path of, does not show the node as the next downstream hop (i.e. the neighbor must reside on another up or downstream branch off its upstream node), the node records its unique upstream hop as next upstream hop and sets hopcount to the receiver to [table entry hopcount + 1], as it is one hop down from its upstream neighbor. Figure 5.8 shows the center neighborhood set's *level* 0 and 1 node's internal Confirm Neighbor Tables after processing *neighbor confirm* packets. After processing its received neighbor confirm table, a level 1 node adds all neighbor reply table entries that it has processed, grouped and is lead for. This internal neighbor confirm table is then compressed and sent out in the *neighbor confirm* packet header to downstream level 2 nodes. On reception, a level 2 node processing this neighbor confirm table has no internal

neighbor reply table to draw from, since it is at the outer boundary of its neighborhood. It simply determines the neighborhood it is a member of, recording all entries for that neighborhood in its internal confirm neighbor table.

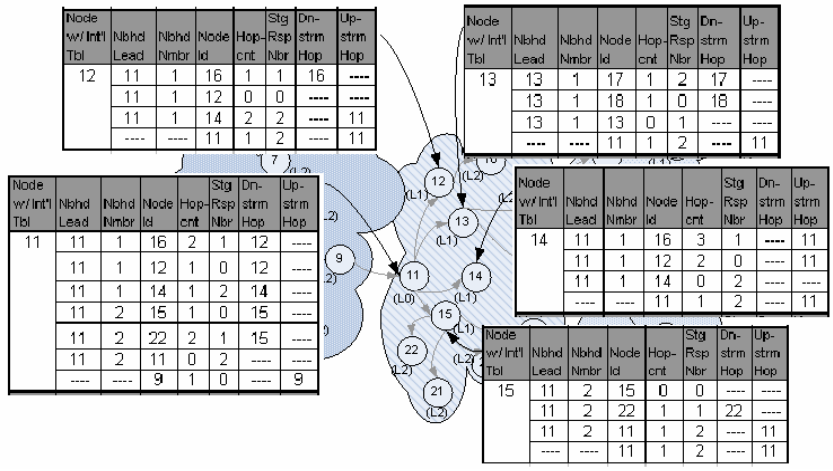


Figure 5.8: Level 0, 1 Neighbor Confirm Tables

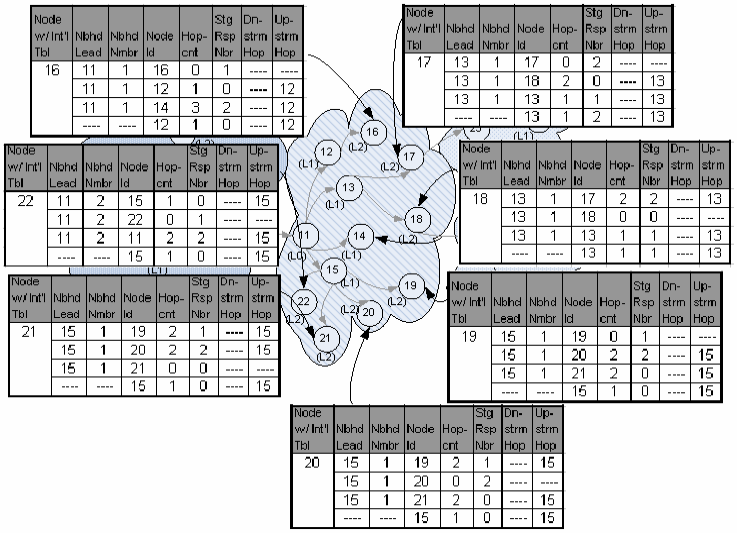


Figure 5.9: Level 2 Neighbor Confirm Tables

After processing its received neighbor confirm table, a *level* 1 node adds all neighbor reply table entries that it has processed, grouped and is lead for, inserting a 'not applicable' signifier for upstream hop, since all those nodes will be downstream of the level 1 node. This internal neighbor confirm table is then sent out in the *neighbor confirm* packet header to downstream level 2 nodes. On reception, a level 2 node processing this

neighbor confirm table has no internal neighbor reply table to draw from, since it is at the outer boundary of its neighborhood. It will simply determine the neighborhood it is a member of, recording all entries for that neighborhood in its internal confirm neighbor table. Figure 5.9 shows the center neighborhood set level 2 node's Confirm Neighbor Tables after processing.

Figure 5.10 depicts the neighborhood partitioning of the network shown in Figure 2 after the *neighbor confirm* packet has propagated through the network.

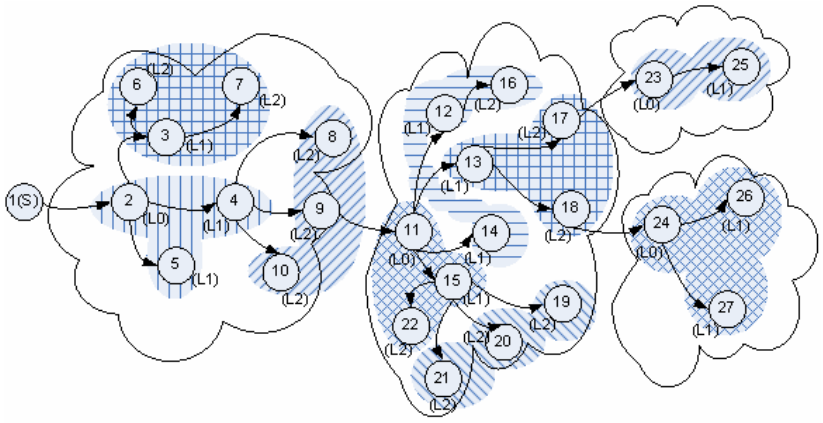


Figure 5.10: Network Neighborhood Partitioning

5.6 Reyes Data Request Mechanisms

5.6.1 Packet Header Request Mechanism

For a node's packet loss due to link contention, a clear answer is to simply request the upstream hop to resend the missing data immediately. Given the dissemination rate of a packet broadcast by the source, it would be a rare condition that original packets arrive at a given node out of sequence. Hence the mechanism is to have a node, upon receiving a data packet, check for a gap between its sequence number and that of the previously received data packet. A gap found indicates a resend is required. If the network is congested in the local area, the lowest overhead way for the node to request resends from its upstream node is to add the first and last sequence numbers of the noticed gap to the *header request* portion of the data packet causing the node to have noticed the gap. Since every outgoing data packet from the node contains the current upstream node's id (required for other parts of the protocol), the upstream node identifies the request by the

listed gaps with itself as the upstream node, and resends the requested packets. This mechanism has the lowest overhead and latency of the Reyes resend request mechanisms. This mechanism can be used any time, even upon receiving a *path discovery* packet from a brand new upstream node. Packet sequences for header requests are shown in Figure 5.11.

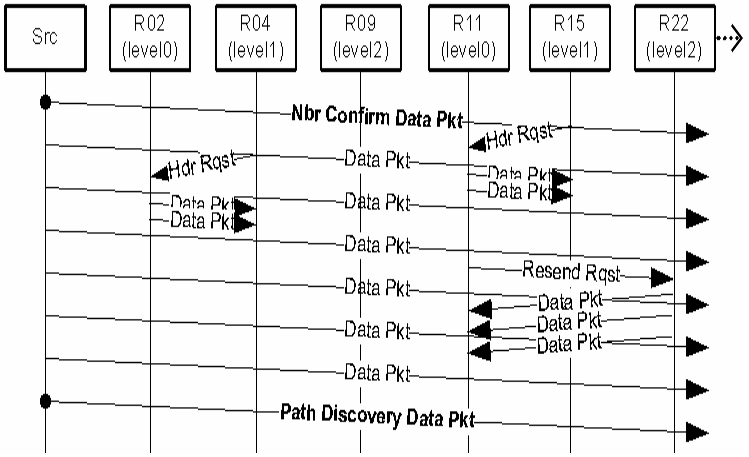


Figure 5.11: Packet Request Mechanisms

5.6.2 Resend Request Mechanism

A stronger request mechanism with a higher overhead cost was developed for packets lost due to missing links. This mechanism requires a node to have knowledge of local neighborhood receivers, their storage responsibilities and their up or downstream next hop paths. It is triggered by a node's reception of a *neighbor confirm* packet. After processing a new *neighbor confirm* packet all network data paths are established and each node has freshly learned local neighborhood knowledge that is likely to last for a brief period of relative stability until the next path discovery packet is received.

In this mechanism, a node first examines its list of missing packets and determines all storage responsibility buckets they fall into. For each bucket, the node does a lookup in its neighbor confirm table, loading a found receiver node id and next hop into a resend request packet. If a receiver is not found for the storage bucket, the node will load the id of its upstream hop into the packet, with no receiver id, signifying that the upstream hop must check its own neighbor confirm table to locate a receiver with the needed storage responsibility number. If the upstream hop can't locate such a receiver, it forwards the

request to its upstream hop, until one is found. The requesting node loads its list of gaps into the packet, sending it out to nodes on the next hop list. Nodes receiving this packet that identify themselves as next hops but not targets will lookup in their neighbor confirm table to find the next hop from their position, forwarding the resend request packet on. They will also store the previous hop id in an internal routing table to forward reply packets.

The goal of the resend request mechanism is that when a node is missing packets, for each bucket of missing packets it needs, at most one single responsible receiver node that is located as close as possible will be found and will unicast back the data packets, on fresh data paths. This is intended to reduce the network overhead of resend requests as much as possible, while giving them a high likelihood of succeeding. An example of resend request packet sequences is shown in Figure 9.

5.6.3 Beacon Request Mechanism

This third data request mechanism in Reyes has the highest network bandwidth costs and latency. It is intended to handle network partitions where bandwidth costs are not the most important consideration.

In Reyes the Beacon Request process begins with a beacon timer initially configured to expire relatively infrequently. On expiration, a node checks to see if any data packets have been received in the last cycle. If none were the node goes into beacon mode, creating and sending out a Beacon Request packet that is very similar to the Resend Request packet, with a few exceptions. No upstream hop id is listed, all next hops and storage responsibility nodes are set as “needed but unknown”, and a “last packet received” sequence number is added. A node in beacon mode initiates Beacon Requests more frequently, since overhead is not an issue.

If the recipient of a beacon packet is a receiver or the source, it will send back all listed missing packets found in its reliable storage, including packets with sequence numbers greater than the requestor’s “last packet received”. If the recipient is actively receiving new packets, it will turn on its forwarding flag, if not already enabled. If the recipient is a non-receiver, with or without its forwarding flag on, it enters the requestor as a previous

hop into its internal routing table and forwards the beacon request on. If the request ultimately reaches a receiver node, resent data packets will have a fresh path to return to the requestor on, and nodes along the path will turn on their forwarding flag if the receiver is actively receiving. If the recipient is another receiver in beacon mode, the packet is dropped and no further action is taken.

5.7 Protocol Discussion

In protocol design an initial priority was placed on each receiver node's storage responsibilities being permanently assigned. A node makes every effort to obtain and place in long term reliable storage each packet it is responsible for storing. This makes a unicast data request to the storing node a meaningful event likely to be successfully fulfilled. The network neighborhoods dynamically formed during Reyes' operations do not always include nodes with all storage responsibilities of 0, 1 and 2. In fact it often happens that neighborhoods have no coverage for one or two storage responsibilities at any point in time. When this occurs, and a node needs packets but has no neighbors assigned to store them, the request is unicast to the upstream node, where another search is made for a node with the required storage responsibility. This is necessary in order to guarantee that resend requests and data resends will always be unicast, in order to not congest regions of the network with multiple same data resends from different senders.

Initial experimentation proved the importance of this temporary regional congestion issue in another aspect of Reyes operations. Since the first *path discovery* packet of a new cycle is always flooded, a large amount of link contention exists during an upstream level 1 node's reception of a downstream level 2 node's *path reply/neighbor reply* packet that acts as its *path discovery* packet downstream, when other nodes are nearby. A generous time allocation of 0.6 seconds was set to allow upstream level 1 nodes another chance to receive downstream level 2 node's *neighbor reply* packets, after the temporary congestion caused by the flooding of the initial *path discovery* packet had cleared away. Level 0 nodes receive downstream *neighbor reply* packets well after this congestion has passed, and forwarding paths are established, and so do not require as much time.

Experimentation also showed that within a neighborhood set it was important to require that each node obtain its *neighbor confirm* packet from the same node it received its *path discovery* packet from in the current cycle, in order to enforce correct operation of the neighborhood building algorithm, and have all nodes correctly record receiver neighbors and up/downstream next hop links. This requirement however was shown not to be necessary between neighborhood sets. In other words, a level 0 node's neighborhood confirmation phase could be triggered by receipt of the correct *neighborhood confirmation* packet from any node, not just its recorded upstream hop. In this way, *neighborhood confirmation* recognition problems arising from link issues between nodes in one neighborhood set during transmission of a particular *neighborhood confirmation* packet were not propagated to downstream neighborhood sets.

In experimentation, it was interesting to compare the data paths established by Reyes with those established by ODMRP, using the same scenario files. Reyes data paths tended to drive straight through areas of greater network congestion, where ODMRP paths tended to skirt around these congested areas. This was due to the fact that in Reyes data paths are established by a node receiving a *path discovery* packet and broadcasting it out, with a flag in the header to signal the upstream node to set its forwarding flag and timer. In ODMRP the process is for a node to send a Join Reply packet to its upstream node and a Join Query packet to its downstream nodes. In denser areas of the network the consequence of sending two packets back to back is multiple send collisions between nodes, backoffs and resends, greatly slowing data path establishment. In ODMRP paths skirting areas of greater density are established faster, and tend to win out.

Consequently, the data paths in Reyes tend to be more 'backbone' oriented, resulting in fewer data forwarder nodes over the network. During normal operations the network tends to be less loaded with duplicate packets at any one node, and conditions are easier for Resend Request operations after data path establishment. With this configuration however, it is also easier for receiver nodes at the edge of the mesh to move out of range.

Trial runs showed that this condition could be easily identified and solved in Reyes operations, with little harmful effect. As nodes headed out of the mesh reception area

they would receive fewer and fewer duplicate packets for a given packet sequence number. Just before leaving the mesh they would receive their final packets from a single forwarder with no duplicates at all. A mechanism was established to notice this condition. Duplicate packet reception was counted on the fly on an ongoing basis, and when the count dropped to one for a given data packet, the receiver turned on its data forwarding flag. The consequence of this was that network areas on the borders of the mesh that had light data delivery or were momentarily partitioned would suddenly have a strong delivery mesh extended into them. Other areas inside the mesh were unaffected, since nodes always received at least one duplicate packet in these areas, so increased link contention was not an issue.

5.8 Performance Evaluation

5.8.1 Simulation Environment

Reyes was implemented in the ns-2 network simulator [FV02], developed by the University of California, Berkeley, and the VINT project, along with Carnegie Mellon's Monarch Project mobile and wireless ns-2 extensions [C99]. Ns-2 was used to compare Reyes with ODMRP [LSG00], Flooding and R-ODMRP. ODMRP was selected because it is a well documented high reliability protocol, and flooding is currently a standard for reliability in MANETs. R-ODMRP was used as a reliability protocol comparison point.

The Reyes, ODMRP and Flooding simulations all executed with the same randomly generated network traffic and node movement files. All node movement files utilized random waypoint, and were generated with the Colorado School of Mine's "Mobgen-ss" application [NCB04a], which implements a method to guarantee a random waypoint simulation begins in a steady-state distribution, documented in [NCB04]. All movement scenario files establish a varying number of mobile nodes moving within a 1000m x 1000m area at varying speeds with a delta of 10m/sec, and 0 seconds pause time. The radio range for each node was 250 meters, and channel capacity was 2 Mb/sec. All simulations ran for 600 seconds. Node speeds for the traffic rate and sparse network scenarios were 20m/sec, while the mobility scenarios varied node speeds from 20m to

70m/sec. Multiple runs were executed for each data point in all scenarios, with the results averaged.

The number of nodes was set at 50 nodes per network with 25 receivers for the mobility and traffic rate scenarios, while the sparse network scenarios varied the number of nodes from 45 nodes with 23 receivers, down to 5 nodes with 3 receivers. Data rate was set at three 512 byte packets per second for the mobility and sparse network scenarios, while it moved from 3 to 24 packets per second in the traffic rate scenarios. In the simulations, parameters for ODMRP were set to 3 seconds for the Join Query interval and 9 seconds for the forwarding state timeout. Parameters for Reyes set a level 1 node reply table timeout to 0.6 seconds, and a level 0 reply table timeout to 1.0 seconds, with the Reyes source period set to 1.25 seconds from *path discovery* to *neighbor confirm* packet, then 1.25 seconds from *neighbor confirm* to *path discovery* packet. Typically packets in a node's storage responsibility are stored until the storage threshold is reached. A node will store all other packets as well for 25 seconds or until purge, in order to fulfill packet header requests, excepting sparse network scenarios, where the storage time is 250 seconds. Since nodes are in multiple ongoing partitioned states in sparse networks, the threshold is reached less often. In all Reyes scenarios, if more packets need to be purged than have expired, the oldest non-responsibility packets are purged first, and the newest responsibility packets are purged last. These metrics hold for all scenarios except data rate, where Reyes reply timeouts, cycle periods and data storage capacity are all scaled to match the data rate change.

5.8.2 Performance Metrics

Three metrics were used to evaluate performance of the three protocols in the various scenarios:

1. Reliabililty Percentage – the number of packets actually received by all receivers compared to the total number of packets that could have been received.
2. Control Overhead – the number of control bytes sent by all network nodes compared to the number of data bytes delivered to all receivers.

3. Data Delivery Latency – the difference between the time when the source sends a data packet and a receiver receives it, averaged across all network receivers for reception of all data packets.

5.8.3 Sparse Medium Mobility Network Results

In the Sparse Network Figures, 5n_3r denotes a network of 5 nodes, with 3 being receivers. Sparse networks are where the strengths of Reyes have the most impact. In Figure 5.12, in the sparsest network with five nodes, Reyes greatly outperforms flooding, ODMRP and R-ODMRP in reliability.

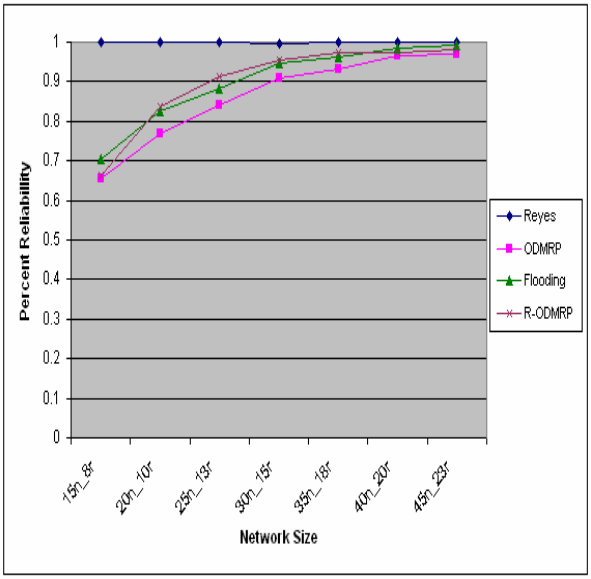


Figure 5.12: Sparse Medium Mobility Network Reliability

Reyes retains full reliability across the sparse scenarios shown. R-ODMRP delivery drops off even more sharply than the best effort protocols. Once the network becomes sparse enough it provides no advantage in terms of reliable delivery. Its centralized mechanisms are more fragile, and the control overhead turns from a benefit to a liability, dropping delivery to below that of best effort protocols.

Here it is likely that if Reyes continued to operate after the source halts packet sends at 600 seconds, 100 percent packet delivery would be obtained even for the sparsest

networks, since nodes would continue moving around the network exchanging missing packets with each other through beacon requests.

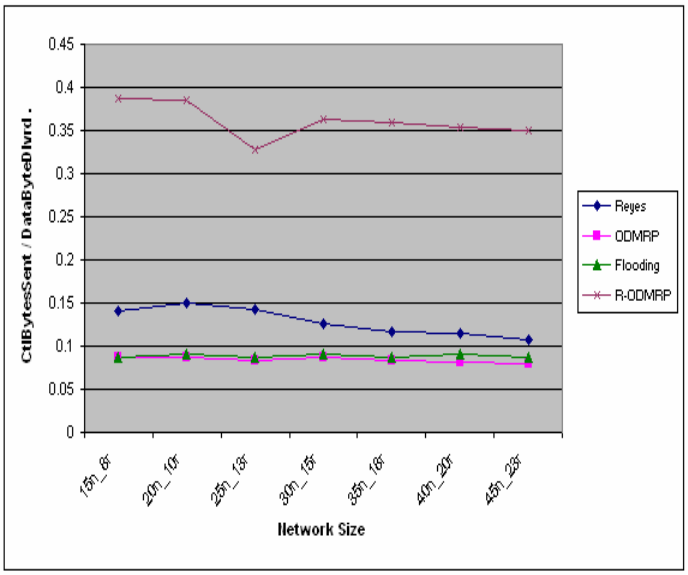


Figure 5.13: Sparse Medium Mobility Network Control Overhead

As the scenarios become sparser, Figure 5.13 shows Reyes control overhead rising gradually, since many more data packets are missed in the initial delivery due to missing links, and more work must be done to handle this. The number of data packets received due to packet header requests, resend requests and beacon requests all show significant increases as the network becomes sparser. Flooding shows control overhead holding steady in the sparsest network scenarios due to the steep dropoff in packets that reach receivers (note: flooding's control overhead consists of the bytes of the IP header, which are also counted as part of the overhead in the other protocols). ODMRP shows a similar holding steady in control overhead, being best effort.

The control overhead metric for Reyes shown in Figure 5.13 is a bit misleading. The increasing frequency and number of partitions triggers Reye's mechanism for creating datapaths at the edges of networks, enhancing reliable delivery across what would have been partitions. R-ODMRP, with its less targeted operations, generates a far greater and more steeply increasing amount of control overhead, interfering with the increasingly difficult task of packet delivery, with fewer links and more partitions. R-ODMRP must

work harder to overcome the negative effects of the overhead, resulting in increasing latency as the network becomes more sparse.

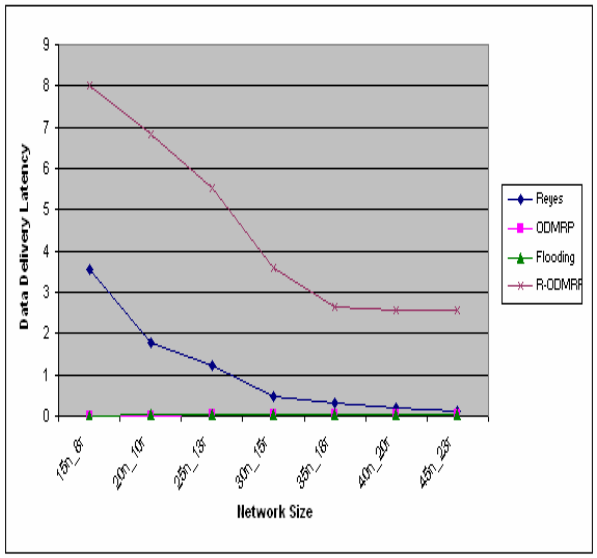


Figure 5.14: Sparse Medium Mobility Network Latency

Figure 5.14 shows latency for Reyes rising as expected, since the amount and length of network partitioning increases as the network becomes sparser and nodes must reconnect to receive missed packets. However, it increases far less than R-ODMRP, with much better results for reliable delivery. It not only delivers more packets than R-ODMRP, it delivers them faster on average as well. Latency for the best effort protocols is expected to be low here, since fewer packets are delivered, and those delivered are strictly a result of the initial source send. For Reyes, not only does the count of packets received for all three request types show a steep increase here, but the average latency per packet for each request type shows a steep increase as well. In sparse networks high latency seems unavoidable for Reyes. Sparse networks are a weak point for R-ODMRP, since its operations generate high overhead which works directly against reliable delivery.

5.8.4 Sparse High Mobility Network Results

In the Sparse High mobility Network Figures, 5n_70ms denotes a network of 5 nodes, 3 receivers, moving at 70 m/sec.

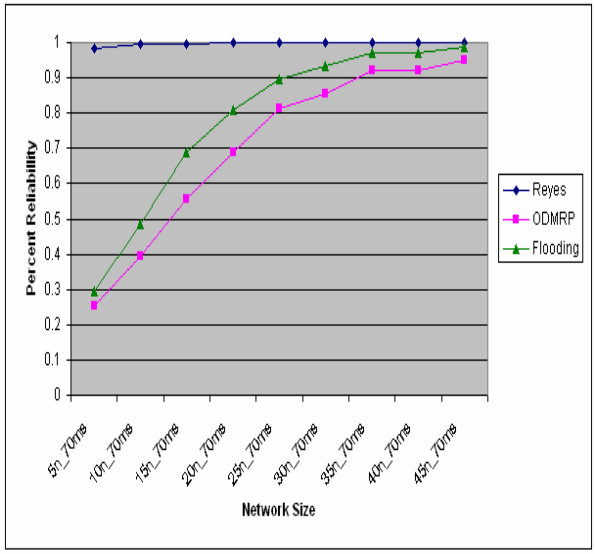


Figure 5.15: Sparse High Mobility Network Reliability

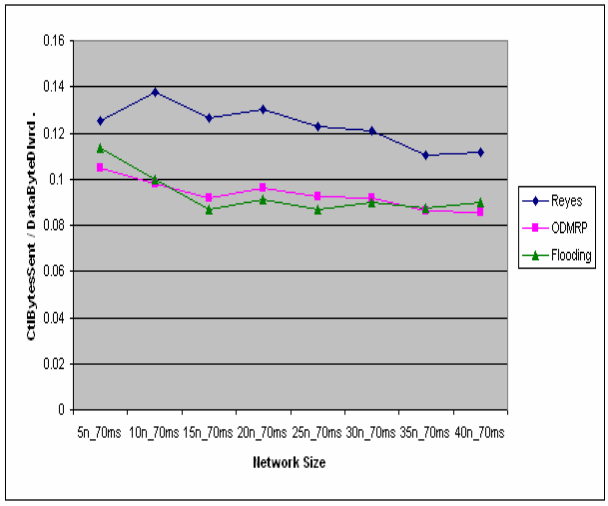


Figure 5.16: Sparse High Mobility Control Overhead

Figures 5.15, 5.16 and 5.17, show that increasing mobility increases reliability for Reyes in sparse networks, while decreasing reliability across the board for flooding. Another advantage for Reyes with high mobility, latency decreases significantly in the sparsest networks when compared to medium mobility.

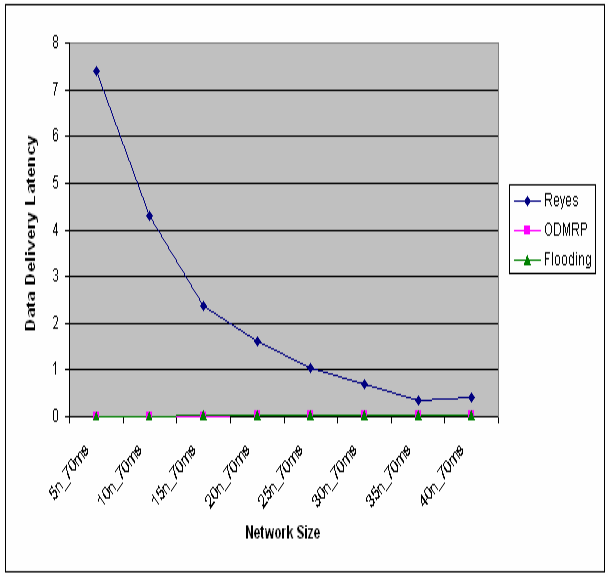


Figure 5.17: Sparse High Mobility Network Latency

5.8.5 Dense Medium Mobility Network Results

In the Dense Network Figures, 450n_225r denotes a network of 450 nodes, with 225 of them being receivers. As Figure 5.18 shows, Reyes performs very well in dense networks in terms of reliability, outperforming both flooding and ODMRP.

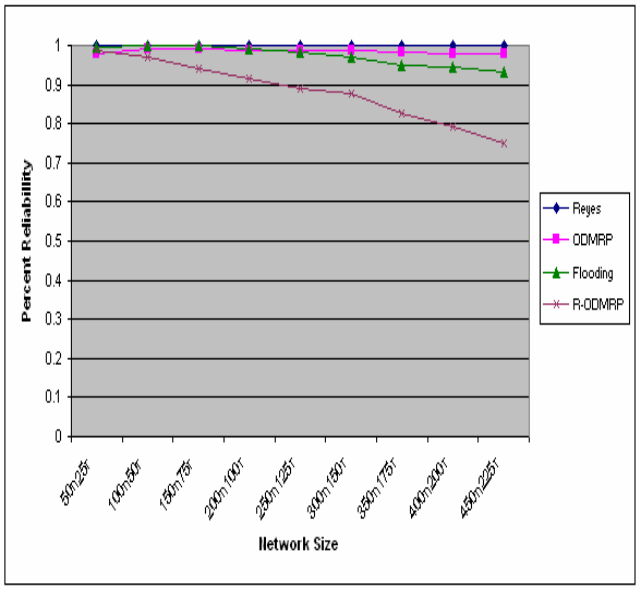


Figure 5.18: Dense Medium Mobility Network Reliability

In the worst test run of the set for Reyes, only 48 packets were not yet delivered to receivers. This could be due to the last packet not yet propagating through the network, or to recent missed packets. In either case, this score would likely rise to 100% if the protocol were allowed to operate, via beacon requests for a few seconds after the source stops sending at 600 seconds.

ODMRP and flooding also perform well, ODMRP's restricted set of data paths provide for better reliable delivery than flooding, where every data path is active all the time. In terms of reliable delivery, R-ODMRP has a restricted set of data paths similar to ODMRP, but unfortunately the large amount of overhead required for control operations only increases with network density, causing reliable delivery to drop off at a steeper rate than the best effort protocols as the network becomes denser. As the network grows denser, ODMRP experiences a gradual decline in reliability, due to a greater amount of link contention, while flooding shows a more significant drop off due to link contention operating at a much higher degree, since no data paths are ever disabled.

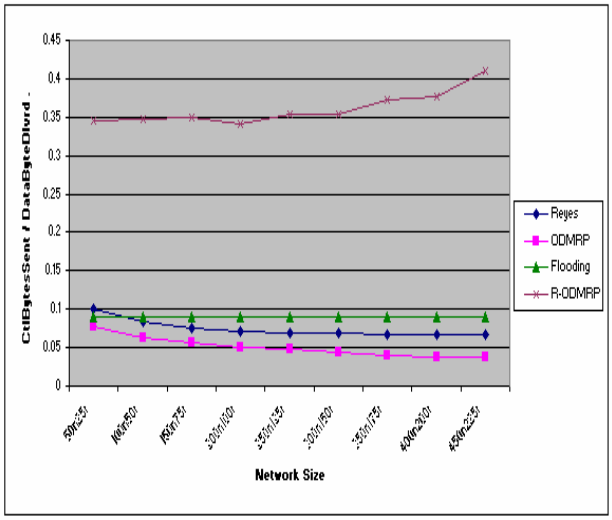


Figure 5.19: Dense Medium Mobility Network Control Overhead

As the scenarios become denser, Figure 5.19 shows Reyes and ODMRP control overhead actually dropping below that of flooding. Dense networks are a weak point for flooding. While more datapaths are enabled and the total control bytes sent increases across the Reyes experiments, the number of data bytes delivered per datapath due to an

increasing receiver count grows at an even faster pace, so the overall overhead metric decreases.

Since Reyes has more overhead than ODMRP, given its reliability mechanisms, dense networks could have been a weak point for Reyes reliable delivery, but its increased overhead is very targeted and controlled, to impact reliable delivery as minimally as possible. The downside of the slight overhead increase is outweighed by the increase in reliability. The flooding metric shows an almost imperceptably slight increase across the experiments, since its control bytes sent number is growing very slightly faster than its data bytes delivered number. R-ODMRP shows control overhead increasing with increased network density, in turn causing a steeper dropoff in reliable delivery compared to the best effort protocols.

Reyes and the two best effort protocols show similar low metrics for delivery latency, even though Reyes delivers more packets through resends, though none attributable to beacon packets. Dense networks are a strong point for Reyes. On the other hand, R-ODMRP's metric shows how hard the protocol actually has to work in dense networks. As the network becomes more dense R-ODMRP control overhead increases and latency sharply increases. While the reliable resend mechanism works harder, more packets are dropped due to increasing contention, so more resend requests must be sent.

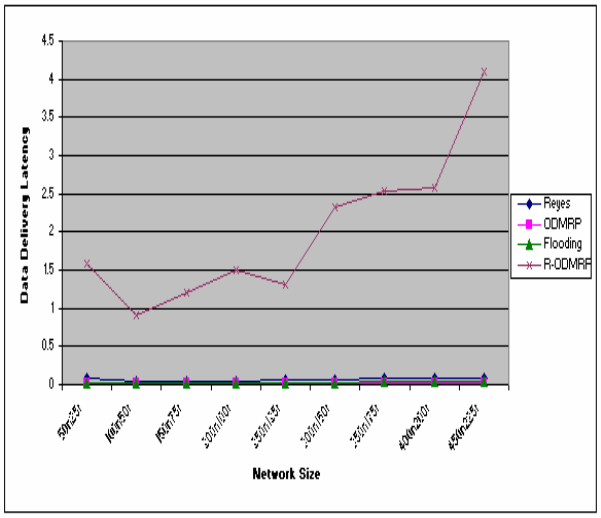


Figure 5.20: Dense Medium Mobility Network Latency

5.8.6 Dense High Mobility Network Results

In the Dense High Mobility Figures, 50n_70ms denotes a network with 50 nodes, 25 receivers, moving at 70 m/sec.

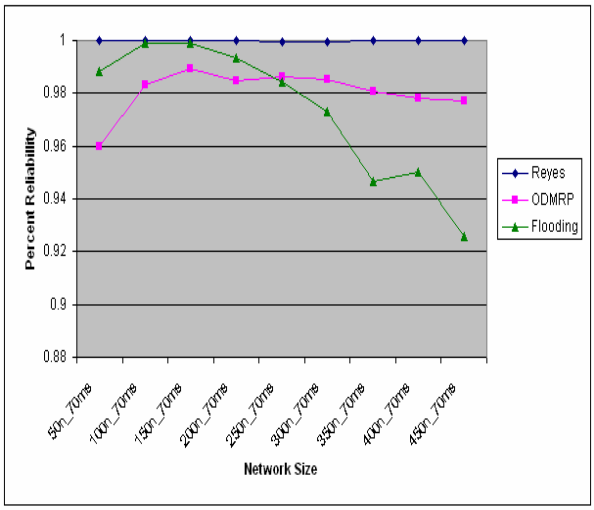


Figure 5.21: Dense High Mobility Network Reliability

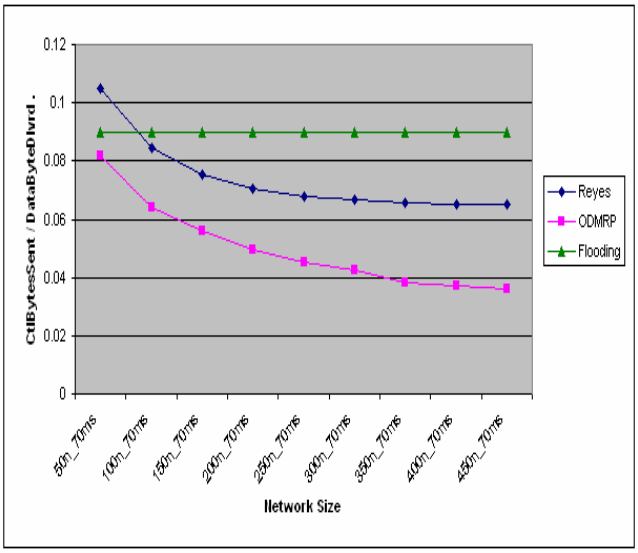


Figure 5.22: Dense High Mobility Control Overhead

Figure 5.21 shows Reyes retaining its high reliability, ODMRP losing reliability in the less dense and more dense data points, and flooding with slightly less reliability across all data points when compared with medium mobility reliability. Figure 5.22 shows flooding and Reyes to have similar control overhead, but ODMRP to have slightly

higher control overhead in the less dense data points when compared with the dense medium mobility scenario. For Reyes, the real difference in high mobility versus medium mobility dense networks is the increase in latency, seen in Figure 5.23. More resend requests were dropped due to contention, delaying eventual redelivery. No increase was seen for ODMRP and flooding.

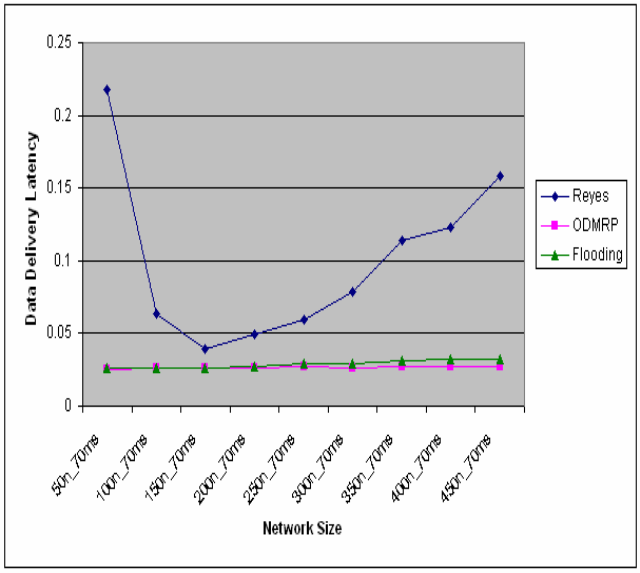


Figure 5.23: Dense High Mobility Network Latency

5.8.7 Mobility Results

In the Mobility Figures, 50n25r_70s denotes the entry for 50 nodes, 25 being receivers, moving at 70 meters per second.

Figure 5.24 shows reliability was excellent across all mobility scenarios for Reyes, with the worst case being 32 packets not yet received (.999601% reliability) at the 599.95 second mark. This is a number expected under any network conditions, since source sends are stopped after the metrics are taken, at 600 seconds. Usually packets have not yet propagated to all nodes, and recently missing packets have either not yet been requested or received. Higher mobility provides each node a greater chance of being neighbors with other nodes storing needed packets. Flooding shows good reliability

across all mobility tests, dropping only slightly at the highest mobility. ODMRP results show reliability decreasing as node speed increases, due to links broken more often with no means to make up for missed packets. This scenario is ideal for R-ODMRP, allowing its reliability mechanism to operate well, providing enhanced reliability when compared to ODMRP. The network is of light density, and the traffic rate is relatively low.

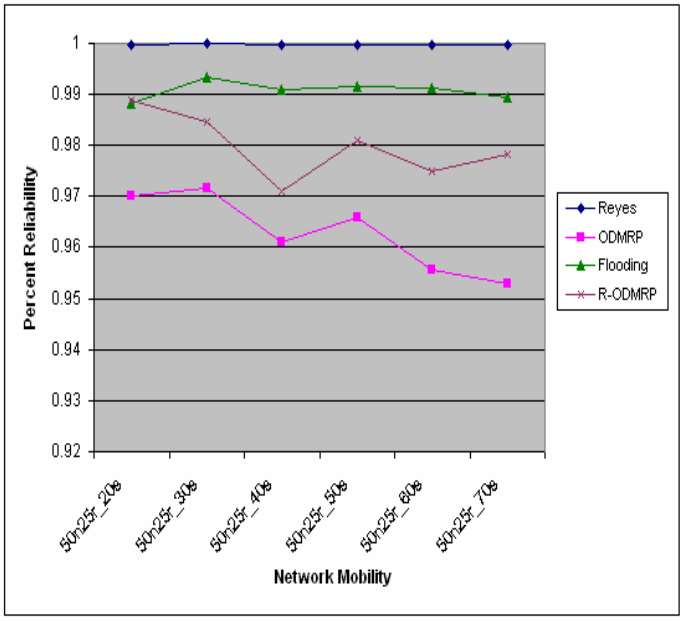


Figure 5.24: Mobility Reliability

Figure 5.25 shows control overhead metrics that are consistent for all three protocols. Though Reyes had a greater number of control bytes used than ODMRP or flooding, its higher packet delivery metric decreased its control bytes sent per data byte delivered metric to be close to both flooding and ODMRP. Since ODMRP is best effort, there are no protocol changes triggered by increased mobility. Link breaks due to mobility tend to reduce the number of control bytes sent as mobility increases, since the packets with the control bytes are more likely to be dropped. Flooding control overhead is relegated to the IP portion of the packet header, with no control overhead added for increased mobility. ODMRP control overhead is consistently less than flooding, because of the reduced amount of data packets sent due to fewer operational data paths. Figures 23 and 24 together show the effects of both R-ODMRP's source based centralized algorithm, and its

non-targeted data request operations as increased mobility puts greater stress on communication. As mobility increases the links carrying control information for both these operations are broken more frequently, delaying reception of initially missed packets and causing the gradually increasing overhead and more sharply increasing latency seen.

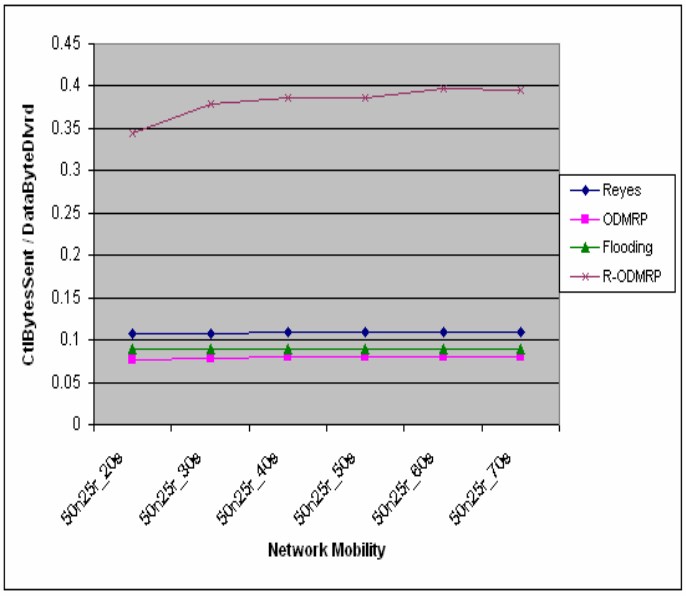


Figure 5.25: Mobility Control Overhead

Surprisingly, Reyes also shows consistent control overhead as mobility increases. The expectation was that as mobility rose, Reyes would have to use more control overhead to make up for more missing packets. What actually happened was that though packets received from the initial send decreased slightly as mobility rose, resent packets sent in response to packet header requests and resend requests rose slightly to account for this. Both mechanisms create very little additional overhead: resend request packets just had more gap sequence numbers in packets that would have been sent anyway, and the flag bytes used for packet header requests were counted as control bytes whether or not they were used. The very slight increase in Reyes control overhead was mostly generated from the resent data packets.

In Figure 5.26, Reye's gradual but increasing latency as mobility rose reflects the greater amount of work it performed due to the rising number of packets eventually received from resend requests. If a node is missing packets, but is a connected member of a neighborhood, it must wait for the stability phase to request packets. Then, if its first unicast request is unanswered, it must wait for the next stability phase. Beacon requests would be answered faster with other nodes in range, but in a connected state the impact of multicast beacon requests and multiple replies on network bandwidth would be unacceptable, impacting all metrics. Latency for Flooding and ODMRP are very low, since they operate on 'best effort' delivery.

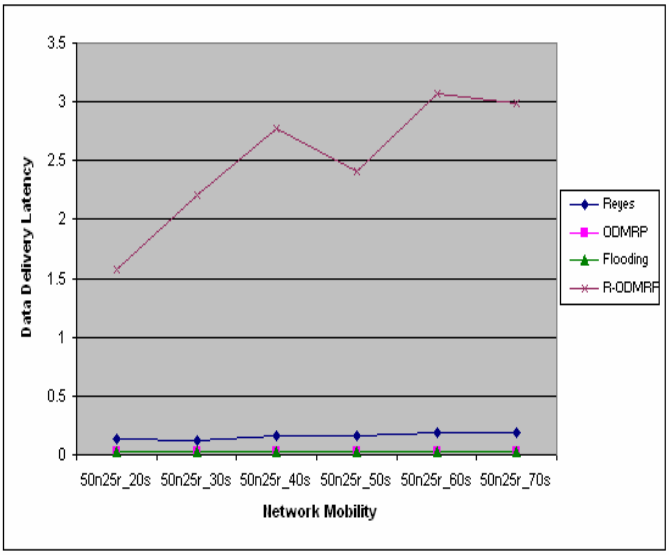


Figure 5.26: Mobility Latency

5.8.8 Traffic Rate Results

In the Traffic Rate Figures, 50n25r_24ps denotes a network of 50 nodes, 25 of them receivers, with packets sent from the source at a rate of 24 packets per second. In these scenarios network link contention became an increasingly difficult factor to deal with. Figure 5.27 shows flooding was the worst performer in terms of reliability as the traffic rate rose. In low traffic rate scenarios flooding's overly redundant packet delivery adds reliability, but as the data rate rises and links become congested this becomes a liability. ODMRP, with its restricted number of data paths, is less affected by traffic rate, but still

gradually deteriorates with rising data rates. Reyes maintained its reliability for higher data rates, but eventually declined as well. A "cool down" period after the source halts sending at 600 seconds could bring reliability up to 100% even for high traffic scenarios.

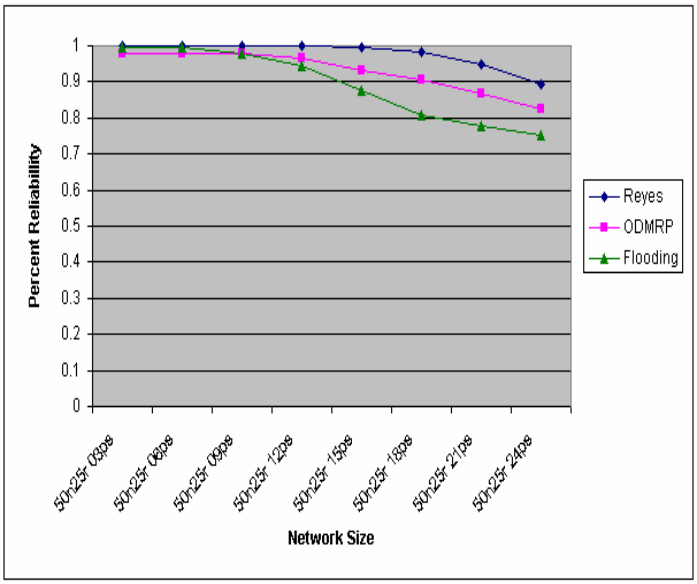


Figure 5.27: Traffic Rate Reliability

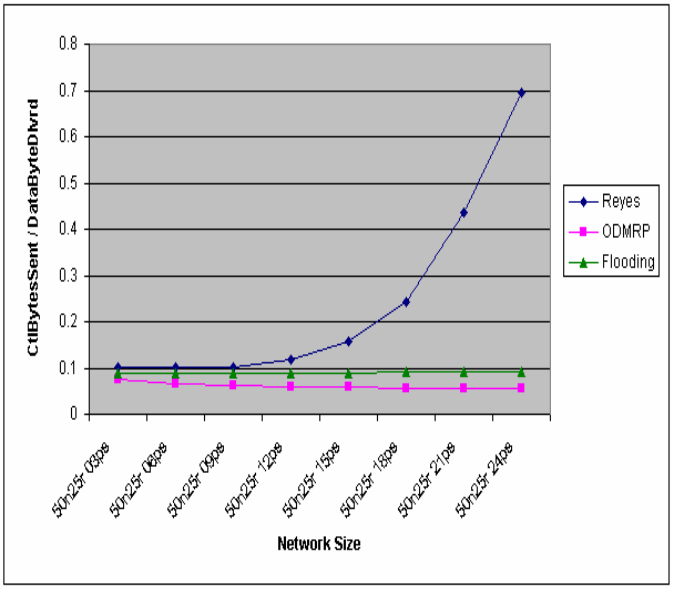


Figure 5.28: Traffic Rate Control Overhead

Figure 5.28 shows that Reyes reliability here comes at a much greater overall cost. Since the mechanisms of the protocol all consume some amount of network bandwidth,

they act in direct competition with the increasing load of ongoing data delivery. Reyes must work proportionately harder to overcome link contention and missing packets as data rates rise. As expected, control overhead rises at an increasing rate as traffic rates rise, even though Reyes maintains a greater delivery ratio.

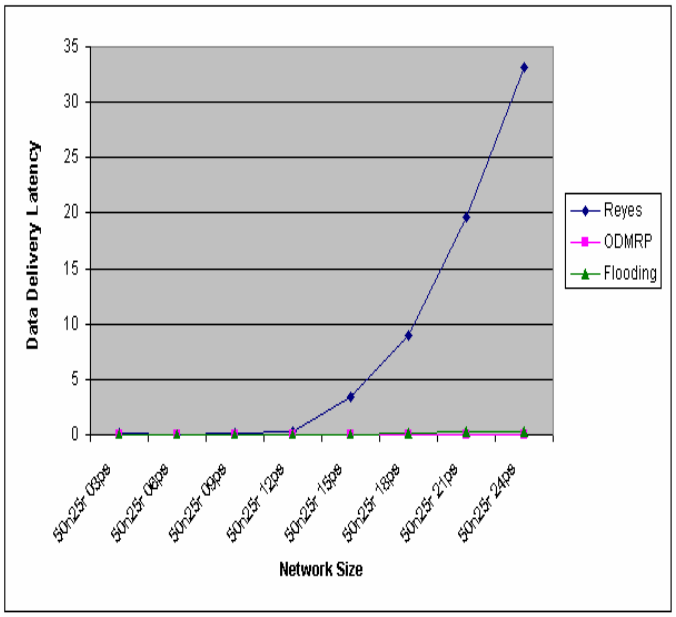


Figure 5.29: Traffic Rate Latency

Figure 5.29 shows a sharp rise in latency as traffic rate increases for Reyes. This is mostly due to an ongoing increase not only of the percentage of received packets due to resend requests, but also to the increase in time for these packets to reach the requestor. As link contention increases, a much greater percentage of these unicast requests and replies do not reach the intended recipient. It is likely that this increasing delay in reception of requested packets accounts for the lower reliability metric for Reyes at higher data rates, and that if the protocol were allowed to operate after the last packet is sent at 600 seconds, reliability would reach one hundred percent even at the highest data rate scenarios.

5.9 Conclusions for Reyes

This section introduced Reyes, a reliable multicast routing protocol for MANETs. Reyes is a low overhead, scalable protocol that allows a node missing packets three mechanisms for requesting them, based on network conditions local to the node. These mechanisms are each targeted to the specific causes of missed packets in ad hoc multicast communications, broken and missing links in sparse networks, temporary and ephemeral links in high mobility networks and links overloaded with contention in dense and high traffic networks.

There are several ad hoc reliable multicast protocols that have been developed in recent years. Of these, only a few have been evaluated in a standard detailed simulator with reproducible results. For these, the scenarios tested have either been unrealistic (e.g., 1 packet per second for 500 seconds, with 1500 seconds 'cleanup time'), or only partial results were shown, or the authors knowingly traded off one performance metric for another (e.g., reliability is kept high by consciously sacrificing latency).

While Reyes is developed to provide high reliability, its reliability mechanisms were designed specifically to also provide low latency and low bandwidth consumption. Reyes has been extensively evaluated across a wide variety of common network scenarios, including increasing mobility and increasing data traffic rate scenarios, and both increasingly sparse and increasingly dense networks, under both medium and high mobility conditions. Performance evaluations of Reyes show it to perform very well under all these scenarios, achieving close to 100% reliable packet delivery to all receivers in most scenarios using the ns-2 network simulator. Reyes reliability is higher than the current standards of flooding and ODMRP across all scenarios, and higher than R-ODMRP as well. Its control overhead and latency are within range of flooding and ODMRP, and better than R-ODMRP, except in the extremely stressful scenarios of very high data rates and very sparse and dense networks.

6 General Conclusions and Future Work

The majority of current research in reliable ad hoc multicast communication protocols to date appears to fall into three distinct categories: **Probabilistic Protocols**, **Deterministic Topological Protocols based on global store and resend techniques**, and **Deterministic Non-Topological Protocols based on local store, link determination and send/resend techniques**.

From current research, results indicate that **Probabilistic protocols** can increase reliability with lower associated overhead, but ultimately have great difficulty providing fully reliable packet delivery under multiple scenarios. High latency is also a difficult issue for Probabilistic protocols to deal with, given the basic premise of link determination. In these protocols, it is often the case that the probabilistic mechanisms defined to determine packet transmit status are modified by a series of patches needed to overcome the issues caused by the basic probabilistic technique itself.

The second category of **Deterministic Topology-Based Protocols based on global store and resend techniques** has been proven to have a much greater potential for fully successful reliable packet delivery as a result of the protocols developed and described in this dissertation, R-ODMRP and Reyes. R-ODMRP proved that a global store and resend technique could result in measurably increased reliable packet delivery when compared to the most reliable "best effort" protocols available. Once implemented, it became a very successful testbed from which to run many experiments designed to discover impediments to reliable packet delivery, and to develop techniques to overcome them in ad hoc networks. These experiments resulted in a design from scratch of the protocol Reyes, which was specifically targeted to maximize reliable delivery while minimizing associated latency and overhead. With these goals in mind from the very beginning, Reyes turned out to be very successful across the metrics of reliability, overhead and latency, and across the scenarios of dense networks, sparse networks, highly mobile networks and high data traffic networks. At the start of this work it appeared counterintuitive that a multicast communication protocol could successfully work to increase reliability specifically under

the scenarios where the negative impacts of its added control overhead would have their greatest effect, namely dense networks and high traffic networks, but the techniques in Reyes accomplish this by specific targeting to minimize not only the control overhead but also the network overhead associated with multiple data paths. Essentially, Reyes proved the potential of this area of research to realize the goal of full reliability across all feasible mobile ad hoc network scenarios.

After developing Reyes, it is clear that this category of research still has many possibilities for maximizing reliability and minimizing the costs in terms of latency and overhead. Reyes operations essentially take time to set a network state where nodes have the local (and extended) knowledge needed to enhance reliable reception, then take advantage of that state with a period of time during which receiver nodes make use of the ability to request and receive missed packets. An enhanced approach could be that a much smaller portion of packet bytes in every packet header let both upstream and downstream nodes know about nearby receiver nodes and their responsibilities on an ongoing fluid basis, with no requirement for the periodic operations that potentially increase latency and control overhead such as those implemented by Reyes. This information could be transmitted in a much smaller header, with a limited number of hops after which the information is dropped. Resend requests could operate in a similarly fluid basis, sent locally to receivers most recently learned, with a greatly minimized amount of data in the header of a standard data packet that travels no more than a few hops before being purged. This type of operation would work within the bounds of normal data packet dissemination, with very little need for additional protocol control data. The global packet dissemination method itself could be examined in much greater detail through experimentation, to find optimal techniques for data path creation for every type of network scenario, along the lines of the experiments used to design Reyes' data path creation techniques, which worked to minimize data paths in dense networks and maximize data paths in sparse networks. The goals here would be a greater amount of optimization of data paths, specifically looking at ways to do this with little, or even no, control overhead.

The third category of **Deterministic Non-Topological Protocols based on local store, link determination and send / resend techniques** also seems to be rich with possibilities for maximizing reliability while minimizing latency and overhead. This area of investigation is one of the least explored in terms of current research. One central problem here is global determination of fully reliable delivery on a per packet basis. This problem is dealt with in different ways by the Scribble and EraMobile protocols. This problem is a difficult one to solve, since the protocols in this category use mechanisms that operate locally at the level of an individual node, and aligning a global view with this is counterintuitive.

Local link determination on a per node basis is a key issue for this category as well, being a very important factor in both the degree of reliable delivery, and the amount of network overhead used. Link determination also plays a big part in the latency metric, since links that could have been established but are not could eventually result in the triggering of some sort of request/resend mechanisms that will always increase latency above that of reception of original source sent packets. Storage and resend mechanism development would require a whole new approach for this category than for deterministic protocols based on global techniques, as the requirements for mechanisms will be very different.

In summary, there are many avenues to explore on the path to developing a mobile ad hoc multicast protocol with fully reliable delivery that minimizes latency and network overhead in every network scenario, but the work done in the research for this dissertation has shown concrete results along this path.

References

[BM05] Biswas, S., Morris, R., ExOR, Wireless Network Protocol, with standard 802.11 radios. Commercially available from Meraki, start-up company founded by the authors. 2005.
ExOR, Opportunistic Multi-Hop Routing for Wireless Networks.

[B91] Birman, K. "Building Secure and Reliable Network Applications", Manning Publishing Company, Greenwich, CT, and Prentice Hall. 1991.

[BMJHJ98] Broch, J., Maltz, D., Johnson, D., Hu, Y., Jetcheva, J. "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", Proc. of the Fourth Annual ACM/IEEE International Conference on Mobile Computing And Networking, Dallas, TX., October, 1998.

[BM98] Bagrodia, R., Meyer, R., et al, "PARSEC: A Parallel Simulation Environment for Complex Systems", Computer Magazine, 1998.

[CRB01] Chandra, R., Ramasubramanian, V, Birman, K. "Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks", Proc. of the 21st Int'l Conference on Distributed Computing Systems, Phoenix, Arizona. April, 2001.

[CGZ98] Chiang, C.C., Gerla, M., Zhang, L. "Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks", *ACM-Baltzer Journal of Cluster Computing: Special Issue on Mobile Computing*, vol. 1, no. 2, pages 187 – 196. December, 1998.
FGMP - Forwarding Group Multicast Protocol.

[C99] CMU Monarch Project "The CMU Monarch Projects wireless and mobility extensions to ns", The CMU Monarch Project, August 1999. Available at http://www.monarch.cs.cmu.edu/.

[CU] Cornell University, JiST/SWANS Java in Simulation Time / Scalable Wireless Ad Hoc Network Simulator. Available at: http://jist.ece.cornell.edu/.

[DB08] Detti, A., Blefari-Melazzi, N. OBAMP, "Overlay, Boruvka-based, Ad-hoc multicast Protocol: description and performance analysis", Wireless Communications and Mobile Computing, Wiley, 2008
Available from:
http://netgroup.uniroma2.it/Andrea_Detti/obamp/index.html

[DMM02] Das, S.K., Manoj, B.S., Murthy, C.S.R. "A Dynamic Core Based Multicast Routing Protocol for Ad hoc Wireless Networks", In *Proc. of the 3rd ACM International Symposium on Mobile and Ad-hoc Networking & Computing (MobiHOC)*, pages 24 - 35, Lausanne, Switzerland. June, 2002.
DCMP - Dynamic Core Based Multicast Routing Protocol.

[DMM02b] Das, S.K., Manoj, B.S., Murthy, C.S.R. "Weight Based Multicast Routing Protocol for Ad hoc Wireless Networks", In *Proc. of IEEE GlobeCom 2002, vol. 1, page 17-21*. November, 2002.
WBM -

[DSS01] Devarapalli, V., Selcuk, A.A., Sidhu, D. "MZR: A Multicast Protocol for Mobile Ad Hoc Networks", In *Proc. of the IEEE International Conference on Communications (ICC)*, pages 886 - 891, Helsinki, Finland. June, 2001.
MZR - Multicast Zone Routing.

[DFKS06] Drabkin, V., Friedman, R., Kliot, G. Segal, M. ”RAPID: Reliable Probabilistic Dissemination in Wireless Ad-Hoc Networks”, Technical Report CS-2006-19, Department of Computer Science, The Technion, December, 2006.

FJMLZ97 Floyd, S., Jacobson, V., McCanne, S., Liu, C., Zhang, L., “ A reliable multicast framework for light-weight sessions and application-level Framing”, IEEE/ACM Transactions on Networking, vol. 5, no. 6, December 1997, pages 784-803.

[FV02] Fall, K, Varadhan, K,. Editors “The ns Manual”, The VINT Project, UC Berkeley, LBL, USC/ISI, and XEROX PARC. April, 2002.
Available at http://www-isi.edu/nsnam/ns/.

[GM99] Garcia-Luna-Aceves, J.J., Madruga, E.L. "The Core Assisted Mesh Protocol", *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, vol. 17, no. 8, pages 1380 – 1394. August, 1999.
CAMP - Core-Assisted Mesh Protocol.

[GO07] Genc, Z., Ozkasap, O. “EraMobile: Epidemic-based Reliable and Adaptive Multicast for MANETs”, *Proc. of the Wireless Communications and Networking Conference (WCNC),* Hong Kong, China. March, 2007.
Available from:

http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=4224245&arnumber=4225046&count=810&index=800

[GS99] Garcia-Luna-Aceves, J.J., Spohn, M., "Source-Tree Routing in Wireless Networks", Proceedings of IEEE ICNP 1999, page 273-282, October 1999.

[GSPS02] Gopalsamy, T., Singhal, M., Panda, D., Sadayappen, P. "A Reliable Multicast Algorithm for Mobile Ad Hoc Networks", *Proc. of the Distributed Computing Systems Workshops,* pp. 563-570, Vienna, Austria. July, 2002.

[HK09] Hsiu, P.C., Kuo, T.W. "A Maximum-Residual Multicast Protocol for Large-Scale Mobile Ad Hoc Networks", IEEE Transactions on Mobile Computing, 2009.
MRMP – Maximum-Residual Multicast Protocol.
Available from:
http://ieeexplore.ieee.org/xpls/pre_abs_all.jsp?isnumber=4358975&arnumber=4796204

[JHMJ01] Jetcheva, J.G., Hu, Y., Maltz, D., Johnson, D. "A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks", *Internet Draft draft-ietf-manet-simple-mbcast-01.txt.* July, 2001.
DSR-MB - Simple Protocol for Multicast and Broadcast using DSR.

[JJ01] Jetcheva, J.G., Johnson, D. "Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks", In *Proc. of the 2nd ACM International Symposium on Mobile and Ad-hoc Networking & Computing (MobiHOC)*, pages 33 - 44, Long Beach, CA. October, 2001.
ADMR - Adaptive Demand-Driven Multicast Routing.

[JC98] Ji, L., Corson, M.S. "A Lightweight Adaptive Multicast Algorithm", In *Proc. of the IEEE Global Telecommunications Conference (Globecom)*, pages 1036 - 1042, Sydney, Australia. November, 1998.
LAM - Lightweight Adaptive Multicast.

[JC01] Ji, L., Corson, M.S. "Differential Destination Multicast-A MANET Multicast Routing Protocol for Small Groups", In *Proc. of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1192 - 1202, Anchorage, AK. April, 2001.
DDM (Differential Destination Multicast)

[KC05] Kaan, B.; Cem, E., AQM, “Ad Hoc Quality of Service Multicast Routing", *Elsevier Science Computer Comms.*, vol. 29, no. 1, pg 136 – 148. December, 2005.

[KR05] Klos, L., Richard III, G.G., “ Reliable Ad Hoc Group Communication using Local Neighborhoods”, *Proceedings of the IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, 2005.

[LJMVCA03] Laouiti, A.; Jacquet, P.; Minet, P.; Viennot, L. ; Clausen, T.; Adjih, C. "Multicast Optimized Link State Routing", INRIA research report,RR-4721. February, 2003.
MOLSR (Multicast Optimized Link State Routing)
(http://www.inria.fr/rrrt/rr-4721.html).

[LGC99] Lee, S.J., Gerla, M., Chiang, C.C. "On-Demand Multicast Routing Protocol", In *Proc. of the Wireless Communications and Networking Conference (WCNC)*, pages 1298 - 1302, New Orleans, L.A. September, 1999.
avail:http://www.cs.ucla.edu/NRL/wireless/PAPER/odmrp-wcnc99.ps.gz
ODMRP - On-Demand Multicast Routing Protocol.

[LK00] Lee, S., Kim, C., “Neighbor Supporting Ad Hoc Multicast Routing Protocol”, Proceedings of ACM MOBIHOC 2000, Page 37-50, August, 2000.

[LSG00] Lee, S.J., Su, W., Gerla, M. Internet Draft, “On-Demand Multicast Routing Protocol(ODMRP) for Ad Hoc Networks”, draft-ietf-manet-odmrp02.txt. January, 2000.

[LSHGB00] Lee, S. J., Su, W., Hsu, J., Gerla, M., Bagrodia, R. “A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols”, Proc. of IEEE INFOCOM 2000, Tel Aviv, Israel, March 2000.

[LTMB99] Liu, M., Talpade, R.R., McAuley, A., Bommaiah, E. "AMRoute: Adhoc Multicast Routing Protocol", *University of Maryland CSHCN Technical Report 1999-1*, College Park, MD.
AMRoute - Adhoc Multicast Routing Protocol.

[LEH03] Luo, J., Eugster, P.T., Hubaux, J.P. “Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks”, INFOCOM, San Francisco, CA. March, 2003.
Mobgen-ss app. at: http://toilers.mines.edu/Public

[M09] Macker, J., editor, SMF Design Team, SMF, "Simplified Multicast Forwarding for MANET", work in progress, 2009.
Available from http://www.ietf.org/id/draft-ietf-manet-smf-09.txt.

[ML02] Moustafa, H., Labiod, H., "SRMP: A Mesh-based Protocol for Multicast Communication in ad hoc networks", In *Proc. of the 2002 Intn'l Conf. on Third Generation Wireless and Beyond*, pg 43 - 48, San Francisco, CA. May, 2002.
SRMP (Source Routing-based Multicast Protocol)

[NCB04] Navidi, W., Camp, T., Bauer, N. "Improving the Accuracy of Random Waypoint Simulations Through Steady-State Initialization", *Proc. of the 15th Int'l Conf. on Modeling and Simulation*, pp. 319 – 326. March, 2004.

[NCB04a] Navidi, W., Camp, T., Bauer, N. Mobgen-ss app at: http://toilers.mines.edu/Public

[OTV01] Obraczka, K., Tsudik, G., Viswanath, K. "Pushing the Limits of Multicast in Ad Hoc Networks", *Proc. of the 21st International Conference on Distributed Computing Systems*, Phoenix, Arizona, April, 2001.

[OKS99] Ozaki, T., Kim, J.B., Suda, T. "Bandwidth-efficient multicast routing protocol for ad-hoc networks", In *Computer Communications and Networks, Conf. Proceedings*, pages 10-17, Boston, MA. September, 1999.
BEMRP - Bandwidth-Efficient Multicast Routing Protocol.

[PB94] Perkins, C.E., Bhagwat, P., "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", Proceedings of ACM SIG-COMM 1994, page 234-244, August 1994.

[ROLTG03] Rajendran, V., Yi, Y., Obraczka, K., Lee, S.J., Tang, K., Gerla, M. "Reliable, Adaptive Congestion Controlled Ad Hoc Multicast Transport Protocol: Combining Source Based and Local Recovery", UCSC Tech. Report. University of California, 2003.

[RG04] Ravindra, V., Garcia-Luna-Aceves, J.J. "Efficient and Robust Multicast Routing in Mobile Ad Hoc Networks", In *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 304- 313, Fort Lauderdale, FL. October, 2004.
PUMA - Protocol for Unified Multicasting Through Announcements.

Available from: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1392169. A NS-2 implementation by Sidney Doria is available in: <http://puma-adhoc.cvs.sourceforge.net/puma-adhoc/Puma/.

[RP99] Royer, E.M., Perkins, C.E. "Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol", In *Proc. of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 207 - 218, Seattle, WA. August, 1999. MAODV - Multicast Ad-hoc On-Demand Distance Vector routing.

[SMM02] Sisodia, R. S., Manoj, B. S., Murthy, S. R., "A Preferred Link-Based Multicast Protocol for Wireless Mobile Ad Hoc Networks", Journal of Communications and Networks, vol. 4, no. 1, page 14 – 21, March 2002.

[SN] Scalable Networks: http://www.scalable-solutions.com

[SP02] Shu, L., Poppe, D. "Assuring Message Delivery in Mobile Ad Hoc Networks with Packet Erasure Recovery", Proc. of the Distributed Computing Sys. Workshops, pp 14-19, Vienna, Austria. July, 2002.

[SSB99] Sinha, P., Sivakumar, R., Bharghavan, V. "MCEDAR: Multicast Core-Extraction Distributed Ad hoc Routing", In *Proc. of the Wireless Communications and Networking Conference (WCNC)*, pages 1313 – 1317, New Orleans, LA. September, 1999. MCEDAR - Multicast Core-Extraction Distributed Ad hoc Routing.

[TOLG02] Tang, K., Obraczka, K., Lee, S.J., Gerla, M., "A Reliable Congestion-Controlled Multicast Transport Protocol in Multimedia Multi-hop Networks", *The 5th International Symposium on Wireless Personal Multimedia Communications*, pp 252-256, Honolulu, USA. October, 2002.

[TGB00] Toh, C.K., Guichal, G., Bunchua, S. "On-demand associativity-based multicast routing for ad hoc mobile networks (ABAM)", In *Proc. of the 52nd IEEE VTS Vehicular Technology Conference (VTC) 2000 Fall*, vol. 3, pages 987- 993, Boston, MA. September, 2000. ABAM - On-Demand Associativity-Based Multicast.

[TFWME04] Transier, M.; Füßler, H.; Widmer, J.; Mauve, M.; Effelsberg, W. "Scalable Position-Based Multicast for Mobile Ad-hoc Networks", In *Proc. of the 1st International Workshop on Broadband Wireless Multimedia: Algorithms, Architectures and Applications (BroadWim)*,

San José, CA. October, 2004.
SPBM (Scalable Position-Based Multicast)
Available from: http://www.informatik.uni-mannheim.de/pi4/publications/Transier2004c.pdf. A NS-2 implementation available from:http://www.informatik.uni-mannheim.de/pi4/projects/pbm/kernel.html

[TG01] Gopalsamy, T. B., "Multicasting in Mobile Ad Hoc Networks", Technical Report, 2001.

[UC01] University of California website: The USC/ISI ns-2 version of ODMRP,. Supported by NSF's NGE Program and the IMAHN Project. Copyright 1991-1997, Regents of the University of California. 2001.

[VE04] Vollset, E., Ezhilchelvan, P. "Scribble: an Efficient Reliable Manycast Protocol for Ad-hoc Networks", The 1st IEEE Int'l Conference on Mobile Ad-hoc and Sensor Systems, Fort Lauderdale, Florida, USA. October, 2004.

[MG96] Murthy, S., Garcia-Luna-Aceves, J.J., "An Efficient Routing Protocol for Wireless Networks", ACM Mobile Networks and Applications Journal, Special Issue on Routing in Mobile Communication Networks, Vol. 1, no. 2, page 183-197, October 1996.

[WT99] Wu, C.W., Tay, Y.C., "AMRIS: A Multicast Protocol for Ad Hoc Wireless Networks", In *Proc. of the IEEE Military Communications Conference (MILCOM)*, pages 25 - 29, Atlantic City, N.J. November, 1999.
AMRIS - Ad hoc Mcast Routing protocol utilizing Incrsg id-numberS.

[ZS00] Zhou, Hl, Singh, S., "Content based multicast (CBM) in ad hoc networks", In *Proc. of the 1st ACM International Symposium on Mobile and Ad-hoc Networking & Computing (MobiHOC)*, pages 51 – 60, Boston, MA. August, 2000.
CBM - Content Based Multicast.

[ZBG98] Zeng, X., Bagrodia, R., Gerla, M. "GloMoSim: a Library for the Parallel Siulation of Large-scale Wireless Networks", PADS, 1998.

Vita

Lawrence Klos was born in Vacaville, California, and raised in Klamath Falls, Oregon. He attended the University of Oregon, receiving a B.S. in Mathematics in 1984, then worked for a time in the Information Technology field in Portland, Oregon.

Following that, he attended Harvard University, receiving a Master of Architecture degree in 1991, then worked for a time with architecture firms in both Washington D.C. and New Orleans.

After this, he attended the University of New Orleans, receiving an M.S. degree in Computer Science in 1999. On completing this degree, he entered the DENAS Ph.D. program at the University of New Orleans, finishing his coursework and comprehensive exam while in attendance locally. Moving to Sacramento, California, he joined Intel as a senior software engineer, while working on his dissertation research. He finally completed all requirements for his Ph.D., including defense, in the fall semester of 2009.