ROMR: ROBUST MULTICAST ROUTING IN MOBILE AD-HOC NETWORKS

by

Gretchen H. Lynn

BS, BSEd , Concord College, 1978

MS, University of Tennessee - Knoxville, 1984

Submitted to the Graduate Faculty of

the School of Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2003

UNIVERSITY OF PITTSBURGH

SCHOOL OF ARTS AND SCIENCES

This dissertation was presented

by

Gretchen H. Lynn

It was defended on

November 25, 2003

and approved by

Taieb Znati, Ph.D. (Computer Science)

Shi-Kuo Chang, Ph.D. (Computer Science)

Prashant Krishnamurthy, Ph.D.(Telecommunications)

Rami Melhem, Ph.D.(Computer Science)

Dissertation Director: Taieb Znati, Ph.D. (Computer Science)

ROMR: ROBUST MULTICAST ROUTING IN MOBILE AD-HOC NETWORKS

Gretchen H. Lynn, PhD

University of Pittsburgh, 2003

Support for multicast services is crucial for *mobile ad-hoc networks (MANETs)* to become a viable alternative to infrastructured networks. Efficient multicasting in MANETs faces challenges not encountered in other types of networks such as the mobility of nodes, the tenuous status of communication links, limited resources, and indefinite knowledge of the network topology. This thesis addresses these challenges by providing a framework and architecture with proactive and reactive components to support multicasting in MANETs emphasizing reliability and efficiency of end-to-end packet delivery. The architecture includes the *Robust Multicast Routing protocol* (RoMR) to provide multicast services to multicast applications. RoMR's proactive component calculates multiple multicast trees based on the prediction of future availability of the links and the assumption that the trees will become disconnected over time. The reactive components respond to changes in the network topology due to the mobility of the nodes and to changes in the multicast group's membership.

Sending redundant data packets over multiple paths further enhances the reliability at the cost of an increase in the use of network resources. RoMR uses approximations to Steiner trees during tree formation and forward error correction encoding techniques during packet transmission in order to counteract this increase. To avoid additional network traffic, trees are distributed only when the existing trees cannot be easily patched to accommodate changes in topology or group membership.

The novelty of the proposed protocol stems from integrating techniques that have not previously been combined into a multicasting protocol and a unique method to calculate the

relative weights of the links.

In addition to the specifications of the protocol, a simulation framework was developed to test different implementations of the various components of RoMR. Simulations compared the performance of the basic version of RoMR to a version that ignored link weights, and to a link-state multicast protocol currently being considered by the Internet Engineering Task Force. A statistical analysis of the results showed that RoMR performed better, overall, than the other two protocols.

Keywords: multicast, mobile ad-hoc networks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION AND PROBLEM DEFINITION

1.1 INTRODUCTION

Today the use of wireless devices is becoming increasingly popular. Many people communicate using a cellular phone, often eliminating the land-based phone in their residences. Computing networks, both private and corporate, are adding wireless network cards to laptop computers and incorporating access points into their topologies. Personal digital assistants (PDAs) with wireless network accessibility have also become popular with a variety of people ranging from students to executives. As the devices become more prevalent, the users will come to expect greater capabilities from the devices and networks and will also expect services that are comparable to those received in a wire-based network such as using point-to-point communications, receiving streaming multimedia and taking part in conferencing capabilities.

The technical problems associated with wireless communications are different from the problems encountered in a wire-based network. First, the transmission of a radio signal through the air is much less reliable than transmission of an electrical signal over wires. The wireless signals can be affected by environmental conditions such as rain and can possibly be blocked by objects in the signal's path. It is also affected by interference from other radio signals in the area or interference from its own signal reflected off of objects. Second, the devices may have limited resources such as battery life and memory. Third, the bandwidth available to wireless communications is a precious commodity since it is limited by federal regulations. Speeds of 11Mbps have only recently been realized in unlicensed wireless local area networks although we may soon see speeds of 54 Mbps become more common with the latest IEEE 802.11g standard [2]. Also, the fact that the amount of available bandwidth

may not be constant as the network changes is another source of complication in wireless communications.

Many of the problems have been addressed in infrastructured networks in which typically the last hop to the user is the only wireless link the signal must traverse. An example of an infrastructured wireless network is a cellular phone company's network which is divided into cells. The communications transpiring over a wireless link is handed off to another cell as the user leaves one cell and enters another.

An *ad hoc network* is a network that is dynamically reconfigurable, rapidly deployable, and does not depend on a fixed infrastructure or a central administration [36, 29]. Ad hoc network topologies can be the basis of wireless networking in fixed or mobile environments. For example, a set of students may be sitting at stations in a science lab sharing information via a wireless peer-to-peer network that they quickly set up upon the start of the investigation. This is an example of a use in a fixed environment since the students are not moving their computers around the room. On the other hand an example of a use of an ad-hoc network in a mobile environment would be a network that is set up to be used by rescue personnel as they search for a missing person over a wide area that might not have cellular communications available. Communications need to be maintained during the time the devices are mobile whether they are being carried by a person walking at a slow speed or by a vehicle travelling at a higher speed.

Ad hoc networks still have the problems associated with any wireless network mentioned previously as well as problems associated with the characteristics due to the ad hoc nature. How can a network be quickly established? How are messages forwarded without the use of routers or switches when a destination is not within the transmission range of the sender? Such a network is called a *multihop* network and uses intermediate nodes as temporary routers between a source and a destination. How can the network be easily maintained as users are added or deleted from the network?

An ad hoc network in which the communicating nodes can be mobile is referred to as a Mobile Ad Hoc Network (MANET). The addition of mobility to an ad hoc network adds even more difficulties to the communications problems. Routes are no longer static. Signal strength varies. The network may become disconnected. A link that exists at one moment

may not exist at the next moment since the link only exists when a node can receive a sufficiently strong recognizable signal from its neighbor. The strength of the received signal depends on the power of the transmitted signal, the antenna gains of both the sender and receiver, the distance between the two nodes, the obstacles between them, and the number of different paths the signals travel due to reflection [54]. As a consequence each node in a multihop mobile ad hoc network must continuously monitor the radio signals it receives in order to determine a list of one-hop neighbors composing a localized view of the network topology to use with routing protocols. Can the problem get worse?

Communications can be categorized based on the number of senders and receivers. One-to-one communications in which a single user communicates with another single user is referred to as *unicast* communications. One-to-all communications, in which a single user sends a message to everyone in the network, is referred to as *broadcast* communications. The routing protocols that have been developed to support both unicast and broadcast communications in wired networks have evolved into standardized protocols. A set of routing protocols developed for unicasting and broadcasting in MANETs is in the process of becoming standardized. In both cases the algorithms used to find the paths used in the routing of messages are efficient. The remaining categories of communications are one-to-many and many-to-many, both of which fall under the heading of *multicast* communications. Let us examine a sequence of progressively more efficient techniques to perform multicasting in which a message is distributed from a single user to a group of receivers in a wireless network. Suppose the solid lines in Figure 1 represent a subset of the available links in a communications network that have been selected to serve as a distribution tree from the sender, node A, to each of the receivers, nodes E, F, G, and H. The dotted lines indicate links that exist, but are not part of the distribution tree. Notice that nodes B, C, and D have been selected to act as routers even though they are not in the set of receivers.

The first technique is one of brute force. Node A will send the message to E over the path A, B, C, E and will then send a second copy of the message to F over the path A, B, C, F. Node A repeats this process for each of the receivers. The link from A to B will carry four copies of the message, the link from B to C will carry two copies, as will the link from B to D, and each of the last hop links will carry one copy of the message. Summing over all of the

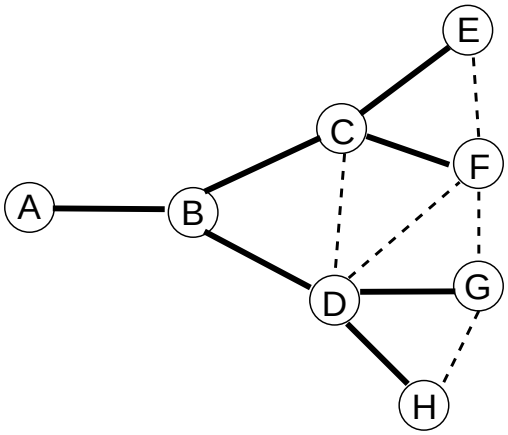


Figure 1: A Distribution Tree

links in the distribution tree a total of 12 messages has been transmitted over the individual links.

The second technique assumes the intermediate nodes can copy an incoming message and forward it over the links leading to the receivers. In this case A will send one copy of the message to B. B will then send a copy of the message down the link to C as well as on the link to D. The latter two nodes will duplicate the message and send it to the final receivers. In this scenario, the total number of messages over the individual links is 7.

The third technique is based on the assumption that multiple nodes can receive a message as long as each is in the transmission range of the sender. In this case A sends one copy of the message to all of its neighbors as a result of one transmission. B receives the message and forwards it to all of its neighbors. C and D then forward the message via one transmission each to the final recipients. In this final example, the total number of messages transmitted is 4.

Thus we see that given a distribution tree, the final multicasting technique can result in a significant reduction of the bandwidth. In the examples one particular distribution tree was considered, but other possible trees could have been chosen. For example, the tree shown by the solid lines in Figure 2 is an alternate distribution tree with the same number of links as in Figure 1.

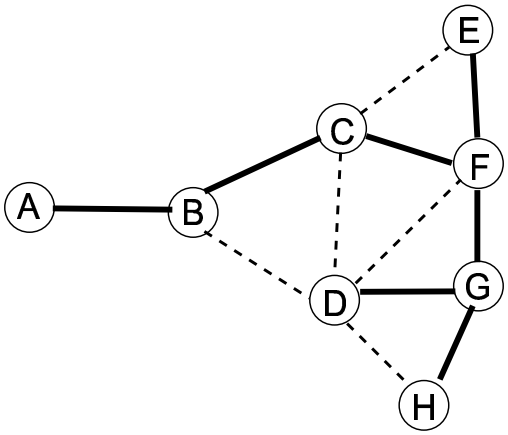


Figure 2: An Alternate Distribution Tree

Finding an optimal distribution tree for multicasting purposes is difficult due to the combinatorial explosion of the number of ways to use intermediate nodes as relay nodes to forward the data to the intended set of recipients. In fact, the problem of finding an optimal multicast tree has been shown to be NP-complete [26].

Now consider the challenges of *multicasting in a mobile ad hoc network to a dynamically changing group of receivers*. All of the problems discussed above are present - complexities stemming from the use of wireless devices transmitting through the air, mobile nodes, the lack of an infrastructure and multicast communications. In addition we now face additional complexity resulting from possible changes in group membership. Can all of these problems be overcome to provide a reliable and efficient multicasting service to users in a MANET? This thesis addresses some of these very problems and proposes a protocol to provide such a service.

1.2 PROBLEM STATEMENT, CHALLENGES AND RESEARCH GOALS

1.2.1 The Multicasting Problem

The goal of this research is to develop a multicast protocol to deliver the multicast data with a high degree of reliability from a single sender to each of the receivers of a dynamic multicast group in a mobile ad hoc network. The protocol should have proactive as well as reactive components to increase the applicability to a variety of circumstances. The proactive component should set up the multicasting in anticipation of future events and the reactive component should deal with events that have not been foreseen. A secondary goal is to perform the multicasting efficiently, using network resources, especially bandwidth, wisely. Let us define the following:

$N(t)$**:** the set of nodes in the network at time t.

$E(t)$**:** a set of directed edges representing the known radio links between nodes in $N(t)$ at time t.

$G(t) = (N(t), E(t))$**:** a model of a time-varying network at time t as a directed graph

$s(gid) \in N(t)$**:** the source node for group gid.

$R(gid, t) \subseteq N(t)$**:** the set of receiver nodes of the multicast group with group address gid at time t.

The problem of multicasting in a mobile ad hoc network from a sender to a dynamic group of receivers can now be stated more formally.

Given: a time varying network modelled as a graph $G(t) = (N(t), E(t))$, and a multicast group gid having a sender, $s(gid)$, and a the set of receivers, $R(gid, t)$, which varies over time.

Objective: to develop a multicasting protocol to use in routing data packets from the sender to the receivers in a dynamic multicast group gid which emphasizes reliability and efficiency. In order to achieve the objective we need to find

- a dynamic sequence of sets of multicast trees, $TS_1, TS_2, \ldots$ Each TS_i is active from time t_i until the number of intact trees in TS_i falls below a threshold.

- the individual multicast trees, $T_{i,1}, T_{i,2}, \ldots T_{i,n} \in TS_i$ rooted at the sender connecting the current set of receivers over which will be sent data packets.

1.2.2 Challenges

Some of the challenges facing multicasting in a MANET mentioned in the previous paragraphs result from characteristics of the environment whereas other challenges are inherent to the complexity of the problem itself. The challenges resulting from the environment come from three sources: the medium, the devices, and the network organization. The multicasting problem itself is the source of the fourth challenge.

- Network-centric Challenges: Characteristics of an ad hoc network contribute to the challenge of multicasting in such an environment. The lack of infrastructure and the lack of dedicated routers require nodes to assume router duties and to cooperate with each other.

- Device-centric Challenges: Limited transmission range, limited memory, limited storage capabilities, and limited power due to the use of batteries are characteristics of the devices acting as nodes in an ad hoc network which contribute to the challenges of multicasting in a MANET.

- Medium-centric Challenges: Use of radio waves as the transmission medium also adds to the problems encountered in multicasting in a MANET. The limited bandwidth, the limited range of available frequencies due to federal regulations and interference and packet collision caused by multiple signals are aspects that must be considered during the development of a multicasting protocol in a mobile ad hoc network.

- Challenges due to the Multicasting Problem: The difficulty of the problem is inherent to the nature of multicasting in a MANET. It is well known that the problem to find an optimal multicast tree is NP-complete. This is further compounded by the need to create and manage a group of nodes as a multicast group in which the nodes may join and leave the group over time.

1.2.3 Research Questions and Goal

Techniques established for routing in fixed wired networks are not adequate for use in mobile ad hoc networks in unicast or multicast communications since they do not deal with the problems inherent to MANETs. For example, a change in the topology in a wire-based network is assumed to be a rare event which may require the intervention of a network administrator to update the affected routers whereas a change in the topology of a mobile ad hoc network is a common event and needs to be handled by cooperating nodes in order to be effective. Also consider the difference in reliability in the two different environments. The medium in wired networks is much more consistently reliable than in MANETs so the protocols developed for the wired networks do not adequately deal with the limited bandwidth and interference problems. Is multicasting possible in the mobile ad hoc environment? In order to achieve the goal of delivering data packets with a high probability to members of a multicast group in a MANET, heuristics must be developed that address the following set of questions:

- How can we make end-to-end delivery of data in multicast group communications highly reliable?
- How can we adapt to the variability in bandwidth limitations when sending packets to a multicast group?
- How can we reduce the impact due to mobility to the overall performance of the multicast protocol?

In this thesis we propose the ***Ro****bust* ***M****ulticast* ***R****outing protocol* (RoMR) as a solution to the problem of multicasting in a mobile ad hoc network with a dynamic group of receivers. How does the proposed protocol address the issues of reliability and efficiency with proactive and reactive components? RoMR includes a component which assign weights to the links in a network to reflect the probability that the link will be available in the next time frame given that it is available in the current time frame. It then uses these weights during the construction of the multicast trees so that the more reliable links are chosen over less reliable links to be included in the multicast trees. Another method RoMR uses to increase the chance of a group member receiving the sender's packet is through the use of multiple

trees. Data packets are sent down the links in all of the trees in a tree set. If a member node receives duplicate packets, the extra packets are discarded. In order to reduce the overhead associated with the redundancy to support increased reliability, RoMR addresses efficiency in two ways. First, instead of recreating the set of trees as soon as one of the trees becomes nonfunctional due to disconnectedness, RoMR waits until the number of intact trees reaches a threshold before recalculating and redistributing the trees, thus reducing the number of tree set packets that must be distributed. Second, RoMR uses a forward-error correction code to encode k packets from the sender as n packets where n is the number of multicast trees. The i^{th} packet of the encoded group of k packets will be sent down the i^{th} multicast tree. The receiver only needs to receive *any* k of the packets in order to determine the contents of the original k packets. Using such an encoding technique can significantly reduce the number of additional packets injected into the network when the network is fairly stable. Since k is computed dynamically based on the current network conditions, it is a flexible way to control the overhead.

RoMR's proactive component computes the multicast trees based on the weights of the links reflecting predicted conditions in the near future. It computes multiple trees in a tree set in anticipation that some of the trees will become disconnected and will not deliver the packets to one or more of the destinations. The use of an underlying link state unicast protocol also adds to the proactive nature of the protocol since the topology of the network is readily available for use in calculations. In addition to the proactive component, RoMR has a reactive component. When network conditions change resulting in the number of connected multicast trees falling below a threshold level, RoMR reacts to the situation and creates and distributes a new set of trees. RoMR also reacts to changes in group membership as nodes join and leave the multicast group.

Thus far we have introduced the multicasting problem in mobile ad hoc networks and have seen that solutions associated with wired networks are not adequate to the environment under consideration. We have considered important questions that must be answered in order to develop an adequate multicasting protocol in MANETs. The rest of the chapters will address these issues and examine the proposed solution. In Chapter 2 background material on routing protocols is presented as well as a discussion of several multicast protocols that

have been proposed for use in wireless ad-hoc networks. In addition to the protocols is a section pertaining to multicast trees, concentrating on Steiner tree heuristics. Chapter 3 gives an overview of the Robust Multicast Routing protocol (RoMR), our solution to the multicasting problem in MANETs. Chapter 4 derives the formulas used to compute the weights associated with the links in a network and Chapter 5 presents the algorithms of RoMR. The parameters, results and analysis of the data generated by the simulation are given in Chapter 6. Finally, the conclusion and future work are discussed in Chapter 7.

2.0 RELATED WORK

In order to better understand the issues of multicasting in mobile ad-hoc networks and the ideas behind the proposed solution to the multicasting problem, this chapter will review the classifications of routing protocols, unicasting in mobile ad hoc networks and multicasting in both wired and wireless networks. Unicast protocols are included in the review since some of the multicasting protocols depend on a particular type of underlying unicast protocol and they provide insight into the routing problem in general. Some unicast protocols are based on calculations of link availability which is briefly discussed. The sections dealing with multicasting include discussions of flooding, multicast trees based on Steiner trees, and a variety of multicast protocols.

2.1 CLASSIFICATIONS OF ROUTING PROTOCOLS

2.1.1 Table-driven vs On-demand Protocols

Routing protocols are classified as either *table-driven* or *on-demand* [47, 15]. The table-driven protocols, also referred to as *proactive* protocols, maintain current information on all reachable destinations in anticipation of the use of the information. The nodes are required to maintain one or more tables of routing information which are propagated throughout the network as changes in the topology occur. The protocols differ in the necessary tables related to routing and how the tables are exchanged. The on-demand methods, also known as *reactive* protocols, wait to determine the route packets will travel at the time the communication begins. When a route is needed, the source node initiates a route discovery process to the destination. Once established the route must be maintained until it is no longer needed or

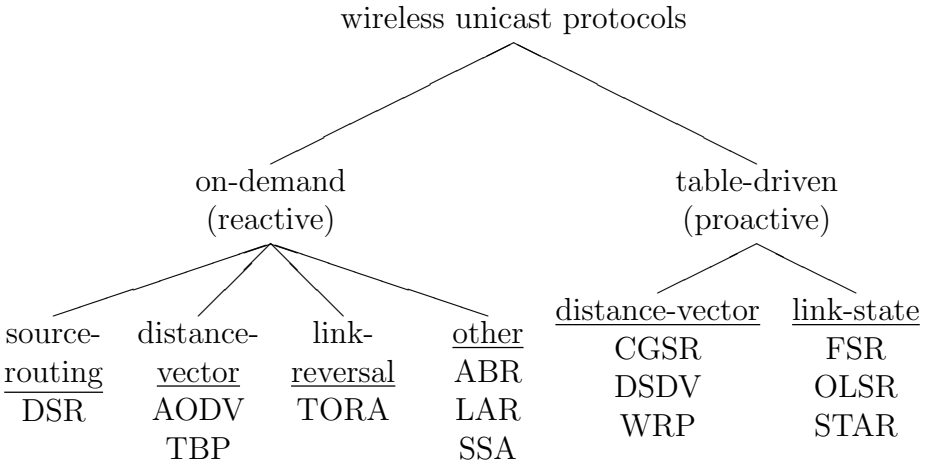
the destination node becomes unaccessible. Proactive and reactive protocols each have advantages and disadvantages. In a proactive protocol the information to determine the routes is immediately available so no additional time is needed to discover the hops in a route, thus the delay of the first packet does not include route discovery time. This is a significant advantage when many routes are needed within a short period of time. A disadvantage of a proactive method is that it requires periodic updating of the routing tables, so if only a few routes need to be determined then the overhead of table exchanges and maintenance may be substantial. The advantages to an on-demand protocol are related to the disadvantages seen in the proactive protocols. If only a few routes need to be determined, using an on-demand method would incur less overhead to discover the hops than the overhead associated with the proactive protocol's exchange of topology information. The disadvantage of the reactive method is that it necessitates a longer delay in getting the packets to the destination since it must first discover the route.

2.1.2 Source Routing, Link-State, and Distance Vector Protocols

A routing protocol can be classified as either a source routing protocol, a link-state protocol or a distance vector protocol based on the type of information the intermediate nodes keep in order to determine the next hop when forwarding packets. A *source routing protocol* requires no information to be kept in the nodes since the source node determines the route between source and destination and lists each hop of the path in the packet header. This is advantageous if the intermediate nodes have limited memory resources, but can be considered a disadvantage since it can add substantially to a packet's size. A *link-state protocol*, on the other hand, requires each node in the network to maintain a database describing either the entire network or a localized subset of the entire network. In order to maintain the database, messages are regularly exchanged between neighbors to determine the local neighborhood and then tables describing the local neighborhood are distributed to other nodes. Thus the advantage is knowledge of the topology of the network so that routing decisions can be made easily, while the disadvantage is the amount of information exchanged between nodes may be significant when the routing tables are large. In order to address the main

disadvantage of the generalized link-state protocol, many implementations only exchange a subset of the routing tables, either exchanging information that applies to a localized area or exchanging only updates that reflect changes in the topology since the previous update. A *distance-vector protocol* keeps track of only the distances to reachable nodes in order to determine the next hop. The advantage is that only next hop information is passed between nodes instead of the entire routing table, but the disadvantage is that only one route is known from a node to each of the accessible nodes instead of the entire or localized topology. The proposed multicasting protocol developed in this project assumes the use of an underlying link-state unicast protocol due to the availability of knowledge of the topology so that multiple multicasting trees can be constructed.

2.2 UNICAST PROTOCOLS



ABR	Associativity-Based Routing	OLSR	Optimized Link State Routing
AODV	Ad-hoc On-demand Distance Vector	SSA	Signal Stability-based Adaptive Routing
CGSR	Clusterhead Gateway Switch Routing	STAR	Source-Tree Adaptive Routing
DSDV	Destination-Sequenced Distance-Vector	TBP	Ticket Based Probing
DSR	Dynamic Source Routing	TORA	Temporally Ordered Routing Algorithm
FSR	Fisheye State Routing	WRP	Wireless Routing Protocol
LAR	Location-Aided Routing		

Figure 3: Unicast Routing Protocols

Unicast protocols are the techniques used to send data from a single source node to a single destination node. A variety of unicast routing protocols have been proposed for ad-hoc networks. (See Figure 3.) The protocols can be viewed as either on-demand or table-driven. The on-demand protocols are further divided into categories based on the techniques used to determine the routes. An on-demand protocol in which the routing is initiated by the source is Dynamic Source Routing (DSR) [25, 24, 47]. An on-demand protocol that is based on the use of a distance vector is Ad Hoc On-Demand Distance Vector Routing (AODV) [38, 40, 47]. Temporally Ordered Routing Algorithm (TORA) [37, 47] uses a technique in which link reversals determine the route and Associativity-Based Routing (ABR) [52] , Signal Stability-based Adaptive Routing (SSA) [13] and Location-Aided Routing (LAR) [27] are examples that use information such as the past availability of the links, a signal's strength over a link or the location of nodes on which to base the routes. Unicast protocols that are table-driven protocols can be further divided into those that use distance-vector tables or those that use link-state information. The distance vector protocols include Destination-Sequenced Distance-Vector Routing (DSDV) [39, 47], Clustered Gateway Switch Routing (CGSR) [7, 47], and the Wireless Routing Protocol (WRP) [35, 47]. Table-driven protocols based on link-state include Source-Tree Adaptive Routing (STAR) [15], Optimized Link State Routing (OLSR) [8], and Fisheye State Routing (FSR) [23]. Recently, Core-Extraction Distributed Ad hoc Routing (CEDAR) [50], Ticket-Based Probing (TBP) [6] , and an extension to AODV have been proposed to provide Quality-of-Service (QoS) guarantees in unicast routing.

In our proposed solution to the multicasting problem discussed in Chapter 3, we assume the availability of an underlying link-state unicast protocol by which to gather information about the topology of the network. STAR, Fisheye State Routing and OLSR are possible candidates due to their efficiencies and are described below. The reader is invited to refer to the indicated sources for information about the other protocols mentioned.

2.2.1 Source-Tree Adaptive Routing

Source-Tree Adaptive Routing (STAR) [15] is a table-driven, link state unicast routing method that can work with any current clustering mechanism. In STAR each node maintains

a source-based tree from itself to each known destination and shares this information with its neighbors. Initially a node only knows its adjacent links and sends this information to neighbors. When a node has knowledge of its own adjacent links as well as those reported by its neighbors then a partial topology of the network is formed which is used to determine preferred paths to known destinations. A node sends updates about its source tree to neighbors when one of three cases occur: a) when a node can no longer reach one or more previously reachable destinations, b) when a node detects a new destination, or c) when potential long term loops are possible. When an update needs to be sent only the links that change are sent, not the entire tree. Sequence numbers are used to validate the update being received. STAR is a link-state protocol that does not rely on periodic updates. Simulation showed STAR generated less control traffic than DSR while delivering the same number of data packets in mobile networks with a relatively low number of senders. When failed nodes were included or when the network had many destinations receiving data, STAR performed significantly better than DSR.

2.2.2 Fisheye State Routing

Fisheye State Routing (FSR) [23] is a link state algorithm that has been modified for use in ad-hoc networks. Since much of the overhead of link state algorithms is due to the propagation of the link state tables, FSR sends out information regarding local nodes more frequently than it sends out information regarding nodes that are farther away. The idea is that a packet will start out in the correct general direction of a distant destination and as it gets closer to the destination, nodes will have more up-to-date information and will be able to deliver it to the destination.

2.2.3 Optimized Link State Routing

Optimized Link State Routing (OLSR) [8] is another modification of a pure link state algorithm. OLSR reduces the amount of state that is periodically exchanged through the use of multipoint relays (MPRs). MPRs are selected independently by each node so that each node has an MPR within one hop and the set of MPRs for the node can reach all of the nodes

within two hops of the node. A proposed heuristic to determine the MPR set for a given node n is given below.

1. The MPR set is initialized to the set, N, of known nodes that are willing to participate in routing.
2. Calculate the degree of each one hop neighbor node.
3. Let $N2$ be the set of all two-hop neighbors.
4. Add to the MPR set those nodes which are the only nodes to provide reachability to a node in $N2$.
5. While there exists nodes in $N2$ which are not yet covered by at least one node in the MPR set:
   a. For each node in N, calculate the number of nodes in $N2$ which are not yet covered by at least one node in the MPR set and which are reachable through this one hop neighbor.
   b. Select as MPR the node which provides reachability to the maximum number of nodes in $N2$. If more than one node is a candidate, select the node whose degree is greater.

In Figure 4 nodes E and F have been selected as MPRs.

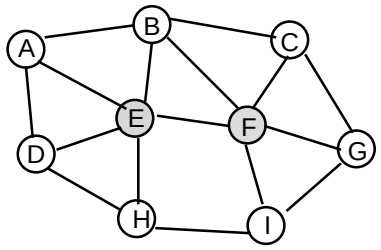


Figure 4: MultiPoint Relays in OLSR

When a packet travels from a source to a receiver it will be sent over links between MPRs whenever possible. In the pure link state protocol each node floods the network with information about *all* of its incident links. In OLSR the amount of flooding is reduced since

only the MPRs forward the link information. Furthermore, the information that is forwarded only includes links between MPRs. As a consequence, the nodes have knowledge of a partial topology graph and thereby routes only use internal nodes that are designated as MPRs.

Any of the three described unicast protocols could be used as the underlying unicast in the proposed multicasting protocol since all are proactive link-state unicast protocols providing the information needed in order to construct multiple multicast trees. While the proposed multicasting protocol is not tightly coupled to a particular unicast protocol, Fisheye and OLSR were both used during the development stage of the project with OLSR being implemented in the final version.

2.3 LINK AVAILABILITY - PROTOCOLS AND MODELS

While most unicast routing algorithms base the formation of routes on the number of hops between source and destination some protocols associate numerical values with links to aid in a prediction of the availability of the link in the near future. Associativity Based Routing (ABR) and Location Aided Routing (LAR) are two unicast protocols which do not simply calculate paths based on shortest path distances, but incorporate aspects of the past behavior of the link.

Associativity Based Routing (ABR) [52] is an on-demand unicast routing protocol for wireless networks that builds routes based on the length of time the link has been active. The protocol selects a more stable route as opposed to a less stable shortest path. ABR assumes the longer a link has existed the more likely it will continue to exist. It may be the case that the longer a link survives, the less likely it is to continue to exist because the transmitting node may be reaching the limits of the receiver's range due to mobility.

Location Aided Routing (LAR) [27] is an on-demand unicast routing protocol that incorporates both the past location of a node and its mobility characteristics in order to predict the location of the node in the next time period. After the prediction is made, the predicted area is flooded with search messages in order to find the destination. Complications arise due to the fact that some of the intermediate nodes that must be used in order to reach the target may not be in the area that receives the search messages in which case a path that

exists might not be discovered. LAR also assumes the knowledge of location which in many cases is unobtainable.

In addition to the protocols mentioned, a model of link availability [33] has been created based on the known mobility parameters of nodes. The parameters used in the calculations for link availability include the average length of time called a mobility epoch during which a node has constant speed and direction, the average speed during the mobility epoch, and the variance in speed of each node during a mobility epoch. While this information may be available in devices integrated into automobiles, this may not be available for simpler handheld devices.

A model of link availability based on the strength of received signals is discussed in [49]. Although the claim is made that the method is appropriate to mobile ad hoc networks the calculations assume that distances can be calculated based on receiving radio signals with known transmission power levels from several base stations which are generally not available in ad-hoc networks. Furthermore, the calculations used to determine the availability of a link rely on an assumption which may not be valid. In the model three measurements of signal strength are gathered from a neighboring node at three consecutive times in order to estimate three distances to the neighboring node while the node is mobile. The method assumes that both the neighboring node and the node performing the analysis are moving in straight lines at constant speeds. From this information the maximum amount of time that the link will continue to survive is computed if the nodes were to continue on their current paths. The likelihood that the link will survive after factoring in change of direction produces a metric for link availability. If the three successive readings did not come from a node travelling in a straight line at a constant speed, it is possible for the calculations to attempt to take the square root of a negative number in which case no further calculations and predictions can be made. Such a scenario was not developed as part of the model.

Both of these models of link availability rely on the knowledge of both speed and direction of the mobile nodes. Another aspect of these and many models is the reliance on the assumption of a specific statistical distribution of the mobility patterns of the nodes. As we will see in Section 4.1, our proposed multicast protocol, RoMR, decouples itself from the dependency of a particular distribution and focuses on the most likely area to be covered by

a node.

2.4 MULTICAST TREES

In order to successfully deliver packets to the group of receivers in a multicast group, routes between the sender and the recipients need to be established. This set of paths forms a *multicast tree*.

Definition 1 *Given a set of nodes N and edges E forming a graph $G = (N, E)$, and a subset of the nodes $Z \subseteq N$, a* multicast tree *T is a tree subgraph of graph G such that T spans the nodes in Z.*

2.4.1 Shortest Path Trees and Steiner Trees

Efficiency is a major concern of multicasting so before proceeding with the descriptions of the multicast protocols, we will first examine various approaches to creating efficient multicast trees. A naive approach to multicasting is for each source node to send a separate message to each receiver in the group of receivers. This can introduce significant overhead traffic in the network and a delay of packet delivery as seen in the example in Section 1.1. A better approach is to construct a tree structure as the basis of the communication distribution system with the source node as the root of the tree and receivers attached to the source using intermediate nodes as duplicators and relay points. Only one copy of each data packet is sent along any branch of the tree. Designated intermediate nodes duplicate the message and send a copy to each of its children in the tree. The message is eventually delivered to each of the receivers in the multicast group. The tree's structure depends on the objective of the particular multicasting scheme. In one approach the cost of *each* of the paths from sender to receiver is minimized producing a shortest-path multicast tree where the cost is the number of hops. Another approach minimizes the *total* cost of all of the links in the tree, thus reducing the use of network resources. The latter approach produces a *Steiner tree*. Both approaches assume the topology of the network is known. Figure 5(a) shows an example of a shortest-path tree and Figure 5(b) shows a Steiner tree for the same set of

sender and receivers. In both cases we assume that all links have equal costs and that the costs associated with the links are additive.

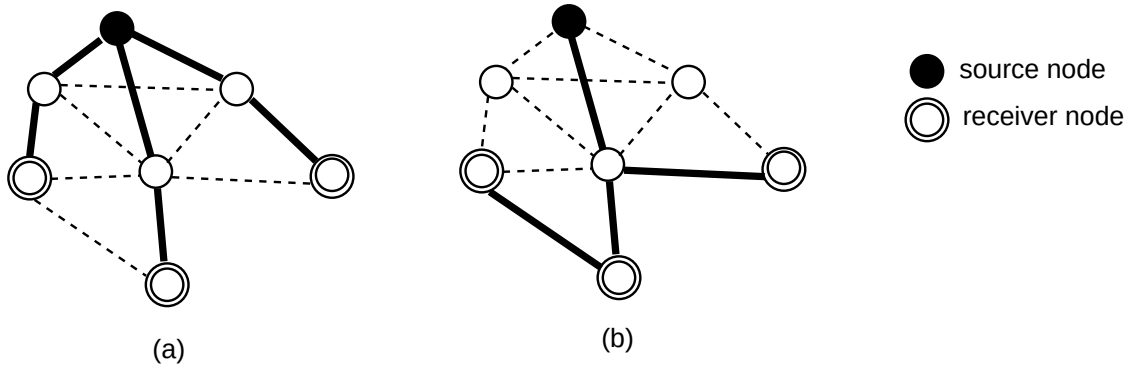


Figure 5: Shortest Path Tree vs. Steiner Tree

A minimal cost spanning tree of the network serves as a distribution tree for *broadcasting* packets throughout the network. A minimal cost spanning tree that spans all of the n nodes in a network using a subset of the E edges can be computed in $O(E \cdot log\ n)$ time. Thus we see that broadcast communications can be efficiently implemented while optimizing total cost. Can we use the minimal spanning tree in the formation of a multicast tree? Constructing a minimal cost tree and removing branches to nodes that are not members of the multicast group does not necessarily produce an optimal multicast tree, though. For example, in Figure 6(a) a minimal cost spanning tree has been computed and is displayed by the thick lines. In Figure 6(b) the links that are not part of a path to a destination node have been removed. The remaining path from source to destination uses multiple links whereas the optimal tree that joins the source to the destination is simply the one hop link between the two nodes.

Multicasting methods may use Steiner trees as the underlying distribution network as mentioned previously. Steiner tree algorithms optimize the total cost to connect a subset of the nodes in a graph by utilizing additional nodes as relay nodes. These relay nodes are referred to as Steiner nodes.

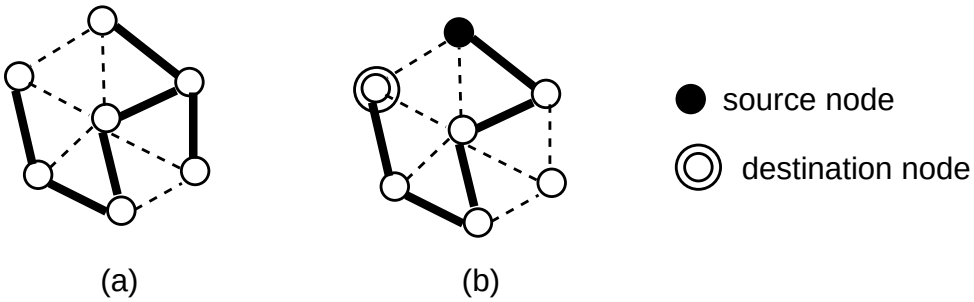


Figure 6: Pruned Minimal Cost Spanning Tree

Definition 2 *Given a graph $G = (N, E)$, a subset of the nodes $Z \subseteq N$, and a cost function $c : E \mapsto \mathbb{R}$ on the edges of G then a* Steiner Tree *is a subgraph of G with minimal cost connecting the vertices of Z [21].*

The problem of constructing a Steiner tree to optimize cost is more complex than finding shortest path trees or minimal cost spanning trees since only a *subset* of the nodes are to be connected. Since the problem is NP-complete the heuristics that create approximations of the Steiner tree in polynomial time are very important. Steiner tree problems have been the subject of much mathematical research with the goal being to find polynomial time algorithms that produce trees with costs that approach the cost of the optimal Steiner tree. The algorithms have reduced the cost of the trees produced from twice the optimal tree in 1980 [51] to 1.55 times the optimal tree in 2000 [45]. In order to use Steiner tree algorithms in realtime calculations of multicast trees, the runtime of the algorithm is extremely important. Unfortunately, in many cases the runtime increases as the Steiner ratio decreases to the point that the algorithms are not suitable for use in forming multicast trees in real-time computer networks due to the amount of preprocessing of topology information that is required. Refer to [55] for a survey of Steiner tree algorithms.

Most of the Steiner tree algorithms assume the graph has undirected edges which would correspond to networks having bidirectional links. A version of the Steiner tree problem exists for directed graphs corresponding to a network with unidirectional links. The directed version is also NP-complete. The asymmetry of the links increases the complexity of finding near-

optimal solutions. In fact, the existence of approximation algorithms that have a constant Steiner ratio for arbitrary directed graphs is highly unlikely [42].

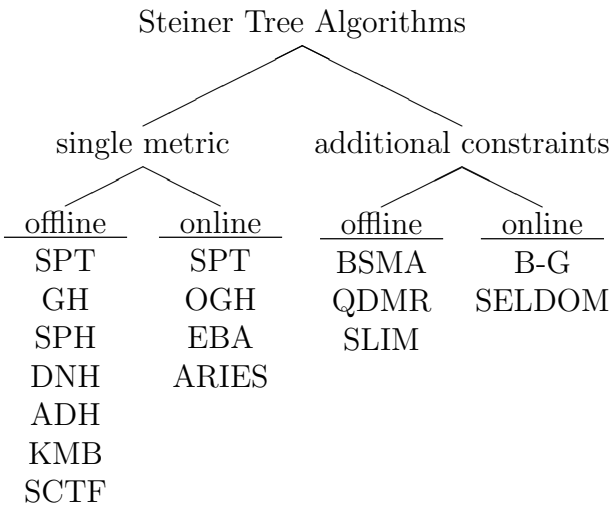
The Steiner tree problem described above assumes that the group of nodes Z is known at the time of tree creation and does not vary. Another version of this problem is known as the *dynamic* Steiner tree problem. A sequence of join and leave requests $\{r_0, r_1, ...r_k\}$ represent the events of nodes being added to or removed from the multicast group. Let Z_i be the multicast group after the i^{th} request.

Definition 3 *The* dynamic Steiner tree problem*: Given an undirected graph $G = (N, E)$, a sequence of requests R, a cost function $c : E \mapsto \mathbb{R}$ on the edges of G at the time of the ith request, find a sequence of trees T_i each of which spans all the nodes in Z_i with optimal total cost [22].*

Any Steiner tree heuristic can be applied to the dynamic Steiner tree problem if the tree is recalculated after each change in group membership. Such an approach may be undesirable if the requests arrive faster than the new trees can be distributed to the nodes in the multicast tree or if the cost of distributing new trees is high. An alternative is to use a nonrearrangeable heuristic. Nonrearrangeable methods change the existing multicast tree by only appending links in order to process an add request and by only deleting links in response to a remove request.

2.4.2 Steiner Tree Algorithms

In this section we examine various Steiner tree algorithms that have been proposed to create a multicast distribution tree in computer networks. (See Figure 7.) As stated previously, Steiner trees attempt to optimize the aggregation of some metric associated with the tree. Usually the total cost of the tree is minimized. The algorithms we examine in this section can be categorized as either offline or online. *Offline* algorithms are those in which the set of multicast nodes is known at the time of the creation of the tree and do not change. *Online* algorithms add and remove multicast nodes from the tree as the nodes join and depart from the group during the group's session. Offline algorithms address the Steiner tree problem while online algorithms apply to the dynamic Steiner tree problem.



ADH	Average Distance Heuristic	OGH	Online Greedy Heuristic
ARIES	A Rearrangeable Inexpensive Edge-based Steiner (Algorithm)	QDMR	QoS Dependent Multicasting Routing
		SCTF	Selective Closest Terminal First
BSMA	Bounded Shortest Multicast Algorithm	SELDOM	Simple and Efficient, Low-cost, Delay-bounded Online Multicasting
DNH	Distance Network Heuristic	SLIM	delay-conStrained, Low-cost, Inexpensive Multicasting
EBA	Edge Bounded Algorithm		
GH	Greedy Heuristic	SPH	Shortest Path Heuristic
KMB	Kou, Markowsky, and Berman	SPT	Shortest Path Tree

Figure 7: Steiner Tree Algorithms

2.4.2.1 Offline Single Metric Heuristics Examples of offline heuristics that find approximations to Steiner trees based on a single metric include the Greedy Heuristic (GH), the Shortest Path Heuristic (SPH), the Distance Network Heuristic (DNH), the Average Distance Heuristic (ADH), the Selective Closest Terminal First algorithm (SCTF), and an algorithm proposed by Kou, Markowsky, and Berman (KMB). The Greedy Heuristic [53] starts with a randomly selected multicast node as a single node tree and adds the multicast nodes one at a time in random order using the shortest path based on the metric being used from any node in the current tree to the node that is being added. The Shortest Path Heuristic (SPH) is similar to the Greedy Heuristic, but adds the multicast nodes in ascending order of distance to any node in the current tree. The Distance Network Heuristic (DNH) [28] is similar to SPH, but adds the multicast nodes using the shortest path from any one

of the multicast group members (as opposed to any node) in the current tree to the node being added. The Average Distance Heuristic (ADH) [43] iteratively finds the node that is most central to a set of subtrees and joins the two closest trees to that central node using the shortest paths from the central node to each of the two trees. Initially single node trees are created from the set of multicast nodes. Any of these could be used in our solution to the single-source multicast problem.

The Selective Closest Terminal First algorithm (SCTF) [42] is an algorithm designed for single-metric *directed* graphs that is very similar to SPH. A parameter k specifies which of the nodes in the current multicast tree to consider when finding the node closest to the tree. A low-value of k decreases the running time, but increases the total cost of the tree. Since the problem at hand specifically calls for the minimization of the tree, SCTF is not considered for use with RoMR.

Many of the algorithms that result in lower Steiner ratios start with the formation of a complete graph based on the network configuration. The KMB [48]algorithm is one such algorithm. It is composed of five steps:
Given a graph $G = (N, E)$,

1. Create a complete graph $G' = (Z, E')$ with the multicast nodes $Z \subseteq N$ such that for every pair of nodes u and v in Z, the edge (u, v) in G' has cost equal to the cost of the shortest path between u and v in G.
2. Find a minimum spanning tree T' of G'.
3. Expand each edge (u, v) of T' with the corresponding links in the shortest path between u and v in G, giving G''.
4. Find the minimum spanning tree T'' of G''.
5. Prune T'' so that all the leaves are in Z.

The algorithms that use an approach similar to KMB have not been used in multicasting protocols as of yet.

The Shortest Path Tree (SPT) [12] is created as the union of all the shortest paths from the source node to each of the receiver nodes. This minimizes the individual paths from sender to each receiver, but doesn't attempt to use common links to reduce the total cost of

the tree further. Dijkstra's shortest path algorithm [10] is an efficient single-source shortest-paths algorithm. Given one source node and n nodes in a network, the algorithm computes shortest paths from the source node to each of the nodes in the network in $O(n^2)$ time. Using a Fibonacci heap implementation of the priority queue in the algorithm can further reduce this to $O(n \cdot log\ n + E)$ where E is the number of edges in the network.

SPT can be used in both offline and online computations since the shortest path from sender s to receiver r does not depend on the existence of paths from s to the other receivers. Either cost or delay may be used as the metric of interest. Many of the current multicasting protocols implement some form of SPT, despite the fact that the union of shortest paths may use more network resources than other approximations to the optimal Steiner tree.

In this section we have examined offline heuristics designed for producing trees based on static groups. These may also be applied to dynamic groups if the tree is to be recalculated as a result of a change in group membership. In the next section we look at online heuristics to deal with dynamic groups.

2.4.2.2 Online Single Metric Heuristics. The set of *online* heuristics address the dynamic Steiner tree problem. Online heuristics based on a single metric include the Online Greedy Heuristic (OGH), A Rearrangeable Inexpensive Edge-based Steiner algorithm (ARIES), and the Edge-Bounded Algorithm (EBA). The Online Greedy Heuristic [53], also referred to as the Dynamic Greedy Algorithm (DGA) [22], is the dynamic version of GH using a nonrearrangeable heuristic. When a node joins the group the shortest path from the tree to the node is added to the tree. When a node leaves the group the node is marked as a non-multicast node. If the departing node is a leaf node then the branches of the path to it that are not included in other paths are deleted from the tree. If the requests are restricted to join requests then the ratio of the cost of the trees formed by OGH to the cost of the optimal Steiner trees is less than or equal to $\lceil \lg(n) \rceil$ where n is the number of leaf nodes that were added to the tree. If both join and remove requests are considered, the ratio is unbounded[22]. OGH is implemented in the some of the multicast protocols in section 2.5. ARIES [5] in particular is a multicast algorithm that modifies a region of the multicast tree when the number of join and leave requests reaches a threshold.

The Edge-Bounded Algorithm (EBA) [22] is a rearrangeable heuristic which creates a complete graph G' from the original graph G as in the first step of KMB. The shortest path between nodes u and v in G is represented as a single edge between u and v in G'. A minimum spanning tree T' of G' is computed and branches leading to non-source and non-destination nodes are pruned from the tree. For each join request, EBA finds the least-cost path from the new node v to the closest node u in the minimum spanning tree. If the cost of the maximum-cost edge in the path from v to *any* node u in T' is not more than α times the cost of edge (u, v) in G' the path is added to T' . If the path under consideration is more than the bound then u and v are connected via the least-cost path. For each leave request, the degree of the node is examined. If the degree is one then the node and branch between it and the attachment point to the tree are removed from the tree. If the degree of the leaving node is three or more then the node is marked for deletion. If the node has degree two, then both of the edges incident to the node are removed from the tree creating two unconnected subtrees. The two subtrees are connected with the least-cost path between the two trees.

One of the components in RoMR creates and maintains multicast trees. Any of the offline or online single metric heuristics discussed could be used in the construction of a single multicast tree. An online version of SPH, similar to OGH, is implemented in the current version of RoMR.

2.4.2.3 Heuristics With Constraints In all Steiner trees the total cost is optimized, but in some cases the final tree may be subject to additional constraints. For example, each path from sender to receiver may be required to meet a certain end-to-end delay or each path may have a minimum bandwidth requirement associated with it. The Bounded Shortest Multicast Algorithm (BSMA) [57], the QoS Dependent Multicast Routing (QDMR) algorithm [16] , and the delay-conStrained, Low-cost, Inexpensive Multicasting (SLIM) heuristic [3] are three examples of offline algorithms that create delay-bounded Steiner trees. The maximum delay value may be common to the group or each receiver may specify a delay bound. B-G[20] and SELDOM [3] are examples of online constrained Steiner tree algorithms that create multicast trees. Since the project does not incorporate quality-of-service constraints these algorithms are not discussed here. The reader is invited to read the cited papers for

further information.

2.5 MULTICAST PROTOCOLS

In this section we consider a variety of multicast routing protocols designed for use in wired and wireless networks. The protocols are based on a range of methods from using flooding techniques to disseminating the packets based on an optimal tree structure. One of two types of trees are generally implemented - source-based shortest path trees (see SPT in Section 2.4.2.1) or shared trees. A new approach using a mesh falls between the two extremes of flooding and optimal trees. We will discuss the generalized techniques and then cover the multicasting protocols in both wired and wireless environments.

2.5.1 General techniques

The original techniques for multicasting were developed for wired networks and thus refer to routers, edge-routers, and subnets. These same ideas can be applied to ad-hoc networks where nodes take on the responsibilities of routers.

Some protocols use a general technique known as "broadcast-and-prune" in which data packets are initially broadcast to all routers in the network, thus they are termed "data-driven" [32]. If an edge router does not have any group members in its subnet after receiving the broadcast data, it will request to be pruned from the multicast tree. A router that receives pruning requests on all links other than the one towards the source sends its own pruning request upstream. A modification of broadcast-and-prune requires nodes to keep track of individual downstream links and to forward group data only over the links that lead to receivers. After a specified time (minutes to hours) the pruning is no longer in effect and the process of broadcasting and pruning is repeated. Broadcast-and-prune methods work best on dense networks since the default action is to forward a packet and to only quit forwarding when an explicit prune message is received on that link. The advantage of such a technique is that shortest path trees are formed from each source to each receiver. The disadvantages are the broadcasting consumes bandwidth in areas of the network that may

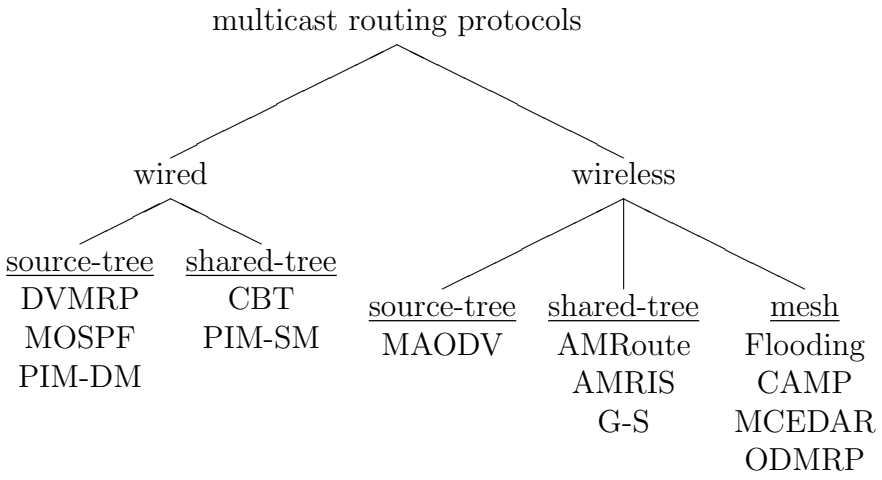
not have any receivers and the method requires that routers not on the multicast tree keep track of pruned branches emanating from them. Another disadvantage is that routers that are on the multicast tree need to keep entries in the routing tables for each source in the group so the approach does not scale well as more senders are added to the group.

Other protocols are based on the concept of shared trees which have a central node [32]. The sources send data to the central point which in turn send it to the receivers. Unlike the broadcast-and-prune approach receivers must explicitly join a shared tree prior to receiving data. The advantages are that non-tree nodes do not need to maintain any state information about the group and that member nodes only need to keep track of information pertaining to the entire group as opposed to each sender in the group. A disadvantage is that the central node is a single point of failure for the entire group. Another disadvantage is that the path from sender to a receiver may not be optimal.

When resources are plentiful or the time required to calculate more efficient trees is unacceptable, flooding is a technique that can be used to disperse messages to the receivers. The sender sends a copy of the message out on every available channel. Each node receives a copy and forwards it to its neighbors other than the source of the previous hop. A node caches identifying information regarding the packets received, thus recognizing duplicate packets and discards any repeat packets that it receives. Flooding is advocated in [19] as the preferred forwarding method in highly mobile fast-moving ad-hoc networks to increase the chance that a node receives a packet. Some multicast routing protocols are based on limited flooding.

2.5.2 Multicasting in Wired Networks

The generalized multicast routing techniques discussed above are used in specific protocols for IP-based wired networks. The main protocols are Distance-Vector Multicast Routing Protocol (DVMRP), Protocol-Independent Multicast-Dense Mode (PIM-DM), Multicast operation of Open-Shortest Path First (MOSPF), Core-Based Trees (CBT), and Protocol-Independent Multicast-Sparse Mode (PIM-SM). (See Figure 8.) The first three rely on source-based trees while the last two use shared trees. CBT is designed for use with group communication,



AMRIS	Ad Hoc Multicasting Routing protocol utilizing Increasing ID Numbers	MAODV	Multicast operation of AODV
AMRoute	Ad-hoc Multicasting Routing	MCEDAR	Multicast version of CEDAR
CAMP	Core-Assisted Mesh Protocol	MOSPF	Multicast version of OSPF
CBT	Core Based Trees	ODMRP	On-Demand Multicast Routing Protocol
DVMRP	Distance-Vector Multicast Routing Protocol	PIM-DM	Protocol-Independent Multicast – Dense Mode
G-S	Gupta and Srimani's reliable protocol	PIM-SM	Protocol-Independent Multicast – Sparse Mode

Figure 8: Multicast Routing Protocols

where nodes can act as both senders and receivers. Networking textbooks such [32] provide excellent coverage of these protocols.

2.5.3 Multicasting in Wireless Ad-Hoc Networks

Multicast algorithms designed for wired networks do not adequately deal with problems that arise due to a node's mobility or to the less reliable transmission medium. Even though the methods for multicasting over wired networks described in the previous section allow a node to join and leave a group, they assume that a node that remains in the tree will not move and cause the topology of the network to change. Although some of the wired protocols can deal with link or node failures as rare events, they would not be able to handle the increased frequency of failures found in a wireless ad-hoc network in an efficient manner.

In this section we will examine a number of multicast protocols that have been designed specifically to address the characteristics of a wireless ad-hoc network. Some are based on source-rooted trees; others on shared trees. MAODV is an example of the former and AMRoute, AMRIS, and G-S are examples of the latter. Several of the latest proposals including MCEDAR, CAMP, and ODMRP, are based on meshes to increase the protocols' robustness. (See Figure 8.)

2.5.3.1 MAODV Multicast operation of AODV (MAODV) [46] is not an extension of AODV, but is considered to be an integral part of AODV which can perform unicasting, multicasting and broadcasting.

The process of joining a multicast group is similar to discovering a route to another node when sending a unicast message. The node wishing to join the group sends a route request message with the join flag set to the group leader in a unicast message if the leader is known. If the node does not know of a route to the leader or does not know the identity of the leader the request is broadcast. All nodes that receive the request record which node sent the request and on which link. If the node receiving the request is already part of the tree, it sends a reply message to the requester via unicast; otherwise it rebroadcasts the request to its neighbors. After an attachment node to the tree has sent the reply, each node that

forwards the reply message towards the requester updates the message. By the time it is received by the requester, the reply message indicates the number of hops from the requester to the node on the multicast tree. Each node that relays the reply message also records the next hop towards the multicast tree. The requester will wait an amount of time, gathering one or more reply messages and will choose the route with the fewest hops to the tree. Once the requester's next hop to the tree has been selected the requester activates the branch to the tree by sending a unicast activate message to the next hop neighbor which is forwarded along the path to the attachment point on the tree. The nodes that receive the activation message become part of the multicast tree. Sequence numbers in messages allow nodes to ignore outdated information and thus to avoid the formation of loops.

If a node does not receive any reply, it becomes the group leader. A group leader is responsible for maintaining the group's sequence number and periodically broadcasting group hello messages throughout the network. The information disseminated includes the group address and the latest sequence number for each group for which the node is leader and is recorded by all nodes in the network.

When a leaf node leaves the multicast group the node sends a prune message to its parent node on the multicast tree which removes the link between the two nodes from its routing table for the multicast group. The prune message continues upstream until it arrives at a node that is either a member of the group or one that does not become a leaf node as a result of the pruning. The failure of a link on the multicast tree causes the downstream node to send a join request via an expanding ring search in order to find a node that is on the tree.

When the network becomes partitioned a new group leader is selected for the section without a leader. At a later time two nodes from different trees for the same group may be within transmission range of each other. The node in the group having the leader with the lower IP address initiates reconnecting the trees after receiving permission to do so from its leader. To reconnect, the node sends a route request to the other group leader with a flag to indicate this is a repair. When the node receives a route reply the two trees become reconnected with the node sending the reply message as group leader.

2.5.3.2 AMRIS Ad Hoc Multicasting Routing protocol utilizing Increasing ID NumberS (AMRIS) [56] is based on a shared tree and is geared towards long-lived multicasting sessions so that route reconstruction is emphasized over route discovery. Each node is assigned an ID number that increases with the number of hops from the core, usually the first source node. In order to reduce the number of join requests that propagate through the network, nodes only forward requests to lower numbered nodes. Each node on the tree periodically sends a one-hop broadcast containing its ID number as well as the ID numbers for its parent and children. When a link breaks, the higher numbered node (downstream from the core) is responsible for recovery actions. If it knows of a neighboring node that is a potential parent, it will send a join request to that potential parent node, otherwise it will broadcast a join request using an expanding ring search technique. If the upstream node of the broken link has no other downstream children on the tree then it will request to be pruned from the tree.

2.5.3.3 AMRoute Ad-hoc Multicasting Routing protocol (AMRoute) is another protocol based on the idea of a shared tree, but only the senders and receivers are nodes of the tree; there are no relay nodes. This is accomplished through the use of IP-in-IP unicast tunnels. If a tree contains a virtual link between node A and node B AMRoute uses a unicast route between the two nodes. If the path from A to B changes, the multicast tree is not affected as long as some path between the two nodes exists. Dynamically chosen core nodes detect new members and manage the tree but they are not central data distribution points as in other shared tree protocols.

Tree creation is a two-step process. Each core node sends out a join request message to discover a close member node using an expanding ring search. (Initially each node is a core of its own single node tree.) When another member node is discovered, the meshes are merged and one core is designated as the unique core of the new structure. Once the mesh has been formed, the core sends out a tree creation message to each of the group members that are adjacent to the core by a virtual link. Group members that receive the message remember the incoming link and forward the tree creation message to other virtually adjacent group members. Transient loops can be formed in the tree as a result of node mobility, but are eliminated once the network becomes less dynamic.

Several restrictions are imposed on tree nodes - a node is not allowed to graft itself to its own logical core, a node is only allowed to have a limited number of tree links, and a node's designation as a core can be changed due to characteristics of the multicast tree or an expired time period. Disadvantages of the scheme are reduced efficiency since the intermediate nodes between A and B do not participate in packet replication and an increase in the delay of the receipt of a packet, although it is no more than twice the delay in a protocol in which the unicast tunnels are not used.

2.5.3.4 Gupta and Srimani (G-S) Gupta and Srimani describe a protocol for mobile multi-hop radio networks [17] based on the CBT protocol that increases the reliability of a multicast message arriving at each of the intended receivers. When a node becomes disconnected from the multicast tree, it sends a REJOIN message downstream along the old tree branches. A REJOIN message causes a node to destroy its current routing information for the multicast tree and to have member nodes send a join request to the core. The use of the old path prevents loops from occurring. While the node is in the process of becoming reattached to the tree, any messages that are sent to any of the multicast nodes in the disconnected fragment are flooded into a selective area. The calculation of the flooding area guarantees that the message will be delivered to the destinations if possible.

2.5.3.5 ODMRP On-Demand Multicast Routing Protocol (ODMRP) [4] is a multicasting scheme which uses a mesh over which packets are forwarded through the use of scoped flooding. Moreover, since ODMRP builds its own link state tables on demand it can function as a unicast routing protocol as well as a multicast protocol without the need to maintain global link state information.

As found in many of the multicast protocols, ODMRP has a request phase and a reply phase. A source periodically broadcasts a join request throughout the network. When a join request is first received by a non-member node, it stores the incoming link as the link from the source and rebroadcasts the packet. When a node that wishes to become a member of the group receives the join request, it updates its own join table and periodically broadcasts it to its neighbors. If a node that receives a join table is listed as one of the entries, it becomes

a forwarding node, updates its own table and sends it to its neighbors, thus propagating the multicast group link state information. The method of joining a group produces shortest paths from each source to each receiver. The intermediate nodes simply flood a non-cached data packet if the node is on any one of these paths. The flooding increases the robustness since a node that is mobile has a better chance of receiving one of the redundant data packets.

No explicit leave messages are needed since all routes are kept as soft state. If a node wishes to leave the group it simply does not send join tables for that group. If a forwarding node does not receive a join table within a certain time interval it will become a non-forwarding node. If a source node wants to leave the group it quits sending out join request packets. The timeout periods must be carefully determined in order to avoid excess flooding.

An extension of ODMRP incorporates predictions based on traffic patterns gathered from a global positioning system (GPS) to determine how long a route is good and to compute the timeout values for the soft state and the frequency of join request broadcasts.

2.5.3.6 CAMP Core-Assisted Mesh Protocol (CAMP) [14][31] creates a mesh of shared trees and shortest path trees. It relies on a distance vector unicast routing protocol such as WRP to determine paths to cores and to source nodes. Core nodes may be designated at the time the multicast group is set up or they may be dynamically chosen. The former assumes that a node can find out initial group information from a service similar to a domain name server (DNS). The latter requires the core to periodically broadcast its own advertisement messages.

A node that wishes to join a group sends a join message towards its nearest core node. If a node that is a group member receives the join message on its way to the core, the member node returns an acknowledgement to the new member. If the join message gets all the way to the core, then the core returns the acknowledgement. Once a node can be attached to the mesh, it announces this fact to its neighbors. A neighbor to a node on the mesh records which neighbors are on the mesh in case it needs to attach itself later. This ploy reduces the number of join/ack message pairs in the network. Another technique to reduce traffic is to allow nodes to join the group as a sender-only node in which case data packets from other sources are not forwarded to the node.

Intermediate and destination nodes accept a data packet from any incoming link; receipt of a packet is not restricted to a particular incoming link as in other multicast tree schemes. This feature allows a node to periodically check to see if the unicast route to a source that sent a cached data packet is shorter than the path currently used. If so, it sends a join message towards the source, eventually resulting in a reverse shortest path route to the sender. During the time the new route is being constructed the data continues to be forwarded along the existing route. Depending on the time out intervals, the node may receive the data along two different paths increasing the robustness of the multicast.

2.5.3.7 MCEDAR Multicast Core Extraction Distributed Ad hoc Routing (MCEDAR) [50] is a multicast version of the unicast routing protocol CEDAR. MCEDAR uses a subset of the core nodes established by CEDAR for group communication. A "robustness factor" controls the number of core neighbors a core may have in a multicast group's mesh. The group's core nodes perform multicast tree operations on behalf of the nodes in their domain. Communications pertinent to the multicast group are efficiently broadcast over the mesh. Each core node implicitly determines a source-based tree from itself to the other core nodes over the mesh established for the group as a result of the core broadcasts. Data forwarding can be done over these trees, but an optimization is available to reduce the congestion on the mesh and the length of the paths. In the optimization each core node determines a source-based tree from the nodes in its domain to those in the neighboring downstream domains.

2.5.3.8 Comparisons The multicast protocols discussed in this section can be compared based on their reliance on a unicast protocol, the primary structure used for distributing data, the amount of flooding that is done, and the amount of data redundancy provided. See Table 1.

What are the shortcomings of these multicast protocols? MAODV and AMRIS react to link failures by discovering a new route to the multicast tree. Data may be lost during the discovery period. AMRoute depends on a unicast protocol to forward data, so if the underlying unicast method finds a route on-demand, similar packet loss will occur. G-S

Table 1: Comparison of Wireless Multicast Protocols

	MAODV	AMRIS	AMRoute	G-S	ODMRP
Does it need a unicast protocol? If so, which one?	yes, AODV	no	yes, any (to make tunnels)	yes, any	no
Primary Structure	source trees	shared tree rooted at first sender	shared tree of virtual links	shared tree	mesh of shortest paths
Is an advertisement periodically flooded through net? If so, from whom?	yes, group leader	no	yes, each core	yes, the core	yes, each sender
Does a member receive (nonlooped) redundant data?	no	no	no	yes, during repair	yes

Continuation of Table 1	CAMP	MCEDAR	MOLSR	RoMR
Does it need a unicast protocol? If so, which one?	yes, any distance vector	yes, CEDAR	yes, OLSR	yes, any link-state protocol
Primary Structure	mesh of shortest paths	mesh of cores	MPRs	multiple trees
Is an advertisement periodically flooded through net? If so, from whom?	yes, new cores	no	yes, the source	no
Does a member receive (nonlooped) redundant data?	yes, at times	no	yes	yes

reacts to a link failure that partitions the core-based tree by selective flooding so the data will arrive at the disconnected nodes, but the use of the shared tree may cause more traffic overall for the group. ODMRP uses the concept of a forwarding group when links fail which results in data redundancy, but otherwise relies on shortest paths between members. CAMP converges to shortest paths between members so if an important link fails and another path is not active, it would have to wait for the unicast protocol to determine the new route.

The existing proposed multicast protocols react to a link failure after it occurs. None attempt to predict link availability and provide reliability using data redundancy based on the prediction before the link fails. Our solution to the multicasting problem in mobile ad-hoc networks uses a proactive approach to tree creation and maintenance in order to reduce the overhead encountered at the time of link failures without sacrificing optimality of the tree structure. Even though we construct multiple trees, those trees are based on approximations of optimal trees in order to counteract the effect of allocating resources to multiple trees. An additional improvement over the current protocols is the specification of system parameters that can be fine-tuned to specific environments and network conditions.

In this chapter we have examined background material dealing with routing protocols in general and related work concerned with algorithms for tree formation as well as particular unicast and multicast protocols appropriate for use in mobile ad hoc networks. In the next chapter we propose RoMR with variations as our solution to the multicasting problem. It incorporates the use of a proactive link-state unicast protocol and multiple Steiner tree approximations in a novel multicasting protocol.

3.0 ROBUST MULTICAST ROUTING (ROMR) ARCHITECTURE

The responsibilities associated with network communications are commonly viewed as a set of layers with the topmost layer dealing with communications between applications running on a different hosts down to the bottom layer dealing with the details of how to put the individual bits out onto the communications link. The first section of this chapter examines the general layering model and how adding multicasting capabilities to the networking communications affects several of these layers. The rest of the chapter discusses the architecture of the ***Ro****bust* ***M****ulticast* ***R****outing protocol* (RoMR) we developed to address the multicasting problem.

3.1 NETWORK COMMUNICATIONS FRAMEWORK

Early in the history of networking the International Organization for Standardization (ISO) developed a 7-Layer Reference Model to describe the tasks and responsibilities needed in network communications [9]. Here we examine the well known historic seven layer model and show how multicasting affects various layers.

3.1.1 A Generalized Layered Framework

The seven layers of the ISO Reference Model are the application layer, the presentation layer, the session layer, the transport layer, the network layer , the data link layer, and the physical layer as shown in Figure 9.

- Application Layer: The topmost layer of the model deals with applications. An application layer protocol specifies how an application program on one machine makes a request to another machine and how the application on the other machine responds.

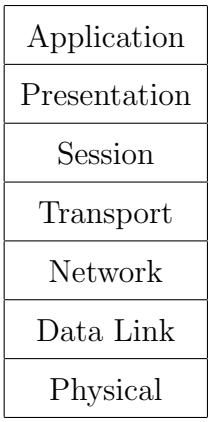


Figure 9: ISO 7-Layer Reference Model

- Presentation Layer: The protocols used in the presentation layer specify how to represent data and how to translate from one computer's representation to another.
- Session Layer: The session layer lies between the presentation layer and the transport layer and is responsible for maintaining aspects of communication that pertain to the entire session from the time of establishment until the session is over.
- Transport Layer: Below the session layer is the transport layer which is responsible for end-to-end delivery of data and delivers packets to the upper layers. Today two protocols are commonly used at the transport layer for communications over the Internet: User Datagram Protocol(UDP) or Transmission Control Protocol (TCP). TCP provides error-free delivery of data desired in unicast communications by using a system of acknowledgements. If a packet is not received correctly, an acknowledgement is not received by the sender and the packet is sent again. UDP, on the other hand, does not incorporate acknowledgements. If a packet is in error or is lost UDP does not cause the packet to be resent.
- Network Layer: Below the transport layer is the network layer which is responsible for routing packets. A node that receives a packet must be able to determine if the packet needs to be forwarded or not. This is done by consulting the routing tables maintained by network layer protocols. The Internet Protocol (IP) provides this service in unicast communications over the Internet.

- Data Link Layer: The data link layer is responsible for the transport of data over a particular link. The data link layer has a sublayer called the Medium Access Control (MAC) sublayer which is responsible for determining who goes next on a multiaccess channel.
- Physical Layer: The physical layer is responsible for putting the individual bits onto the link using methods appropriate to the actual media used.

3.1.2 Multicasting and the Layered Framework

Multicast communication is the sending of data from a single sender to a set of recipients or from multiple senders to a set of recipients. *Multicast networking* provides support in the network so that multicast communication is able to occur. Let us consider multicasting in relation to the layers of the 7-layer model above.

- Multicasting and the Application Layer: *Multicast applications* are the applications that rely on the multicast network services to establish and maintain multicast communication. Multicast applications send data to members of a multicast *group*, which is a set of nodes designated by a common group address, cooperating to achieve a common goal.
- Multicasting and the Session Layer: In terms of multicast communications, one of the responsibilities of the session layer is to keep track of the multicast groups. It must inform other layers of changes in group membership so that appropriate actions can be taken.
- Multicasting and the Transport Layer: Many unicast applications that run on top of IP use TCP as the transport protocol due to the reliability of TCP, but TCP can introduce problems when used with multicast communications. Suppose a subset of the receivers do not receive the packet. The sender under TCP would need to resend the packet either to the entire group or to the individual members that did not receive it. If only a few receivers need the copy sending it to all receivers could create network congestion problems. The acknowledgements that are required to determine which nodes received the packet and which did not would also add to the network congestion. Time and delays also come into play when considering TCP for multicast. Should the need to

resend to some of the receivers cause a delay in the transmission to the nodes which successfully received the packet? These problems show that TCP is not adequate to support multicasting. As a result multicast applications usually run on top of UDP, although a few run on top of IP directly. If multicasting is not supported by any of the protocols in the lower layers of the protocol stack, then the application will have to provide its own set of routines for multicast communications essentially creating its own multicast transport layer.

- Multicasting and the Network Layer: A message directed to a multicast group is addressed with a group address instead of an individual host's address so the network layer must be modified to work with these group addresses.
- Multicasting and the Data Link Layer: Two data link layer protocols, frame relay and Asynchronous Transfer Mode (ATM), have been developed to support multicasting in wired networks. The problem with data link layer multicast protocols is that the multicast groups must be set up by the network provider and changes to the groups result in a reconfiguration of the network. Such data link layer multicast protocols are clearly not appropriate for use in mobile ad hoc networks.
- Multicasting and the Physical Layer: The physical layer is not affected by whether or not the communications is point-to-point unicast or if it is multicast.

In section 3.2.4 we will see which of these layers are modified by proposed multicasting protocol, RoMR.

3.1.3 Group Management

In multicast communications the concept of the multicast group is very important. These groups may be semi-permanent or dynamic [34] therefore we need to establish a set of primitives to create, dissolve, and manage the group allowing nodes to join or leave a group as desired.

status=FormGroup(grpId, grpStruct, grpType)

status=TerminateGroup(grpId, grpOwnerId)

status=JoinGroup(grpId, mbrName, ntfyList, timeout)

status=LeaveGroup(grpId, mbrName, ntfyList)

Sometimes a multicast group manager has the additional capabilities to invite a member to join a group or to limit group membership by excluding certain nodes from joining the group. In this case, the command set would be extended with the following additions to the commands list above.

status=InviteToGroup(grpId, mbrName, grpOwnrName, ntfyList)

status=ExcludeFromGroup(grpId, mbrName, ntfyList)

The arguments grpId, mbrName and grpOwnr identify the name of the group, the name of its members, and the name of the owner. A group is created using the primitive FormGroup(). Upon its creation, the group is owned by its creator who becomes automatically a member of the group.

The grpStruct argument determines the relationship among the members of the group. In a coordinated group, the owner of the group has a special privilege in accepting new members. In a peer group, however, all processes are treated equally. The grpType determines whether the group is closed, or open. In a closed group communications is restricted to only the members of the group. In an open group, non-members may send messages to the members of the group. The terminate primitive allows the destruction of the group. Only the owner of the group may execute such a primitive.

The primitives JoinGroup(), and LeaveGroup() allow a process to become a member of the group, or to leave the group. For some join methods, the knowledge of the group name may suffice to become a member of the group. For others, the owner of the group or an election process may be required before the process candidate can become a member of the group.

The last primitive ExcludeFromGroup() causes the membership of a group member to be terminated. For some applications, the right to exclude a member is only granted to the owner. For others, a vote must be taken before a member is excluded. The argument ntfyList contains a list of group members that need to be notified.

Certain information must be maintained to manage the group. This information includes group names, group members, and incoming messages.

In future chapters we shall see that RoMR implements the first four of these primitives

in a closed group setting.

3.2 OVERVIEW OF ROMR

The problems associated with multicasting in mobile ad hoc networks were introduced in Chapter 1. The general multicasting problem was then cast as a problem to find multiple sets of multicasting trees. The objective of the research problem at hand is to develop a multicasting scheme with proactive and reactive features which emphasizes reliability and efficiency to use in routing data packets from the sender to the receivers in a multicast group of a mobile ad hoc network. Since an optimum solution to the multicasting problem is NP-hard, due to the Steiner tree problem, the multicasting problem is difficult to solve in a static wired environment and even more so in a dynamic mobile ad hoc network. Instead of seeking optimal solutions we will develop heuristics to achieve the objective of reliably delivering packets while meeting the constraints to avoid network congestion due to redundant packets as closely as possible. In the remainder of this chapter we describe the Robust Multicast Routing protocol (RoMR) as a set of heuristics for the problem.

3.2.1 RoMR Approach

The main goals of RoMR are to deliver packets to the recipients in a reliable and efficient manner. First let us address the reliability factor. Instead of creating a single dynamic multicast tree over which all of the packets from the source are routed as found in many of the multicast protocols reviewed in Section 2.5, RoMR creates multiple multicast trees. This approach provides alternate paths for packets to travel in the event that some of the paths become disconnected. Packets travel down *each* one of the trees in the set of trees increasing the likelihood that a receiver receives the packet sent from the sender node as the topology of the network changes. Another aspect of RoMR which affects reliability in a positive manner is that it does not wait until all trees become disconnected before it computes a new set. Rather it tracks the number of trees in the current set of trees that remain connected and when that number reaches a threshold, new trees are computed and distributed before all

distribution paths are lost.

The second major goal of RoMR is to deliver the packets reliably without placing an undue burden on the network. Inefficiency stems from two sources: i) routing may result in many unnecessary redundant packets and ii) recomputing and distributing the multicast trees may occur too often. In order to accomplish the subgoal of efficient multicasting, RoMR incorporates two ideas into its heuristics. First it makes use of an encoding scheme whereby k packets are encoded as n packets, where n is the number of multicast trees in the current set of multicast trees. When $k = 1$ and n is large, the number of extra packets injected into the network is high producing an n-fold increase in the number of packets travelling in the network due to multicasting. RoMR attempts to find values of k and n such that the reliability due to redundancy is high, yet the amount of extra network traffic is not overly burdensome. Second, RoMR makes the trees based on the reliability of the links in hopes that the trees will remain intact as long as possible. This second approach to efficiency must not make the paths from sender to receiver overly long.

The questions that RoMR must answer in an effort to meet the objective and to satisfy the subgoal as closely as possible are:

- How many trees, n, should be created in a multicast tree set?
- Should links be shared among trees in a tree set? If so, how many should be shared?
- What value of k should be used in the encoding scheme?
- How can the trees be created so that they will remain intact as long as possible while not introducing overly long paths?
- How can weights be assigned to known links to aid in the computation of the trees?

How many trees need to be created? The actual number of trees created is a function of the level of reliability specified by the user as well as the topology of the network at the time of tree creation. An elastic application will specify three values when using RoMR as its multicast routing protocol: μ, a minimum number of trees to be formed, τ, a reliability threshold associated with the set of trees, and ρ, the percentage of links that can be reused from one tree to the next. A value of 0 for ρ would ensure disjoint trees which may be appropriate in networks with very few or no strongly reliable links. Default values for the

parameters are used if the application does not specify them.

Should links be shared among trees in a tree set? RoMR shares only those links that are deemed to be the more reliable links in the set of candidate links at the time of tree creation, again supporting reliability as well as efficient use of network resources. When constructing a set of trees, RoMR will reject $(100-\rho)$ percent of the links in the most recently constructed tree from being candidate links in the other remaining trees in the tree set.

How does RoMR determine the value of k? It starts with an initial value and then examines the time interval between the formation of the current set of trees and the previous set. If this is smaller than a given threshold then the recomputation of the trees has occurred too frequently. As a result the value of k is reduced when possible so that the next tree set will be recomputed when the number of intact trees reaches this new smaller value. If the degree of dynamicity of the network remains somewhat constant, this should result in a longer time interval between tree set computations. This also results in the recipient needing to receive a smaller fraction of the packets sent along the n trees. Similarly if the time interval between successive tree computations is extremely long, then the amount of redundancy may be reduced, which is caused by an increased value of k. Section 5.1.5 provides more detail in the determination of the value of k.

How can the trees be created without introducing overly long paths? When creating the trees, we will use a Steiner tree heuristic in which we attach branches to the tree in such a way to increase its "bushiness" as opposed to its depth when such a choice is available. The particular algorithm is an enhancement of an online version of the Shortest Path Heuristic (SPH) mentioned in Section 2.4.2.1. It is guaranteed to have an upper bound of twice the number of branches as the optimal tree [51] and is easy to implement. The enhancement addresses the question about overly long paths. The paths that are added to the tree will maximize the degree of the relay nodes in the multicast tree when multiple choices exist for the attachment of the next node. This minimizes the number of retransmissions and reduces the chance for packet collision. More details of the algorithm are given in Section 5.1.3.

How are weights assigned to links? RoMR assigns weights to the known links in the network to reflect the probability that the link will continue to exist into the next time frame. The underlying unicast protocol must be slightly modified in order to include these

weights of links during the exchange of topological information. During the creation of a single multicast tree within a set of trees these weights are taken into account so that only links that are likely to remain in existence for the longest time are eligible to be shared among trees. The method to calculate the weights of the links is another feature unique to RoMR and is discussed in Section 4.1.

3.2.2 RoMR Framework

3.2.3 Formal Specification of RoMR

Given:

$G(t) = (N(t), E(t), W(t))$**:** A time-varying network modelled as a directed graph with

$N(t)$**:** a set of nodes that comprises the network at time t. It is assumed that all nodes in the network have the capabilities to receive and forward transmissions so a node is able to act as a router in a multicast tree as well as a receiver in a multicast group.

$E(t)$**:** a set of directed edges representing the radio links at time t. If link (u, v) exists then we assume link (v, u) also exists.

Q**:** a dynamic set of join and leave requests. Each request is a triplet (*node, action, t*) to describe the action of a particular node joining or leaving the set of receivers of a group at a given time t.

ρ**:** a percentage of the links that can be reused from one tree in the formation of the next tree in a set of trees.

τ**:** a threshold that represents the lowest level of desired reliability of an entire set of trees.

μ**:** a minimum number of trees to create.

Compute:

- $W(t)$: a set of weights $w_e : E(t) \mapsto [0.0, 1.0]$. Each link $e \in E(t)$ is associated with a weight $w_e(I_{k+1})$ which corresponds to the probability that link e will continue to exist over the next time interval I_{k+1} given link e existed during time interval I_k.
- a dynamic graph $G'(t) = \bigcup_{i=1}^{m} M_i(t)$ for group *gid* of m multicast trees such that
  - each tree may have at most ρ percent of its links in common with the successive tree

  - the value of m is based on the topology of the current network, as well as τ and μ.
  - the set of trees are adjusted to reflect the group membership changes specified in Q.
  - whenever possible, $m \geq \mu$
  - the cumulative weight of $G'(t) \geq \tau$ whenever possible
  - the degree of interior nodes of $G'(t)$ is as large as possible given the other constraints
- an encoding method that will encode a group of packets from the source as n packets (where $n = m$, the number of multicast trees) such that a receiver that receives a subset of a given size of the n packets will be able to decode the original source packets.
- a group management scheme that will maintain group membership information and update the multicast trees when a change in membership occurs.

3.2.4 RoMR Architecture

In Section 3.1 we discussed the general 7-layer networking model. In this section we will examine RoMR's framework in relation to the general model.

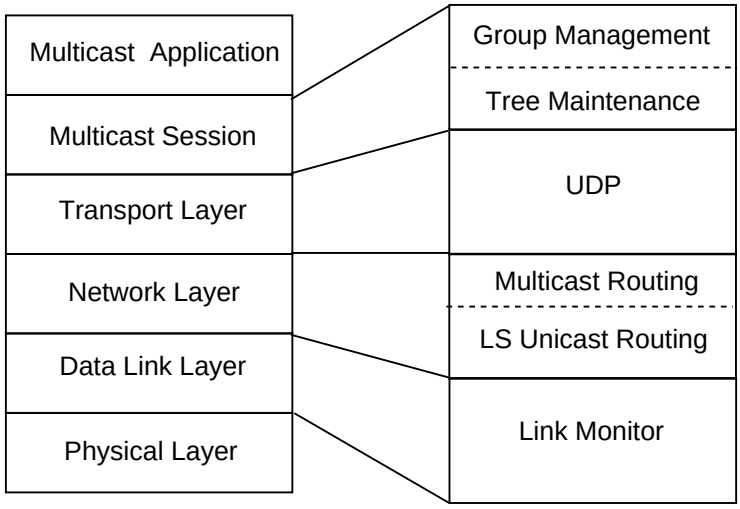


Figure 10: RoMR Framework

RoMR has four main components that work together to accomplish the goal under the set of constraints of the problem. The *link component* is responsible for monitoring the signal

strengths of links, calculating weights for the links, and exchanging the link information with neighbors using the underlying unicast protocol. The *tree component* is responsible for the making, distributing, and maintaining the multicast trees, the *routing component* is responsible for the routing of packets from other applications to the group members over the multicast trees, and the *membership component* is responsible for servicing requests for nodes to join and leave the group and to maintain the membership lists. The routing component uses the trees formed in the tree component, the tree component uses the topology tables maintained by the unicast protocol as well as the list of member nodes kept by the membership component, and the unicast component includes the weights of the links determined by the link component during its exchange of topology tables.

3.2.4.1 The Link Component The link component in each node monitors the strength of the radio signal from neighboring nodes and determines the weights associated with adjacent links. Using values of readings over a time interval the node will determine an area in which the neighboring node could be located in the near future. The ratio of the overlapping area in which the neighbor could possibly hear the transmission from the original node to that of the entire area of the neighbor's locus results in a weight in the range $[0, 1]$ to assign to the link. This will reflect the probability that the link will exist during the next time interval given that it existed in the most recent time interval. The link component will work in conjunction with the underlying table-driven link-state unicast protocol to distribute these values during the exchange of topology tables. The appropriate formulas and their derivations are given in Section 4.1.

3.2.4.2 The Tree Component RoMR computes multiple multicast trees that may be interconnected through the sharing of reliable links and then maintains these trees as the group membership and network topology changes. The idea is to make a set of n possibly interconnected multicast trees connecting a single source to a set of receivers.

Figure 11 shows an example of a network with a multicast group that is connected by two interconnected trees with a maximum of two common edges. The large circles represent members of the multicast group and the small circles represent nodes that are possible relay

nodes. The nodes and all possible radio links are shown in (a). The first multicast tree (b) is created using a Steiner tree approximation algorithm. In this example the two most reliable links in the first tree (shown as thick segments) are eligible to be included in the second multicast tree. The other links in the tree are considered to be weaker and are removed from the set of possible edges under consideration for the second tree. The second multicast tree using both of the two most reliable links from the first tree is shown in (c).

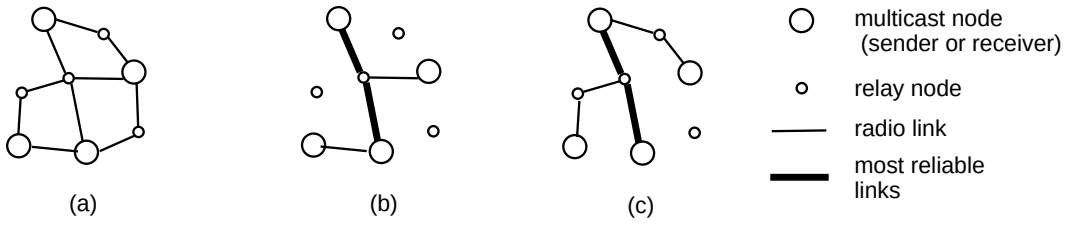


Figure 11: Interconnected Multicast Trees

Specifically the functions of the tree component include:

1. Determine n, the number of trees to create.
2. Build the multiple multicast trees based on information gathered by the underlying unicast protocol and the link component.
3. Determine the best value of k to use in the (n, k) encoding scheme based on the dynamicity of the network. A higher value of k results in lower overhead but reliability may be adversely affected in highly dynamic networks, whereas a lower value of k may result in increased reliability with a higher overhead cost. Any k of the n packets received will be sufficient for the member node to be able to decode the k original packets.
4. Distribute the trees to the relay nodes and group members
5. Monitor the topology of the network by examining the topology tables provided by the unicast link state protocol. The number of intact trees is expected to decrease with time. When the number of intact trees reaches k, remake and distribute the trees. Special actions are taken when $k = 1$.
6. Send messages to relay nodes to use current trees when new trees are not made within the timeout interval, thus avoiding premature timeouts from occurring.

The value of k mentioned above is dynamically determined as a tradeoff between the quality of the tree set and the overhead. If the trees are being made too frequently then the value of k will decrease resulting in the receiver nodes needing to receive fewer of the n packets in order to decode the originals, increasing the time needed between tree creations. Given the dynamics of the network, RoMR will strive to provide enough redundancy to increase the likelihood that a node receives the packets according to the desired level of performance while controlling the amount of redundancy to avoid overloading the network. The algorithms of the tree component are given in Section 5.1

3.2.4.3 The Routing Component The routing component has two complementary parts. The first subcomponent processes packets that come from upper layers of the protocol stack that are addressed to a group address. In this subcomponent the router component buffers k of the packets addressed to the group, then applies a (n, k) forward error correction encoding scheme before sending the n encoded packets on to the MAC layer. The second part of the routing component examines a data packet that is received from the MAC layer from another node. If the packet is addressed to a group and if the node is a relay for that group for one of the trees then the node forwards the packet. The use and caching of sequence numbers prevents the occurrence of possible loop formation. If the node is a member of the group then the node buffers the incoming packets addressed to the group until k packets have been received at which time the routing component decodes the packets and passes them up the protocol stack to the upper layers. All routing tables are kept as soft state. If a node does not receive periodic updates the entries will timeout and be deleted from the multicast part of the routing table. The algorithms associated with the routing component are discussed in Section 5.2.

3.2.4.4 The Membership Component The membership component services the join and leave requests for the group and keeps the tree component informed of the current membership lists. A node sends the requests to the multicast manager. Group management commands for creating, destroying, joining, and leaving a group were discussed in Section 3.1.3.

3.2.5 RoMR Node States

A node may be in one of five basic states: default, multicast manager, sender, receiver, or relay. Other composite states exist and are combinations of the last four states listed. The state of the node reflects the responsibilities and thereby the actions of the node.

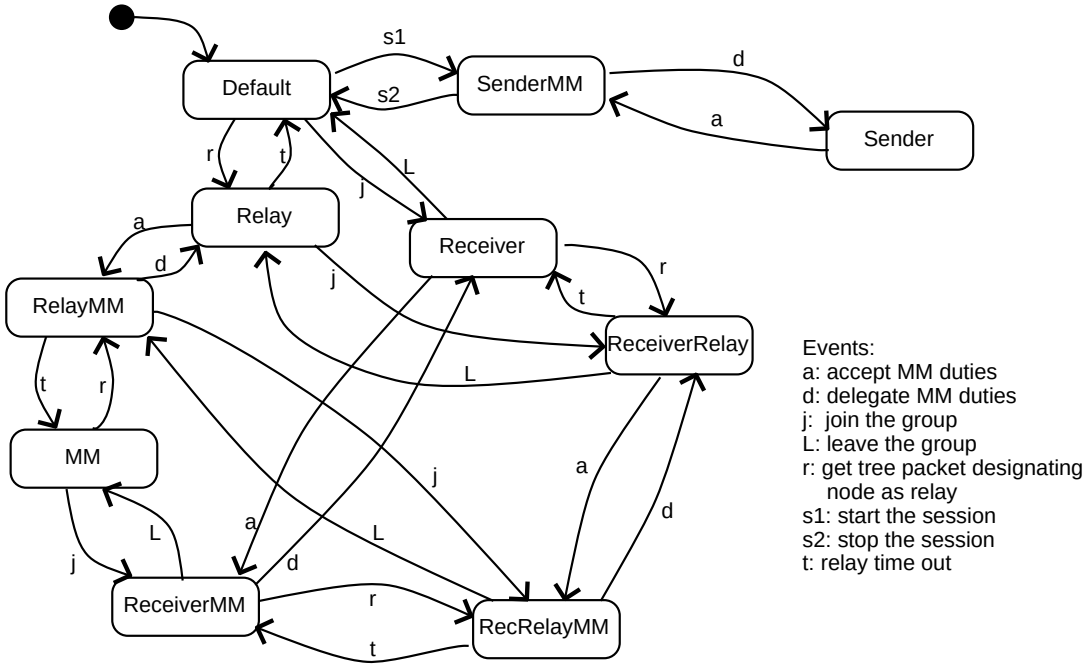


Figure 12: RoMR State Diagram

3.2.5.1 The Default State A node in the default state is not associated with a multicast group in any way. It is not a sender, receiver, relay node, or a multicast manager. All nodes are in the default state before multicast communications are initiated.

3.2.5.2 The Multicast Manager State Each multicast group includes a node which acts as the multicast manager (MM) for the group. This node is responsible for performing the duties of the tree and membership components. Initially the first source node will be designated as the MM. If the source becomes a bottleneck, it will advertise to its neighbors

that it is looking for a node to take over the MM duties. The willing neighboring nodes will respond and the current MM will select one of those willing nodes, sending it the group information. The selection can be based simply as the lowest IP number or may involve more sophisticated load distribution analysis. RoMR is designed on the basis of closed groups as discussed in Section 3.1.3 although the existence of the group is public knowledge. Any node may join a group by contacting the group's MM. The node will be able to determine the MM node for a group as that information will be available to all nodes through a registration system similar to a local domain name server (DNS).

3.2.5.3 The Sending State A node switches to the sending state becoming a source node when an application in the node begins to send data to a multicast group address. In order to use RoMR as the multicast protocol, the application either must specify the values of the parameters used by RoMR through an API or accept the default values for μ, τ, and ρ which are the minimum number of trees to create, the minimum cumulative weight of the set of trees, and the reuse factor respectively. The sender will obtain the values of k, the number of trees that must remain intact before forming a new set of trees, and n, the number of trees in the current set of trees, as well as current set of multicast trees from the multicast manager. Once the values of k and n are known the network layer in the sender node is responsible for the encoding of the outgoing packets. When the network layer of the source node receives a packet to send to a specific group address from the transport layer the source node proceeds as follows:

1. Buffer k successive packets addressed to that group.
2. Encode the k packets as n unique packets.
3. Broadcast the n packets to neighboring relay nodes and receivers.

3.2.5.4 The Relay State The relay nodes are determined when the multicast trees are formed. They are the nodes that are the interior nodes on at least one of the multicast trees. Each relay node is associated with one or more of the trees in the current tree set. If the relay is part of the i^{th} tree, then the relay node will forward packets designated as belonging to the i^{th} tree. If a relay node does not belong to tree j, then it will not forward packets

that are designated as being routed on tree j. Note that some or all of the receivers may also act as relays. In the simplest version of RoMR a relay node does not save a copy of the tree in order to reduce the demands on the relay node. The responsibilities of a relay node include:

1. Update the forwarding information when a new set of n trees arrives, recording the trees to which it belongs.
2. Forward all tree packets to downstream relay nodes and receivers.
3. If a data packet is designated as being routed on the i^{th} tree and if the relay node belongs to the i_{th} tree, then forward the packet to the neighboring nodes.
4. Delete the entry in the forwarding table for tree j of the specified multicast group if no data or new multicast trees has arrived since the previous time-out for tree j.

3.2.5.5 The Receiver State A receiver node in the multicast group has the following responsibilities:

1. When it receives a (nonduplicate) data packet
   a. Keep track of the number of packets received for the indicated packet group.
   b. If k packets have previously been received for the indicated packet group then ignore the packet since it has already been decoded.
   c. Otherwise if the received packet is the k^{th} packet decode the group of k encoded packets and send the k recovered packets to the application.
   d. Otherwise buffer the packet until enough of the other packets in the packet group arrive or a time out occurs.
2. When it receives a new set of multicast trees, determine the encoding scheme and create the corresponding decoding matrix.

3.2.5.6 Composite States A node may act both a sender as well as a multicast manager thus giving rise to a composite state senderMM. The other combination states are receiverMM, relayMM, receiverRelay, receiverRelayMM.

3.2.5.7 All States All nodes in the network will participate in the exchange of topology information in the underlying unicast protocol regardless of their state. Thus all nodes will:

1. Gather data for link availability calculations.
2. Perform the link availability calculations.
3. Exchange the link availability information with neighbors in the unicast protocol's topology table exchange.

4.0 ROMR LINK AVAILABILITY

Various approaches to predicting a link's availability were discussed in Section 2.3. Those either overly simplified the prediction calculations, assumed the availability of unobtainable information, or relied on statistical distributions of mobility patterns on which to base predictions. As such, those models may be unrealistic. RoMR uses a different approach. A node computes an area in which a neighboring mobile node is likely to be in the near future and then examines the portion of that area which overlaps the transmission range of the node performing the calculations. A weight is assigned to the link to the neighboring node reflecting the ratio of the overlap area to the possible location area. In order to perform the calculations RoMR relies on signal strength readings from neighboring nodes. Once the link calculations have been performed, the values are distributed with the topology tables by the underlying link-state protocol, thus RoMR requires a slight modification to the unicast protocol.

4.1 CALCULATION OF LINK WEIGHTS IN ROMR

Link weight is a function of two accumulated measures of signal strength received from a neighboring node. The first is the average signal strength received from the transmitting node from time $t_i - \Delta t$ until time t_i. In the ensuing discussions this is simply referred to as the received signal strength at time t_i. The second measure is the average signal received during the time interval from t_i to $t_i + \Delta t$ from the same transmitting node. Likewise, this will be referred to as the signal strength of the received signal at time $t_i + \Delta t$. The power of the transmitted signal at the sender is assumed to be constant over the small interval

of time Δt. A locus of possible destinations for the transmitting node at time $t_i + 2\Delta t$ is determined relative to the receiving node based on the strengths of the previously received signals. The intersection of this locus set with the area in which the receiving node is able to receive a reliable signal determines the weight of the incident link corresponding to the probability that the link will be available over the next time interval.

Most of the reduction in a signal's strength from the time of transmission to the time of reception is due to the distance between the two nodes, the obstacles between them, and the number of different paths the signals travel due to reflection. In a free space environment the average received power at the receiving antenna is

$$P_r = P_t \left(\frac{\lambda}{4\pi d} \right)^n g_t g_r$$

where P_t is the transmitted power, λ is the carrier wavelength, d is the distance between transmitter and receiver, n is the path loss coefficient, g_r is the antenna gain at the receiver and g_t is the antenna gain at the transmitter[54]. The value of n depends on the environment, varying from 2 in a free-space environment, 4 in a typical urban environment up to 6 when there are many obstacles in indoor transmissions.

In the idealistic free-space formula for the received power, we assume that the path loss is a function of distance, but in reality other sources of path loss exist. Radio waves travel over multiple paths from the source to the destination and as a consequence other more complex models have been developed. The two-ray model assumes the radio signals travel over two paths - one part of the signal travels in a direct line-of-sight path and the other is bounced off of the surface of the earth. Another model incorporates the effects of shadow fading due to the signal being blocked by objects in the environment. Several additional path loss models based on the size of the radio cells are discussed in [36]. In addition to these large-scale effects, rapid fluctuations in the signal result from small-scale fading. Small-scale fading is caused by the movement of the transmitter, receiver, and/or objects between them. Multipath fading and effects of the Doppler spectrum are two sources of small-scale fading.

For the calculations that follow we assume $n = 2$ and there is no other source of path loss. For a particular pair of nodes during a short time span the values of P_t, λ, g_t, and

g_r can be considered to be constant and can be combined with the 4π into one constant K giving

$$P_r = \left(\frac{K}{d^2}\right) \tag{4.1}$$

Notice that the value of K may need to be recomputed periodically as the transmission power decreases due to a node's battery becoming depleted.

Lemma 1 *If p_0 is the power of the signal received by node n from node m at time t_i, p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$, and $p_0 < p_1$, then the area of the locus of the possible locations of node m at time $t_i + 2\Delta t$ is*

$$\frac{K}{p_1}\frac{4\left(\arcsin\left(\frac{1}{p_1}\sqrt{p_1 p_0}\right)\right)p_0 + 6\sqrt{\left(-p_0\left(-p_1+p_0\right)\right)} + 2\left(\arcsin\left(\frac{1}{p_1}\sqrt{p_1 p_0}\right)\right)p_1 + 2\pi p_0 + \pi p_1}{p_0}$$

Proof. $p_0 = \left(\frac{K}{d_0^2}\right)$ and $p_1 = \left(\frac{K}{d_1^2}\right)$ from Equation 4.1

Solving for the distances:

$$d_0 = \sqrt{\frac{K}{p_0}} \tag{4.2}$$

$$d_1 = \sqrt{\frac{K}{p_1}} \tag{4.3}$$

$d_1 < d_0$ since we know $p_0 < p_1$.

Let point N be the location of node n. Let points P_0 and P_1 represent the perceived locations of node m at times t_0 and t_1 respectively relative to node n. (See Figure 13.) P_0 could be any point on circle C_0 with center N and radius d_0 and P_1 could be any point on circle C_1 with center at node n and radius d_1. The analysis is equivalent for all possible positions of P_0 on circle C_0 at distance d_0 from node n so fixing P_0 will not affect the results. In order to perform the analysis we assume that the net effect of the node's mobility during one time frame is greater than or equal to the net effect during the next time interval in which case node m will be on or in the interior of circle C_2, the circle centered at P_1 with radius P_0P_1 at time $t_{i+2} = t_0 + 2\Delta t$. The union of all possible circles where node m could be located at time t_{i+2} forms the interior region of a limaçon. A limaçon has two basic shapes. When $d_0 \leq d_1 < 2d_0$ the limaçon has a dimple as in Figure 14(a) and when $d_1 < d_0$

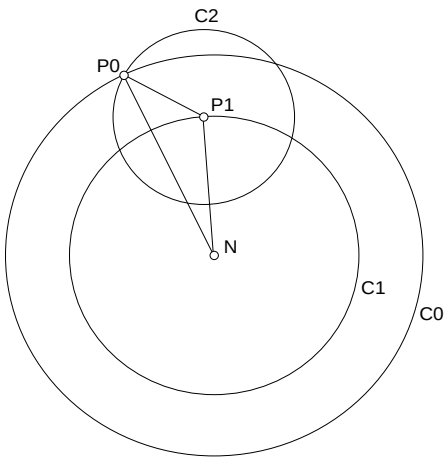


Figure 13: Relative Positions of Nodes

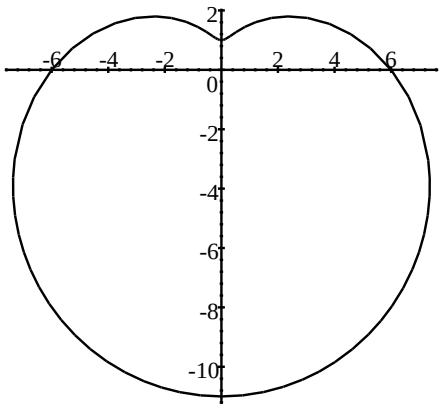
the limaçon has a keyhole as in Figure 14(b). The dimple disappears when $d_1 \geq 2d_0$ and the limaçon becomes convex. We will assume that the general characteristics of the node's movement change an insignificant amount over the interval $2\Delta t$, that is, the node will not experience a significant deviation in speed.

A limaçon with the orientation as shown has the formula

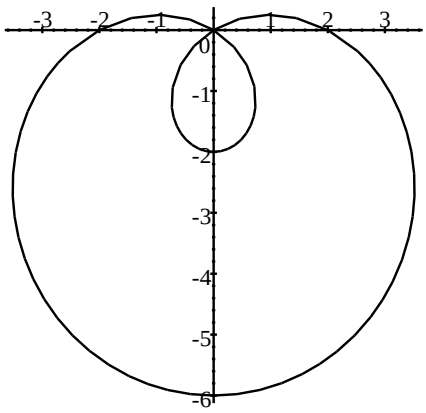
$$r = 2d_1 - 2d_0 \sin\theta \tag{4.4}$$

in a polar coordinate system with the center of the two circles at $(0,-d_0)$ in rectangular coordinates. From calculus we know the region bounded by lines $\theta = \alpha$, $\theta = \beta$, and the curve $r = f(\theta)$ has area $A = \frac{1}{2}\int_\alpha^\beta [f(\theta)]^2 d\theta$.

In the case when $d_0 > d_1$ the limaçon has a keyhole area that begins when r is first equal to zero as θ increases from 0 and the keyhole region continues while r is negative, until the point when r is zero once again. Let ϕ be the angle at the beginning of the keyhole, thus the keyhole occurs from ϕ up to $\pi - \phi$ where $\phi = \arcsin(d_1/d_0)$ so the formula for the area of the limaçon that does not duplicate the keyhole is



(a) r=6 - 5 sin(theta)



(b) r = 2 - 4 sin(theta)

Figure 14: Graphs of Limaçons: (a) $d_0 \leq d_1 < 2d_0$ (b) $d_1 < d_0$

$$\begin{aligned} \text{A of limaçon} &= \frac{1}{2}\int_0^{\phi} (2d_1 - 2d_0 \sin\theta)^2 \, d\theta + \frac{1}{2}\int_{\pi-\phi}^{2\pi} (2d_1 - 2d_0 \sin\theta)^2 \, d\theta \\ &= 4d_1^2\phi + 8d_0 d_1 \cos\phi - 2d_0^2 \cos\phi \sin\phi + 2d_0^2\phi + 2d_1^2\pi + d_0^2\pi \\ &\quad \text{where } \phi = \arcsin(d_1/d_0) \end{aligned}$$

Replacing d_0 and d_1 with the expressions involving p_0 and p_1 from Equation 4.2 and Equation 4.3 yields

$$\frac{K}{p_1} \frac{4\left(\arcsin\left(\frac{1}{p_1}\sqrt{p_1 p_0}\right)\right) p_0 + 6\sqrt{(-p_0\,(-p_1 + p_0))} + 2\left(\arcsin\left(\frac{1}{p_1}\sqrt{p_1 p_0}\right)\right) p_1 + 2\pi p_0 + \pi p_1}{p_0}$$

□

Lemma 2 *If p_0 is the power of the signal received by node n from node m at time t_i, p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$, and $p_0 \geq p_1$, then the area of the locus of the possible locations of node m at time $t_i + 2\Delta t$ is*

$$A \text{ of limaçon} = 2K\pi \frac{2p_0 + p_1}{p_1 p_0} \tag{4.5}$$

Proof: The proof is basically the same as above changing the bounds of the integration to match the limaçon corresponding to Figure 14(a).

$$\begin{aligned} \text{A of limaçon} &= \frac{1}{2}\int_0^{2\pi} (2d_1 - 2d_0\sin\theta)^2\, d\theta \\ &= 4d_1^2\pi + 2d_0^2\pi \\ &= 2K\pi\frac{2p_0 + p_1}{p_1 p_0} \end{aligned}$$

□

Suppose the minimum power at which node n can reliably read a packet is p_m. Then the maximum distance the node sending the signal can be from n is

$$D = \sqrt{\frac{K}{p_m}} \tag{4.6}$$

Assume the reception area for node n to receive a signal from node m is a circle centered at n with radius D. The intersection of the interior of this circle with the interior of the limaçon is where node m could be at time t_2 if node n were able to receive an adequate signal from node m.

Lemma 3 *Given a limaçon with center at $r = d_0, \theta = 3\pi/2$:*

$$r = 2d_1 - 2d_0\sin\theta$$

and a circle with center at $r = d_0, \theta = 3\pi/2$ and diameter D:

$$r^2 + 2d_0 r\sin\theta + d_0^2 = D^2$$

if $\frac{1}{4}\frac{4d_1^2+d_0^2-D^2}{d_1 d_0} \in [-1, +1]$ then the circle and limaçon intersect, otherwise they do not intersect.

Proof: Solve the simultaneous equations:

$$\left.\begin{aligned} r &= 2d_1 - 2d_0 \sin\theta \\ r^2 + 2d_0 r \sin\theta + d_0^2 &= D^2 \end{aligned}\right\}$$

The solution of the simultaneous equations is:

$$\begin{aligned} \theta &= \arcsin\left(\frac{4d_1^2 + d_0^2 - D^2}{4d_1 d_0}\right) \\ r &= -\frac{d_0^2 - D^2}{2d_1} \end{aligned}$$

When the argument of the arcsin function is not in $[-1, +1]$ the arcsin is not a real number indicating that the two shapes do not intersect. □

Lemma 4 *If p_0 is the power of the signal received by node n from node m at time t_i,*
p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$,
p_m is the minimum power level that node n can receive,
$p_0 < p_1$, and $\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}} \in [-1, +1]$ then the area of the locus of the possible locations of node m at time $t_i + 2\Delta t$ that is inside node n's receiving area is

$$\begin{aligned} &\frac{-K}{p_1p_0p_m}\big(- 4\beta p_0p_m - 8\sqrt{p_0}\sqrt{p_1}\,(\cos\beta)\,p_m + 2\,(\cos\beta\sin\beta)\,p_1p_m - 2\beta p_1p_m + 4\gamma p_0p_m \\ &+ 8\sqrt{p_0}\sqrt{p_1}\,(\cos\gamma)\,p_m - 2\,(\cos\gamma\sin\gamma)\,p_1p_m + 2\gamma p_1p_m - \sqrt{p_m}\sqrt{p_0}\,(\sin\alpha)\,p_1 \\ &- p_1p_0\pi + p_1p_0\alpha\big) \end{aligned}$$

where $\beta = \arcsin\sqrt{\frac{p_0}{p_1}}$

$$\gamma = \arcsin\left(\tfrac{1}{4}\tfrac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}}\right)$$

$$\alpha = \arccos\left(\tfrac{1}{8}\tfrac{-p_1p_m^2 + 2p_1p_0p_m - p_1p_0^2 + 4p_0p_m^2 + 4p_0^2p_m}{\left(\sqrt{p_0}\right)^3\left(\sqrt{p_m}\right)^3}\right)$$

Proof: The receiving area for node n is assumed to be a circle with center at $(0, -d_0)$ and radius D where D is defined in Equation 4.6. The equation for the circle in polar coordinates is

$$r^2 + 2r\,(\sin\theta)\,d_0 + d_0^2 = D^2 \tag{4.7}$$

The intersection of the circle and the limaçon is determined from the simultaneous solution of Equation 4.7 and Equation 4.4

$$\left.\begin{aligned} r &= 2d_1 - 2d_0 \sin\theta \\ r^2 + 2d_0 r \sin\theta + d_0^2 &= D^2 \end{aligned}\right\}$$

The solution of the simultaneous equations is:

$$\begin{aligned} \theta &= \arcsin\left(\frac{1}{4}\frac{4d_1^2 + d_0^2 - D^2}{d_1 d_0}\right), \\ r &= -\frac{1}{2}\frac{d_0^2 - D^2}{d_1} \end{aligned}$$

Again, we put the expressions in terms of p_0 and p_1 and simplify:

$$\begin{aligned} \theta &= \arcsin\left(\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{(p_0p_1)}}\right), \\ r &= -\frac{1}{2}K\frac{p_m - p_0}{p_0p_m}\frac{p_1}{\sqrt{(Kp_1)}} \\ &= -\frac{1}{2}K\frac{p_m - p_0}{p_0p_m}\frac{p_1}{\sqrt{(Kp_1)}} \end{aligned}$$

We know from Lemma 3 if the argument of the arcsin is in the range $[-1, +1]$ then the circle and limaçon intersect. This is guaranteed from the assertions of the lemma.

Let P_2 be the intersection point which occurs in either quadrant I or quadrant IV. Due to the symmetry there is also an intersection point in quadrant II or III.

Since $p_0 < p_1$ the limaçon has a keyhole which begins when $r = 0$. Substituting 0 for r in the equation for limaçon (Equation 4.4) we have

$$0 = 2d_1 - 2d_0 \sin\beta$$

Solving for β:

$$d_1/d_0 = \sin\beta$$

$$\beta = \arcsin(d_1/d_0) = \arcsin\sqrt{\tfrac{p_0}{p_1}}$$

Thus the keyhole begins at polar coordinate (0, β).

Now consider the triangle $\triangle NP_0P_2$, the triangle shown in Figure 15. We know

$$NP_0 = d_0 = \sqrt{\frac{K}{p_0}},$$

$$P_0P_2 = r = -\frac{1}{2}\frac{d_0^2 - D^2}{d_1}, \text{ and}$$

$$NP_2 = D = \sqrt{\frac{K}{p_m}}$$

The measure of $\angle P_0NP_2$ can be found using the Law of Cosines: Let $m\angle P_0NP_2 = \alpha$

$$r^2 = d_0^2 + D^2 - 2d_0D\cos\alpha$$

Solving for α yields:

$2d_0D\cos\alpha = d_0^2 + D^2 - r^2$

$\cos\alpha = \frac{d_0^2+D^2-r^2}{2d_0D}$

$\alpha = \arccos\frac{d_0^2+D^2-r^2}{2d_0D}$

Substituting the expressions using the power levels for the distances we get

$$\alpha = \arccos\left(\frac{1}{8}\frac{-p_1p_m^2 + 2p_1p_0p_m - p_1p_0^2 + 4p_0p_m^2 + 4p_0^2p_m}{(\sqrt{p_0})^3(\sqrt{p_m})^3}\right)$$

Now we can find the area of $\triangle NP_0P_2$:

$$\begin{aligned} A \text{ of } \Delta NP_0P_2 &= \frac{1}{2}Dh \\ &= \frac{1}{2}Dd_0\sin\alpha \end{aligned}$$

so area of triangle $= \frac{1}{2}Dd_0\sin\alpha = \frac{1}{2}Dd_0\sin\left(\arccos\left(\frac{1}{8}\frac{-p_1p_m^2+2p_1p_0p_m-p_1p_0^2+4p_0p_m^2+4p_0^2p_m}{\left(\sqrt{p_0}\right)^3\left(\sqrt{p_m}\right)^3}\right)\right)$

$= \frac{1}{16}\frac{1}{p_m^2}\frac{K}{p_0^2}(p_m - p_0)\sqrt{(-16p_m^2p_0^2 - p_1^2p_m^2 + 8p_m^2p_1p_0 + 8p_mp_1p_0^2 + 2p_mp_1^2p_0 - p_1^2p_0^2)}$

The area of the portion of the limaçon that is inside the receiving circle consists of six parts, three parts in quadrants I and IV and congruent corresponding parts in quadrants II and III. Figure 15 shows the parts in quadrants I and IV: the area bounded by the limaçon curve and the segment P_0P_2, the area of $\triangle NP_0P_2$, and the area of the sector of the circle.

Area of sector of circle $= \frac{(\pi-\alpha)}{\pi}\frac{1}{2}\pi D^2 = \frac{(\pi-\alpha)}{2}D^2 = \frac{(\pi-\alpha)}{2}\frac{K}{p_m}$

Let γ be the angle at start of limaçon piece: $\gamma = \arcsin\left(\frac{1}{4}\frac{4p_0m+p_1m-p_1p_0}{m\sqrt{p_0p_1}}\right)$

Let β be the angle at the start of keyhole: $\beta = \arcsin\sqrt{\frac{p_0}{p_1}}$

Area of intersection (when $p_1 > p_0$) = 2 limaçon pieces + 2 triangles + 2 sectors

$$= \left(2 * \frac{1}{2}\int_{\gamma}^{\beta} \left(2d_1 - 2d_0 \sin\theta\right)^2 d\theta\right) + \left(2 * \frac{1}{2} D d_0 \sin\alpha\right) + \left(\pi - \alpha\right) D^2$$

After simplification and substitution we find the area of intersection to be

$$\begin{aligned} = \quad & \frac{-K}{p_1 p_0 p_m}(- 4\beta p_0 p_m - 8\sqrt{p_0}\sqrt{p_1}\left(\cos\beta\right) p_m + 2\left(\cos\beta \sin\beta\right) p_1 p_m - 2\beta p_1 p_m + 4\gamma p_0 p_m \\ & + 8\sqrt{p_0}\sqrt{p_1}\left(\cos\gamma\right) p_m - 2\left(\cos\gamma \sin\gamma\right) p_1 p_m + 2\gamma p_1 p_m - \sqrt{p_m}\sqrt{p_0}\left(\sin\alpha\right) p_1 \\ & - p_1 p_0 \pi + p_1 p_0 \alpha) \end{aligned}$$

$$\text{where } \beta = \arcsin\sqrt{\frac{p_0}{p_1}}$$

$$\gamma = \arcsin\left(\frac{1}{4}\frac{4p_0 p_m + p_1 p_m - p_1 p_0}{p_m\sqrt{p_0 p_1}}\right)$$

$$\alpha = \arccos\left(\frac{1}{8}\frac{-p_1 p_m^2 + 2p_1 p_0 p_m - p_1 p_0^2 + 4p_0 p_m^2 + 4p_0^2 p_m}{\left(\sqrt{p_0}\right)^3\left(\sqrt{p_m}\right)^3}\right)$$

□

Now perform the same analysis for the dimple case:

Lemma 5 *If p_0 is the power of the signal received by node n from node m at time t_i, p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$, p_m is the minimum power level that node n can receive, $p_1 \leq p_0$, and $\frac{1}{4}\frac{4p_0 p_m + p_1 p_m - p_1 p_0}{p_m\sqrt{p_0 p_1}} \in [-1, +1]$, then the area of the locus of the possible locations of node m at time $t_i + 2\Delta t$ that is inside node n's receiving area is*

$$\begin{aligned} & \frac{K}{p_1 p_0 p_m}(2\pi p_0 p_m + \pi p_1 p_m - 4\gamma p_0 p_m - 8\sqrt{p_0}\sqrt{p_1}\left(\cos\gamma\right) p_m + 2\left(\cos\gamma \sin\gamma\right) p_1 p_m - 2\gamma p_1 p_m \\ & \quad + \sqrt{p_m}\sqrt{p_0}\left(\sin\alpha\right) p_1 + p_1 p_0 \pi - p_1 p_0 \alpha) \end{aligned}$$

$$\textit{where } \gamma = \arcsin\left(\frac{1}{4}\frac{4p_0 p_m + p_1 p_m - p_1 p_0}{p_m\sqrt{p_0 p_1}}\right)$$

$$\alpha = \arccos\left(\frac{1}{8}\frac{-p_1 p_m^2 + 2p_1 p_0 p_m - p_1 p_0^2 + 4p_0 p_m^2 + 4p_0^2 p_m}{\left(\sqrt{p_0}\right)^3\left(\sqrt{p_m}\right)^3}\right)$$

Proof: The only difference in the calculations is in the limits of integration for the area of the two pieces of the limaçon that are inside the circle. Previously the angle went from γ to β, now it goes all the way from γ to $\pi/2$.

$$
\begin{aligned}
&= \left(2 * \frac{1}{2}\int_{\gamma}^{\pi/2} \left(2d_1 - 2d_0\sin\theta\right)^2 d\theta\right) + \left(2 * \frac{1}{2} D d_0 \sin\alpha\right) + (\pi - \alpha) D^2 \\
&= 2d_1^2\pi + d_0^2\pi - 4d_1^2\gamma - 8d_0d_1\cos\gamma + 2d_0^2\cos\gamma\sin\gamma - 2d_0^2\gamma + Dd_0\sin\alpha + (\pi-\alpha)D^2 \\
&= \frac{1}{p_1p_0p_m}(2K\pi p_0p_m + K\pi p_1p_m - 4K\gamma p_0p_m - 8K\sqrt{(p_0)}\sqrt{(p_1)}\left(\cos\gamma\right)p_m \\
&\quad + 2K\left(\cos\gamma\sin\gamma\right)p_1p_m - 2K\gamma p_1p_m + K\sqrt{(p_m)}\sqrt{(p_0)}\left(\sin\alpha\right)p_1 + Kp_1p_0\pi - Kp_1p_0\alpha) \\
&= \frac{K}{p_1p_0p_m}(2\pi p_0p_m + \pi p_1p_m - 4\gamma p_0p_m - 8\sqrt{p_0}\sqrt{p_1}\left(\cos\gamma\right)p_m + 2\left(\cos\gamma\sin\gamma\right)p_1p_m \\
&\qquad -2\gamma p_1p_m + \sqrt{p_m}\sqrt{p_0}\left(\sin\alpha\right)p_1 + p_1p_0\pi - p_1p_0\alpha) \\
&\quad \text{where } \gamma = \arcsin\left(\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}}\right) \\
&\qquad \alpha = \arccos\left(\frac{1}{8}\frac{-p_1p_m^2 + 2p_1p_0p_m - p_1p_0^2 + 4p_0p_m^2 + 4p_0^2p_m}{\left(\sqrt{p_0}\right)^3\left(\sqrt{p_m}\right)^3}\right)
\end{aligned}
$$

□

Finally we can consider the ratio of the area of the locus of locations of node m at time $t_i + 2\Delta t$ that are inside node n's receiving range to that of the area of node m's entire locus to be the probability that node n will be able to receive a signal from node m at time $t_i + 2\Delta t$ given that it received signals at t_i and $t_i + \Delta t$.

Theorem 1 *If p_0 is the power of the signal received by node n from node m at time t_i,*
p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$,
p_m is the minimum power level that node n can receive,
$p_0 < p_1$, and $\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}} \in [-1, +1]$, then the probability that link (m, n) will exist at time $t_i + 2\Delta t$ is

$$
\begin{aligned}
&-\Big(- 4\beta p_0p_m - 8\sqrt{p_0}\sqrt{p_1}\left(\cos\beta\right)p_m + 2\left(\cos\beta\sin\beta\right)p_1p_m - 2\beta p_1p_m \\
&\quad + 4\gamma p_0p_m + 8\sqrt{p_0}\sqrt{p_1}\left(\cos\gamma\right)p_m - 2\left(\cos\gamma\sin\gamma\right)p_1p_m + 2\gamma p_1p_m \\
&\quad - \sqrt{p_m}\sqrt{p_0}\left(\sin\alpha\right)p_1 - p_1p_0\pi + p_1p_0\alpha\Big) \\
&/\Big(p_m\left(4\beta p_0 + 8\sqrt{(p_1p_0)}\cos\beta - 2\left(\cos\beta\sin\beta\right)p_1 + 2\beta p_1 + 2\pi p_0 + \pi p_1\right)\Big)
\end{aligned}
$$

$$
\begin{aligned}
&\textit{where } \beta = \arcsin\sqrt{\frac{p_0}{p_1}} \\
&\gamma = \arcsin\left(\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}}\right) \\
&\alpha = \arccos\left(\frac{1}{8}\frac{-p_1p_m^2 + 2p_1p_0p_m - p_1p_0^2 + 4p_0p_m^2 + 4p_0^2p_m}{\left(\sqrt{p_0}\right)^3\left(\sqrt{p_m}\right)^3}\right)
\end{aligned}
$$

Proof: The probability that link (m, n) will exist at time $t_i + 2\Delta t$ corresponds to the ratio of the area of the locus of locations of node m at time $t_i + 2\Delta t$ that are inside node n's receiving range to that of the area of node m's entire locus so we divide the result from Lemma 4 by the result from Lemma 1 giving the indicated expression. □

Theorem 2 *If p_0 is the power of the signal received by node n from node m at time t_i,*
p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$,
p_m is the minimum power level that node n can receive,
$p_0 < p_1$, and $\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}} \notin [-1, +1]$, then the probability that link (m, n) will exist at time $t_i + 2\Delta t$ is 1.0.

Proof: If the fraction is not between -1 and +1 inclusive, then the limaçon representing node m's locus does not intersect node n's receiving circle. Since node n did receive signals p_0 and p_1 the corresponding points must be located inside the receiving range, thus the entire limaçon must be inside the receiving range. □.

Likewise we can examine the case when $p_1 \leq p_0$:

Theorem 3 *If p_0 is the power of the signal received by node n from node m at time t_i,*
p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$,
p_m is the minimum power level that node n can receive,
$p_1 \leq p_0$, and $\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}} \in [-1, +1]$, then the probability that link (m, n) will exist at time $t_i + 2\Delta t$ is

$$
\begin{aligned}
= \quad & \frac{1}{2p_m\pi\left(2p_0 + p_1\right)}(2\pi p_0p_m + \pi p_1p_m - 4\gamma p_0p_m - 8\sqrt{p_0}\sqrt{p_1}\left(\cos\gamma\right)p_m \\
& + 2\left(\cos\gamma\sin\gamma\right)p_1p_m - 2\gamma p_1p_m + \sqrt{p_m}\sqrt{p_0}\left(\sin\alpha\right)p_1 + p_1p_0\pi - p_1p_0\alpha) \\
& \textit{where } \gamma = \arcsin\left(\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}}\right)
\end{aligned}
$$

$$\alpha = \arccos\left(\frac{1}{8}\frac{-p_1 p_m^2 + 2p_1 p_0 p_m - p_1 p_0^2 + 4p_0 p_m^2 + 4p_0^2 p_m}{\left(\sqrt{p_0}\right)^3\left(\sqrt{p_m}\right)^3}\right)$$

Proof: Using the same argument as in Theorem 1 we divide the result from Lemma 5 by the result from Lemma 2 to achieve the indicated expression. □

Theorem 4 *If p_0 is the power of the signal received by node n from node m at time t_i,*
p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$,
p_m is the minimum power level that node n can receive,
$p_1 \leq p_0$, and $\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}} \notin [-1, +1]$, and $p_m < \frac{p_0p_1\sqrt{p_0p_1}}{4p_0\sqrt{p_0p_1} - 4p_0p_1 + p_1\sqrt{p_0p_1}}$ then the probability that link (m, n) will exist at time $t_i + 2\Delta t$ is 1.0

Proof: There are two cases when the dimpled limaçon does not intersect the transmission circle. When $p_m < \frac{p_0p_1\sqrt{p_0p_1}}{4p_0\sqrt{p_0p_1} - 4p_0p_1 + p_1\sqrt{p_0p_1}}$ the limaϙn is entirely inside the circle so the probability is 1.00. □

Theorem 5 *If p_0 is the power of the signal received by node n from node m at time t_i,*
p_1 is the power of the signal received by node n from node m at time $t_i + \Delta t$,
p_m is the minimum power level that node n can receive,
$p_1 \leq p_0$, $p_m > \frac{p_0p_1\sqrt{p_0p_1}}{4p_0\sqrt{p_0p_1} - 4p_0p_1 + p_1\sqrt{p_0p_1}}$ and $\frac{1}{4}\frac{4p_0p_m + p_1p_m - p_1p_0}{p_m\sqrt{p_0p_1}} \notin [-1, +1]$, then the probability that link (m, n) will exist at time $t_i + 2\Delta t$ is $\frac{p_0p_1}{p_m(4p_0 + 2p_1)}$

Proof: The second case when the dimpled limaϙn does not intersect the circle is when the circle is entirely inside the limaϙn in which case the probability is the ratio of the area of the circle to the entire limaϙn. Area of the circle $= \pi D^2$ and area of the limaϙn $= 4\pi {d_1}^2 + 2\pi {d_0}^2$. Dividing and substituting yields $\frac{p_0p_1}{p_m(4p_0 + 2p_1)}$ □

Although the calculations are somewhat complex, since they only rely on p_0, p_1, and p_m a table of values for various values of p_0 and p_1 given the device's receive threshold sensitivity p_m can be hard-coded into the device so that the determination of the link reliability is simply a table lookup operation requiring constant time.

In this chapter we have derived the formulas used by RoMR to calculate the weights associated with the links in the network topology. These weights reflect the probability that the link will be available in the next time frame. We shall see in Section 5.1.2 that the weights of the links are used in the computation of the multicast trees.

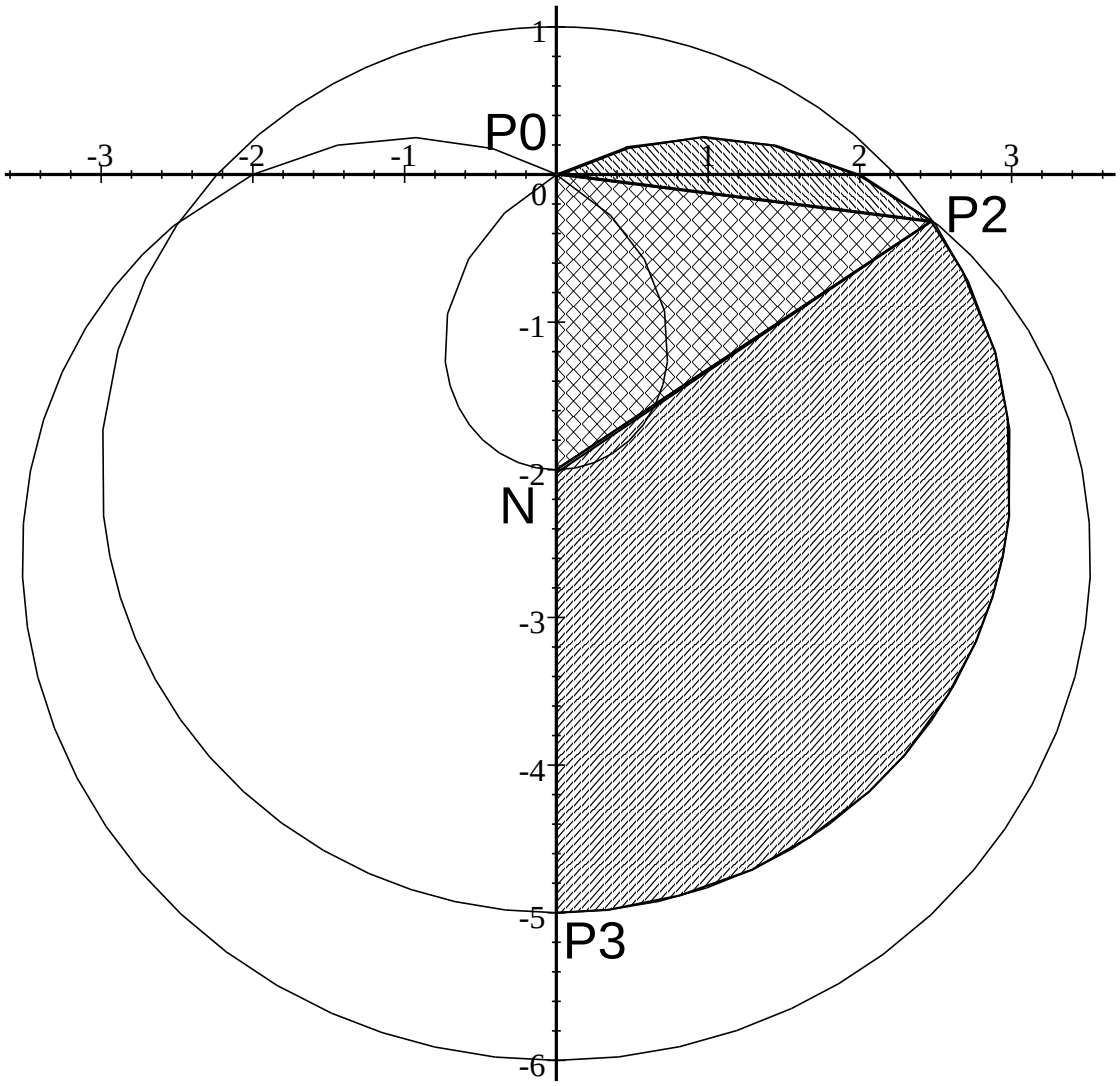


Figure 15: Intersection of Circle and Limaçon

5.0 ROMR MULTICAST AND TREE MANAGEMENT ALGORITHMS

In this chapter we will examine the more important algorithms used to perform the tasks associated with each of the states of nodes under RoMR.

5.1 MULTICAST MANAGER ALGORITHMS

5.1.1 Procedure CheckTrees

When a multicast manager receives a message from the unicast protocol that the topology has changed it needs to determine how many of the trees specified in the current tree set are still connected. If the number of connected trees has fallen below a threshold the tree set needs to be recalculated and distributed. Pseudocode for *CheckTrees* is listed in Figure 16.

5.1.2 Function MakeTrees

A node in the Multicast Manager (MM) state is responsible for creating the set of trees to be used in RoMR when the need to do so is determined by *CheckTrees*. Three parameters are associated with a multicast session when it starts: the Reuse Factor (RF), the Weight Threshold (WT), and a minimum number of trees (MT). In Section 3.2.3 these parameters were referred to as ρ, τ, and μ respectively. Combined, these values reflect the desired reliability level to be achieved and can be specified by the application. If the application does not specify values for these parameters, RoMR will use default values.

The Reuse Factor (RF) is the percentage of links in a tree that are eligible to be used in the next tree. The RF value must be less than 1 (100%). A value of 0 would result

Procedure CheckTrees()

for each group g this node manages **do**

  $nIntactTrees$ = number of intact trees in current tree set

  if $nIntactTrees \leq kPacketsNeeded$ **then**

    $treeSetID \leftarrow NextTreeSetID(treeSetID, g)$

    $nTrees \leftarrow MakeTrees(treeSetID, g)$

    $kPacketsNeeded \leftarrow DetermineKPacketsNeeded(treeSetID)$

    $distributeTrees(treeSetID)$

  else

    $minutes \leftarrow$ minutes elapsed since last tree message sent

    if $minutes > TREE\_INTERVAL$ **then**

      send msg to continue to use tree set with $treeSetID$

      reset tree interval timer

    end if

  end if

end for

schedule next time to check trees

Figure 16: Procedure CheckTrees()

in disjoint trees, increasing the chance that a node would receive packets in an extremely dynamic network, but at the same time also increasing the use of network resources.

The weight threshold (WT) and minimum number of trees (MT), similar to a Quality of Service (QoS) parameter, are specified by the node initiating the multicast session. The tree module will attempt to make a sufficient number of trees so that the probability of at least MT trees existing in the next time frame is above WT.

The steps in creating a tree set are outlined in Figure 17. In Figure 18, node A is the sender and nodes D, E, and G are the receiver nodes. The diagram shows the first tree created as a result of the modified SPH algorithm. The order in which the nodes are added to the first tree is A, D, E, G. This example assumes the reuse factor $RF = 0.5$ so the branch from D to C, with weight 0.7, and the branch from G to E are eliminated from consideration in the construction of the second tree. The modified SPH algorithm is once again followed to produce the second tree.

5.1.3 Function MakeSingleTree

The MM makes a single tree in the process of creating the current tree set for the group. The basic part of the algorithm stems from the shortest path heuristic (SPH) described in Section 2.4.2.1 with a modification. If two paths from a given member to the existing tree have the same hop count then the path that connects to the node in the tree with the highest degree relative to the tree is selected. If the degrees of the nodes in the current tree are the same and if one of the tree nodes is the sender the path to the sender is selected. The same proof to show that SPH produces a tree having at most twice the number of links as the optimal tree [51] applies to the modified version. The steps of the algorithm are listed in Figure 19. The choice of SPH to create the trees addresses the concern that RoMR should use network resources efficiently by making trees that may be closer to optimal than simply using the union of shortest paths. The modification to attach paths to nodes having a higher degree in the current tree prevents the branches in the tree from becoming overly long.

The computation of the best path from a single node to all of the other nodes in the network is based on Dijkstra's shortest path algorithm which has $O(n^2)$ complexity so the

Function MakeTrees($treeSetID, groupID$)

Returns: a set of trees

$i \leftarrow 0, \quad n \leftarrow \infty, \quad done \leftarrow false, \quad treeSet \leftarrow \phi$

$links \leftarrow$ set of known links in the network with corresponding link weights

$members \leftarrow$ set of reachable members of group $groupID$

repeat

  $T \leftarrow MakeSingleTree(sender, members, links)$

  if $T = \phi$ **then**

    $done \leftarrow true$

  else

    $i \leftarrow i + 1$

    $treeSet \leftarrow treeSet \cup T$

    $treeWeight \leftarrow$ minimum weight of edges of T

    $cumulativeWeight \leftarrow ProbAtLeastKAreGood(MT, treeSet)$

    $done \leftarrow (cumulativeWeight \geq WT)\, or\, (i = MAX\_TREES)$

    if not *done* **then**

      $n \leftarrow \lceil (1 - RF) * |T| \rceil$ where $|T|$ is the number of branches in tree T

      delete the n links of T having lowest weights from $links$

      {n will be ≥ 1 so duplicate trees cannot be formed}

    end if

  end if

until *done*

return $treeSet$

Figure 17: Function MakeTrees()

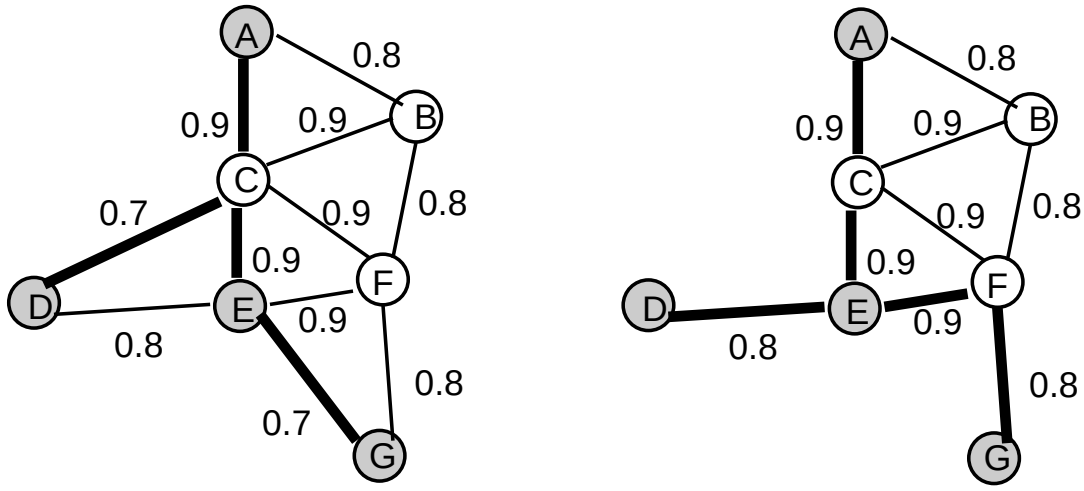


Figure 18: Making Two Multicast Trees

complexity of the *makeSingleTree* as stated is $O(|R|n^2)$ where $|R|$ is the number of reachable receivers in the group.

An alternative method was considered to construct the trees with the goal of producing the most reliable tree. The above technique was modified using the greatest product of the link weights as opposed to the least sum of distances. In dense networks this created two problems. The first problem was the increase in the length of time a packet took to reach a receiver. The second problem was caused by the long paths in the local area that resulted in a large number of packet collisions due to the fact that the broadcast medium is shared by all nodes in the local area. Figure 20 illustrates these problems. Node A is the sender and node F is the receiver. The values on the links represent the weights of the links. The shortest path between A and F is clearly the one hop path from A to F with the product of the weights equal to 0.8 whereas the most reliable path based on the product of the weights would be A, B, C, D, E, F using 5 hops with a reliability product of $0.99^5 = 0.95$.

An enhancement was added after initial testing of RoMR. When two paths have the same length to a node in the current tree, choose the node in the current tree that is closest to the sender. This has the advantage of decreasing the overall path length to the member.

Function MakeSingleTree($sender, members, links$)

Returns: a set of edges making up the tree

{$SP(m,n)$ returns the shortest path from m to n}

$nodes \leftarrow \{sender\}, \quad edges \leftarrow \phi$

while $members \neq \phi$ **do**

  $pathLength \leftarrow \infty$

  for each $m \in members$ **do**

    for each $n \in nodes$ **do**

      if $length(SP(m,n)) < pathLength$ **then**

        $pathLength \leftarrow length(SP(m,n))$

        $treeNode = n$

      else if $length(SP(m,n)) = pathLength$ **then**

        if $degree(n) > degree(treeNode)$ **then**

          $treeNode \leftarrow n$

        else if $n = sender$ **then**

          $treeNode \leftarrow n$

        end if

      end if

    end for

  end for

  $edges \leftarrow edges \cup$ edges of $SP(m, treeNode)$

  $nodes \leftarrow nodes \cup$ nodes of $SP(m, treeNode)$

  $members \leftarrow members - \{m\}$

end while

return $edges$

Figure 19: Function MakeSingleTree()

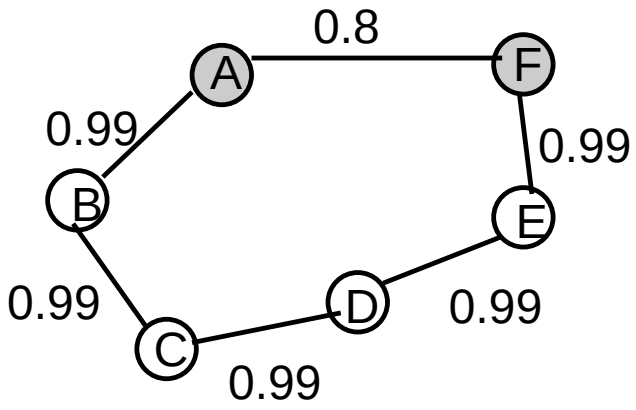


Figure 20: Problem Using Product of Link Weights

5.1.4 Function ProbAtLeastKAreGood

In the procedure *MakeTrees* the cumulative weight of all the trees currently in the tree set was computed with a call to *ProbAtLeastKAreGood*. Recall from Section 4.1 that a weight of a link reflects the probability that the link would continue to exist from time $t_i + \Delta t$ until time $t_i + 2\Delta t$. If the existence of all links in a tree continuing to exist during the same interval were considered to be independent events then the probability of tree T remaining intact until time $t_i + 2\Delta t$ would be

$$\prod_{link \in T} weight(link)$$

However, the duration of the links are *not* independent. Suppose nodes A, B, and C are linearly arranged with B being between A and C. As B moves slowly toward A the weight of AB increases while the weight of BC decreases. Simulation confirms that the product is a vast underestimate of the probability. As an approximation to the probability that all of the links in the tree will survive until the next time interval we selected the minimum weight of all of the links in the tree to represent the probability that the tree would remain connected until time $t_i + 2\Delta t$.

If every tree in a tree set has the same probability p associated with it, then the probability of exactly k out of the n trees in the tree set being connected at time $t_i + 2\Delta t$ is

$$\binom{n}{k} p^k q^{n-k} \text{ where } q = 1 - p$$

and the probability of k or more of the n trees being connected at time $t_i + 2\Delta t$ is

$$\sum_{i=k}^{n} \binom{n}{i} p^i q^{n-i}$$

$$= \sum_{i=0}^{n-k} \frac{n!}{(n-i)!i!} p^{n-i} q^i$$

Changing this to a form in which each of the p_i values may differ, the probability that at least k trees will remain intact is

$$\sum_{j=0}^{\binom{n}{k-1}} \prod_{i=0}^{n-1} \left(q_i * j_i + p_i * \overline{j_i} \right)$$

where j_i is the i^{th} bit of j

and $\overline{j_i} = 1 - j_i$

Clearly we can see that for a fixed value of k, the larger the value of n the larger the sum will become while being bounded from above by 1.00.

The above formula can be stated as a recursive routine as shown in Figure 21. The complexity of the computation is exponential in the general case due to the recursion, but in the actual implementation the number of trees that will be created is bound by a small constant MAX_TREES thus eliminating exponential execution time concerns.

Function ProbAtLeastKAreGood($k, treeSet$)

Returns: a floating point value representing the probability that k out of the n trees in the $treeSet$ will be intact at the end of the next time interval

$n \leftarrow$ number of trees in $treeSet$

make array p to be the weights of the trees in $treeSet$

return f$(k, n, 1.0, 0, p)$

Function f(k, n, val, i, p)

if $n = 0$ **then**

  if $k \leq 0$ **then**

    return val

  else

    return 0

  end if

else

  return f$(k, n-1, val * (1 - p[i]), i+1, p) + f(k-1, n-1, val * p[i], i+1, p)$

end if

Figure 21: Function ProbAtLeastKAreGood

5.1.5 Procedure DetermineKPacketsNeeded

Recall from Section 3.2.1 RoMR attempts to make the length of time between the creations of tree sets through the use of forward error correction codes as large as possible while meeting the primary goal. Every k packets that the application sends will be encoded as n unique packets by RoMR at the routing layer. The i^{th} packet of the group of n packets will be sent to receivers over the i^{th} tree. The receiver only needs to receive *any* k of the n packets that were sent in order to recreate the original k packets. The encoding is performed by a node in the sender state and the decoding by a group member in the receiving state. It is the responsibility of the the node in the multicast manager (MM) state for the group to determine the value of k.

Every time a new tree set is calculated the value of k is recomputed based on the previous value of k and the length of time that elapsed since the creation of the previous tree set. The unicast protocol specifies an interval of time, TC_INTERVAL, between sequential topology control messages. In RoMR if the elapsed time since the previous tree set was created is greater than or equal to 4 times the TC_INTERVAL then the network is considered to be stable and will increase the the value of k if possible to reduce the overhead due to redundancy. The unicast protocol also specifies the length of time it will maintain information about a neighbor in the event it does not receive any update information from the neighbor. This time interval, NEIGHB_HOLD_TIME, is also used in the calculation of k. If the elapsed time since the creation of the previous tree set is less than the NEIGHB_HOLD_TIME then the trees were being made too often, so decrease the value of k if possible, so that fewer of the trees in the tree set need to remain intact before recomputing the entire set of trees.

5.2 SENDER AND RECEIVER ALGORITHMS

5.2.1 Forward Error Correction Codes

The techniques employed by RoMR to reduce the amount of packets travelling through the network for redundancy purposes is based on Forward Error Correction (FEC) techniques developed for use in telephone systems to recover from noise in the lines so that switching

signals can still be understood despite the loss of some of the pieces of the signal.

5.2.1.1 Erasure Codes and Galois Fields One class of forward error correction codes is based on erasure coding methods. This refers to a coding technique wherein the original data may be recovered even if some of the data that was sent did not arrive at the destination (i.e. it had been erased). Rabin [41] discusses a method based on a finite field, mentioning the fact that Galois Fields could be used in order to avoid increasing the number of bits in the encoded packets. Rizzo [44] provides details in using the Galois Fields.

Galois Fields, GF, are also called *prime fields*. $GF(p)$ is a set of integers from 0 to $p-1$ closed under + and * with p being a prime number. $GF(p^r)$ is defined to be an *extension field* where p is prime and $r > 1$. $GF(p^r)$ is a set of integers from 0 to $p^r - 1$ closed under + and *. In order to do the calculations efficiently on a computer let $p = 2$ and define + and * using bit operations. Rizzo recommends a value of $r = 8$ since two 256×256 tables for the + and * operations are small enough to be hard-coded avoiding the calculation of the values when needed.

$a + b$ is defined to be a XOR b. Let the values of a and b be integers in the range [0,255] a and b in order to represent a byte of data each.

The definition of * is based on the concept of a *primitive polynomial*, a polynomial of degree r with coefficients in $GF(p)$ that is not divisible by any polynomial of smaller degree having coefficients in $GF(p)$. $x^8 + x^4 + x^3 + x^2 + 1$ is an appropriate primitive polynomial for $p = 2, r = 8$. This polynomial can be expressed as the bit string 100011101 where the n^{th} bit is the coefficient of the n^{th} degree term. Define $a * b$ to be the multiplication of the polynomials corresponding to a and b modulo the primitive polynomial with the coefficients expressed modulo p.

The following is an example of multiplying two polynomials:

Let $a = 100$ and $b = 13$ be integer representations of the two polynomials to multiply.

$a = 1100100_2$ which corresponds to $x^6 + x^5 + x^2$

$b = 1101_2$ which corresponds to $x^3 + x^2 + 1$

$$
\begin{aligned}
a * b &= ((x^6 + x^5 + x^2)(x^3 + x^2 + 1)) \bmod (x^8 + x^4 + x^3 + x^2 + 1) \\
&= (x^9 + (2 \bmod 2)x^8 + x^7 + x^6 + (2 \bmod 2)x^5 + x^4 + x^2) \bmod (x^8 + x^4 + x^3 + x^2 + 1) \\
&= (x^9 + x^7 + x^6 + x^4 + x^2) \bmod (x^8 + x^4 + x^3 + x^2 + 1) \\
&= (x^9 + x^7 + x^6 + x^4 + x^2) \bmod (x^8 + x^4 + x^3 + x^2 + 1) \\
&= x^7 + x^6 + (-1 \bmod 2)x^5 + (-1 \bmod 2)x^3 + x^2 + (-1 \bmod 2)x \\
&= x^7 + x^6 + x^5 + x^3 + x^2 + x \\
&= 11101110_2 \\
&= 238
\end{aligned}
$$

A property of all prime fields and extension fields simplifies the process of computing the product of two values. There exists an element $\alpha \in GF(p^r)$ such that successive powers of α yield all of the nonzero elements in $GF(p^r)$. Therefore for each nonzero $n \in GF(p^r)$, there exists an integer $i \in [0, p^r - 2]$ such that $\alpha^i = n$. Define $\log(\alpha^i) = i$. Now multiplication of x and y can be defined as

$$x * y \equiv \alpha^{(\log(x)+\log(y)) mod (q-1)} \quad \text{where } q = p^r$$

The inverse of an element $x \in GF(p^r)$ is defined as

$$\frac{1}{x} \equiv \alpha^{q-1-\log(x)} \text{ where } q = p^r$$

For $GF(2^8)$ with + and * as defined above, $\alpha = 2$.

Function MakeEncoderMatrix(n, k)

Returns: a $n \times k$ encoder matrix E

{Rows and columns are numbered starting with 0}

{n : number of blocks of encoded data (maximum $n = 2(2^r - 1)$) }

{k : number of blocks of source data (maximum $k = 2^r - 1$) }

Construct a $(n - k) \times k$ matrix A with

$A(i, j) = (\alpha^{n-k+i})^j$ for $i = 0..n - k - 1$ and $j = 0..(k - 1)$

Construct a $k \times k$ matrix B with $B(0, 0) = 1$, $B(0, j) = 0$ for $j = 1..(k - 1)$ and

$B(i, j) = (\alpha^{i-1})^j$ for $i = 1..(k - 1)$ and $j = 0..(k - 1)$

Find the inverse of B, B^{-1}, a $k \times k$ matrix

Calculate $C = A \cdot B^{-1}$, a $(n - k) \times k$ matrix

Construct a $n \times k$ matrix E with the first k rows being the identity matrix and

the last $(n - k)$ rows being matrix C.

return E

Figure 22: Function MakeEncoderMatrix

Function MakeDecoderMatrix(n, k, P)

{P is a $k \times 1$ matrix of the k packets to decode. }

Returns: $k \times k$ decoder matrix

matrix E = MakeEncoderMatrix(n, k)

Form a $k \times k$ matrix, D, from the rows of E that correspond to the k packets in P

return D^{-1}

Figure 23: Function MakeDecoderMatrix

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

*	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	3	1	7	5
3	0	3	6	5	7	4	1	2
4	0	4	3	7	6	2	5	1
5	0	5	1	4	2	7	3	6
6	0	6	7	1	5	3	2	4
7	0	7	5	2	1	6	4	3

Figure 24: Addition and Multiplication Tables for p=2, r=3

x	α^x	$\log(x)$	$1/x$
0	1	-	-
1	2	0	1
2	4	1	5
3	3	3	6
4	6	2	7
5	7	6	2
6	5	4	3
7	1	5	4

Figure 25: Log and Inverse Functions for p=2, r=3

5.2.1.2 Encoder and Decoder Matrices The encoding process takes k packets of source data and produces n unique packets such that the original data can be determined from any k of the n encoded packets. The product of an $n \times k$ encoder matrix E and the $k \times 1$ matrix of original packets yields a $n \times 1$ matrix of encoded packets. Matrix arithmetic operations are based on + and * defined for $GF(2^r)$. Maximum values of n and k are $2(2^r-1)$ and $2^r - 1$ respectively. The algorithm shown in Figure 22 specifies how to construct the encoding matrix E. The decoding matrix depends on which subset of size k of the n packets was received. The steps are outlined in the algorithm given in Figure 23. Note that $\binom{n}{k}$ different decoding matrices are possible.

The following is an example of the encoding/decoding process with $p = 2$ and $r = 3$. The maximum values of n and k are $n_{\max} = 2(2^3 - 1) = 14$, and $k_{\max} = 2^3 - 1 = 7$ respectively. A primitive polynomial for $GF(2^3)$ is $x^3 + x + 1 = 1011_2$ which has an α value of $2 = 10_2$ representing the polynomial $1x + 0$. Successive powers of α are $\alpha^0 = 1$, $\alpha^1 = 2$, $\alpha^2 = 4$, $\alpha^3 = 3$, $\alpha^4 = 6$, $\alpha^5 = 7$, and $\alpha^6 = 5$. The addition and multiplication tables are given in Figure 24 and the powers, log and inverse functions are listed in Figure 25.

Let $n = 5$ and $k = 3$ be the encoding parameters. Following the steps of the algorithm we find matrices A, B, B^{-1}, C, and E:

$$A = \begin{bmatrix} (\alpha^2)^0 & (\alpha^2)^1 & (\alpha^2)^2 \\ (\alpha^3)^0 & (\alpha^3)^1 & (\alpha^3)^2 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 0 \\ (\alpha^0)^0 & (\alpha^0)^1 & (\alpha^0)^2 \\ (\alpha^1)^0 & (\alpha^1)^1 & (\alpha^1)^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix}, \quad B^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 7 & 3 \\ 5 & 6 & 3 \end{bmatrix}$$

$$C = AB^{-1} = \begin{bmatrix} 1 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 4 & 7 & 3 \\ 5 & 6 & 3 \end{bmatrix} = \begin{bmatrix} 4 & 3 & 6 \\ 1 & 1 & 1 \end{bmatrix}$$

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 4 & 3 & 6 \\ 1 & 1 & 1 \end{bmatrix}$$

Suppose each of 3 packets contain 8 values. Each value must be in the range $0..(p^r - 1) = 0..7$

packet 0: 3 4 5 6 7 0 1 2

packet 1: 4 3 6 5 0 7 2 1

packet 2: 5 6 3 4 1 2 7 0

$$E \times \begin{bmatrix} 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 \\ 4 & 3 & 6 & 5 & 0 & 7 & 2 & 1 \\ 5 & 6 & 3 & 4 & 1 & 2 & 7 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 4 & 3 & 6 \\ 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 \\ 4 & 3 & 6 & 5 & 0 & 7 & 2 & 1 \\ 5 & 6 & 3 & 4 & 1 & 2 & 7 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 \\ 4 & 3 & 6 & 5 & 0 & 7 & 2 & 1 \\ 5 & 6 & 3 & 4 & 1 & 2 & 7 & 0 \\ 3 & 1 & 2 & 4 & 7 & 5 & 6 & 0 \\ 2 & 1 & 0 & 7 & 6 & 5 & 4 & 3 \end{bmatrix} \begin{array}{l} \leftarrow \text{encoded pkt 0} \\ \leftarrow \text{encoded pkt 1} \\ \leftarrow \text{encoded pkt 2} \\ \leftarrow \text{encoded pkt 3} \\ \leftarrow \text{encoded pkt 4} \end{array}$$

Now suppose the receiver gets encoded pkt 4, then encoded pkt 3,and lastly encoded pkt 2.

$$D = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 3 & 6 \\ 0 & 0 & 1 \end{bmatrix} \text{ corresponding to rows 4, 3, and 2 of } E$$

$$D^{-1} = \begin{bmatrix} 7 & 4 & 2 \\ 6 & 4 & 3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D^{-1}\times \text{ encoded packets} = \begin{bmatrix} 7 & 4 & 2 \\ 6 & 4 & 3 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 1 & 0 & 7 & 6 & 5 & 4 & 3 \\ 3 & 1 & 2 & 4 & 7 & 5 & 6 & 0 \\ 5 & 6 & 3 & 4 & 1 & 2 & 7 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 \\ 4 & 3 & 6 & 5 & 0 & 7 & 2 & 1 \\ 5 & 6 & 3 & 4 & 1 & 2 & 7 & 0 \end{bmatrix} \text{ the original data packets!}$$

5.2.2 FEC in the Sender State

After the n trees of a tree set have been created the source node creates an $n \times k$ encoding matrix, E, as described in Figure 22. Since this matrix only depends on the values of k and n it can be calculated once and used any time in the future with the same values of k and n. In fact, if k and n are limited to a small number of values, all possible matrices that might be needed could be computed offline and stored in memory.

When an application layer program sends a data packet, RoMR's routing layer component in the source node examines the destination address. If the destination is a node address, RoMR passes the packet on to the unicast protocol with no further processing. If the destination is a group address, the source node will buffer the outgoing packets. (A group address has an IP address between 224.0.0.0 inclusive and 240.0.0.0 exclusive [32].) When the number of packets in the buffer reaches k, it encodes the packets as n unique packets. The group of k packets to encode may be viewed as a $k \times 1$ matrix P. The matrix product of $E \times P$ yields a $n \times 1$ matrix of the n encoded packets. After performing the encoding, a header is added to each packet containing information needed for the routing and decoding processes. Specifically, the header includes the sequence number of the tree set, a value to indicate which tree of the tree set should carry this packet, and the value of k. After the header has been attached then the packets are broadcast to the node's one-hop neighbors.

It may be the case that RoMR's routing component in the source node does not receive k outgoing packets from the application within a timeout period or that the values of k and/or n have changed due to a new set of trees being used. RoMR will simply send these few remaining packets out with $k = 1$ over the existing set of n trees so that no encoding needs

to take place on either the sender's or receiver's end of the path. Thus in highly dynamic networks with rapidly changing sets of trees RoMR degenerates into sending a copy of each packet down each tree with none of the advantages associated with encoding. The encoding scheme produces a n/k-fold increase in network load. For example, when $k = 1$, RoMR is sending n packets for every one packet originally produced by the application. On the other hand, when $k = n$, as in the case of a static network, no extra packets are injected into the network for redundancy purposes.

The algorithm in RoMR's network layer component that receives a data packet addressed to a multicast group from the transport layer is given in Figure 26.

5.2.3 FEC in the Receiver State

When the network layer receives a packet from the MAC layer it examines the destination address. If it is not a group address then the underlying unicast routing protocol routes the packet. If the destination is a group address then RoMR performs a sequence of actions. First, it checks a message cache to see if it is a duplicate packet. If it is a duplicate then the packet is simply ignored. If the packet is not a duplicate, RoMR checks to see if the node that received the packet is a member of the indicated group. If so, RoMR buffers k of the incoming packets belonging to the indicated packet group sent by the source. It then selects or creates the appropriate $k \times k$ decoding matrix, D as described in 5.2.1.2. The buffered encoded packets form a $k \times 1$ matrix, P and the matrix product $D \times P$ yields a $k \times 1$ matrix of the original data packets which are then sent on to the upper layers of the protocol stack. Once k packets of a particular group of packets have been received then all of the other possible $n - k$ packets that are received belonging to the indicated packet group are ignored. Since the first k packets of the group of packets are encoded using the identity matrix if the first k packets to arrive do arrive via one of the first k trees, then no decoding is necessary, slightly speeding up the overall delivery of the packets to the application layer.

RoMR's network layer component's algorithm that deals with the buffering and routing of incoming data packets addressed to a multicast group that are received from the MAC layer is given in Figure 27.

Procedure getMsgFromTransport(n, k, msg)

Purpose: buffers and encodes outgoing messages {k is the number of packets to encode}

{n is the number of trees in the current set of trees}

Put the msg in the outgoing buffer

if the number of messages in the outgoing buffer $\geq k$ **then**

  if $n > 1$ **then**

    E = MakeEncoderMatrix(n, k);

  end if

  while the number of messages in the outgoing buffer $\geq k$ **do**

    if $n > 1$ **then**

      form the k original packets into a matrix P

      compute $P' = E \times P$, producing the n encoded packets

      add the $n - k$ new packets listed in P' to outgoing buffer

    end if

    add the headers and send each of the n packets to the MAC Layer

  end while

end if

Figure 26: FEC in Sender

Procedure getMsgFromMAC(n, msg)

Purpose: buffers and decodes inbound messages { n is the number of trees in the current set of trees}

put the msg in the inbound buffer

if msg in message cache **then**

  discard packet

else

  put message into the cache

  determine the tree set number from the message's header

  if this node is a relay node for any of the trees in the tree set **then**

    send the message to the MAC layer to be forwarded to neighbors

  end if

  if this node is a member of the multicast group **then**

    determine the value of k from the message

    if $k = 1$ **then**

      send msg up to UDP layer

    else

      add the message to the incoming buffer for the group

      while there are k or more packets in the buffer **do**

        arrange the k packets into a matrix P

        P' = MakeDecoderMatrix(n, k, P)$\times P$

        deliver the packets of P' to the UDP layer

        remove the k packets from inbound buffer

      end while

    end if

  end if

end if

Figure 27: Procedure getMsgFromMAC

5.3 ALGORITHMS FOR RELAY NODES

A node in the Relay state has three responsibilities. First when a node receives a *tree packet*, it parses the packet to determine if it is a relay node. If so, it updates its forwarding information and forwards the tree packet. Second, when the time since the arrival of a new set of trees (or a message to use the current tree set) reaches a threshold, the appropriate entries for the forwarding table are deleted, and the node's state changes accordingly. Third, when a node in one of the relay states associated with a particular group receives a *data packet* addressed to the group from the MAC layer, it checks to see if it has already received a copy of this packet. If not, it then checks to see if it is a relay for the tree value indicated in the packet's header. If so, the node forwards the packet without modification so that any receiver or relay node that receives the packet can process it. Since RoMR does not restrict the processing of the packet to the specific child on a given branch, a node might be able to receive it earlier than if it went the entire path. The caching of packet ID numbers prevents broadcast storms from occurring. The caches are periodically flushed. Since a receiver node may also act as a relay node the algorithm for a relay is embedded in the algorithm given in Figure 27.

5.4 JOIN AND LEAVE ALGORITHMS

When a node wishes to join a multicast group it sends a JOIN message to the multicast manager for the group as determined by a DNS-like lookup. The MM determines the receiver that is closest to the new member and sends an UPDATE message to the receiver. The recipient of the UPDATE message extends *all* trees in the tree set to the new member over the shortest path between the new member and the receiver by resending the packet from the MM as a tree set packet in the same manner as in distribution of new tree sets. At the time of the next tree set calculation performed by the source, the new nodes will be included in the tree calculations.

When a node wishes to leave a multicast group it sends a LEAVE message to the MM. The change in the tree occurs when the next tree message is sent from the sender due to a

topology change or a topology time-out.

5.5 ROMR VARIATIONS

Several variations of RoMR may be considered by giving the relay nodes a more active role in tree formation and maintenance. In the first variation the relay nodes make local repairs to the trees as needed until a new tree set arrives. In the second variation the multicast manager only determines the trees within a local scope and assigns proxy nodes to be responsible to create the rest of the tree. The third variation deals with an alternate method of selecting the multicast manager (MM) for the group. The fourth variation examines ways to reduce the size of the packets used to distribute the tree sets to the relay nodes. The last variation discusses an alternate way to calculate the weight of an entire tree.

5.5.1 Local Repair by Relay Nodes

A relay node records the set of downstream one-hop neighbors that act as relays or are member nodes when it receives a new tree set. When a link to one of these neighbors becomes unavailable, the node can determine from the topology table obtained from the unicast protocol if a two-hop path to the former neighbor exists. If so, the relay node will send a "mini" tree set packet to the new intermediate node to inform it that it is to act as a relay for the given set of trees. Again, the use of sequence numbers will prevent loop problems from occurring. The node will send a message to the multicast manager (MM) informing of the repair. The new relay node will continue to forward the packets until a new tree set is received. The advantage is that the tree set will remain intact for a longer period of time reducing the number of tree set packets being sent. The disadvantage is the requirement for relay nodes that are not group members to store the subtrees rooted at the relay node and to monitor them.

5.5.2 Tree Creation using Proxies

The multicast manager(MM) will calculate the trees based on the current topology tables available from the unicast protocol. Typically in unicast link state protocols each node knows all of the links within a certain local scope (usually 2 hops), while it may or may not know of all of the links outside of this range. The MM will consider each of the relay nodes on the periphery of its scope as proxy nodes and send the proxy node a list of trees to be extended to connect the indicated members to the indicated tree. The advantage is that the proxies will have more recent local information and can form more reliable trees than the MM alone. The disadvantage is the increased computational demands required of the relay nodes selected as proxy nodes.

5.5.3 Multicast Manager Selection

When the current multicast manager (MM) becomes a bottleneck and wishes to delegate the duties to another node, how should that node be selected? Instead a merely using neighbor nodes as in the simpler RoMR case, a more complex method could be utilized. The MM determines the three candidate nodes that are most central to the set of receivers and begins negotiations with those nodes to determine the new MM. Let M be the set of senders and receivers of the multicast group and N be the set of all nodes in the network, find $v_1, v_2, v_3 \in N$ such that

$$\sum_{m \in M} d(v_i, m)^2 \text{ is minimized}$$

Once the set of candidate nodes has been determined, poll each one to see if they are willing to assume the MM responsibilities.

5.5.4 Forwarding Tree Packets

In the simplest version of RoMR a node that receives a new set of trees examines the list of branches in the trees and if it finds that it is a parent of a child for some branch then it updates its records and forwards the tree packet unmodified. What are other techniques for handling the tree packet? One packet format would omit the branches that lead to leaf

nodes. In this situation the node would be a relay node if it found itself listed as a parent or a child in the tree list. The result would be a reduction in the size of the tree packet. Another packet handling technique increases the involvement of the relay nodes. In this second method a relay node would only forward the tree information that applied to the tree(s) rooted at that node. The disadvantage is the analysis required by the relay and the repeated repackaging of the tree data which may slow dissemination of the trees.

5.5.5 Tree Weight

In the simplest version of RoMR, the weakest link in the tree is used as the tree's weight. Another technique would be to estimate the lifetime of each link based on past behavior and to calculate the weight of the tree as being the ratio of the number of nodes with a predicted lifetime greater than a threshold τ to the total number of links in the tree.

6.0 ROMR SIMULATION AND RESULTS

The purpose of simulation is to study the effectiveness of the protocol in a variety of environments. Two of the freely available programming packages, ns and GloMoSim, are designed to provide a basis for network simulations including wireless mobile ad hoc networks and their features are given. After performing a few experiments with each, GloMoSim was selected as the simulation package based on ease of use. Once the simulator was selected, additional C code was added to the provided code in order to specify the algorithms of RoMR. Once the code was written and debugged, the parameters of the simulations were determined for a variety of network densities, multicast group sizes, and speeds of the nodes. The simulations were run to assess the performance of the protocol and to compare it to another proposed multicast protocol.

6.1 SIMULATORS

A number of network simulators exist, but only a few of the freely available ones are specifically designed for wireless communications in mobile networks. The two that are most often used in the academic research community are *ns* and GloMoSim.

6.1.1 ns

ns (network simulator) was originally developed at the University of California (UC) at Berkeley. Today's version, $ns-2$, is part of the Virtual InterNetwork Testbed (VINT) Project, a collaboration between researchers at UC Berkeley, Lawrence Berkeley National Laboratory (LBNL), the Information Sciences Institute at the University of Southern Cal-

ifornia, and Xerox's Palo Alto Research Center (PARC). The goal of the VINT Project is "to build a network simulator that will allow the study of scale and protocol interaction in the context of current and future network protocol" [18]. Carnegie Mellon University's Monarch Project [1] has provided wireless and mobility extensions to *ns*. The extensions provide radio propagation models and a shared media model for the network interface at the physical layer, IEEE 802.11 and carrier sense multiple access (CSMA) media access control (MAC) protocols at the link layer, and several unicast routing protocols at the network layer. A Tcl script runs the simulation with inputs from three files: a movement pattern file, a communication pattern file, and a router configuration file.

6.1.2 GloMoSim

The Global Mobile Information System Simulator (GloMoSim) was developed by the Parallel Computing Laboratory within the Computer Science Department of University of California, Los Angeles(UCLA) [30]. It is a simulation library for wireless networks that is built using the Parallel Simulation Environment for Complex Systems (PARSEC), the underlying discrete-event simulation package which was also developed at UCLA and is available on a variety of platforms. GloMoSim closely models the OSI layered approach in the design of network protocols and provides several options at the various layers. Simple APIs are used to communicate between the layers which allow developers to easily integrate new models with the existing code. Many of the environment's parameters are specified in a configuration file. The parameters include:

- Dimensions of the geographical area
- Number of nodes
- The placement of nodes (random, uniform, positions read from a file)
- The mobility model (random waypoint, random drunken)
- Details about the radio signals used (frequency, bandwidth, noise, power levels)
- The MAC protocol (802.11, CSMA, MACA, TSMA)
- The routing protocol
- A traffic-generating application (HTTP, FTP, CBR)

While both GloMoSim and *ns* are widely used, the former was selected as the simulator for this project after an initial experimental period using both packages. Both provided similar network simulator capabilities. Both have a significant learning curve. GloMoSim seemed to have a cleaner interface, whereas *ns* seemed to have been patched numerous times which led to the decision to use GloMoSim.

6.2 ROMR AND GLOMOSIM

A simulator framework was written to test the RoMR protocol as it evolved over time. The framework included the basic simulator built on top of GloMoSim, a visualization program written using Borland C++ Builder, as well as several additional helper program to generate the test cases and analyze the results. The C++ code to calculate the forward error correction codes based on Vandermonde matrices was written by Luigi Rizzo.

GloMoSim was selected as the simulator due to its being freely available and the fact that it was designed to simulate wireless networks. The underlying unicast protocol was initially chosen to be the Fisheye State Routing protocol since it was provided as part of the GloMoSim package. Later simulations used the Optimized Link State Routing (OLSR) protocol as the unicast protocol since an Internet draft detailing its operation as well as a multicast version (MOLSR) were available which could be used for comparison purposes. The unicast protocol was slightly modified in order to use it with RoMR. Specifically, the topology tables were modified so they included the links' probabilities when they were exchanged between neighbors.

Since GloMoSim is written using APIs to communicate between layers, only files that were directly related to the implementation of RoMR needed to be modified providing sufficient hooks for the RoMR code to execute. These files are written in C which are preprocessed by PARSEC and then compiled using Microsoft's Visual C++ version 6.0 on an IBM Thinkpad computer with a 700MHz Celeron processor running the Microsoft Windows 2000 Professional operating system.

The overall objective of the simulation is to examine and assess the performance of the basic version of RoMR. In order to isolate the effects of the use of multiple trees, link

weights, and the forward error correction encoding scheme, the initial simulations assume static groups for the duration of the simulations. This assumption is quite common in the published papers on other multicast protocols. We also assume the multicast manager is the sender for the duration of the simulation. The main item of interest is the percentage of the packets sent by the source node that were received by the group members. This metric will reflect the reliability of the protocol. The delay of the packet is inversely proportional to the overhead associated with the data redundancy. The higher the average delay, the fewer the number of redundant packets. The performance of the protocol is to be examined under a number of varying conditions, specifically the mobility of the nodes, the density of the network, and the size of the multicast group relative to the size of the network. Once the data has been gathered and examined a statistical analysis of the data will determine if the average number of packets received in variations of RoMR is significantly different from the number of packets received under MOLSR. The appropriate test to use in this case is a t-test of two samples assuming unequal variances. Excel 2000 from Microsoft Corporation is used to perform the analysis.

6.3 SIMULATION PARAMETERS

All of the simulations were run in a terrain of 1000 meters by 1000 meters. The nodes were randomly placed within the terrain. One node was randomly selected as the sender. The mobility was based on a random waypoint model with nodes travelling at constant speeds. In a random waypoint model a point in the terrain is randomly selected and the node travels to that point at the assigned speed. Once the node arrives at the point another point is selected and travel continues to the new point at the same speed. Input files were randomly generated for both the initial positioning of the nodes as well as the mobility trace files so that both a RoMR simulation and the corresponding MOLSR simulation would be based on the same initial node placements and subsequent node movements. Each group was initialized with its members at the beginning of the simulation. The group membership did not change during the simulation in order to reduce the number of aspects under consideration. The sender node generated a 512 byte constant bit rate packet every 2 seconds for 9 minutes starting 30

Table 2: Constant Simulation Parameters

Parameter	Value
Terrain	1000 × 1000 meters
Mobility Model	Random Waypoint
Speed	constant
Radio Frequency	2.4GHz
Bandwidth	11Mbps
Transmit Power	20 dBm
Signal-to-Noise ratio	10 using the accumulated noise model
Minimum Power for Received Signal	-78 dBm
Traffic Generator	constant bit rate (1 packet every 2 seconds)
Packet Size	512 bytes
Simulation Time	10 minutes

seconds into the simulation to allow time for an initial set of topology tables to be exchanged before multicasting began. Each simulation was run for 10 minutes of simulated time.

For all of the simulations the parameters were set to reflect the radio communications of a typical laptop computer circa 2002-2003: frequency of 2.4GHz, bandwidth of 11Mbps, and transmit power of 20dBm. The minimum power of a received packet is set to -78dBm. A packet is processed only if the signal to noise ratio is above 10 using an accumulative noise model.

To narrow the number of aspects affecting the results only three factors were varied in any given given group of simulation runs while the other parameters remained fixed. The three factors were network density, group size relative to the size of the network, and the speed of the nodes as suggested in RFC 2501 from the Network Working Group of the IETF [11]. Three densities of the network were tested: a sparse network with 10 nodes, a medium network with 30 nodes and a dense network with 50 nodes. With each of the density values, three sizes of groups were examined: 1/10, 1/5, and 1/2 of the network nodes. The simulations were run with the nodes travelling at six different speeds: 0.5, 1, 5, 10, 20, and 30 meters per second to test rates resembling slow walking up to highway vehicular speeds. Finally, each configuration was run using multiple starting seed values for

Table 3: Variable Simulation Parameters

Parameter	Value
Network Density	sparse with 10 nodes medium with 30 nodes dense with 50 nodes
Group Size	1/2, 1/5, or 1/10 of network size
Speeds (in meters per second)	0.5, 1, 5, 10, 20, 30
Minimum Number of Trees (μ)	2
Minimum Cumulative Weight (τ)	0.15
Reuse Factor (ρ)	0.5

the random number generator.

The first set of data was run with the set of multicast trees begin formed under the condition that at least two trees must exist in the next time epoch with a cumulative weight of at 0.10 or better. Other simulations varied the μ and τ and input parameters.

A second set of experiments was performed to investigate the advantages of the use of the Steiner tree as the basic type of tree as opposed to a tree composed of the union of shortest paths from each of the receivers to the sender. The percentage of packets successfully delivered to the receivers as well as the differences in the overhead due to control packets carrying tree set information and the duration of the tree set were examined.

6.4 SIMULATION RESULTS

The simulations tested the basic version of RoMR to investigate the performance under varying network conditions and RoMR parameters stated in Section 6.3 and to compare the results to the performance of MOLSR. The graphs on the following pages are arranged in pairs. The graphs shown at the top of the pages indicate the percentage of packets that were successfully delivered to the members with the nodes in the network moving under the set of 6 different speeds. The graphs at the bottom show the average end-to-end delay at the application layer for the same simulation. Each point on the graph represents the average of five different runs of the simulation using 5 different random number seeds. The i^{th} run

of MOLSR used the same mobility trace files and initial positions as did the i^{th} run of any of the version of RoMR so that the only difference was the routing protocol.

The captions on the graphs have the following meanings:

- molsr: Multicast version of OLSR
- romrOff: An experiment in which all link were assumed to have equal weight as a test of the effectiveness of calculating the weights and using them in the tree calculations. In the original version of RoMR the links with the lesser weights were discarded from consideration in the construction of the next tree. In romrOff, the links to be discarded were chosen at random.
- romrOn: Reuse factor = $\rho = 0.5$, cumulative weight threshold = $\tau = 0.15$, and minimum number of Trees = $\mu = 2$
- romr25Thresh: $\rho = 0.5$, $\tau = 0.25$, and $\mu = 2$
- romr3Trees: $\rho = 0.5$, $\tau = 0.15$, and $\mu = 3$

Overall, RoMR delivered more packets to the group members than MOLSR. See Figure 28. Examining the data for the individual cases, shows a general increase in the number of packets delivered to the group members when using RoMR as opposed to MOLSR along with an increase in the delay in the slow-moving network configurations. The increase in delay resulted from RoMR increasing the value of k to approach n with the purpose of reducing the overhead due to redundant packets. Since more packets comprise a packet group, the application had to wait longer for all of the packets in the packet group to arrive in order to decode them, but then all of the packets in the group were delivered with no delay to the upper layers. Indirectly this can be viewed as a measurement of the efficiency of the protocol. The higher the delay indicates a lower number of packets injected into the network for redundancy purposes.

A number of questions can be answered by performing statistical analysis of the data.

- Did the use of the link weights affect the performance of RoMR?
- Was the performance of RoMR significantly different from that of MOLSR?
- Did increasing the value of τ result in a significant difference in the number of packets received?

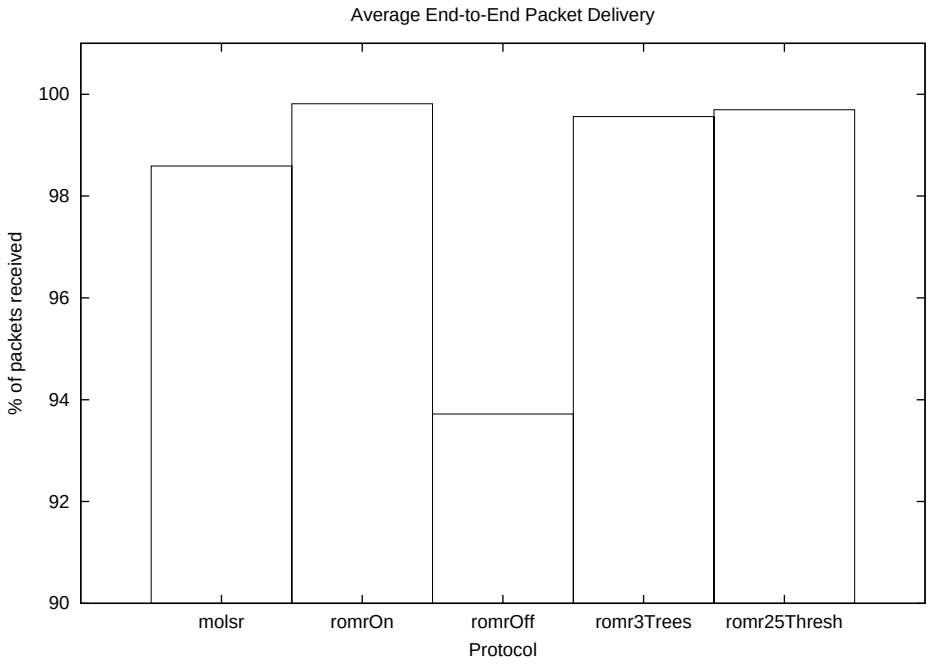


Figure 28: Average Percentage of Packets Delivered to Group Members

- Did increasing the value of μ result in a significant difference in the number of packets received?

In order to answer these questions a two-sample t-test assuming unequal variances with a 95% confidence interval was performed on the accumulated data under each of the various versions of the protocol. Tables 5 - 7 show the results of the tests. In all cases the differences in the mean number of packets received was shown to be significant so the answer to each of the above questions is "yes".

The second set of experiments compared the effectiveness of using a Steiner tree as the basic type of tree as opposed to a tree made up of shortest paths from each of the receivers to the sender. The parameters were the same as used in the romrOn case described above. The results, given in Table 4, showed that the successful delivery of packets to the receivers was very similar in both of the two cases, but other statistics gathered indicate the superiority of the use of the Steiner trees. The average amount of time a tree set remained active before a new tree set was distributed was 8% shorter in the shortest paths trees than in the Steiner tree version. This is due to the likelihood that more links are in the shortest paths tree than in the Steiner tree and thus the tree set becomes obsolete sooner in the shortest paths version. Another indicator that Steiner trees performed better than the shortest paths trees was in the examination of the amount of overhead. The overhead due to the control packets which carried the tree set information increased by an average of 16% when using the shortest paths trees. See Figure 29.

Table 4: Steiner vs Shortest Paths Trees

	Steiner Tree	Shortest Paths	% Increase/Decrease
Packets received	99.7%	99.7%	0.0%
lifetime	189.9 sec	174.2 sec	-8.3%
# of tree sets	11.0	11.4	3.9%
# of control bytes	20,023	23,297	16.4%

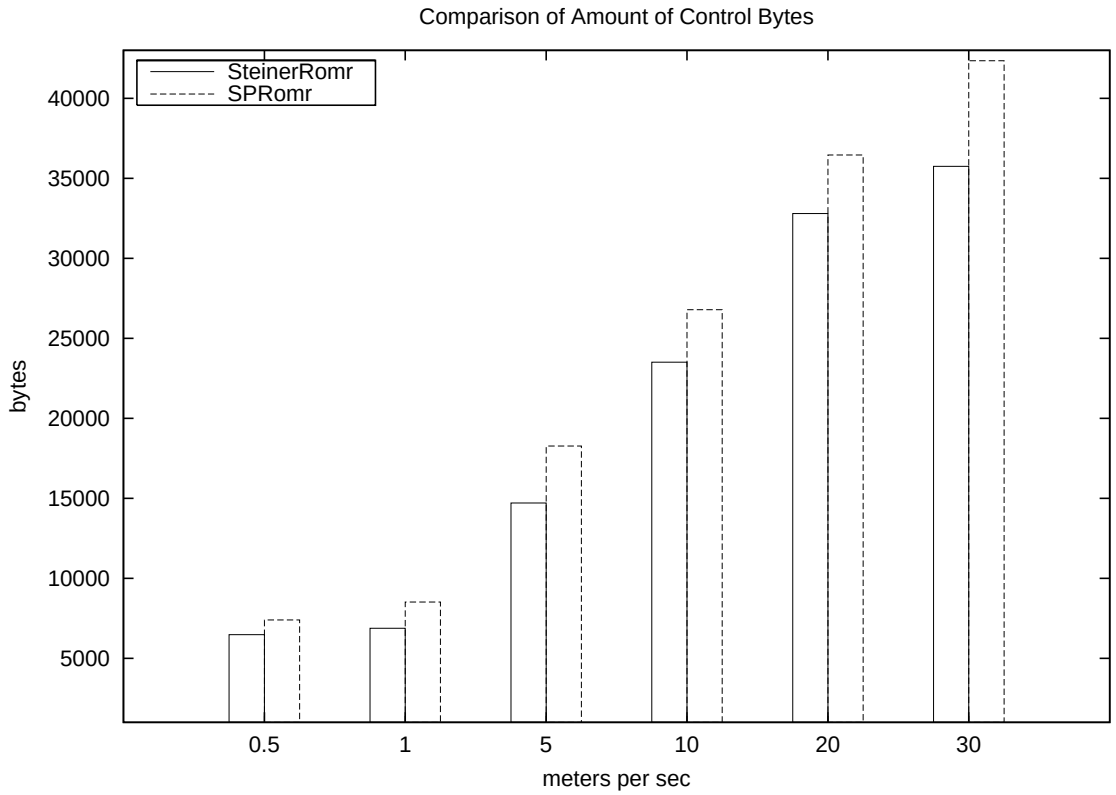
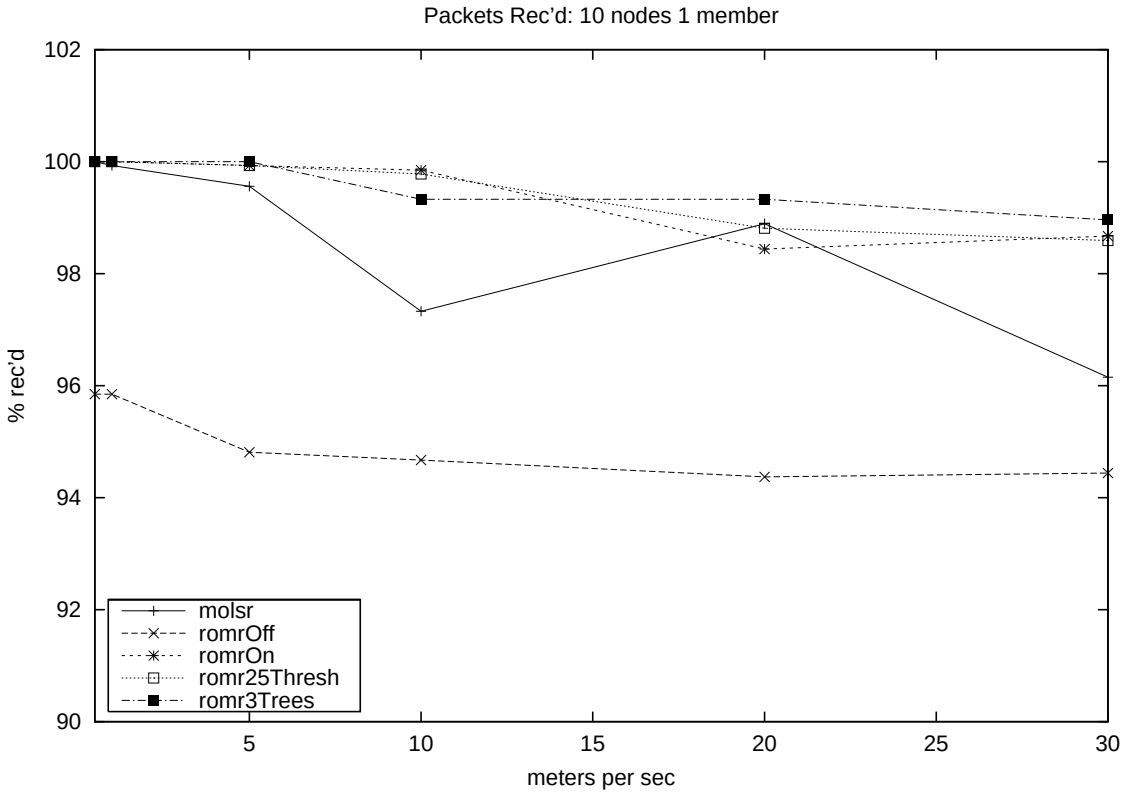
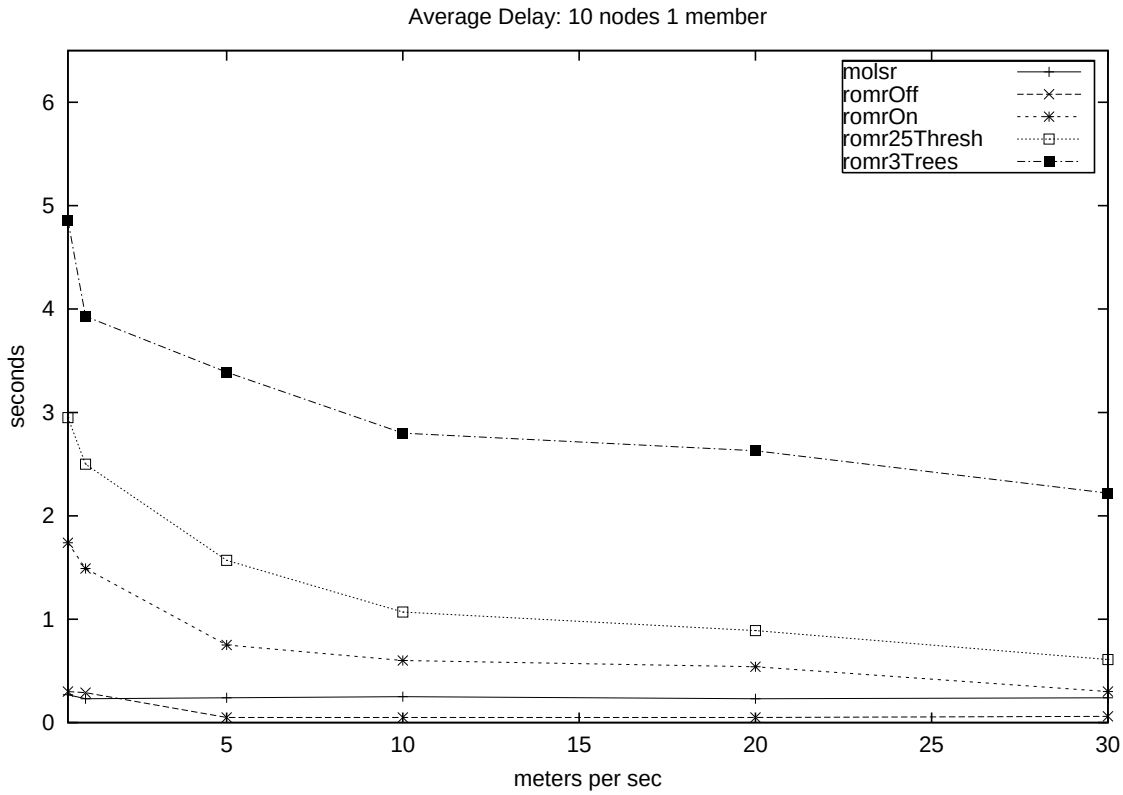
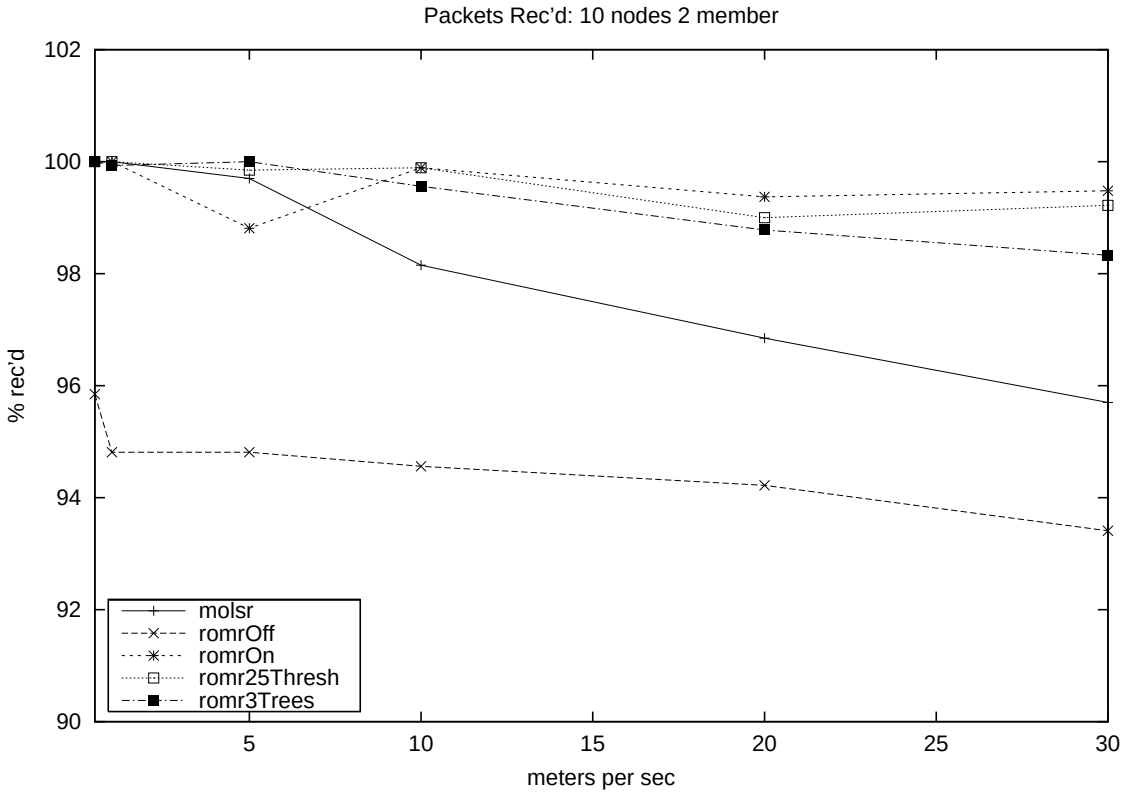
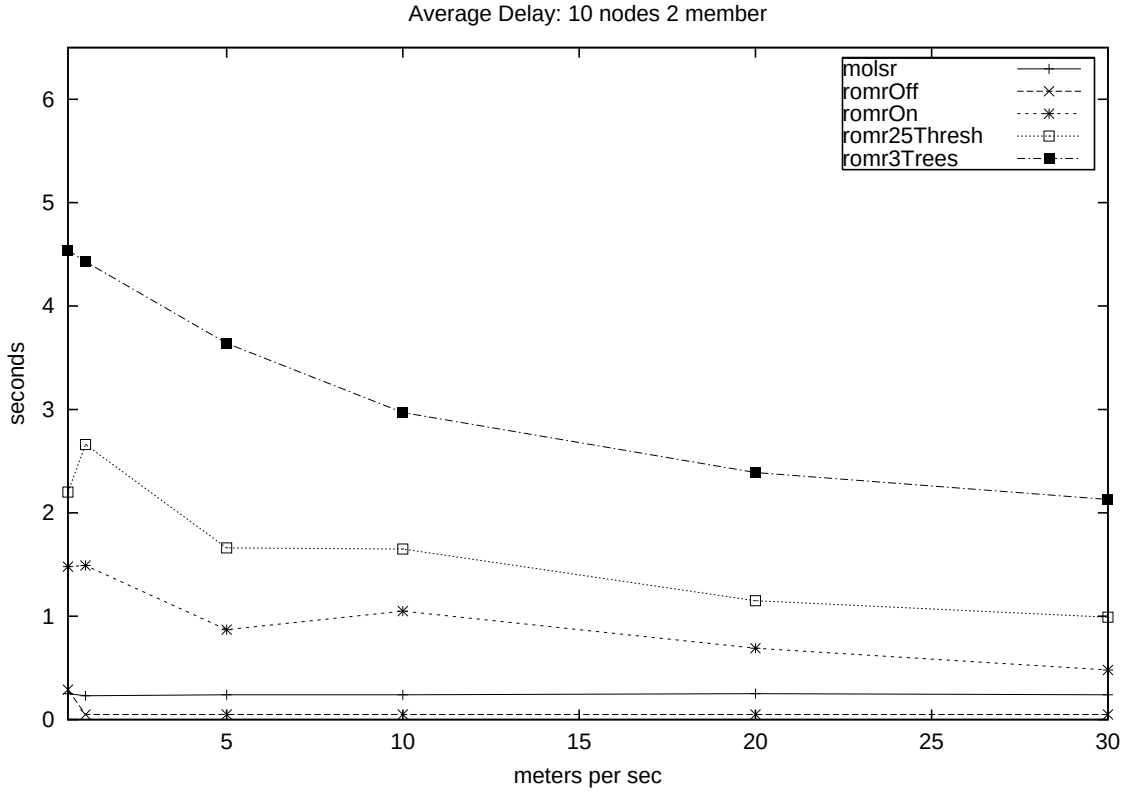


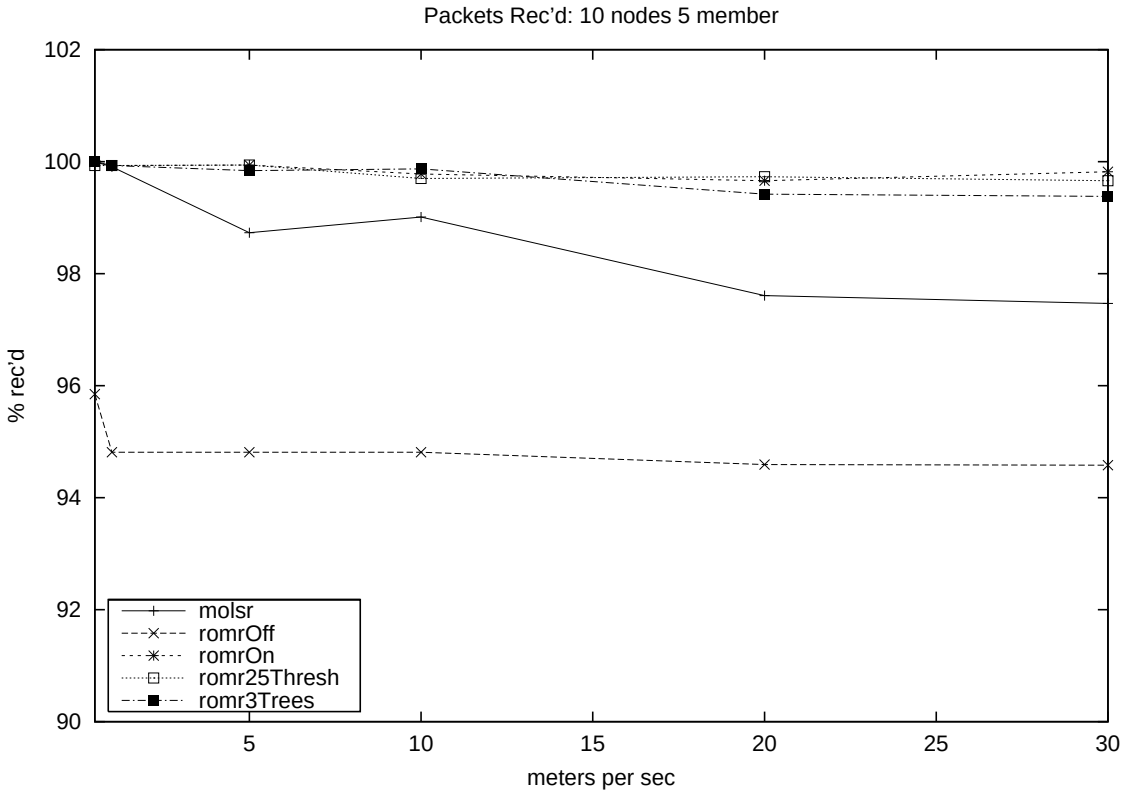
Figure 29: Overhead Comparison

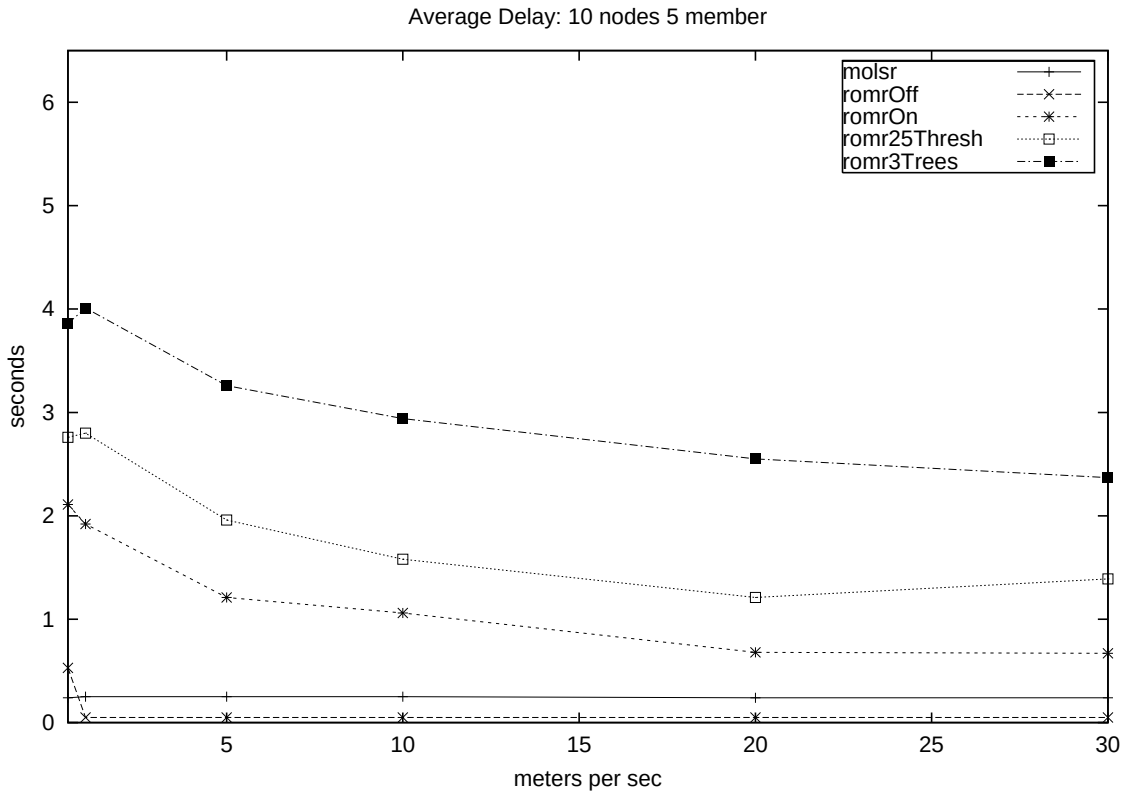

Packets Rec'd: 10 nodes 1 member
% rec'd
meters per sec
102
100
98
96
94
92
90
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

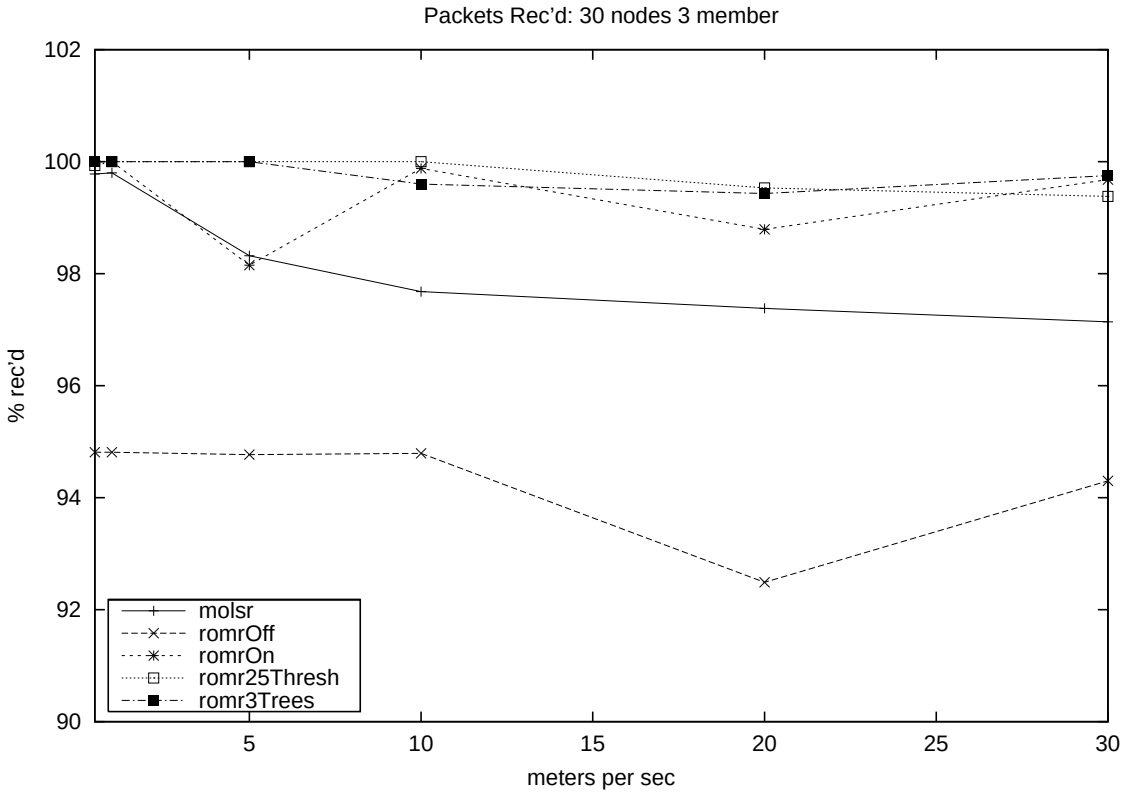

Average Delay: 10 nodes 1 member
seconds
meters per sec
6
5
4
3
2
1
0
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

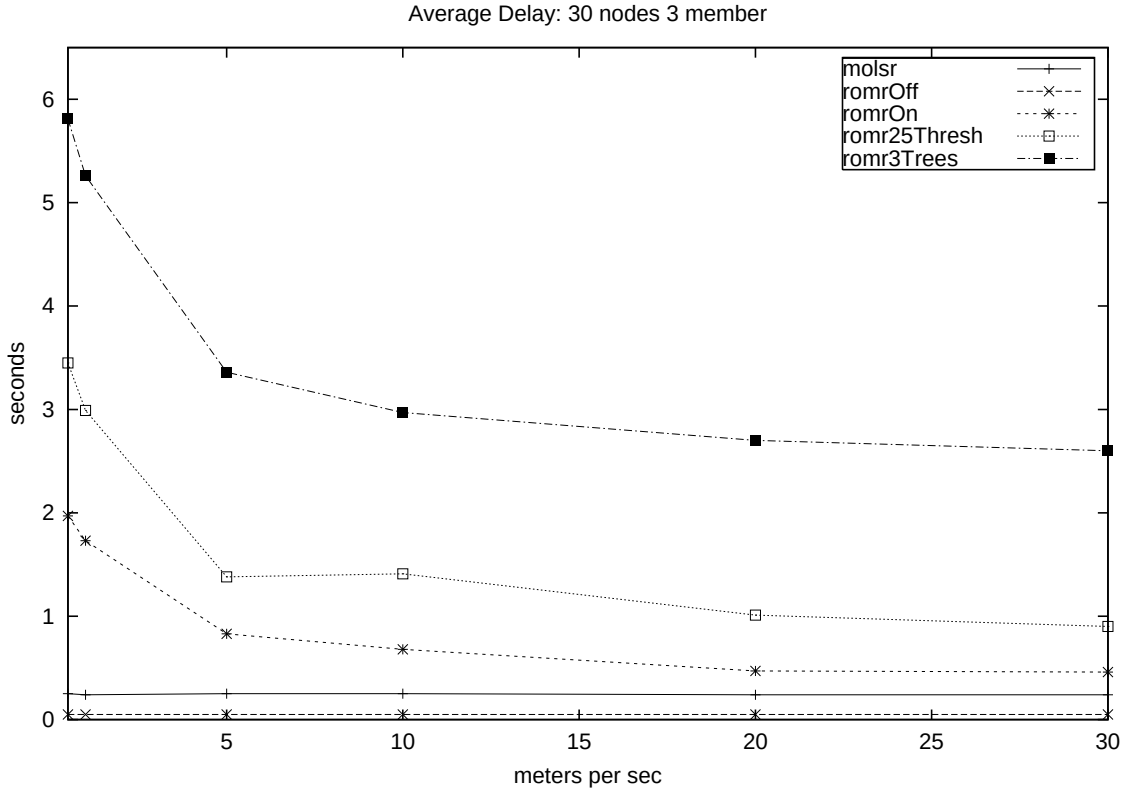

Packets Rec'd: 10 nodes 2 member
% rec'd
meters per sec
102
100
98
96
94
92
90
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

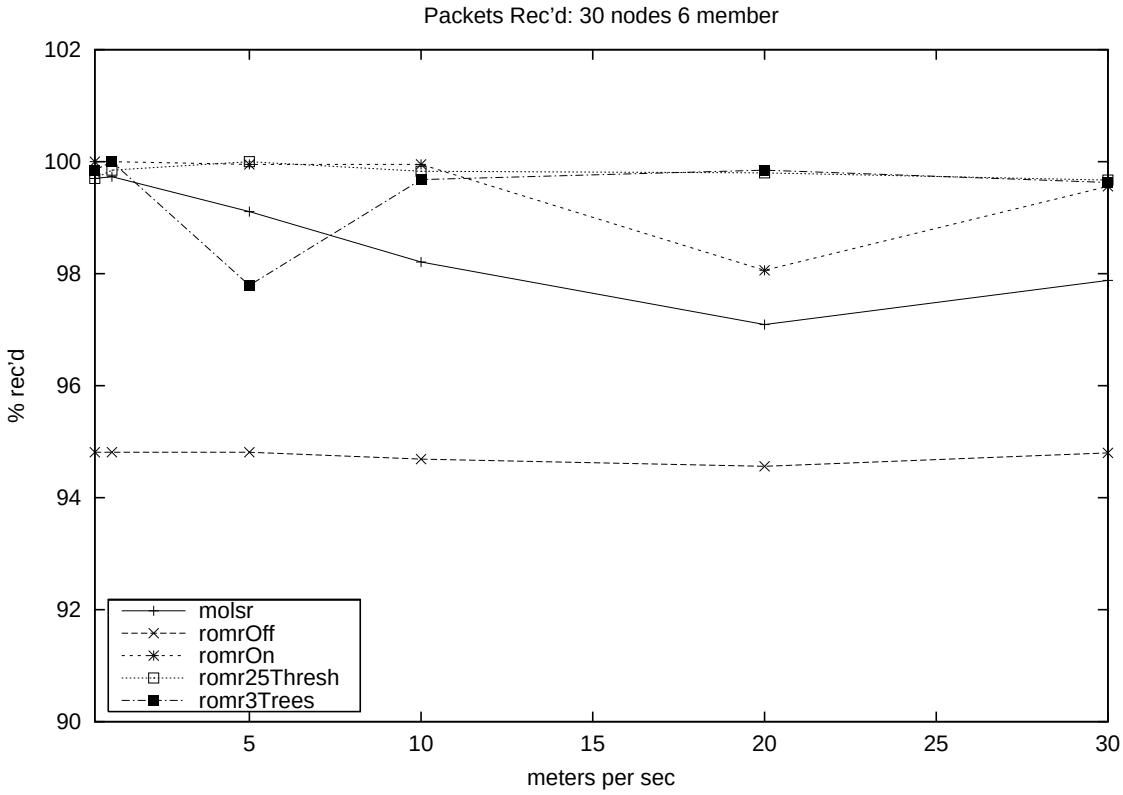

Average Delay: 10 nodes 2 member
seconds
meters per sec
6
5
4
3
2
1
0
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

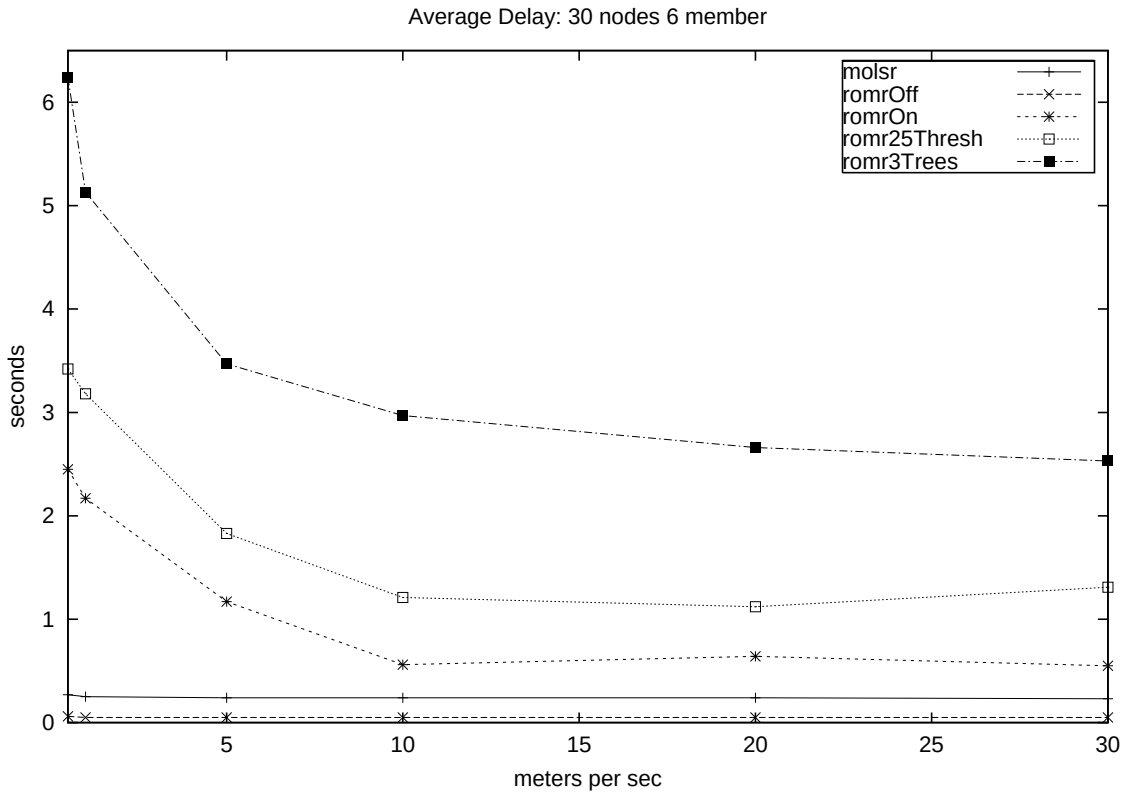

Packets Rec'd: 10 nodes 5 member
% rec'd
meters per sec
102
100
98
96
94
92
90
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

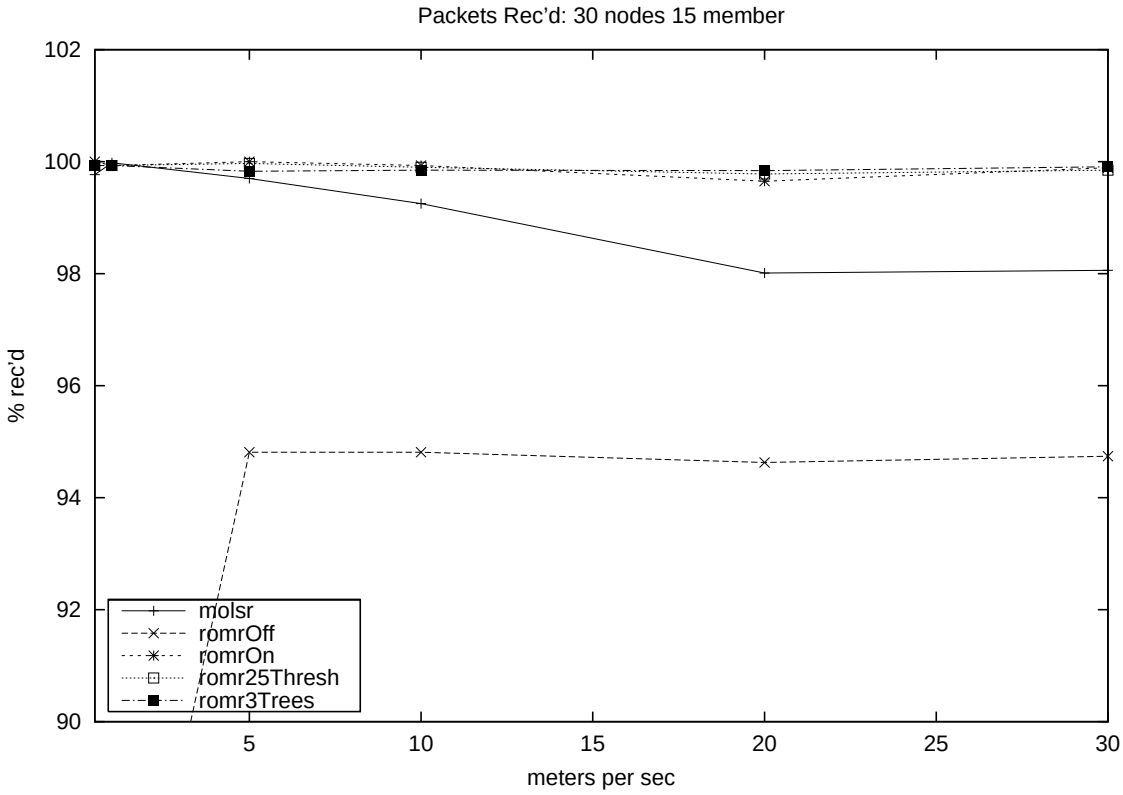

Average Delay: 10 nodes 5 member
seconds
meters per sec
6
5
4
3
2
1
0
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

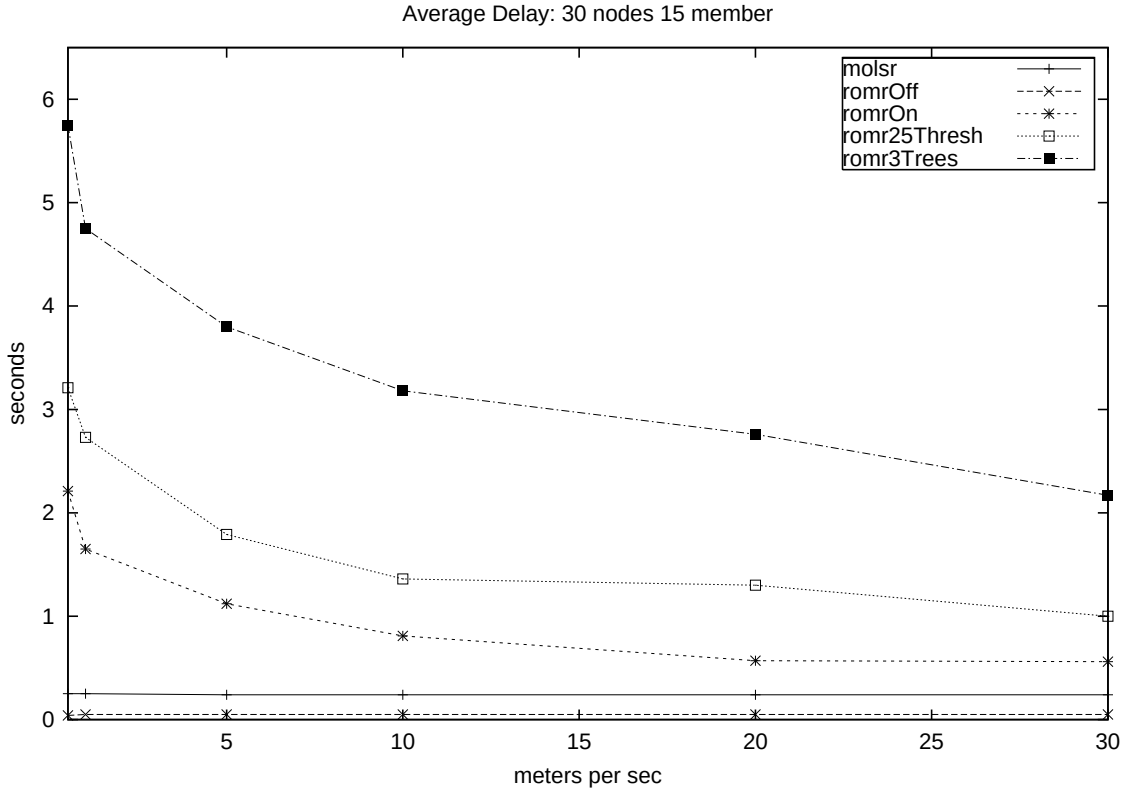

Packets Rec'd: 30 nodes 3 member
% rec'd
meters per sec
molsr
romrOff
romrOn
romr25Thresh
romr3Trees
102
100
98
96
94
92
90
5
10
15
20
25
30

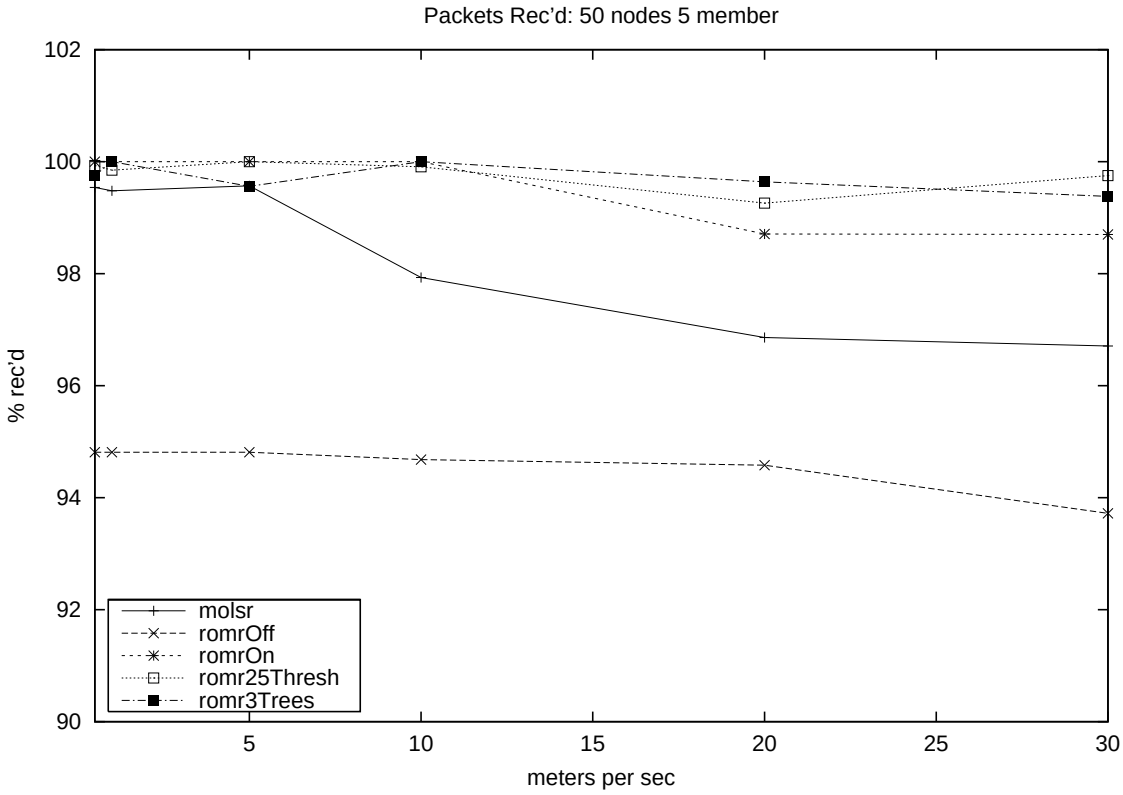

Average Delay: 30 nodes 3 member
seconds
meters per sec
molsr
romrOff
romrOn
romr25Thresh
romr3Trees
6
5
4
3
2
1
0
5
10
15
20
25
30

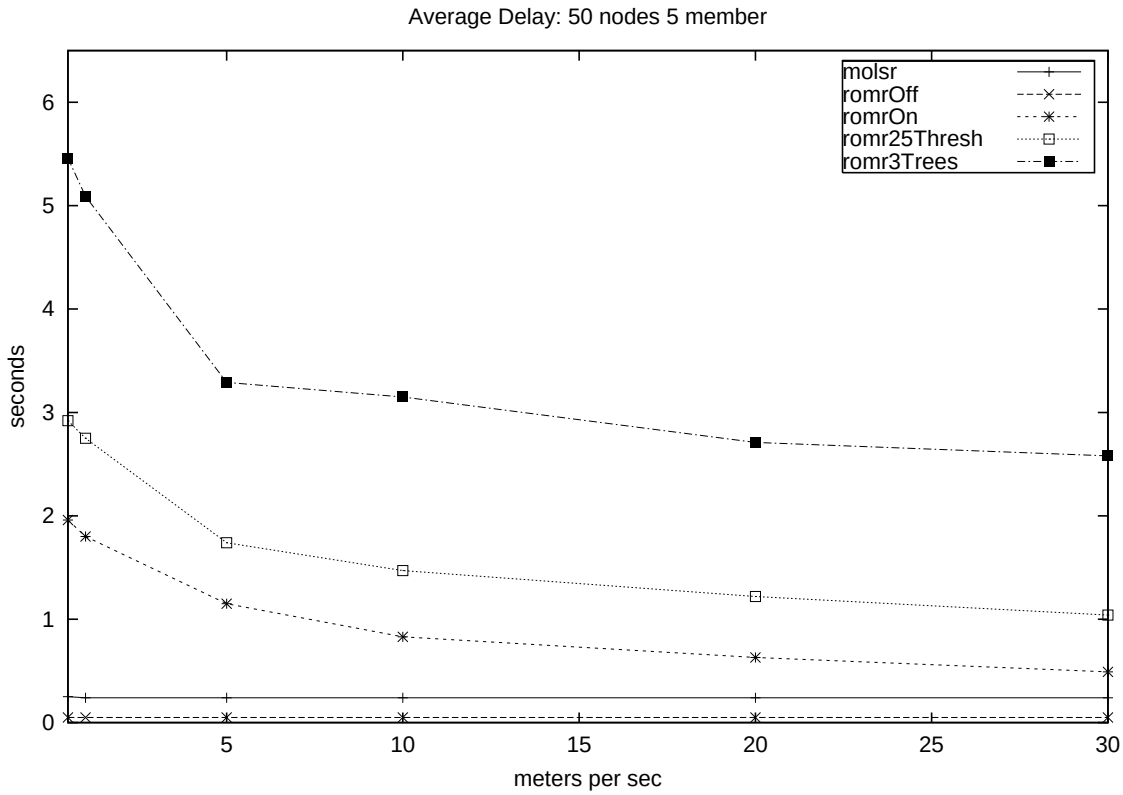

Packets Rec'd: 30 nodes 6 member
% rec'd
meters per sec
102
100
98
96
94
92
90
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

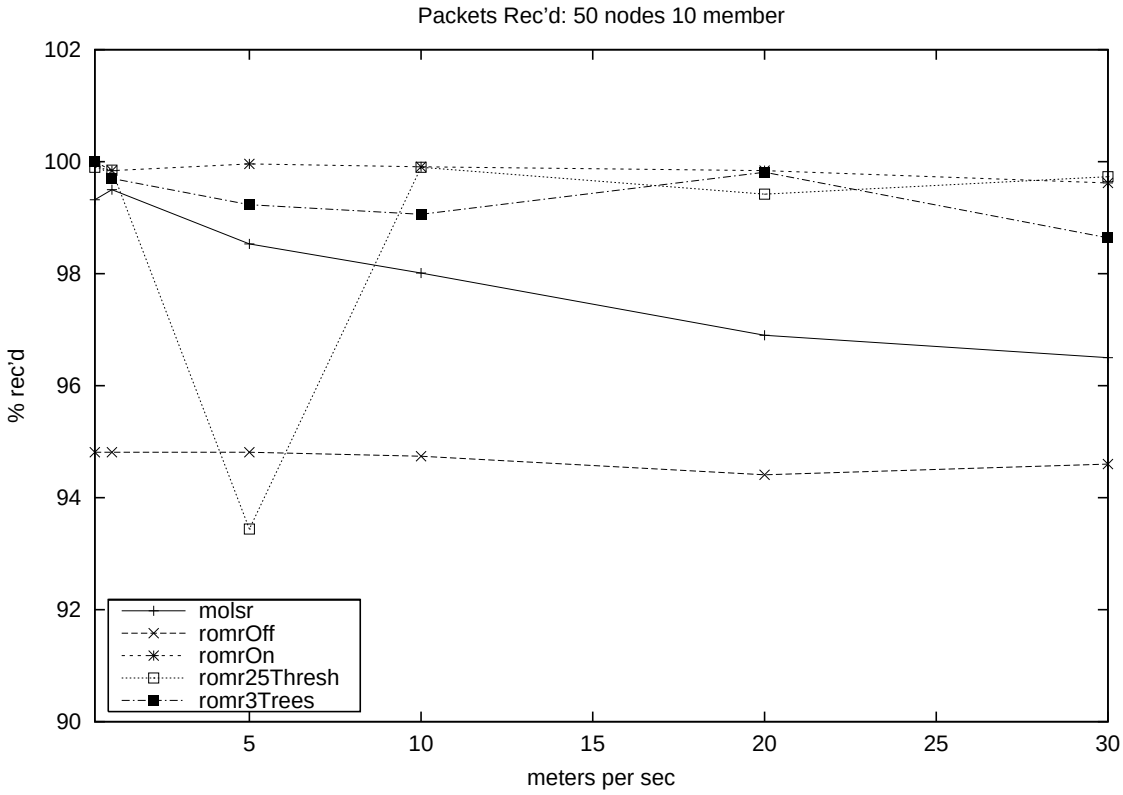

Average Delay: 30 nodes 6 member
seconds
meters per sec
6
5
4
3
2
1
0
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

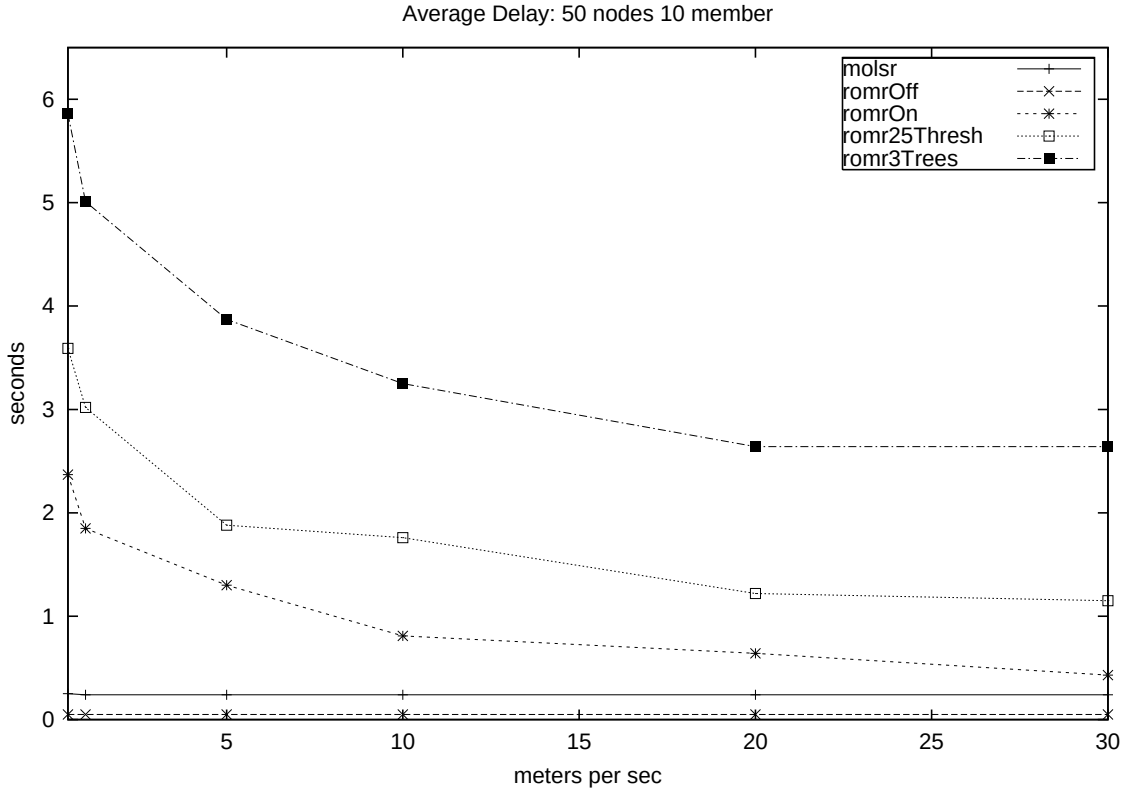

Packets Rec'd: 30 nodes 15 member
% rec'd
meters per sec
102
100
98
96
94
92
90
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

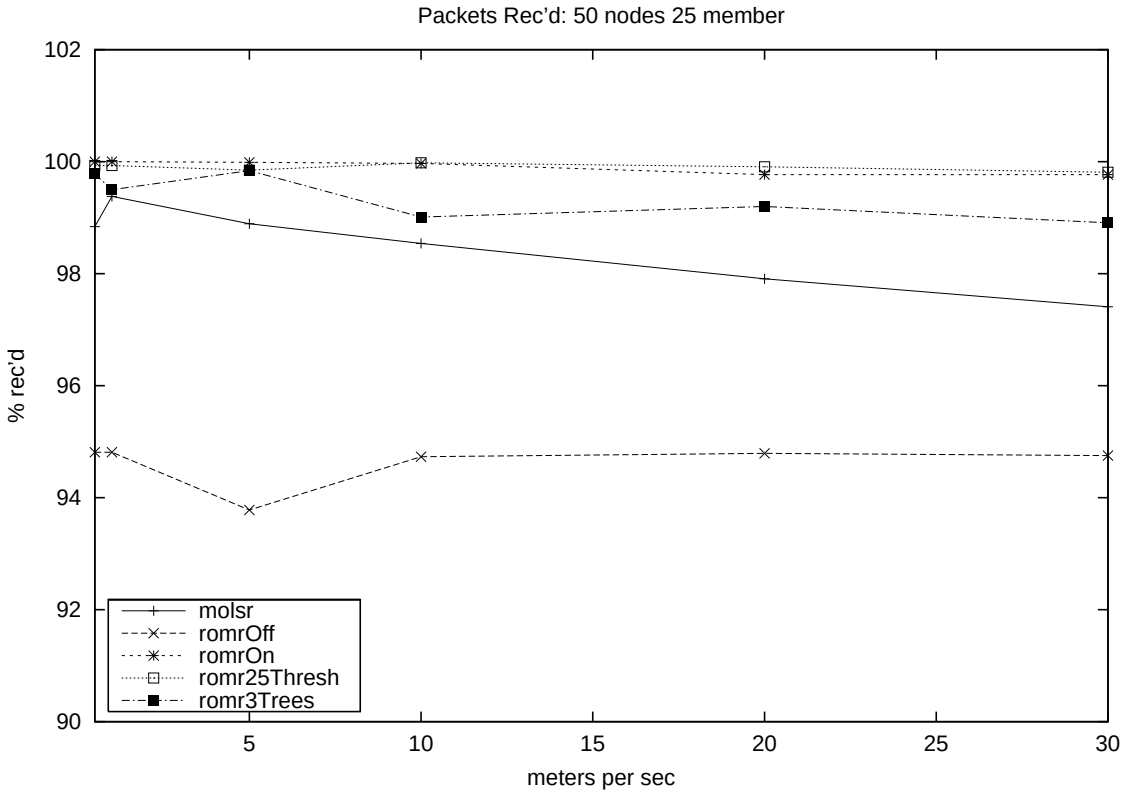

Average Delay: 30 nodes 15 member
seconds
meters per sec
6
5
4
3
2
1
0
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees


Packets Rec'd: 50 nodes 5 member
% rec'd
meters per sec
102
100
98
96
94
92
90
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees


Average Delay: 50 nodes 5 member
seconds
meters per sec
6
5
4
3
2
1
0
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees


Packets Rec'd: 50 nodes 10 member
% rec'd
meters per sec
102
100
98
96
94
92
90
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees


Average Delay: 50 nodes 10 member
seconds
meters per sec
6
5
4
3
2
1
0
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees


Packets Rec'd: 50 nodes 25 member
% rec'd
meters per sec
102
100
98
96
94
92
90
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

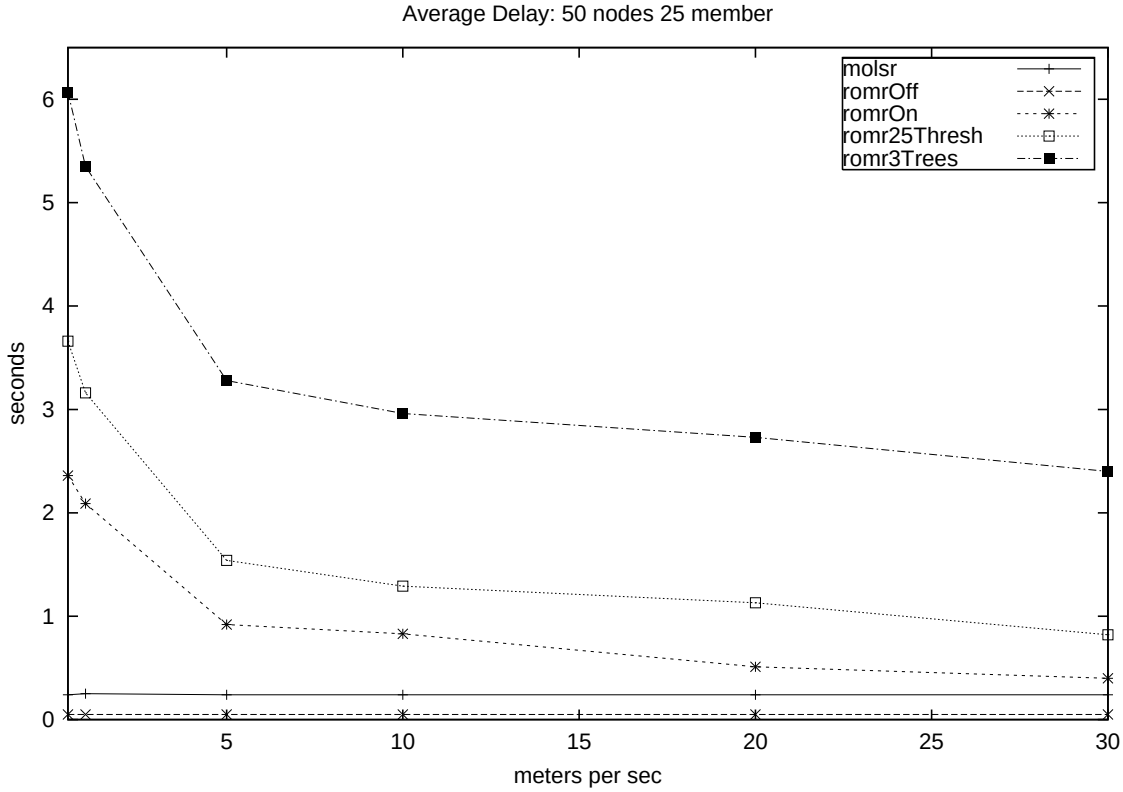

Average Delay: 50 nodes 25 member
seconds
meters per sec
6
5
4
3
2
1
0
5
10
15
20
25
30
molsr
romrOff
romrOn
romr25Thresh
romr3Trees

Table 5: t-tests 1 and 2

RomrOn vs RomrOff		
	RomrOn PktsRec	RomrOff PktsRec
Mean	269.50	253.04
Variance	4.50	553.92
Observations	2160	2160
Hypothesized Mean Difference	0	
df	2194	
t Stat	32.36	
one-tail P(T $\leq$ t)	1.72E-188	
one-tail t Critical	1.65	
two-tail P(T $\leq$ t)	3.45E-188	
two-tail t Critical	1.96	

RomrOn vs MOLSR		
	RomrOn PktsRec	MOLSR PktsRec
Mean	269.50	266.19
Variance	4.50	26.45
Observations	2160	2160
Hypothesized Mean Difference	0	
df	2874	
t Stat	27.59	
one-tail P(T $\leq$ t)	3.93E-149	
one-tail t Critical	1.65	
two-tail P(T $\leq$ t)	7.87E-149	
two-tail t Critical	1.96	

Table 6: t-tests 3 and 4

RomrOn vs Romr3Trees		
	RomrOn PktsRec	3Trees PktsRec
Mean	269.50	268.82
Variance	4.50	9.01
Observations	2160	2160
Hypothesized Mean Difference	0	
df	3886	
t Stat	8.55	
one-tail P(T ≤ t)	8.52E-18	
one-tail t Critical	1.65	
two-tail P(T ≤ t)	1.70E-17	
two-tail t Critical	1.96	

RomrOn vs Romr25Thresh		
	RomrOn PktsRec	25Thresh PktsRec
Mean	269.50	269.18
Variance	4.50	40.66
Observations	2160	2160
Hypothesized Mean Difference	0	
df	2632	
t Stat	2.22	
one-tail P(T ≤ t)	1.33E-02	
one-tail t Critical	1.65	
two-tail P(T ≤ t)	2.66E-02	
two-tail t Critical	1.96	

Table 7: t-tests 5

Romr3Trees vs Romr25Thresh		
	Romr3Trees PktsRec	Romr25Thresh PktsRec
Mean	268.82	269.18
Variance	9.01	40.66
Observations	2160	2160
Hypothesized Mean Difference	0	
df	3071	
t Stat	-2.34	
one-tail P(T $\leq$ t)	9.55E-03	
one-tail t Critical	1.65	
two-tail P(T $\leq$ t)	1.91E-02	
two-tail t Critical	1.96	

7.0 CONCLUSION

Multicasting in a mobile ad hoc network faces many challenges - the mobility of the nodes, the unreliable transmission medium, the lack of dedicated routers and infrastructure, the limited transmission range of the devices, and the limited available bandwidth. In addition to these problems the task of finding the optimal multicasting tree is NP-complete. These challenges and the questions that are are raised as a result of the challenges were discussed in Chapter 1. Chapter 2 examined the areas that are related to the ideas used in the development of the protocols and algorithms, specifically those relating to other proposed unicast and multicast protocols designed for use in mobile ad hoc networks, and compared the features of each to the proposed protocol. The chapter also discussed various heuristics used in the calculations of approximations to the optimal tree known as a Steiner tree. While RoMR could have used a number of these algorithms, an modification of the online version of the Shortest Path Heuristics was selected based on its simplicity and the bound of twice the cost of the optimal tree. In Chapter 3 we laid the basic foundation of the framework into which RoMR would eventually fit, describing how the layers in a layered model of network communications are affected by multicasting and how the architecture of RoMR is based on these layers. Chapter 4 derives the calculations used in the data link layer to compute link weights. The data link layer monitors of the strength of the signals from neighboring nodes and calculates a link weight based on a ratio of the area in which the node could receive the neighbor's signal to the area in which the neighboring node is predicted to occupy in the near future based on the strengths of the signals previously received. Chapter 5 provides an increasing level of detail about the algorithms use in RoMR including the steps in formulating the sets of multicast trees and the details of the forward error correction encoding and decoding process. Finally, Chapter 6 describes the simulations that were run

and the results they produced.

In summary, the Robust Multicast Routing (RoMR) protocol overcomes the challenges of multicast communications in mobile ad hoc networks and achieves the desired results of increasing the overall delivery of packets to members of a multicast group while keeping the number of extra data packets for redundancy purposes low and thereby conserving the use of network resources. Any multicast protocol is not expected to work in all circumstances, but the major strength of the RoMR lies in its flexibility to adapt to a variety of network conditions during the course of execution as exemplified in the simulation results. The novelty of RoMR is due to the development of the link weight calculations and the integration of methods not previously combined into a multicast protocol including forward error correction encoding techniques, Steiner tree approximations, and link availability estimates.

The contributions of this thesis include:

- A novel method used to compute the weights of the links in a mobile ad-hoc network based on previous power levels of the received signal by a node.
- The development of a tree-based multicast protocol incorporating ideas of
  - a modified Steiner tree approximation algorithm
  - the use of multiple interconnected trees
  - forward error correction codes used in the telecommunications industry
- The formulation of a simulation framework.

Work on RoMR can be continued in several areas. Simulations incorporating more variation in group membership and the change in the designation of the multicast manager need to be written, run, and analyzed. In the future RoMR could be expanded to support Quality-of-Service (QoS) with realtime guarantees in elastic applications. Another area is to address the scalability issue in RoMR. In extremely large networks, seeking to compute the set of trees based on shortest paths or Steiner trees may not be feasible. How can RoMR be adapted to work in such an environment? One possibility that addresses this last question is to disseminate the packets to strategically positioned nodes in the network and to have the interested parties seek the information based on content, name or other attributes.

BIBLIOGRAPHY

[1] The CMU Monarch Project's Wireless and Mobility Extensions to NS, August 1998. Available from http://www.monarch.cs.cmu.edu/.

[2] Popular wireless local area networks gain large boost in speed. http://standards.ieee.org/announcements/80211gfinal.html, June 2003.

[3] Tawfig Alrabiah. *Multicast Routing For Real-Time Communication in Wide-Area High Speed Networks*. PhD thesis, University of Pittsburgh, 1999.

[4] Sang Ho Bae, Sung-Ju Lee, William Su, and Mario Gerla. The design, implementation, and performance evaluation of the on-demand multicast routing protocol in multihop wireless networks. *IEEE Network*, pages 70–77, January/February 2000.

[5] F. Bauer and A. Varma. Aries: A rearrangeable inexpensive edge-based on-line steiner algorithm. *IEEE Journal on Selected Areas in Communications*, 15(3):382–397, April 1997.

[6] Shigang Chen and Klara Nahrstedt. Distributed Quality-of-Service Routing in Ad Hoc Networks. *IEEE Journal On Selected Areas in Communications*, 17(8):1488–1505, August 1999.

[7] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in Clustered Multihop Mobile Wireless Networks with Fading Channel. In *The IEEE Singapore International Conference on Networks (SICON97)*, pages 197–211, Kent Ridge, Singapore, April 1997.

[8] Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized Link State Routing Protocol, April 2003. Internet draft available from http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-09.txt, work in progress.

[9] Douglas E. Comer. *Computer Networks and Internets*. Prentice-Hall, second edition, 1999.

[10] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, Massachussetts, 1990.

[11] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. IETF RFC 2501, January 1999.

[12] Matthew Doar and Ian Leslie. How bad is naive multicast routing. In *Proceedings of IEEE INFOCOM*, pages 82–93, San Francisco, CA, April 1993.

[13] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi. Signal Stability based Adaptive Routing (SSA) for Ad-Hoc Mobile Networks. *IEEE Personal Communications*, pages 36–45, February 1997. Accessed on March 25, 2000. Available at: http://www.cs.umd.edu/projects/mcml/papers/pcm97.ps.

[14] J. J. Garcia-Luna-Aceves and Ewerton L. Madruga. The Core-Assisted Mesh Protocol. *IEEE Journal on Selected Areas in Communications*, 17(8), August 1999.

[15] J. J. Garcia-Luna-Aceves and Marcelo Spohn. Efficient Routing in Packet-Radio Networks Using Link-State Information. In *Proceedings of IEEE Wireless Communications and Networking Conference 1999 (WCNC '99)*, pages 1308–1312, New Orleans, Louisiana, September 1999.

[16] Liang Guo and Ibrahim Matta. QDMR: An Efficient QoS Dependent Multicast Routing Algorithm. In *Proceedings of the Fifth IEEE Real-Time Technology and Applications Symposium (RTAS '99)*, Vancouver, British Columbia, Canada, June 1999. Available at: http://www.ccs.neu.edu/home/matta/Papers/rtas99.ps.

[17] Sandeep K. S. Gupta and Pradip K. Srimani. An Adaptive Protocol for Reliable Multicast in Mobile Multi-hop Radio Networks. In *Proceedings of Second IEEE Workshop on Mobile Computer Systems and Applications (WMCSA'99)*, New Orleans, LA, February 25–26 1999.

[18] Ahmed Helmy and Satish Kumar. VINT - virtual InterNetwork testbed. Last accessed 11/6/2003. http://www.isi.edu/nsnam/vint/index.html.

[19] Christopher Ho, Katia Obraczka, Gene Tsudik, and Kumar Viswanath. Flooding for Reliable Multicast in Multi-Hop Ad Hoc Networks. In *Proceedings of Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Seattle, WA, August 1999.

[20] Sung-Pil Hong, Heesang Lee, and Bum Hwan Park. An efficient multicast routing algorithm for delay-sensitive applications with dynamic membership. In *Proceedings of INFOCOMM '98*, pages 1433–1440, 1998.

[21] Stefan Hougardy and Hans Jürgen Prömel. A 1.598 approximation algorithm for the steiner problem in graphs. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–453, 1999.

[22] Makoto Imase and Bernard Waxman. Dynamic steiner tree problem. *SIAM Journal of Discrete Math.*, 4(3):369–384, August 1991.

[23] Atsushi Iwata, Ching-Chuan Chiang, Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Scalable Routing Strategies for Ad hoc Wireless Networks. *IEEE Journal on Selected Areas in Communications*, Special Issue on Ad-Hoc, August 1999.

[24] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996. Accessed at http://www.monarch.cs.cmu.edu/monarch-papers/kluwer-adhoc.ps on June 4, 2000.

[25] David B. Johnson and David A. Maltz. Protocols for Adaptive Wireless and Mobile Networking. *IEEE Personal Communications*, 3(1):34–42, February 1996.

[26] Richard M. Karp. Reducibility Among Combinatorial Problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–99, New York, 1972. Plenum Press.

[27] Young-Bae Ko and Nitin H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of MobiCom '98*, Dallas, Texas, October 1998. Accessed at: http://www.cs.tamu.edu/faculty/vaidya/papers/mobile-computing/mobicom98.pdf on March 25, 2000.

[28] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15:141–145, 1981.

[29] Sung-Ju Lee, William Su, Julian Hsu, Mario Gerla, and Rajive Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *Proceedings of IEEE InfoCom 2000*, March 2000. ftp://pcl.cs.ucla.edu/pub/papers/infocom2000.ps.

[30] Mario Gerla Lokesh Bajaj, Mineo Takai, Rajat Ahuja, Rajive Bagrodia. Glomosim: A scalable network simulation environment. Technical Report 990027, University of California, Los Angeles, Computer Science Department, May 13, 1999.

[31] Ewerton L. Madruga and J.J. Garcia-Luna-Aceves. Scalable Multicasting: The Core-Assisted Mesh Protocol. Accepted for publication in *ACM/Baltzer Mobile Networks and Applications Journal,* Special Issue on Management of Mobility in Distributed Systems, First Quarter 2001.

[32] Thomas A. Maufer. *Deploying IP Multicast in the Enterprise*. Prentice-Hall, Inc., Upper Saddle River, NJ, 1998.

[33] A.B. McDonald and T. Znati. A Path Availabaility Model for Wireless Ad-Hoc Networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference 1999 (WCNC'99)*, New Orleans, LA, September 21–24 1999.

[34] C. Kenneth Miller. *Multicast Networking and Applications*. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

[35] S. Murthy and J.J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM Mobile Networks and Applications Journal*, Special issue on Routing in Mobile Communications Networks:183–197, October 1996.

[36] Kaveh Pahlavan and Prashant Krishnamurthy. *Principles of Wireless Networks*. Prentice-Hall, Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2002.

[37] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of INFOCOM '97*, April 1997.

[38] Charles Perkins. Ad Hoc On Demand Distance Vector (AODV) Routing, November 1997. Accessed at http://www.ics.uci.edu/ atm/adhoc/paper-collection/perkins-draft-ietf-manet-aodv-00.txt on June 4, 2000.

[39] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[40] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, New Orleans, Louisiana, February 1999. Accessed at http://www.iprg.nokia.com/ charliep/txt/aodv/aodv.ps on June 4, 2000.

[41] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM (JACM)*, 36(2):335–348, 1989.

[42] S. Ramanathan. Multicast Tree Generation in Networks with Asymmetric Links. *IEEE/ACM Transactions on Networking*, 4(4):558–568, 1996.

[43] V.J. Rayward-Smith. The computation of nearly minimal steiner trees in graphs. *Mathematics, Education, Science, Technology*, 14:15–23, 1983.

[44] Luigi Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM SIGCOMM Computer Communication Review*, 27(2):24–36, 1997.

[45] Gabriel Robins and Alexander Zelikovsky. Improved Steiner Tree Approximations in Graphs. In *Proceedings of the SIAM-ACM Symposium on Discrete Algorithms (SODA)*, pages 770–779, San Francisco, CA, January 2000. Accessed at http://www.cs.virginia.edu/ robins/papers/soda2000_camera.pdf.

[46] Elizabeth M. Royer and Charles E. Perkins. Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom '99)*, pages 207–218, Seattle, Washington, August 1999.

[47] Elizabeth M. Royer and Chai-Keong Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. *IEEE Personal Communications*, pages 46–55, April 1999.

[48] Laxman H. Sahasrabuddhe and Biswanath Mukherjee. Multicast Routing Algorithms and Protocols: A Tutorial. *IEEE Network*, pages 90–102, Jan/Feb 2000.

[49] Dajiang He Shengming Jiang and Janqiang Rao. A prediction-based link availability estimation for mobile ad hoc networks. In *Proceedings of IEEE InfoCom 2001*, pages 1745–1752.

[50] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. MCEDAR: Multicast Core-Extraction Distributed Ad hoc Routing. In *Proceedings of IEEE Wireless Communications and Networking Conference 1999 (WCNC '99)*, pages 1313–1317, New Orleans, Louisiana, September 1999.

[51] Hiromitsu Takahashi and Akira Matsuyama. An Approximate Solution for the Steiner Problem in Graphs. *Math. Japonica*, 24(6):573–577, 1980.

[52] C-K. Toh. Associativity-Based Routing for Ad-Hoc Mobile Networks. *International Journal On Wireless Personal Communications*, 4(2), March 1997. Accessed at: http://users.ece.gatech.edu/∼cktoh/so_new.ps.gz on March 25, 2000.

[53] Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1611–1622, December 1988.

[54] Ellen Kayata Wesel. *Wireless Multimedia Communications - Networking Video, Voice, and Data*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1998.

[55] Pawel Winter. Steiner problem in networks: A survey. *NETWORKS*, 17:29–167, 1987.

[56] C.W. Wu, Y.C. Tay, and C-K. Toh. Ad Hoc Multicast Routing Protocol Utilizing Increasing Id-numberS (AMRIS) Functional Specification, November 1998. Acessed at: http://www.freenic.net/drafts/drafts-i-j/draft-ietf-manet-amris-spec-00.html on March 18, 2000, work in progress.

[57] Qing Zhu, Mehrdad Parsa, and J. J. Garcia-Luna-Aceves. A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting. In *Proceedings of IEEE InfoCom '95*, Boston, Massachussetts, April 1995. Available at http://www.cse.ucsc.edu/research/ccrg/publications.html.