Routing in Ad hoc Mobile Networks: On-Demand and Hierarchical Strategies

Elizabeth M. Royer

UNIVERSITY OF CALIFORNIA
Santa Barbara

Routing in Ad hoc Mobile Networks: On-Demand and Hierarchical Strategies

A Dissertation Submitted in Partial Satisfaction
of the Requirements for the Degree of

Doctor of Philosophy
in
Electrical and Computer Engineering
by

Elizabeth Michelle Royer

Dissertation Committee:
Professor P. Michael Melliar-Smith, Chairperson
Professor Louise E. Moser
Professor Steven E. Butner
Professor Kevin C. Almeroth
Mr. Charles E. Perkins

December 2000

The dissertation of Elizabeth Michelle Royer
is approved:

Committee Chairperson

October 2000

December 2000

To Mom and Grandpa Ben

ACKNOWLEDGMENTS

I would first like to thank my advisors, Professor Michael Melliar-Smith and Professor Louise Moser, for all their wisdom and guidance during my years as a graduate student. I also thank them for recruiting me into UCSB and for helping me secure funding during my graduate work. I greatly appreciate all the freedom they gave me to pursue whatever research interested me, and for offering their insight whenever I needed it.

I would also like to thank the other members of my thesis committee, Professor Steven Butner, Professor Kevin Almeroth, and Charlie Perkins, for taking the time to participate on my committee and for providing additional guidance and insight into my research. I particularly thank Kevin for all the advice he has given me in the past year, and for always taking the time to meet with me whenever I needed anything.

I give extra special thanks to Charlie Perkins. Thank you for taking an interest in my work and inviting me to intern with you that first summer. Thank you for having the confidence in me to allow me to conduct my research however I saw fit, and for giving me the opportunity to learn both from you and from my own mistakes. Your guidance has been absolutely invaluable to me.

I thank my fellow labmates of the past few years for making lab such an interesting place to work. I give special thanks to Nitya and Priya for being such expert system administrators and for keeping our lab up and running, and to Ruppert for always being willing to help find solutions to any problems I had with my work, and for never complaining that I dragged him away from his.

My undergraduate advisor, Professor Lois Hawkes, first introduced me to the topic of Women in Computing during my search for an honors thesis topic. I thank her for sparking my interest in this important topic, and for taking a special interest in me and the success of my career.

I give special thanks to Phyllis Brady for, first, her early support of the establishment of the UCSB WICSE group, and later for always being ready to lend an ear when I had something I needed to say. Thank you so much for all your effort in the Women in Engineering program. It is more appreciated than you will probably ever know.

I thank Tracy Camp for showing such an interest in my professional development and for being willing to help me however she could in times when I needed it. Thank you for all the nominations and invitations you have given to me. I appreciate your efforts to further my career more than I can express, and I hope that I am someday able to help someone else in the ways that you have been able to help me.

I thank all my wonderful friends who have always been willing to lend an ear when I needed someone to talk to, or an evening when I wanted to have some fun. I have had some truly wonderful times at UCSB, and it would not have been the same without all of you. You all have made Santa Barbara a wonderful place to be.

To my aunt Carol, thank you for being such a wonderful friend. Thank you for always believing in me and for being so proud of my accomplishments. Thank you for always being someone I could confide in, and for being such a wonderful role model.

I thank all my grandparents for all they love they have always given me, and for all the pride they have always taken in my accomplishments. In particular I thank my Grandpa Ben for instilling in me early on the importance of an education in being able to reach my dreams, whatever they were at the time.

Mark, where would I be today if it wasn't for you? Since my very first day of graduate school, you have always been there whenever I needed anything. Thank you for always rejoicing in my accomplishments, for reassuring me when I worried about failure, and for loving me no matter what I did.

Lastly, and most importantly, I thank my mom for all the love, guidance, and support she has always given me. Thank you for being such a wonderful role model, and for always believing in my abilities. Thank you for giving me the courage it took to take the path in life that I have chosen, and for always believing that I would be successful in whatever I decided to do.

Financial support for this research was provided by a National Science Foundation Graduate Fellowship, a University of California Doctoral Scholar Fellowship, Lucent Technologies under grant 97-116, Cubic Corporation under grant A14789, and a Sun Microsystems, Inc. gift.

VITA

1996	Bachelor of Science, Computer Science Florida State University
1996	Bachelor of Science, Applied Mathematics Florida State University
1996-2001	Doctoral Scholar Fellowship University of California, Santa Barbara
1996-2000	Graduate Student Researcher Department of Electrical and Computer Engineering University of California, Santa Barbara
1997-2000	National Science Foundation Graduate Fellowship
1997	Master of Science, Electrical and Computer Engineering University of California, Santa Barbara

PUBLICATIONS

- E. M. Royer, P. M. Melliar-Smith and L. E. Moser. "Hierarchical Routing in Ad hoc Mobile Networks." *Submitted for Publication.*

- S.-J. Lee, E. M. Royer, and C. E. Perkins. "Ad hoc Routing Protocol Scalability." *Submitted for Publication.*

- E. M. Royer, P. M. Melliar-Smith, and L. E. Moser. "An Analysis of the Optimum Node Density for Ad hoc Mobile Networks." *Submitted for Publication.*

- S. R. Das, C. E. Perkins, E. M. Royer, and M. K. Marina. "Performance Comparison of Two On-Demand Routing Protocols for Ad hoc Networks." To appear in *IEEE Personal Communications Magazine* special issue on Ad hoc Networking.

- E. M. Royer and C. E. Perkins. "Transmission Range Effects on AODV Multicast Communication." To appear in *ACM Mobile Networks and Applications* special issue on Multipoint Communication in Wireless Mobile Networks.

- E. M. Royer, S.-J. Lee and C. E. Perkins. "The Effects of MAC Protocols on Ad hoc Network Communications." To appear in *Proceedings of the IEEE Wireless Communications and Networking Conference*, Chicago, IL, September 2000.

- E. M. Royer and C. E. Perkins. "An Implementation Study of the AODV Routing Protocol." To appear in *Proceedings of the IEEE Wireless Communications and Networking Conference*, Chicago, IL, September 2000.

- C. E. Perkins and E. M. Royer. "Ad hoc On-Demand Distance Vector Routing." Ad hoc Networking, Addison-Wesley Publishers, Ed. Charles Perkins, 2000.

- E. M. Royer, L. E. Moser, and P. M. Melliar-Smith. "Wireless Mobile Networking." Encyclopedia of Distributed Computing, Kluwer Academic Publishers, 2000.
- S. R. Das, C. E. Perkins, and E. M. Royer. "Performance Comparison of Two On-demand Routing Protocols for Ad hoc Networks." *Proceedings of the IEEE Conference on Computer Communications*, Tel Aviv, Israel, March 2000, pp. 3-12.
- E. M. Royer and C. E. Perkins. "Multicast Operation of the Ad hoc On-Demand Distance Vector Routing Protocol." *Proceedings of the* 5^{th} *Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle, WA, August 1999, pp. 207-218.
- E. M. Royer and C.-K. Toh. "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks." *IEEE Personal Communications Magazine*, April 1999, pp. 46-55.
- C. E. Perkins and E. M. Royer. "Ad hoc On-Demand Distance Vector Routing." *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90-100.
- C. E. Perkins, E. M. Royer, and S. R. Das. "Ad hoc On-Demand Distance Vector (AODV) Routing." *IETF Internet Draft*, draft-ietf-manet-aodv-07.txt, November 2000 (Work in Progress).
- C. E. Perkins, E. M. Royer, and S. R. Das. "Ad hoc On-Demand Distance Vector (AODV) Routing for IP version 6." *IETF Internet Draft*, draft-ietf-perkins-aodv6-00.txt, November 2000 (Work in Progress).
- E. M. Royer, C. E. Perkins. "Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing." *IETF Internet Draft*, draft-ietf-manet-maodv-00.txt, July 2000 (Work in Progress).
- C. E. Perkins, E. M. Royer, and S. R. Das. "IP Broadcast in Ad hoc Mobile Networks." *IETF Internet Draft*, draft-ietf-perkins-bcast-00.txt, July 2000 (Work in Progress).
- C. E. Perkins, E. M. Royer, and S. R. Das. "Quality of Service in Ad hoc On-Demand Distance Vector Routing." *IETF Internet Draft*, draft-ietf-manet-qos-00.txt, July 2000 (Work in Progress).
- C. E. Perkins, E. M. Royer, and S. R. Das. "Ad hoc Address Autoconfiguration". *IETF Internet Draft*, draft-ietf-perkins-autoconf-00.txt, July 2000 (Work in Progress).
- K. Berket, R. K. Budhia, K. P. Kihlstrom, R. Koch, N. Narasimhan, P. Narasimhan, E. M. Royer, M. D. Santos, A. Shum, and E. Thomopoulos. "On Technologies in Computer Networks and Distributed Systems," *looking.forward* supplement to *IEEE Computer*, 5(3):2–6 (Fall 1997).

ABSTRACT

Routing in Ad hoc Mobile Networks:
On-Demand and Hierarchical Strategies

by

Elizabeth M. Royer

Mobile wireless communication is becoming increasingly popular due to recent advances in wireless devices and applications. A desirable capability of portable devices is the ability for communication between them. Collectively, these devices can form an ad hoc network. An ad hoc network is a group of mobile nodes with no fixed infrastructure; nodes communicate directly between one another over wireless channels. Because the transmission range of these nodes is limited, a routing protocol is needed to enable communication between them. However, because of the portable nature of these devices and the wireless transmission medium, ad hoc networks have many characteristics that render routing protocols designed for wired networks inapplicable.

This dissertation presents the Ad hoc On-Demand Distance Vector Routing (AODV) protocol. AODV is a reactive protocol in that it discovers routes only when they are needed by a source node. It provides unicast, multicast, and broadcast communication capability and is able to maintain routes even when the topology of the network is dynamic. AODV is well-suited for mobile wireless networks in that it has low processing and memory overhead, and low network utilization. Additionally, AODV provides loop freedom for all routes through the use of sequence numbers.

To improve the scaling potential of AODV, as well as that of many other reactive routing protocols, a set of hierarchical routing protocols has been developed. Hierarchical routing records routes logically across hierarchical levels, rather than across physical links. Hierarchical links tend to be longer-lived, and so these routes are more robust to node movement. This dissertation presents the Adaptive Routing using Clusters (ARC) protocol, as well as the Adaptive Routing using Cluster Hierarchies (ARCH) protocol. The former creates a one-level cluster structure across the network, while the latter can adjust dynamically the depth of the routing hierarchy in response to the network topology.

The performance of a protocol is not based solely upon its design. There are many other factors that contribute to the overall performance of the protocol. These factors include the density of the network, the MAC layer protocol selected, and the number of nodes in the network. Studies of these interactions are presented to determine their effect in ad hoc mobile networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Wireless networking has existed since the early days of the DARPA packet radio network [43]. Although it has been in existence for many decades, mobile wireless networking has only recently begun to gain popularity, due to advances in both wireless devices and applications. These devices, such as laptop computers, Personal Data Assistants, pagers and cellular telephones, have only recently become lightweight and portable enough to be conveniently carried by a user. Advances in battery technology have also enabled these devices to be used for increasingly longer periods of time away from electrical power sources.

Wireless devices can communicate between each other using either infrared ports or radio modems. The applications of infrared are somewhat limited due to the line-of-sight requirement and the low data rate characteristic of these waves. Radio modems, on the other hand, have the capability of higher transmission ranges and data rates, although they still suffer from multipath interference and fading [63]. Current wireless modems offer a wide range of transmission power and connectivity levels. Radio modems can currently transmit at rates up to 11 Mb/s at ranges as great as to 600m, depending on the data rate and the surrounding environmental conditions.

Mobile networks have many unique characteristics that make traditional routing protocols inapplicable. The topology of a mobile network is often highly dynamic due to the mobile nature of the nodes. Whereas a broken link in a wired network is considered an exception and is likely to occur relatively infrequently, links within wireless networks tend to frequently break as nodes move in and out of transmission range of one another. Furthermore, atmospheric effects (such as rain) and physical objects also play a role in limiting the communication between wireless nodes. Additional characteristics of mobile wireless networks include limited power and bandwidth, and high error rates due to the wireless transmission.

Routing protocols design for wired networks generally do not perform well over wireless channels. Traditionally, routing protocols designed for wired networks have been either distance-vector [59] or link-state [61] protocols. These protocols maintain routes from each node to every other node in the network through the periodic exchange of routing table

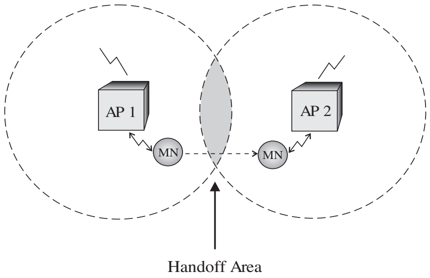


Figure 1.1: Handoff Between Access Points.

messages. Early routing protocols designed for ad hoc networks attempted to adapt these basic protocols for mobile scenarios [62, 71]. However, it was determined that in many mobile networks, link breaks occur too frequently to maintain global topological knowledge. The addition of triggered updates to aid in distributing accurate routing information often results in increased consumption of network bandwidth and makes route convergence difficult, if not impossible. Some current approaches to ad hoc routing make additional modifications to the basic distance vector or link state algorithm to make them more suitable for mobile networks. These modifications include the use of a prioritized connectivity information exchange algorithm, whereby information about parts of the network more distant from the sending node is sent less frequently than information about neighboring nodes [42, 67, 95]. A similar approach [12] sends update information to only those nodes that actually need the information. These protocols have the benefit of a reduction in routing update overhead, as compared with the basic link state and distance vector algorithms. However, they still have the drawback that they require updates based on node movement, which can result in significant control overhead and bandwidth consumption in a mobile network.

For similar reasons, multicast protocols designed for wired networks are not well-suited for operation in mobile networks. For instance, the use of core based trees has the drawback that all branches emanate from a single node, the core [10]. In a mobile network, it may frequently happen that nodes temporarily disappear behind trees or buildings. Algorithms with a single point of failure are likely to suffer from the frequent temporary disconnections witnessed by mobile networks. Protocols designed for *dense mode* multicast are likely to have too much overhead for use in mobile networks, due to the frequent node movements. For instance, in PIM-Dense Mode [26], prune packets are sent whenever a multicast packet arrives via a “wrong” tunnel. Multicast packets are only accepted and forwarded when they arrive over the “right” tunnel. In a mobile environment, the optimal path to some destination may change from moment to moment, and there is no way of knowing the “right” tunnel at any given time. Additionally, due to the inherent difficulty of routing in mobile networks, multicast nodes should accept data packets destined for them from whichever direction they arrive.

Mobile wireless networks have numerous advantages over their traditional wired counterparts. Wireless networks can be established in areas of the world without pre-existing wired infrastructure. Installing cellular infrastructure is much cheaper than burying cables, making wireless networks an attractive option in developing nations [80]. Additionally, wireless systems allow instantaneous network setup. Conference attendees can turn on their laptop computers and immediately network with and share files among the other attendees. Wireless networks allow anywhere/anytime connectivity. These same conference attendees

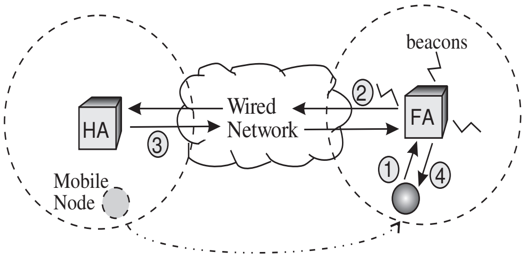


Figure 1.2: Mobile IP.

can sit in the lobby and surf the Web and check their email if the conference center is equipped with wireless access points. Similarly, a business person from one company can check his/her email while at another company without being able to access the company's network files. The mobile piece of the wireless network allows users flexibility of movement while using these devices. A user can walk down a hallway in an office building and maintain connectivity without having to worry about finding an Ethernet connection once the destination is reached. Additionally, a global positioning system (GPS) device can be used in a car to provide the driver with current directions and location information, even while the car is in motion. Finally, wireless networks can result in the elimination of wire clutter in office spaces by reducing the need for printer cables, Ethernet, mouse cables, etc.

1.2 Infrastructured Wireless Networks

The previous section illustrated two distinct types of wireless networks: infrastructured and infrastructureless. Infrastructured wireless networks have a wired backbone of stationary nodes that are connected to the rest of the network or the Internet. These stationary nodes are often called either base stations (BS) or access points (AP). Mobile nodes communicate directly to these access points and generally do not establish point-to-point connections with other mobile nodes. Each AP has a coverage area, or cell, in which it is able to send signals to, and receive signals from, other nodes. This coverage area is dependent on the AP's transmission radius. Nodes within the cell of an AP are able to communicate directly with that AP. Because the mobile nodes are likely to be moving, it is possible that they will not always stay in the coverage area of a single AP. As the mobile node moves from the coverage area of one AP into that of another, a *handoff* occurs, where the node ceases communication with the old AP and begins communicating with the new AP. Figure 1.1 illustrates the handoff of a mobile node from one AP to another. The handoff should be completely seamless so that the user is not aware that a transition in coverage areas has occurred [18, 17]. Infrastructured wireless networks are commonly used in office buildings and college campuses, or locations where the access points can be easily installed and connected to an existing network.

As long as a mobile node stays within its home network, it should be able to access the Internet, receive email, etc, as if it were a wired node on the network. However, once the node leaves its local network, routing difficulties commence because the subnet IP address of the node and the network to which it has moved are likely to differ. Hence, the node can no longer receive packets addressed to it. To solve this problem, MobileIP [69, 70] has been developed. MobileIP is an extension of IP that enables the mobile node to utilize two IP

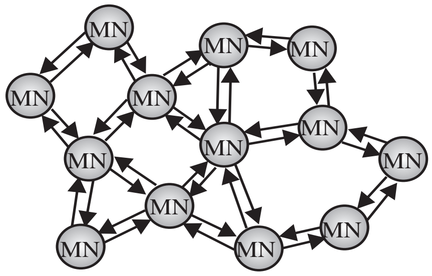


Figure 1.3: Example Ad hoc Network.

addresses. The first is for identification, and the second is for routing. These addresses allow nodes to send and receive data in networks other than their own local home network. Such non-local networks are referred to as foreign networks. Mobile IP works by distinguishing certain nodes as home agents and others as foreign agents. A home agent is a router on a node's home network that tracks the location of mobile nodes while they are connected to other networks. Foreign agents periodically emit beacons indicating their foreign agent status. A mobile node in a foreign network hears these beacons and registers with the foreign agent, as indicated by the line marked '1' in Figure 1.2. The foreign agent then contacts the node's home agent (line '2' in the figure), and a *tunnel* is established whereby packets destined for the mobile node that arrive at the node's home network are encapsulated and then tunneled to the foreign agent on the foreign network (line '3'). Upon receiving the packet, the foreign agent takes the extra header off of the packet and then forwards it to the mobile node (line '4'). Packets initiated by the mobile node are sent directly to the intended recipient. The Mobile IP working group [2] of the IETF [1] oversees the standardization of the Mobile IP protocol.

1.3 Infrastructureless Wireless Networks

In contrast to infrastructured networks, infrastructureless networks, also called ad hoc networks, have no fixed wired backbone; all nodes are capable of movement. Communication in ad hoc networks is peer-to-peer as the mobile nodes communicate directly with one another. Because the transmission range of the nodes is likely to be limited, it is often the case that multiple hops are needed to connect a given source and destination. Hence, the nodes must serve as routers for other nodes in the network so that data packets can be forwarded to their destinations. Figure 1.3 illustrates an example of an ad hoc network. Because there is no wired backbone along which routing can occur, an ad hoc network needs a routing protocol that can establish and maintain routes, even in networks with dynamic topologies.

As previously described, mobile nodes have many unique characteristics that make traditional routing protocols inapplicable. Because of the limitations of mobile wireless nodes, an ad hoc routing protocol should be able to provide routes with a minimum of control overhead and no data overhead, and it should require as little processing time as possible. Furthermore, due to the characteristics of wireless transmission, the range of the nodes is often limited. It is usually the case that paths between sources and destinations require multiple hops. Hence a routing protocol must be able to find multi-hop paths between nodes. The protocol should also be self-starting, and it should be loop free at all times, because even a temporary routing loop wastes the already scarce bandwidth. The Mobile Ad hoc Networking Working Group (`manet`) [58] of the IETF [1] develops and standardizes routing protocols specific to ad hoc networks. Numerous protocols [5, 6, 15, 37, 54, 64, 74] have been proposed for consideration of standardization.

Ad hoc mobile networks generally have reduced administrative cost compared to wired networks. These networks are self-configuring, and so they are able to maintain network connections and routing information without the need for explicit route set-up by a system administrator. Typical examples of ad hoc networks occur in emergency, search and rescue operations, and conference scenarios where attendees want to be able to easily share files. Ad hoc networks are also suitable for networking in locations without an existing wired infrastructure, such as data collection in open fields, and sensor networks [29, 93].

1.4 Contributions and Applicability

This dissertation addresses the problem of providing communication between nodes in an ad hoc mobile wireless network. The topic is addressed at the network layer; methods for the discovery and maintenance of routes between nodes wishing to communicate are presented. The first protocol, the Ad hoc On-Demand Distance Vector (AODV) protocol, provides unicast, multicast and broadcast communication in ad hoc mobile networks. To improve the scalability of AODV and the class of on-demand routing protocols, a set of clustering protocols has also been designed. The first of these, the Adaptive Routing using Clusters (ARC) protocol, creates a one-level cluster topology on the network nodes. The second protocol, the Adaptive Routing using Cluster Hierarchies (ARCH) protocol, creates a multi-level hierarchy which is able to dynamically adjust the depth of the hierarchy in response to the network topology.

These protocols were designed for ad hoc mobile networks with the desired characteristics stated in the previous section taken into consideration. The protocols were designed to provide maximum data delivery with minimal overhead. Although AODV and ARC/ARCH do not depend specifically on particular aspects of the physical medium across which packets are disseminated, their development has been largely motivated by limited range broadcast media such as those utilized by infrared or radio frequency wireless communications adapters. Using such media, a mobile node can have neighbors which hear its broadcasts and yet do not detect each other (the *hidden terminal* problem [96]). The protocols do not make any attempt to use specific characteristics of the physical medium, nor do they handle specific problems posed by channelization needs of radio frequency transmitters. Nodes that need to operate over multiple channels are presumed to be able to do so. The algorithms work on wired media as well as wireless media, as long as links along which packets may be transmitted are available. The only requirement placed on the broadcast medium is that neighboring nodes can detect each others' broadcasts. The protocols use only symmetric links between neighboring nodes; they do not attempt to follow paths between nodes when one of the nodes cannot hear the other one.

Both AODV and the clustering protocols were designed to run over IP. The control packets created by the protocols are given an IP header and normal IP routing and forwarding applies. Additionally, the protocols were designed to operate independent of the protocols used in the other layers of the OSI protocol stack. AODV and ARC/ARCH can operate over any MAC layer protocol. AODV is able to utilize MAC layer feedback about unreachable next hops to maintain an up-to-date routing table. When this information is not available, it is able to determine neighbor connectivity itself. The operation of the clustering protocols is independent of the MAC layer; their operation does not change based upon the available MAC protocol. AODV and ARC/ARCH are also autonomous from the transport layer protocol in that they can provide routing information for either UDP or TCP sessions.

While AODV and the clustering protocols operate independently of the underlying network configuration and MAC protocol, the characteristics of these layers do effect the performance of the protocols. The transmission range of the network nodes and the resulting density of the network effects the performance of the protocols, as does the MAC layer. The effects of these interactions is also presented.

Chapter 2

The Ad hoc On-Demand Distance Vector Routing Protocol

2.1 Introduction

The Ad hoc On-Demand Distance Vector Routing protocol (AODV) can be called a *pure on-demand route acquisition* system; nodes that do not lie on active paths neither maintain any routing information nor participate in any periodic routing table exchanges. Furthermore, a node does not have to discover and maintain a route to another node until the two need to communicate, unless the former node is offering its services as an intermediate forwarding station to maintain connectivity between two other nodes.

When the local connectivity of the mobile node is of interest, each mobile node can become aware of the other nodes in its neighborhood by the use of several techniques, including *local* (not system-wide) broadcasts known as *Hello* messages. The routing tables of the nodes within the neighborhood are organized to optimize response time to local movements and provide quick response time for requests to establish new routes. AODV's primary objectives are:

- To provide unicast, multicast, and broadcast communication to all nodes in the ad hoc network.
- To minimize the transmission of control packets.
- To disseminate information about link breaks to those neighboring nodes that utilize that link.

AODV uses a broadcast route discovery mechanism [23], and relies on dynamically establishing route table entries at intermediate nodes. To maintain the most recent routing information between nodes, the concept of destination sequence numbers is borrowed from from DSDV [71]. Unlike in DSDV, however, each ad hoc node maintains a *monotonically* increasing sequence number counter that is used to supersede stale cached routes. The combination of these techniques yields an algorithm that uses bandwidth efficiently (by minimizing the network load for control and data traffic), is responsive to changes in topology, and ensures loop-free routing.

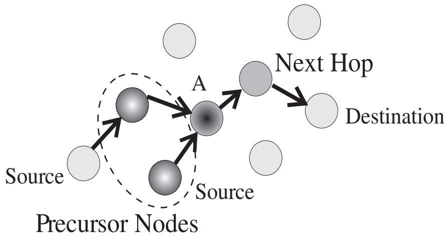


Figure 2.1: Precursor Nodes.

2.2 Unicast Route Establishment

Route discovery with AODV is purely on-demand and follows a route request/route reply query cycle. When a node needs a route to a destination, it broadcasts a *Route Request* (RREQ). Any node with a current route to that destination (particularly the destination itself) can unicast a *Route Reply* (RREP) back to the source node. Route information is maintained by each node in its route table. Information gleaned through RREQ and RREP messages is kept with other routing information in the route table. AODV uses sequence numbers to eliminate stale routes. Routes with old sequence numbers are aged out of the system.

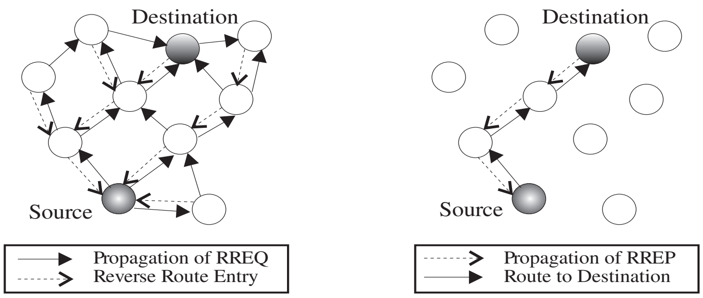
2.2.1 Routing Tables

Each node running AODV maintains a Route Table that is used for recording the next hop for routes to other nodes in the network. The fields of the route table are as follows:

- Destination IP Address
- Destination Sequence Number
- Next Hop IP Address
- Hop Count to Destination
- Last Hop Count
- List of Precursors
- Lifetime
- Routing Flags
- Interfaces

New entries are placed in the route table following the reception of RREQs and RREPs. When a node receives one of the listed message types and it does not already have a route entry for the source of the message, it places an entry in the table listing the indicated information.

Associated with each destination is a sequence number. If a node receives routing information for a destination, and it already has a route table entry for that destination, it updates the entry only if the destination sequence number associated with the new information is greater than that already contained in the route table entry. The use of sequence number guarantees that routing loops cannot form, even under extreme conditions of out-of-order packet delivery and high node mobility (see Appendix A). Additionally, each entry has a corresponding *lifetime* that indicates the length of time the route entry is valid. Routes are invalidated if they are not updated or used within the indicated lifetime.



(a) RREQ Broadcast (b) Path of RREP

Figure 2.2: Route Discovery.

When a route entry expires, the Hop Count value is copied into the Last Hop Count field (see Section 2.2.2), and the Hop Count is set to ∞. The Destination Sequence Number field is also incremented. The List of Precursors is used for sending a Route Error (RERR) message when a link expires (Section 2.2.5). Figure 2.1 illustrates an example of typical routes in a mobile network. In the figure, node A's precursor nodes for the given destination are circled.

2.2.2 Route Discovery

Route discovery with AODV is on-demand and occurs when a node requires a route to a destination for which it does not already have a recorded route. Such a node initiates route discovery by broadcasting a RREQ packet. The message format of the RREQ is as follows:

$$< Flags,\ Hop\_Count,\ Broadcast\_ID,\ Source\_Addr,\ Source\_Seq\#,\\ Dest\_Addr,\ Dest\_Seq\# >$$

The only flag used for unicast is the *Gratuitous RREP* flag. The source of the RREQ sets this flag when a node generating a RREP for this RREQ should send a gratuitous RREP to the destination node (Section 2.2.3). The *Hop_Count* field is initialized to zero and incremented each time the RREQ is forwarded.

Each node in the network is responsible for maintaining two separate counters: a sequence number and a broadcast ID. The sequence number ensures the freshness of routes to the node. The broadcast ID, together with the source node's IP address, uniquely identifies each RREQ. The sequence number is incremented when the node acquires new neighbor information, and the broadcast ID is incremented for each RREQ the node initiates. The node requesting the route places its IP address, current sequence number, and broadcast ID in the *Source_Addr*, *Source_Seq#*, and *Broadcast_ID* fields, respectively. The IP address of the destination and the last known sequence number for that destination are placed in the *Destination IP Address* and *Dest_Seq#* fields.

A node receiving a RREQ first updates its route table to record the sequence number and next hop information for the source node. This *reverse route* entry may later be used to relay a RREP back to the source. The node then checks this table to see whether it has a route to the requested destination. In order to respond to a RREQ, a node must either be the destination itself, or must have an unexpired route to the destination with a sequence number at least as great as that indicated in the *Dest_Seq#* field of the RREQ. A node having such a route is said to have a *fresh enough* route to the destination. If this is the case, the node generates a RREP as described in Section 2.2.3. Otherwise, it rebroadcasts the packet to its neighbors. Figure 2.2(a) illustrates the propagation of the RREQ throughout the network, and the subsequent *reverse route* entries established for the source node.

Whether or not the node is able to generate a RREP, it updates the Destination Sequence Number in the RREQ to the maximum of the existing sequence number in the RREQ and the node's record of the destination's sequence number (if one exists) in its route table. This ensures that the RREQ always carries the greatest known value of the destination sequence number.

A node may receive the same RREQ multiple times. When a node receives a RREQ, it records the source IP address and broadcast ID of the packet. If it later receives a RREQ with this same information, it does not process the packet but instead discards it.

Expanding Ring Search

Each time a node initiates a route discovery process for some new destination, it must broadcast a RREQ across the network. For a small network, the impact of this flooding is minimal. However, for a large network, the impact may become increasingly detrimental. To control network-wide broadcasts of RREQs, the source node can use an expanding ring search technique. This technique allows the source node to search increasingly larger areas of the network if a route to the destination is not found. To use the expanding ring search, the source node sets the Time To Live (TTL) value of the RREQ to an initial value. If no reply is received within the discovery period, the next RREQ is broadcast with a TTL value increased by an increment value. This process of increasing the TTL continues until a threshold is reached. Beyond that value, the RREQ is broadcast across the entire network, up to some maximum number (`rreq_retries`) of additional attempts.

When a new route is established, the distance to the destination is recorded in the routing table. When a route is expired, this distance is copied into the Last Hop Count field of the route entry. If route discovery must later be initiated to this same destination, the initial TTL value for the new RREQ is set to the distance in the Last Hop Count field plus the increment value. This allows the source node to first search the area where the destination was last seen or to which it is likely to have moved. The expanding ring search then continues as previously described.

2.2.3 Forward Path Setup

As stated above, a node can respond to a RREQ if it is the destination itself, or if it has a current route to the destination. When a node fulfills these requirements, it sends a RREP back to the source node. The RREP message format is as follows:

$$< Flags,\ Hop\_Count,\ Source\_Addr,\ Dest\_Addr,\ Dest\_Seq\#,\ Lifetime >$$

The *Acknowledgment* flag is set in cases where the link over which the RREP message is sent may be unreliable. It indicated that a RREP-ACK message should be transmitted by the receiving node to acknowledge receipt of the RREP message. The *Dest_Addr* field is set to the destination address specified in the RREQ, and the *Dest_Seq#* is set to the responding node's record of the destination's sequence number. The *Hop_Count* field is set to the distance of the responding node from the destination, or zero if the destination itself sends the RREP.

The responding node unicasts the RREP to the next hop towards the source node. The node receiving the RREP increments the *Hop_Count* field by one and then creates or updates its entry for the destination node in its route table, thereby establishing the *forward path* to the destination. It then unicasts the RREP to its recorded next hop for the source node. This continues until the RREP reaches the source node. Figure 2.2(b) is an example of the destination node responding by sending a RREP back to the source, and then the subsequent transmission of data packets by the source node. Nodes that are not along the path determined by the RREP delete the reverse pointers after the lifetime associated with that table entry expires.

Once the source node receives the RREP, it can use the route to send data packets to the destination. In the event that the source later receives a RREP with a greater destination sequence number or equivalent sequence number but smaller hopcount, the source node updates its route table information for the destination and instead uses the new route.

If a node forwards a RREP over a link that is likely to have errors, the node can set the *Acknowledgment* flag to require that the recipient of the RREP acknowledge receipt of the RREP by sending a RREP-ACK message back.

It is likely that an intermediate node will receive more than one RREP for a given source/destination pair. In this case, the node checks the *Dest_Seq#* and *Hop_Count* fields against its recorded information. If the destination sequence number is greater than the node's recorded value, or if the sequence number is the same but the *Hop_Count* is smaller, the node updates its information for the destination and forwards the RREP to the source. Otherwise, if the information contained in the RREP is not as good as that which the node already has in its route table and has sent to the source, it discards the RREP and does not forward it.

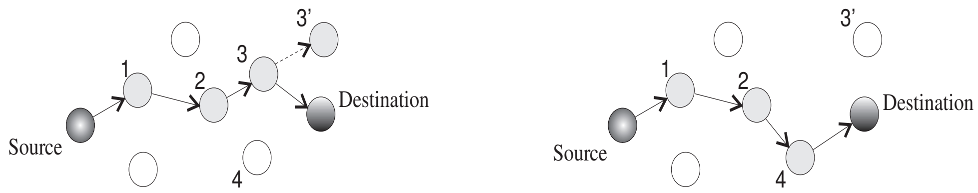
Gratuitous Route Replies

When a node receives a RREQ and responds with a RREP, it does not forward the RREQ any further. If all incarnations of a single RREQ are replied to by intermediate nodes, the destination does not receive a copy of the RREQ. Hence, it does not learn of a route to the source node. This can be problematic if the source is attempting to establish a TCP session. In order that the destination learn of routes to the source node, the source node sets the *Gratuitous RREP* flag in the RREQ if the session is going to be run over TCP, or if the destination should receive the gratuitous RREP for any other reason. Intermediate nodes receiving a RREQ with the *Gratuitous RREP* flag set and responding with a RREP also unicast a gratuitous RREP to the destination node.

The IP address of the destination is placed in the *Source_Addr* field of the gratuitous RREP, while the IP address of the source node is placed in the *Dest_Addr* field. The node initiating the gratuitous RREP places its distance from the source node in the *Hop_Count* field, and then unicasts this packet to the next hop along the path to the destination node.

2.2.4 Local Connectivity Management

Nodes learn of their neighbors through packet transmissions. When a node sends a packet, its neighbors hear the transmission and update their local connectivity information to ensure that it includes this neighbor. Nodes can detect the loss of a neighbor through different



(a) Link Break Along Active Route (b) Repaired Route

Figure 2.3: Route Repair.

mechanisms. If a reliable MAC (level-2) layer protocol such as IEEE 802.11 [27] is used, nodes can rely on MAC layer feedback for the detection of broken links to next hops. Otherwise, if no such MAC layer is available, the proactive method of Hello messages may be used.

To utilize Hello messages, a node broadcasts to its neighbors a Hello message if it has not transmitted anything within the last `hello_interval` msec. This informs its neighbors that the node is still within their transmission range. A Hello message is a special unsolicited RREP which contains a node's IP address and current sequence number. The Hello message is prevented from being rebroadcast outside the neighborhood of the node because it contains a TTL value of 1. Neighbors that receive this packet update their local connectivity information to include the node. The failure to receive any transmissions from a neighbor in the time defined by the periodic transmission of `allowed_hello_loss` Hello messages is an indication that the local connectivity has changed, and the route information for this neighbor should be updated (Section 2.2.5).

2.2.5 The Route Error Message

When a link break with a neighbor occurs, the node first copies the old distance to the destination into the Last Hop Count route table entry for that node. This field is used if a route to that destination must be rediscovered, as described in Section 2.2.2. The node then marks that link as invalid by setting the distance to the lost neighbor equal to ∞ and incrementing the former neighbor's sequence number. It then checks its route table to determine whether the lost node was a next hop for any active routes. An active route is defined as a route which has recently been used to send data packets. If such an active route exists, and if one of the active routes contains one or more precursor nodes, the node generates a *Route Error* (RERR) message, the format of which is as follows:

$$< DestCount,\ Unreach\_Dest\_Addr,\ Unreach\_Dest\_Seq\#... >$$

The RERR message contains an entry for each destination that was lost due to the link break. Associated with each included destination is that destination's sequence number, incremented by one. The *DestCount* field indicates the number of such included destinations. The node broadcasts this packet to its neighbors, and invalidates each of these routes in its own route table.

When neighboring nodes receive the RERR message, they invalidate the routes contained in the RERR *if* these routes use the source of the RERR as a next hop. If this is the case, and if one or more of these routes contains a precursor node entry, then the node broadcasts its own RERR message with the lost destinations.

Figure 2.3 illustrates the process of generating RERR messages. In Figure 2.3(a), the link between nodes 2 and 3 breaks. Node 2 notes the break and generates a RERR containing both node 3 and the node labeled Destination. When node 1 receives the RERR, it invalidates its route table entries for the routes since node 2 was the next hop for those routes. It then broadcasts its own RERR message. When the source node receives the RERR, it similarly invalidates the indicated routes. If it still needs the route to the destination, it reinitiates route discovery. Figure 2.3(b) illustrates the repaired route through node 4.

A node also generates a RERR message if it receives a data packet for which it does not have a route. It places the IP address of the destination node in the RERR message and then broadcasts the message. It sends one RERR message for each data packet it receives with an unreachable destination.

2.2.6 Related Work

Because wireless ad hoc networking has recently seen an increase in research, there are currently a wide variety of routing protocols for providing unicast communication in ad hoc networks [4, 19, 41, 48, 65, 66, 71, 72, 97]. Some of these protocols have been proposed for standardization by the IETF [5, 15, 34, 37, 64, 74]. Numerous studies have attempted to compare the protocols and determine their relative strengths and weaknesses [16, 24, 25, 31, 40, 55, 56, 89]. The majority of these comparisons are based on simulations.

In terms of protocol operation, Dynamic Source Routing (DSR) [41] is the most similar to AODV. Like AODV, DSR is an on-demand protocol that uses a route discovery cycle to determine routes. The primary differences between the two protocols are DSR's use of source routing and its aggressive route caching policy. In DSR, route requests and route replies build source routes as they traverse the network. Nodes that receive these messages learn routes to each node listed in the source route, thereby increasing their knowledge of the overall network connectivity. Additionally, DSR nodes store their routes in route caches, as opposed to AODV's route tables, and hence may have multiple route entries per destination. Also, unlike AODV, these route cache entries have no associated lifetime. The routes stay in the route table until they are proved faulty.

These differences have a substantial effect on the relative performance of the two protocols. In the paper by Das, Perkins, and Royer [25], an in-depth study is performed to determine the effects of these differences on the results achieved by the protocols. The results show that for low mobilities, DSR out-performs AODV in terms of the packet delivery fraction. However, as node mobility increases, particularly for large numbers of sources, AODV is able to maintain a higher packet delivery fraction than DSR. The reason for this occurrence is due to DSR's aggressive use of route caching. Route caching is beneficial only to a certain extent. Because routes are never timed out, stale routes are often selected once one route breaks. Attempting to use an invalid route causes problems that include the consumption of additional network bandwidth, even though the packet is unable to reach its destination, and the pollution of caches in other nodes with a stale route entry.

One additional performance disparity is in terms of the routing load generated by the two protocols. DSR nodes respond to each Route Request they receive, as opposed to AODV nodes which generally only respond to the first such request. Because of this fact, and because DSR utilizes source routing, DSR nodes have to resort to route discovery for routes far less often than AODV nodes. DSR nodes are more likely to already have a route in their cache. Because route requests are flooded, this causes the control overhead of AODV to appear higher. However, while AODV's routing load is dominated by route requests, DSR's is strongly dominated by route reply and route error messages. Route replies and route errors are sent by unicast, as opposed to the broadcast transmission of route requests. While this fact causes DSR's routing load to be lower than AODV's, this does not translate into reduced network load for DSR. Each hop-wise transmission of the unicast route reply and route error packets requires the RTS/CTS/Data/ACK exchanges required by the IEEE 802.11 MAC layer protocol [27]. When factoring this overhead in, DSR generates as much overall network load as AODV.

While DSR is the closest to AODV in terms of protocol operation, there are other protocols that also operate in a similar manner. Many protocols attempt to use nodal positions to prevent the route discovery from flooding the entire network. One such protocol is the Relative Distance Micro-discovery protocol (RDMAR) [4, 5]. Like AODV and DSR, the RDMAR algorithm utilizes a route discovery cycle. However, RDMAR differs from the other protocols in that the TTL field in the route request is adjusted so that the packet is not flooded across the entire network. Instead, the TTL is calculated by knowing the previous distance of the destination from the node initiating the search, together with an estimate of the node's velocity and the average transmission range. Additionally, only the destination node can respond to a route request packet, instead of any node with a current route to the destination, as in AODV and DSR.

Another algorithm that takes advantage of nodal positions to prevent route request floods is the Location-Aided Routing protocol (LAR) [48]. LAR utilizes the Global Positioning System (GPS) to provide location information. The previous location of a node, together with the speed of that node, is used to determine an *Expected Zone* where the node is likely to be at the current time. A *Request Zone* is then calculated, whereby nodes only forward the route request if they are located in this area. The disadvantage to this approach is that all nodes must have a GPS system in order for the protocol operation to be successful.

The Temporally Ordered Routing Algorithm (TORA) [64, 65] approaches routing through a different philosophy by building a directed acyclic graph (DAG) during the route creation and maintenance phases. This DAG is rooted at the destination node and is created through the use of a height metric. TORA provides multiple routes between a given source/destination pair. When the last path to a destination is broken, route maintenance is triggered and the DAG is reestablished at the destination.

The Associativity-Based Routing protocol (ABR) [97] takes a different approach to route selection in that it defines a new metric called the *degree of association stability.* Each node periodically broadcasts a beacon. When received by neighboring nodes, the associativity tick of the current node with respect to the beaconing node is incremented. During the route discovery cycle, nodes append their associativity tick information to the query packet. When the destination receives the query packet, it selects the path with the highest degree of association stability, or the path likely to be the most stable, and then sends a reply along this path. The other distinguishing feature of ABR is that intermediate nodes along a path may perform local repair of broken links, instead of requiring a notification to be sent back to the source of the route, as in the other protocols. The drawback to this protocol, however, is its requirement of periodic beacons.

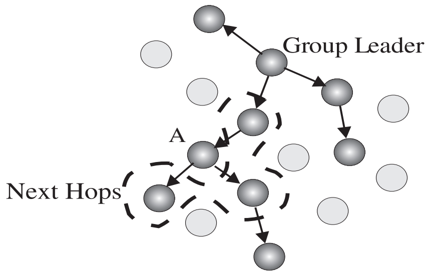


Figure 2.4: Sample Multicast Tree.

2.3 Multicast Route Establishment

The AODV multicast algorithm uses the same Route Request/Route Reply messages as previously described. Only two new messages, the *Multicast Activation* (MACT) and the *Group Hello* (GRPH), are needed. As nodes join the multicast group, a multicast tree composed of group members and nodes connecting the group members is created. Multicast group membership follows the model of the Mbone in that it is dynamic [28]; nodes are able to join and leave at any time. Multicast group members must agree to be routers in the multicast tree. All routers on the multicast tree maintain information about the group in their Multicast Route Table. Each multicast group has associated with it a group leader. The multicast group leader maintains the multicast group sequence number, and disperses the current value to the multicast group through the Group Hello. The following sections describe in detail AODV's multicast operation.

2.3.1 Routing Tables

Each node running AODV must potentially maintain two multicast tables. If a node is either a member of a multicast group or is a router for such a group, it maintains a *Multicast Route Table* (MRT). The MRT is used by nodes to maintain next hop information for the multicast trees. The fields of the MRT are as follows:

- Multicast Group IP Address
- Multicast Group Leader IP Address
- Multicast Group Sequence Number
- Hop Count to Multicast Group Leader
- Hop Count to next Multicast Group Member
- Next hops

There is one entry in the MRT for each multicast group of which the node is either a member or a tree router. Each entry has associated with it a list of one or more next hops, or neighbors on the multicast tree. The next hops field is a linked list of structures, each of which contains the following information:

- Next Hop IP Address
- Link Direction
- Activated Flag

The link direction is defined to be *upstream* if the link is towards the group leader, and *downstream* if it is away from the group leader. Because of the tree structure, a node should have exactly one upstream link at any time. The Activated flag associated with each next hop is an indication of whether the link has been officially added to the multicast tree (see Section 2.3.3). When a link is added to the tree, the flag is set, and only after that time can the link be used for forwarding multicast data packets. In Figure 2.4, node A's next hops on the multicast tree are enclosed by the dashed circle.

Each node running AODV may also maintain a *Group Leader Table* (GLT). The table contains two fields per entry:

- Multicast Group IP Address
- Multicast Group Leader IP Address

When a node receives a Group Hello (Section 2.3.2), it updates its GLT to reflect the multicast group/group leader association indicated in the Group Hello message. If the node later wants to join a multicast group, it first checks its GLT for an entry for that group. If there is such an entry, and if the node has a route to the multicast group leader, it may unicast its Route Request to the group leader instead of broadcasting it across the network. This table is used solely as an optimization; its elimination does not affect the correct operation of the protocol.

2.3.2 The Group Leader

Each multicast group has associated with it a group leader. When a node wishes to join a multicast group, it broadcasts a RREQ and then waits for a reply. If after some maximum number of attempts (`rreq_retries` + 1) it does not receive a reply, it may assume that there are no other members of the group in its connected partition of the network. It then becomes the group leader for that multicast group and initializes the sequence number to one. Once it becomes the group leader, it broadcasts a *Group Hello* (GRPH) message. This message contains the following fields:

$$< Flags,\ Hop\_Count,\ Source\_Addr,\ MGroup\_Addr,\ MGroup\_Seq\# >$$

Currently, there are two defined flags. The first of these is the *Update* flag. This is set when there is a change in group leader information, as described in Section 2.3.5. The second flag is *OffMtree*. When the group leader initiates the GRPH, it leaves this flag unset. Whenever a node *not* on the multicast tree receives the message, it sets this flag. This indicates that the GRPH message has traveled off the tree along this path. When a node on the multicast tree receives a GRPH with the *OffMtree* flag unset, it knows that the GRPH has traveled solely on tree links, and so the *Hop_Count* field can be used to update the node's current distance from the group leader. Otherwise, if a multicast tree node receives the packet with that flag set, it knows it cannot use the *Hop_Count* value as an indicator of its distance from the group leader, because the packet has not traveled only along the tree. The importance of this message being broadcast instead of multicast across the tree is discussed in Section 2.3.5.

The *Hop_Count* field is incremented each time the GRPH packet is forwarded. The *Source_Addr* field is set to the group leader's IP address, and the *MGroup_Addr* and *MGroup_Seq#* fields are set to the multicast group IP address and current sequence number, respectively. The group leader increments the group sequence number each time it initiates a new GRPH message.

When a node receives the GRPH packet, it records the multicast group IP address and sequence number before rebroadcasting the packet. If it later receives a GRPH with this same multicast group IP address/sequence number combination, it knows it has already seen this GRPH message and it can drop the packet. If, on the other hand, a node receives a GRPH packet it has not seen previously, it updates its GLT to reflect the current group/group leader combination.

A group leader change occurs when the current group leader either decides to unsubscribe from the group, or when the multicast tree becomes partitioned. These scenarios are described in Sections 2.3.4 and 2.3.5, respectively.

2.3.3 Subscribing to the Multicast Group

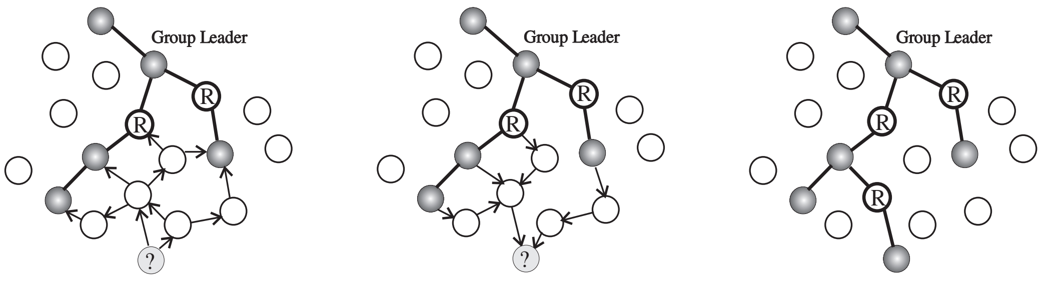
A route discovery cycle is initiated each time a node would like to find a route to a multicast group. A node may decide to initiate route discovery because it would either like to subscribe to a new group, or because it would like to begin sending to a group of which it is not already a member. The node initiates route discovery by broadcasting a RREQ. It then waits the discovery period for the reception of a *Route Reply* (RREP) packet. At the end of the discovery period, the node unicasts a *Multicast Activation* (MACT) message to its selected next hop to activate the route.

Route Requests

When a node wishes to subscribe to a multicast group or to find a route to a group of which it is not already a member, it initiates route discovery by broadcasting a RREQ. The RREQ for multicast has the same structure as that for unicast (Section 2.2.2).

The currently defined flags for the RREQ which are used for multicast are *Join* and *Repair*. The *Join* flag is set when the node wishes to join the group, as opposed to just find a route to the group. The *Repair* flag is set when the RREQ is for a repair to the multicast tree (Section 2.3.5). The *Hop_Count* field is initialized to zero and incremented each time the RREQ is forwarded. The *Broadcast_ID* is an unsigned integer set by the source node. Each time the source initiates a new RREQ, it increments the *Broadcast_ID* by one. The *Broadcast_ID*, together with the *Source_Addr*, forms a unique identifier for the RREQ. The *Dest_Addr* field is the IP address of the desired multicast group, and the *Dest_Seq#* is the source's record of the last known sequence number for the multicast group.

Each node maintains a list of source addresses and broadcast IDs for RREQs it has recently processed. When a node receives a RREQ, it first ensures that it has not already processed this RREQ. If it has, it discards the packet. If it has not already seen the RREQ, it begins by updating both its route to the source node and its record of the source's sequence number. Once a node has created or updated the reverse route entry, it then creates a MRT entry for the multicast group, if one does not already exist. Into this entry it places the identifier of node from which it received the RREQ as a downstream next hop. The Activated flag associated with this entry is left unset.



(a) RREQ Propagation (b) RREPs Returned to Source (c) Multicast Tree Branch Addition

Figure 2.5: Multicast Group Join.

After creating the table entries, the node determines whether it can respond to the RREQ. If the RREQ is *not* a join request, any node with a current route to the multicast group can respond by sending a RREP. A current route is a route to the multicast group whose associated sequence number is no less than that in the *Dest_Seq#* field of the RREQ. If this is the case, the node creates a RREP, as described in Section 2.3.3, and unicasts it back to the source. Otherwise, if the node does not have a current route to the destination, it rebroadcasts the request.

On the other hand, if the RREQ *is* a join RREQ, only a node that is a member of the multicast tree may respond to the RREQ. Because the RREP sets up a potential branch addition to the multicast tree, only nodes that are actually members of the multicast tree may initiate the RREP. When a node that is a member of the multicast tree receives the RREQ, it notes the node from which the RREQ arrived, and creates a next hop entry in its MRT for that previous hop. The Activated flag for that next hop is left unset. It then unicasts the RREP to the source node. Figure 2.5(a) illustrates the propagation of a join RREQ throughout the network. In the figure, the nodes labeled 'R' are tree routers, while the shaded nodes are the group members.

To reduce the impact of route discovery, the RREQ may be sent in an expanding ring search for the destination, as described in Section 2.2.2.

Route Replies

Once a node has decided that it can respond to a RREQ, it creates a RREP and unicasts it back to the source node. The multicast RREP contains the same fields as that for unicast (Section 2.2.3). The only RREP flag needed for multicast is the *Repair* flag. This flag is set when the RREP is in response to a repair request (Section 2.3.5). If the node generating the RREP is a member of the multicast tree, the *Hop_Count* field is initialized to zero. Otherwise, it is set to the responding node's distance from the multicast tree. This field is incremented each time the RREP is forwarded, so that when the source node receives the RREP the field indicates the source's distance from the multicast tree. The *Dest_Addr* is set to the multicast group's IP address, and the *Dest_Seq#* is set to the responding node's record of the group's sequence number. The *Source_Addr* is the address of the node that originated the request. The *Lifetime* field is used when the request is not a join request. It is set to the responding node's current lifetime for the multicast group route entry. For nodes on the multicast tree, the multicast group entry itself does not timeout, and hence does not have a lifetime associated with it. Only the individual next hop links may timeout.

If the RREQ is a join request, the RREP also contains an extension called the Multicast Group Information Extension. This extension contains the multicast group leader IP address and another hopcount field called *MGroup_Hcnt*. This hopcount is set equal to the responding node's distance from the group leader. This field is also incremented each time the packet is forwarded, so that when the source node receives the packet, it indicates that node's distance from the group leader.

When a node receives a RREP, it stores the IP address of the node from which it received this packet. The node then adds a next hop entry for the previous node to its multicast route table entry, and leaves the Activated flag associated with this next hop unset. It then forwards the RREP towards the source. If an intermediate node later receives another RREP for the same source node and destination pair, it only forwards the new RREP if that RREP offers a better route to the multicast tree than was previously known. A better route is one with either a greater destination sequence number or the same destination sequence number but a smaller hopcount to the multicast tree. Figure 2.5(b) shows the path of the RREPs sent back to the source node.

After transmitting the RREQ, the source node must wait the full discovery period before selecting a route. During this period, the source node keeps track of the best route (greatest sequence number and smallest hopcount) to the multicast tree. At the end of the discovery period, it selects its next hop and activates it as described in the next section.

Multicast Activation

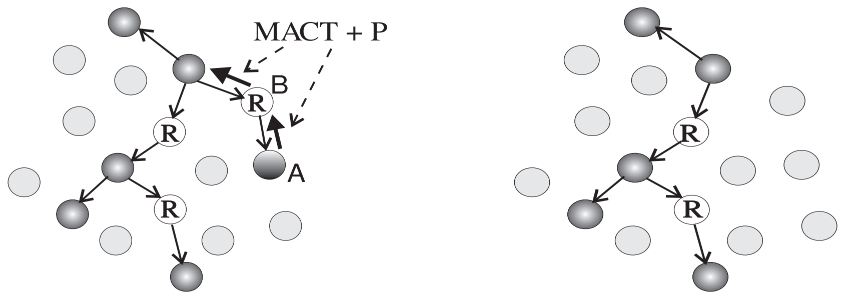
Once the discovery period has ended and the source node has chosen its next hop, it activates this entry in its MRT by setting the Activated flag associated with that next hop. It then creates a *Multicast Activation* (MACT) message, and unicasts this message to its next hop. The MACT message contains the following fields:

$$< Flags,\ Hop\_Count,\ MGroup\_Addr,\ Source\_Addr,\ Source\_Seq\# >$$

The currently defined flags for the MACT message are *Join*, *Prune*, *Grpldr*, *Update* and *Reboot*. The *Join* flag is used when activating a link to indicate whether the link should be grafted on to the multicast tree, while the *Prune* flag is used by a node when it wishes to prune itself from the tree (Section 2.3.4). The *Grpldr* flag is used after a network partition when a new group leader must be selected (Section 2.3.5), and the *Update* field is used after a tree branch repair (Section 2.3.5). The *Reboot* flag is set in the MACT a node broadcasts after rebooting, as discussed in Section 2.6.

The *Hop_Count* field is primarily used after a tree repair (Section 2.3.5). For link activation, this field is set to one. The *MGroup_Addr* field is set to the IP address of the multicast group, and the *Source_Addr* and *Source_Seq#* fields are set to the IP address and current sequence number of the node initiating the MACT, respectively.

If the node is attempting to join the tree, then each RREP it receives sets up a potential tree branch addition. Because only one branch must be added to the multicast tree, that branch must be explicitly selected. The MACT message is used for this purpose. All other non-selected routes to the tree time out and are deleted. Similarly, if the source is finding a route to the tree but not joining the tree, it also must explicitly activate its selected route. Because multicast traffic transmitted as broadcast traffic, neighboring nodes that forward



(a) Pruning of Multicast Group Member

(b) Multicast Tree After Prune

Figure 2.6: Unsubscribing from the Multicast Group.

a RREP to the source node have no way of knowing whether they are selected as next hops for forwarding the multicast data to the multicast tree. When the source broadcasts the data, multiple neighboring nodes could potentially rebroadcast the packet, resulting in an inefficient use of the available bandwidth. Hence, the source node must send a MACT message to its selected next hop to activate that link. Only nodes that receive this MACT message may rebroadcast multicast data packets. This enables unnecessary broadcasts to be suppressed, and bandwidth is more efficiently utilized.

When the source node creates the MACT message, it sets the *Join* flag if it desires to join the multicast tree; otherwise, if it is only finding a route to the tree, it leaves the flag unset. When the next hop receives the MACT message, it activates the next hop entry for the sending node in its MRT. If this node was already a member of the multicast tree, the addition of the new branch to the tree (for a join) or the discovery of a route to the tree (for a non-join) is completed. Otherwise, if this node was not already a member of the multicast tree, then, like the source node, it will also have been keeping track of the best next hop to the multicast tree. It activates this next hop in its MRT, and then unicasts a MACT message to this node. Processing continues in this manner until a node that was already a member of the multicast tree is reached. Figure 2.5(c) shows the multicast tree after the subscribe procedure is completed.

2.3.4 Unsubscribing From the Multicast Group

Multicast group membership is dynamic; nodes can subscribe to or unsubscribe from a multicast group at any time. Because of this, AODV provides a method by which a node may prune itself from the multicast tree. This is accomplished with the MACT message.

Although a multicast group member may unsubscribe from a multicast group of which it is a member at any time, it may exit the *multicast tree* only if it is a leaf node. If a non-leaf node attempted to exit the tree, the tree would then become partitioned. Hence, a non-leaf node that wishes to unsubscribe from the multicast group may change its member status internally, but it takes no overt action to notify any of the other tree members.

When a leaf node decides to unsubscribe from the multicast group, it must also prune itself from the multicast tree. It does this by first deleting the entry for that multicast group from its MRT, and then creating a MACT message with a set *Prune* flag. It then unicasts this message to its next hop.

When the next hop receives the prune message, it deletes the sending node's information from its MRT entry for that multicast group. If, due to the pruning of the sending node, the receiving node is now a leaf node, and if this node is *not* a multicast group member (only a tree router), it can in turn prune itself from the tree in the previously described manner. Otherwise, if it is not a leaf node, or if it is a member of the multicast group, then pruning ends at this node.

Figure 2.6 illustrates pruning. In Figure 2.6(a), node A is a multicast group member that wishes to unsubscribe from the group. Because it is a leaf node, it should also prune itself from the tree. It creates the MACT prune message and unicasts this message to its next hop, node B. When node B receives the message, it deletes the next hop entry for node A, and then notes that it is now a leaf node. Because it is a tree router and not a group member, it prunes itself from the tree as well. Figure 2.6(b) illustrates the multicast tree after pruning.

When the group leader decides to unsubscribe from the group, it operates in a similar manner. If it is a leaf node, it may prune itself from the tree. Otherwise, it must remain a router for the tree. If it is a leaf node, it sends the prune message to its next hop and deletes the multicast group information from its MRT. When the next hop receives the prune message, it is handled as described in Section 2.3.5. Otherwise, if the group leader is not a leaf node, it selects one of its next hops and sends this node a MACT message with set *Grpldr* flag, as is also described in Section 2.3.5.

2.3.5 Multicast Tree Maintenance

Because the network nodes are mobile, links between nodes are likely to break. A multicast tree is maintained for the lifetime of the multicast group. Hence, there must be a way of maintaining the multicast tree in spite of changes in the network topology.

Tree maintenance generally falls into three categories: (i) link break and repair; (ii) link break and subsequent network partition; and (iii) tree merge after a network partition.

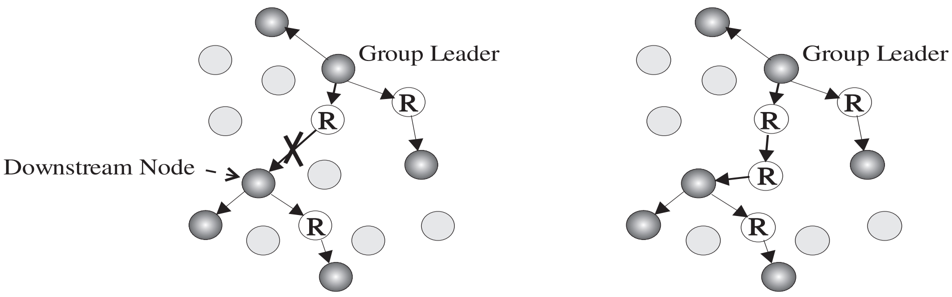
Repairing Link Breaks

Nodes may notice a link break on the multicast tree in one of two ways. If no data packets have been sent recently, then a node must receive a broadcast from each of its next hops every `hello_interval`. This broadcast can be a RREQ, a GRPH, a multicast data packet, or a Hello message. A Hello message is a form of unsolicited RREP, with $TTL = 1$, as described in Section 2.2.4. If a node has not broadcast anything within the last `hello_interval`, it must broadcast a Hello message so that its next hops on the multicast tree know that it is still within transmission radius. Failure to receive any packets from a next hop on the multicast tree for

$$\texttt{hello\_life} = (1 + \texttt{allowed\_hello\_loss}) * \texttt{hello\_interval}$$

indicates that the next hop is out of transmission radius and the link must be repaired.

Alternatively, if the multicast tree has recently been used to send data packets, then a node must hear each of its next hops (except the node from which the packet came) retransmit a data packet within `retransmit_time`. The `retransmit_time` is generally three times the propagation delay through a node. Because reliable packet delivery is not currently



(a) Link Break (b) Repaired Multicast Tree

Figure 2.7: Repair of Multicast Tree Branch.

supported by AODV, a node does not need to hear each of its next hops retransmit each packet; it just must hear it transmit a data packet within that time frame. This special `retransmit_time` is used when the route is active because waiting the full `hello_life` time period to detect a broken multicast tree link would result in a large number of lost packets.

When a link break on the multicast tree occurs, the node *downstream* of the break is responsible for repairing the link. A node knows it is downstream of the break because it knows the direction of each of its next hops in relation to the multicast group leader. Only the downstream node should initiate the repair; if nodes on both sides of the break try to repair the link, they might repair the link through different intermediate nodes, thus forming a loop. The downstream node initiates the repair by broadcasting a RREQ with the *Join* flag set and with a special extension included. This extension, called the Multicast Group Leader Extension, contains a *MGroup_Hcnt* field, which is set equal to the node's current distance from the multicast group leader. Figure 2.7(a) illustrates an example of a multicast tree with a broken link. In the figure, the downstream node sets this field equal to two because it is two hops away from the group leader. When the Multicast Group Leader extension is included, only nodes that are *at least* that close to the group leader can respond. This prevents nodes on the same side of the break as the downstream node from responding to the RREQ, which would prevent the reconnection of the tree. Because the two nodes are likely to still be close by, the downstream node can set the initial TTL value of the RREQ to be small, thereby allowing for a local repair and preventing the RREQ from being broadcast across the entire network.

Because the RREQ has the *Join* flag set, only a node on the multicast tree can respond. When such a node receives the RREQ with this extension field, it checks whether it is at least as close to the group leader as indicated by the *MGroup_Hcnt* field. If so, and if its record of the group sequence number is at least as great as that contained in the RREQ, it can reply to the RREQ by unicasting a RREP back to the initiating node. RREP forwarding and subsequent route activation with the MACT message are handled as described previously. Figure 2.7(b) illustrates the multicast tree after the repair is completed.

Once the repair is finished, it is likely that the node that initiated the repair is now a new distance from the group leader. If so, it must inform its downstream next hops of their new distance from the group leader. The node creates a MACT message, sets the *Update* flag, and sets the *Hop_Count* field equal to its distance from the group leader. It then broadcasts this message. When the downstream nodes receive this message, they increment the *Hop_Count* value and then update their current distance from the group leader. If they are not leaf nodes, they in turn send this update message to their downstream next hops, and so on. Because the MACT is broadcast, the node that is upstream of the broadcasting node also receives the message. In this case, it notes that the packet came from its downstream link, and discards the message.

When a link break occurs, the node upstream of the break also notices the break. It is possible that the tree branch will not be reconnected through that node. If this upstream node is not a group member, and if the loss of that link has made that node a leaf node, it sets a prune timer to wait for the repair. This prune timer should be longer than the route discovery period to allow time for the repair to be completed. If this node does not receive a MACT message activating it as a next hop before the prune timer expires, then it may prune itself at that time.

Network Partitions

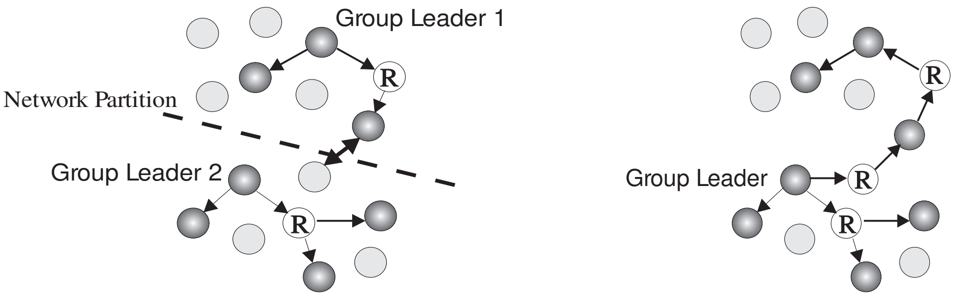
If a node attempting to repair a broken tree link does not receive a RREP within the discovery period, it rebroadcasts its RREQ up to `rreq_retries` additional attempts, using the same expanding ring search as described in Section 2.2.2. If no response is received after this many attempts, the node assumes that the network has become partitioned and that the multicast tree cannot yet be repaired. If this is the case, the multicast tree partition that was downstream of the break is now left without a group leader.

If the node that was trying to repair the break is a multicast group member, then it becomes the new group leader. It broadcasts a GRPH message to announce the group leader change, and sets the *Update* flag in this message to indicate that the change has occurred.

If the node that was trying to repair the break is not a multicast group member, there are two possibilities. The first is that this node has only one downstream link. If this is the case, then the link loss has made this node a leaf on the tree, and so it can prune itself from the tree. It unicasts its next hop a prune message, and then deletes all the group information from its MRT. When the next hop receives the prune message, it deletes the sending node's next hop information from its MRT, and notes that the message came from its upstream link. It is then in the same position as the previous node. If it is a group member, it becomes the new group leader. Otherwise, if it has only one downstream next hop link, it prunes itself from the tree along this link. This process continues until a multicast group member is reached. This node becomes the new group leader.

The second possibility is that the node that was trying to repair the link has multiple downstream branches. In this case, it cannot prune itself from the tree, because doing so would disconnect the tree. Instead, it selects any one of its downstream links and unicasts that next hop a MACT message with set *Grpldr* flag. This flag indicates that the next group member to receive this message should become the new group leader. After unicasting this message, it changes the direction associated with this next hop in its MRT so that the direction is now upstream. If the next hop to receive the MACT is a group member, it becomes the new group leader. Otherwise, it in turn chooses one of its downstream links, and sends that next hop the MACT message with set *Grpldr* flag. It also updates the direction associated with that next hop to be upstream. This process ends once a multicast group member is reached.

Once the new group leader is determined, it broadcasts a GRPH message with set *Update* flag and incremented group sequence number to announce its new status as group leader.



(a) Partitioned Network before the Repair (b) Reconnected Network

Figure 2.8: Merge of Two Components of Multicast Tree.

Merging Two Disjoint Trees

Once a network partition has occurred, there are two group leaders for the same multicast group, each of which periodically broadcasts a GRPH message. If the two network partitions come back into contact with each other, group members learn of this occurrence through the reception of a GRPH message that has information about a new group leader. The two partitions of the multicast tree must then be reconnected. The only node that can initiate the repair of the tree is the multicast group leader with the lower IP address. This distinction is made because if more than one node tried to repair the tree, it is likely that it would be repaired through different intermediate nodes, and thus form a loop.

When the group leader with the lower IP address (GL_1) receives the GRPH, it creates a RREQ with set *Join* and *Repair* flags and *unicasts* this message to the other group leader (GL_2), using the node from which it received the GRPH as the next hop. As the RREQ travels to GL_2, nodes process the packet as they would a regular RREQ with the following exception. If a node that is a member of GL_2's tree receives the RREQ, it *must* forward the RREQ to GL_2 along its upstream multicast tree link. This prevents the formation of routing loops once the RREP is sent.

When GL_2 receives the RREQ, it notes the set *Repair* flag, and creates a RREP to send back to GL_1. It sets the *Repair* flag of this RREP, and then unicasts the RREP back to GL_1. It also updates the multicast group sequence number by taking the larger of its record of the group sequence number and that contained in the RREQ, and incrementing this value by one. The next time GL_2 broadcasts a GRPH message, it includes this new sequence number value, and sets the *Update* flag to indicate a group leader change has occurred.

As the RREP travels back to GL_1, nodes that receive the RREP create the next hop entries and *activate* these entries immediately. Because the RREQ was unicast, there is only one potential tree branch being added to the tree, and so a MACT message does not need to be sent. Hence the next hop entry can be activated immediately. If a node that is on the multicast tree of GL_1 receives the RREP message, it updates its group leader information to reflect GL_2 as the new group leader. This node *must* forward the RREP along its upstream tree link towards GL_1 to prevent routing loops. It then changes the direction of that link to downstream, and marks the link from which the packet arrived as upstream. This link direction change occurs because the new group leader is in a different direction than the previous one. Once GL_1 receives the RREP, it notes that it is no longer the group leader, makes the link addition or direction change, and the tree merge is complete. Figure 2.8 shows an example of a tree repaired in this manner.

2.3.6 Related Work

As with unicast, there have recently been many protocols proposed for providing multicast communication in ad hoc networks [14, 20, 35, 38, 49, 52]. To date there have been only a few attempts to provide a simulation-based comparison of the protocols [55]. The primary difference between the protocols is their approach to connecting the multicast groups. Most protocols provide multicast communication through the formation of either multicast trees or multicast meshes. The multicast trees can be per group or per source, and can be connected either through unicast tunnels or through connecting nodes. One of the primary advantages of multicast trees is the reduction in bandwidth consumed by the forwarding of multicast data packets. Trees generally have only one path to any group member, and so bandwidth consumption can be optimized so that data packets are sent along the direct path to group members. On the other hand, multicast meshes often provide multiple paths between multicast sources and receivers. While this results in excess bandwidth consumption, the increase in redundancy provided by these multiple paths can result in an increased number of data packets that reach the receivers. Multicast meshes are more robust to individual packet losses due to the increased redundancy in data packet forwarding.

The Ad hoc Multicast Routing Protocol (AMRoute) [14] is an example of a tree-based routing protocol. AMRoute creates a shared bi-directional tree that is composed only of group senders and receivers. Tunneling is used to connect the tree members so that non-group members do not need to support multicast. Hence, even when the underlying network topology and routing information changes, the multicast tree stays relatively constant. The primary difference between AMRoute and AODV is AMRoute's use of multicast tunnels. Because multicast packets are encapsulated and sent as unicast packets, network nodes that are not participating in multicast do not need to retain multicast group state or be multicast capable. On the contrary, AODV nodes that are not participating in a multicast group have the potential of being required to be routers for the multicast tree. The drawback to AMRoute, however, is that because it uses tunnels, it is possible that duplicate multicast packet will traverse the same link, thereby reducing efficiency and wasting bandwidth.

The Lightweight Adaptive Multicast protocol (LAM) [38] is an example of another multicast protocol that utilizes the tree-based approach to connecting multicast group members. The LAM protocol is tightly-coupled with the TORA unicast routing protocol [65], and thus takes advantage of TORA's route finding ability to reduce control overhead. Like AODV and AMRoute, LAM creates a shared, bi-directional tree of group members. Where LAM differs, however, is that the trees LAM creates are based at a single core node, and are thus susceptible to a single point of failure.

Chiang, Gerla and Zhang [20] present three different approaches to multicast in wireless networks. Each of the three approaches is based on a shared tree and utilizes a rendezvous point (RP) for establishing group connectivity. In the first approach, a sender encapsulates a multicast data packet and unicasts it to the RP. Upon reception, the RP sends the packet to the multicast group. Their second approach improves upon this model by having the sender actually join the group. In this way, it can transmit the data packet as a multicast packet, and so if any group members lie along the path to the RP, they can process it. The third approach optimizes the multicast trees by allowing the creation of per-source trees, which often have shorter paths between group members than the RP-rooted shared tree. One of the primary differences between these approaches and the AODV protocol is the use of RPs. Because multicast group joins must occur through the RP, the RP is a single point of failure.

Unlike the protocols described so far, On-Demand Multicast Routing Protocol (ODMRP) [52, 54] and the Core Assisted Multicast Protocol (CAMP) [35] diverge from the tree approach and instead create multicast group meshes to provide group connectivity. ODMRP calculates a forwarding group of nodes for each multicast group. The forwarding group is a set of nodes that rebroadcasts multicast packets that they receive. The group is periodically refreshed by a network-wide Join Request message broadcast by each multicast source node. Like ODMRP, CAMP also creates a shared mesh per multicast group. CAMP uses core nodes to limit the amount of control traffic when nodes join a group, and it ensures that the shortest path from receivers to sources are a part of the mesh.

A different approach to multicast is taken by the Location-Based Multicast algorithm, described in [49]. This algorithm defines multicast group membership by location; all nodes in a certain area become multicast group members and hence receive multicast data packets. The protocol creates a forwarding zone, composed of the multicast region and those nodes needed to connect the source to this region. There is not necessarily a single path to the multicast region, and hence the protocol is not tree based. Instead, this is a variation of a mesh-based protocol.

2.4 Broadcast

When a node wishes to generate a broadcast, it sends the broadcast packet to address 255.255.255.255. AODV does not specify transmissions to any directed broadcast address.

Every node maintains a list to keep track of which broadcast packets have already been received and retransmitted. The list contains, for each distinct broadcast packet received, the source IP address, the IP *ident* value and the IP fragment offset from the IP header of the broadcast packet.

When a node receives a packet broadcast to address 255.255.255.255, it checks the source IP address and the IP *ident*/fragment offset value of the broadcast packet's IP header. The node then checks whether the broadcast packet has already been received in the past, and thus whether it has already retransmitted the broadcast packet. If there is no existing list entry containing the same IP source address and IP *ident*/fragment offset value, the node retransmits the broadcast packet. If there is such a list entry with matching source IP address and IP *ident*/fragment offset field, the node silently discards the broadcast packet.

2.5 Interfaces

Because AODV should operate smoothly over wired, as well as wireless, networks, and because it is likely that AODV will also be used with multi-homed radios, the interface over which packets arrive must be known to AODV whenever a packet is received. This includes the reception of all AODV control messages. Whenever a packet is received from a new neighbor, the interface on which that packet was received is recorded into the route table entry for that neighbor, along with all the other appropriate routing information. Similarly, whenever a route to a new destination is learned, the interface through which the destination can be reached is also recorded into the destination's route table entry.

When multiple interfaces are available, a node receiving and rebroadcasting a RREQ or GRPH message rebroadcasts that message on all interfaces. Similarly, when a node needs to transmit a RERR, it should only broadcast it on those interfaces that have precursor nodes for that route. If a node is participating on the multicast tree, and if multicast data packets are sent as broadcast traffic, a multicast tree node should only rebroadcast a packet on those interfaces that have multicast tree next hops.

2.6 Actions After Reboot

A node participating in the ad hoc network must take certain actions after reboot as it will have lost its prior sequence number, as well as its last known sequence numbers for various other destinations. Additionally, there may be neighboring nodes which are still using this node as an active next hop. This can potentially create routing loops. To prevent this possibility, each node on reboot waits for `delete_period`. In this time, it does not respond to any routing packets. However, if it receives a data packet, it broadcasts a RERR, as described in Section 2.2.5, and resets the waiting timer (Lifetime) to expire after current time plus `delete_period`. Appendix F provides a calculation for the suggested value of `delete_period`.

It can be shown that by the time the rebooted node comes out of the waiting phase and becomes an active router again, none of its neighbors will use it as an active next hop. Its own sequence number gets updated once it receives a RREQ from any other node, as the RREQ always carries the maximum destination sequence number seen en route.

A node participating in the multicast tree that reboots (or restarts the routing daemon) loses all of its multicast tree information. Upon reboot, a node should also broadcast a MACT message with set *Reboot* flag to inform neighboring nodes that it has lost its multicast group information. Because the rebooted node does not know whether it was previously a member of the multicast tree, it should broadcast this packet unconditionally upon starting the daemon.

When a node on the multicast tree receives the reboot MACT message, it checks whether this message came from one of its next hops on the multicast tree. If so, one of two situations exists. If the reboot MACT came from a downstream link, the node deletes that link from its list of next hops and sets a prune timer according to the guidelines in Section 2.3.5. Otherwise, if the reboot MACT came from a node's upstream link, it must rebuild the tree branch as is also indicated in Section 2.3.5.

2.7 Security Considerations

Currently, AODV does not specify any special security measures. Route protocols, however, are prime targets for impersonation attacks. These attacks include transmitting RREPs with false routing information and denial of service attacks through the repeated broadcast of RREQ messages. Additionally, wireless transmission is inherently insecure. Packets are received by anyone within transmission range, and, if they are not encrypted, they can also be read by anyone.

To protect against these attacks, authentication techniques involving generation of unforgeable and cryptographically strong message digests or digital signatures can be used. It is expected that, in environments where security is an issue, IPSec authentication headers will be deployed along with the necessary key management to distribute keys to the members of the ad hoc network using AODV.

2.8 Quality of Service

AODV currently provides some minimal controls to enable mobile nodes in an ad hoc network to specify, as part of a RREQ, certain Quality of Service parameters that a route to a destination must satisfy [77]. In particular, a RREQ may include a Maximum Delay extension (see Appendix C.4) or a Minimum Bandwidth extension (see Appendix C.5).

If, after establishment of such a route, any node along the path detects that the requested Quality of Service parameters can no longer be maintained, that node must originate a ICMP QOS_LOST message back to the node that had originally requested the now unavailable parameters (Appendix C.6).

2.9 Conclusion

AODV is an on-demand routing protocol designed for use in ad hoc mobile networks. AODV provides unicast, multicast, and broadcast communication capability, and is able to discover and maintain routes during periods of either static or dynamic network topology. Unicast and multicast routes are determined through the use of a route discovery cycle. Once found, routes are maintained either as long as they are needed by the source node, or as long as the multicast group exists. Link breaks in active routes are efficiently repaired or re-established. Furthermore, all routes are loop-free through the use of destination sequence numbers. These routes are maintained in route tables, and are expired if they are not used within some lifetime period.

Because of the on-demand and low-overhead nature of the protocol, AODV should provide high throughput while generating little control overhead. This is validated by the experiments described in the Chapter 3.

Chapter 3

AODV Performance

3.1 Introduction

One of the primary advantages of on-demand routing protocols is that they minimize control overhead and bandwidth utilization during periods of low mobility or minimal routing demands. When there are no connectivity changes in active routes, AODV plays no role, and hence involves no overhead. The drawback to this class of protocols, however, is that the *route acquisition latency*, or the time to establish the route once it is needed, is often greater than that of table-driven protocols. With on-demand protocols, a route is not usually available the moment it is needed. One goal of an on-demand protocol should be to minimize this acquisition latency.

The validation and evaluation of ad hoc routing protocols can be exceedingly difficult. Testing these protocols in a real-world environment presents numerous difficulties. Creating repeatable scenarios with tens, hundreds, or even thousands of mobile nodes, or creating multiple scenarios with only small variations, is quite challenging. One method by which protocol design can be tested is through a formal verification of the protocol. The use of design verification tools can aid in the examination of each possible usage case and can validate the operation of a protocol in each of these situations. A formal verification of AODV was completed by Bhargavan et. al. [13]. The formal verification confirmed many expectations of AODV's operation in a variety of scenarios. Additionally, it pointed out cases that were not yet considered, such as the operation of the protocol after the spontaneous rebooting of a node. The results of the formal verification were incorporated into the design of AODV to increase the robustness and correctness of the protocol.

Simulation is another tool that can validate the operation of the protocol. Simulation enables a protocol to be tested in numerous scenarios, where one parameter or metric can be isolated to test the effects of that variable. To study the characteristics of AODV, simulations of AODV have been created to test the protocol in a variety of repeatable scenarios. Simulation provides a method by which the basic functionality of the protocol can be validated, and it can be an invaluable tool in refining the protocol design. Numerous simulations, with minor variations that address specific aspects of the protocol, can be run.

To study the performance of AODV, both its unicast and multicast operation have been simulated. For unicast, it is important to establish that AODV is able to find routes whenever such routes exist and are needed. This can be evaluated through the examination

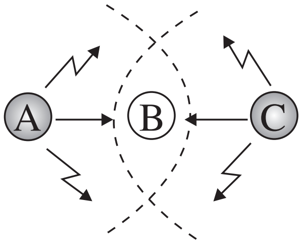


Figure 3.1: The Hidden Terminal Problem.

of the number of data packets that are delivered to their destinations. The more data packets AODV is able to deliver, the better its route finding and maintenance ability. Once it is confirmed that AODV is able to find and maintain routes, the delay needed to find these routes can be determined. The delay should be minimized; it should be small enough that it is not noticed by the user.

For multicast, it is again important to establish that data packets are delivered to their destinations. With this form of communication, however, each data packet must be received by multiple destinations (each member of the multicast group), which is more difficult to achieve. The characteristics of the multicast tree created can be studied to determine the quality of the tree AODV discovers. Does the tree tend to have exceedingly long branches and limited breadth? Or, do the group members tend to be more evenly distributed, with relatively few hops between them and the multicast group leader? Additionally, how much control overhead is generated by the protocol to maintain these trees?

The simulations described here and in the remaining chapters were performed using the GloMoSim Network Simulator developed at UCLA [8]. GloMoSim is based on the parallel discrete-event simulation language PARSEC [7]. This simulator models the OSI seven-layer network architecture and includes models of IP routing and UDP. Additionally, the GloMoSim simulation package includes multiple models at each of the different OSI layers. This enables the investigation of the protocol with, for instance, different MAC-layer protocols, or different channel models. The simulator also allows for network node mobility, thereby providing for simulation of mobile ad hoc networks. GloMoSim was selected as the simulator due to its inclusion of the OSI layer models, its scalability, and its ease of operation. Other network simulators, notably OPNET [3] and NS-2 [30], do not have the scalability of GloMoSim in that their running time for simulating large networks is much greater than that of GloMoSim. Also, because GloMoSim code is written in the C programming language, it is easier to port than simulators that require the use of special-purpose languages.

The following sections present the results of simulations of both AODV's unicast and multicast operation.

3.2 Simulation Environment

The MAC layer protocol used in the simulations is the IEEE standard 802.11 Distributed Coordination Function (DCF) [27]. This standard uses Request-To-Send (RTS) and Clear-To-Send (CTS) control packets for unicast data transmissions between neighboring nodes.

A node wishing to unicast a data packet to its neighbor broadcasts a short RTS control packet containing the length of the data frame that will follow. When its neighbor receives the packet, it responds by broadcasting a CTS packet, which also contains the length of the upcoming data frame. Any node hearing either of these two control packets must be silent long enough for the data packet to be transmitted. In this way, neighboring nodes will not transmit during the data transmission, and the number of collisions is reduced. Once the source node receives the CTS, it transmits the data packet. After receiving this data packet, the destination then sends an acknowledgment (ACK) to the source, signifying reception of the data packet.

The use of the RTS-CTS control packets reduces the potential for the hidden-terminal problem [96]. The hidden terminal problem occurs when two nodes, which are unable to hear each other's transmissions, attempt to transmit to the same neighboring node. Figure 3.1 illustrates an example of this scenario. Nodes A and C transmit simultaneously to node B. Because their transmissions overlap, the packets collide at node B, and, unless one signal is sufficiently stronger than the other, node B is not be able to successfully receive either of the transmissions. Hence both transmissions are lost.

The IEEE 802.11 standard specifies that broadcast data packets and RTS control packets are sent using the unslotted Carrier Sense Multiple Access protocol with Collision Avoidance (CSMA/CA) [27]. When a node wishes to broadcast a packet, it first senses the channel. If it does not detect an on-going transmission, it calculates a short backoff period, and then resenses the channel after this period. If the channel is still idle, it transmits the packet. On the other hand, if the node does detect a transmission, it calculates a backoff time and then waits this amount of time before reattempting the transmission.

The data rate for the simulations is 2 Mb/sec. The propagation model used is the free space model [79] with threshold cutoff included in the GloMoSim simulation package. The free space model has a power signal attenuation of $1/d^2$, where d is the distance between nodes. The radio model used also has capture capability, where it can lock on to a strong signal during interference from other signals and still receive the packet. Other interfering packets with weaker signal strength are dropped.

AODV does not guarantee packet delivery; however, it does find good routes for IP's best-effort delivery. Because data packets are not buffered for retransmission, they are likely to be lost. If a collision involving a data packet occurs at a node and the packet cannot be captured, that packet is lost. Typically, if a link breaks in an active route, data packets are lost before that break is noticed and while the link is being repaired. Hence, it is essential to take steps to monitor active links and provide for immediate and quick repair so that link breaks result in minimum packet loss. For unicast, the 802.11 protocol signals the AODV routing protocol when the data packet transmission was unsuccessful. Hence, packet loss is minimal, as link breaks are immediately noted. On the other hand, if a different MAC layer protocol (such as CSMA [47]) is used, the notification of broken links may not be provided. A notification of the loss of multicast data packets due to broken links is not provided by any of these MAC protocols because the data packets are sent as broadcast traffic; hence, their delivery is not guaranteed. Therefore, AODV provides a method of monitoring these active links that can be used in these scenarios, as described in Section 2.2.4.

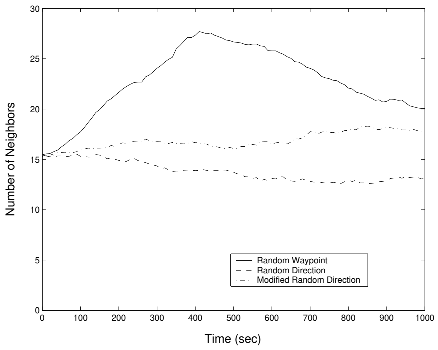


Figure 3.2: Average Neighbors per Node at 1 m/s Mobility.

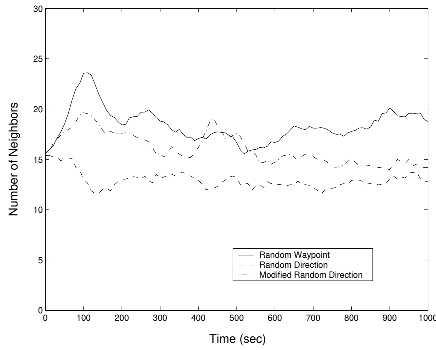


Figure 3.3: Average Neighbors per Node at 5 m/s Mobility.

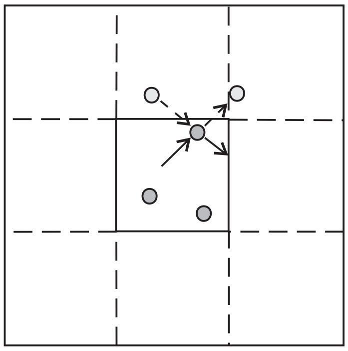


Figure 3.4: Simulated Movement in the Random Direction Mobility Model.

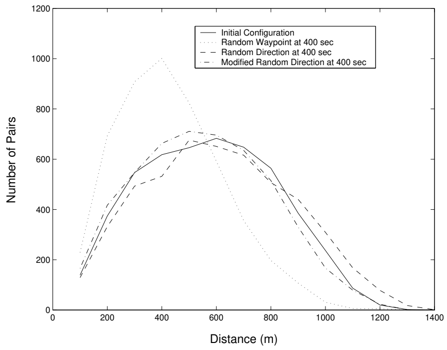


Figure 3.5: Number of Node Pairs at a Given Distance.

3.2.1 The Mobility Model

Node mobility in these simulations was originally modeled using the Random Waypoint mobility model [16]. At the beginning of a simulation, the Random Waypoint mobility model randomly places the nodes within the predefined simulation area. Each node then selects a destination within that area and a speed from a uniform distribution of user-specified speeds. The node travels to its selected destination at the selected speed. Once it reaches the destination, it is stationary for some pre-defined pause time. At the end of the pause time, it selects a new destination and speed combination, and then resumes movement. This model causes continuous changes in the topology of the network.

While running the simulations, an interesting behavior of the mobility model was noticed. The average number of neighbors seen at a given node periodically increases and decreases as the simulation progresses, where the frequency of the change is relative to the speed of the nodes. Figures 3.2 and 3.3 illustrate the average number of neighbors per node in a 100 node network during 1000 seconds of simulated time. The room size for these simulations is 1000m$\times$1000m, and nodes have a transmission radius of 250m. The fluctuation seen in these figures is due to the inherent characteristics of the mobility model. Because a node must choose a destination in the simulation area, the node is most likely to travel in the direction in which there are the most destinations from which to choose. This predisposes

Number of Nodes	50	100	250	500
Room Size	1000m×1000m	1500m×1500m	2400m×2400m	3450m×3450m

Table 3.1: Room Sizes for Network Simulations.

nodes to choose destinations that are either in the middle of the area, or that they reach by traveling through the middle. This characteristic creates a situation in which *density waves* occur; nodes seemingly converge in the center of the area, and then disperse, and then converge, etc.

Because a fairly constant number of neighbors per node was desired, or even necessary, throughout many of the simulations, a new mobility model was needed. The Random Direction model was designed to overcome this characteristic of the Random Waypoint model. Instead of selecting a destination within the area, the nodes choose a *direction* in which to travel, where a direction is measured in degrees. At the beginning of the simulation, each node selects a degree between 0 and 359, and then finds a destination on the boundary of the network area in this direction of travel. It then selects a speed, and travels to that destination at the given speed. Once it reaches the destination, or comes within some distance ε of the destination, it rests for the given pause time, and then selects a new degree between 0 and 180. The direction is limited because the node is already on the boundary, and nodes do not pass through the boundary. Thus, the degree determines a direction relative to the wall of the boundary area on which the node is located. The node then selects a destination on the boundary in this line of direction, selects a new speed, and resumes travel.

This model can be thought of as examining a microcell of a larger area, as illustrated in Figure 3.4. A node travels to the boundary and then selects a new direction, which is similar to a node that continues past the boundary, while a new node traveling in a different direction enters the microcell at that location. The average number of neighbors per node for this model is illustrated in Figures 3.2 and 3.3. As is evident from the figures, the Random Direction mobility model causes many fewer fluctuations in the node distribution than the Random Waypoint model. Also in these figures is the average number of neighbors per node using a variation of the Random Direction model, called the Modified Random Direction model. In this model, nodes select a degree as before, but they may choose their destination anywhere along that direction of travel. They do not need to travel all the way to the boundary.

Figure 3.5 illustrates the average number of pairs of nodes with the given distance of separation after 400 seconds of simulation time. The figure represents nodes moving at 1 m/s. Initially, the nodes are evenly distributed throughout the simulation area. As the simulation progresses, the characteristics of the Random Waypoint model show that after 400 seconds of simulation time, the node density is much higher than the initial configuration. However, the Random Direction model is able to maintain a node density that is similar to the initial node distribution.

3.3 Unicast Simulations

The primary objective of the unicast simulations is to show that AODV is able to discover routes between a source and destination whenever needed, and to maintain these routes

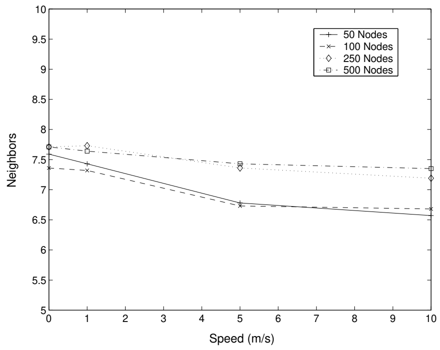
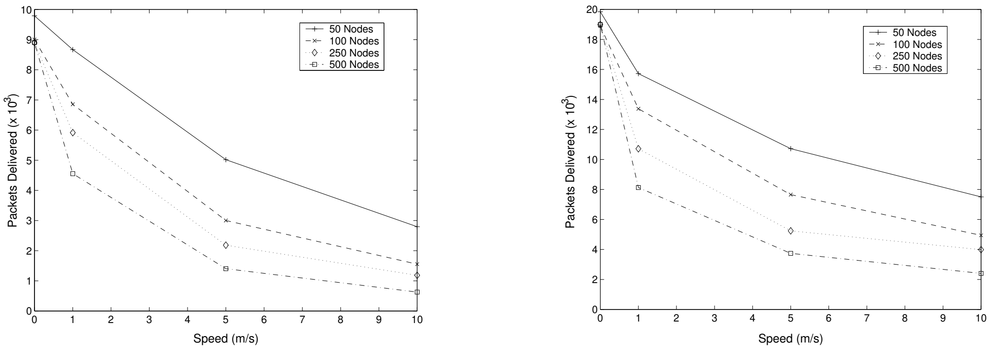


Figure 3.6: Number of Neighbors per Node for 10 Sources.



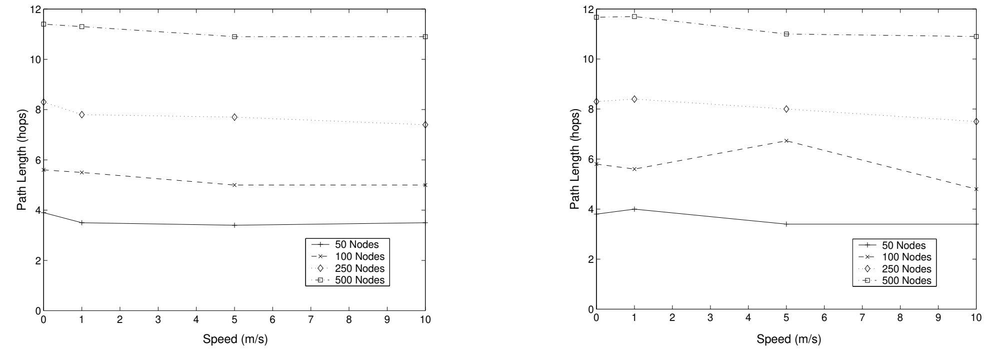
(a) 10 Sources.

(b) 20 Sources.

Figure 3.7: Number of Packets Delivered.

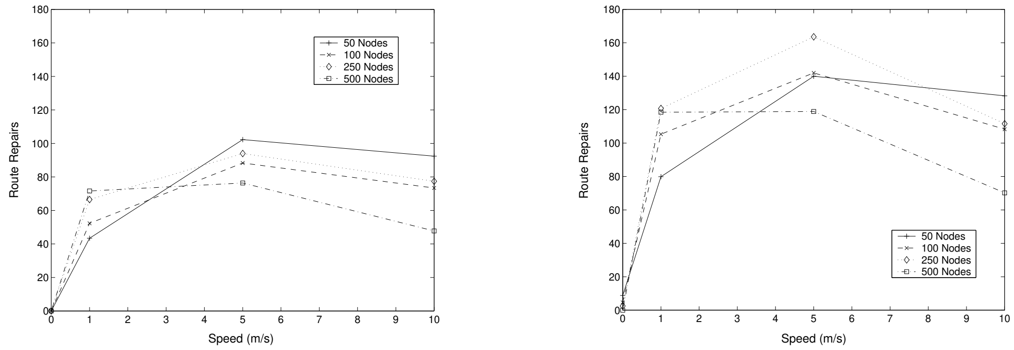
as long as they are required by the source node. AODV should accomplish this with a minimum of control packet overhead. Additionally, it is important to discover the delay for initially finding a route, because this delay is the primary disadvantage of on-demand routing protocols.

To demonstrate the above characteristics of AODV, different network configurations with varying mobility levels are simulated. Networks consisting of 50, 100, 250, and 500 nodes are modeled to investigate AODV's scalability. Table 3.1 shows the respective room sizes for the number of nodes. These room sizes were chosen so that the node density in each of the different networks would be approximately constant. Figure 3.6 illustrates the average number of neighbors per node in the different networks. Additionally, each network size is simulated at four different mobility levels: 0 m/s, to determine AODV's performance in a static network, and then 1.0 m/s, 5.0 m/s, and 10.0 m/s, to determine the effect of mobility on AODV. Each network size/mobility speed combination was run for ten different randomly generated initial node configurations. To investigate AODV's performance further, each of the simulations was run for two different traffic levels: 10 sources, and 20 sources. Sessions commence at the beginning of the simulation and continue



(a) 10 Sources. (b) 20 Sources.

Figure 3.8: Mean Path Length.



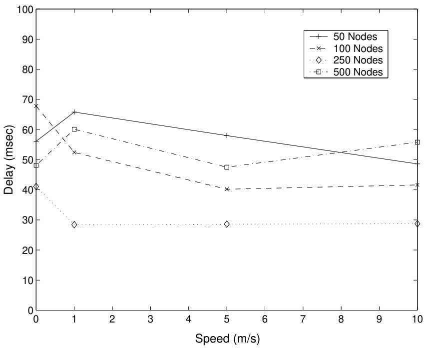
(a) 10 Sources. (b) 20 Sources.

Figure 3.9: Number of Route Repairs.

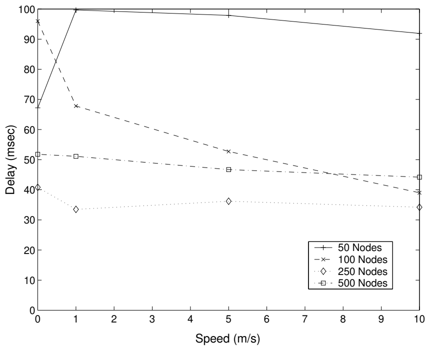
until the simulation ends. The sending rate for each source is four 64-byte data packets per second.

To verify that AODV is able to successfully deliver data packets to their respective destinations, the number of packets delivered is shown in Figure 3.7. The figure shows that for no mobility, AODV delivers virtually the same number of data packets in each of the various sizes of networks. As the mobility speed increases, however, the number of packets that are delivered also decreases. The reduction in delivered data packets is due to the more frequent occurrence of link breaks in the networks with rapidly moving nodes. Once a link break occurs, data packets are lost along that path until the source is notified and can begin repairing the route. Hence, each link break results in a loss of data packets. The larger networks are more affected by mobility than the small networks, due to the increase in path lengths in these networks.

Figure 3.8 illustrates the differences in path lengths for networks of different sizes. Because the node density is kept nearly constant, the room size is scaled as the number of nodes increases; hence, more hops are needed to connect source and destination pairs.



(a) 10 Sources.



(b) 20 Sources.

Figure 3.10: Mean Route Acquisition Latency.

Figure 3.9 shows the number of route repairs per simulation. For low mobilities, the larger networks require more route repairs, due to the longer path lengths. However, once a certain mobility threshold is reached, the number of repairs in the larger networks is actually smaller than that in the smaller networks. This indicates that the long path length/high mobility combination makes routing extremely difficult; so difficult, in fact, that sessions are likely to be aborted due to the inability to acquire a route that is stable enough to use for the transmission of data packets. The route must be stable between the time that the RREQ is transmitted and the RREP is received at the source, so that the RREP is able to be successfully routed back to the source. For long path lengths and high mobilities, the probability of a route break during *route discovery* becomes so high that eventually routes cannot be maintained, and the session must be terminated. It should also be noted that the number of route repairs needed in the higher traffic simulations is greater than that needed in the lower traffic simulations, which makes sense because there are a greater number of routes maintained in the higher traffic simulations.

Finally, the delay needed to acquire the route, the route acquisition latency, is also examined. The route acquisition latency is found by measuring the duration between that time that the source broadcasts its initial RREQ for a route, and the time that it receives the first RREP for that route. If no RREP is received, it does not tabulate the time. Additionally, the time required for repairing routes after a link break is not included. Figure 3.10 shows the route acquisition latencies for the simulations. The graphs indicate that there is no significant difference in the route acquisition latency between the different network sizes and mobility levels. The delay is more a factor of the network topologies and the queueing delays experienced at the individual nodes. In all cases, routes are quickly discovered with minimal delay.

3.4 Multicast Simulations

Similar to unicast, validation of the correct operation of the AODV multicast protocol is needed. Additionally, the amount of overhead created by the protocol, as well as its efficiency at delivering data packets to the multicast group members, should be evaluated. To this

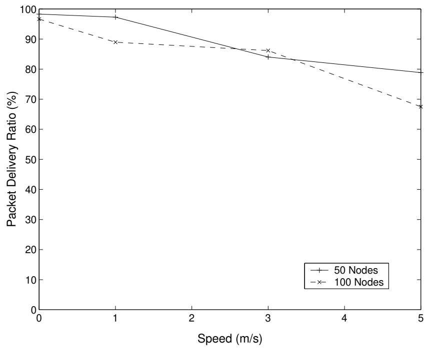


Figure 3.11: Packet Delivery Ratio.

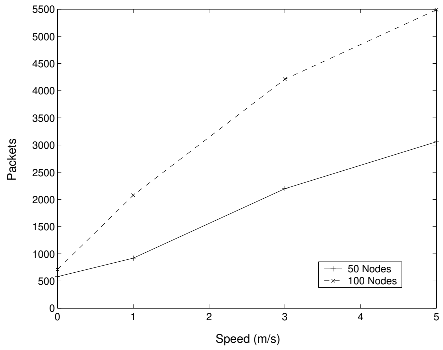


Figure 3.12: Total Control Overhead.

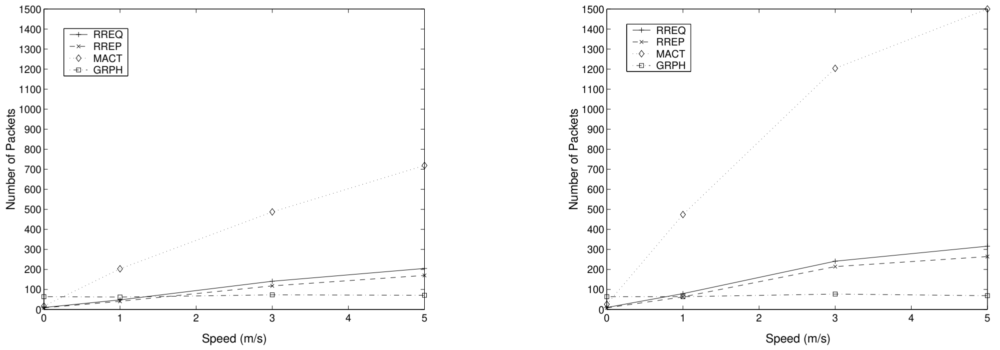
end, 50 and 100 node networks were simulated, with the same corresponding room sizes as for unicast. Four different node mobilities, ranging from 0 m/s to 5 m/s, were modeled. As with unicast, each parameter combination is run for 10 different initial configurations, and the results were averaged to produce the graphs.

In the simulations, there was one multicast group that nodes can join. Ten randomly selected nodes join the group at the beginning of the simulation, and then remain group members throughout the duration of the simulation. Once all ten nodes have joined the group, one of the group members begins sending data packets to the rest of the group, and continues to send data packets throughout the simulation. Data packets are 64 bytes and are sent at a rate of four per second.

To evaluate the performance of the protocol, the packet delivery ratio was studied. The packet delivery ratio is the number of data packets received divided by the number sent. To achieve a 100% packet delivery ratio, each member of the multicast group must receive every data packet that is transmitted by the source. Hence, in effect, each data packet transmission is multiplied by 10, to account for the 10 group members. Each time a group member receives a data packet, the number of received packets is incremented. Because multicast data packets are transmitted as broadcast traffic, there is no guarantee that the next hop has received the transmission, as there is in unicast. Additionally, there is no MAC layer feedback to determine whether the next hop has moved out of range. As in unicast, when a link break occurs, packets are dropped until the link is repaired. Because of these characteristics, and because of the multitude of links that must be maintained across the multicast tree, the packet delivery ratio is not likely to be 100%.

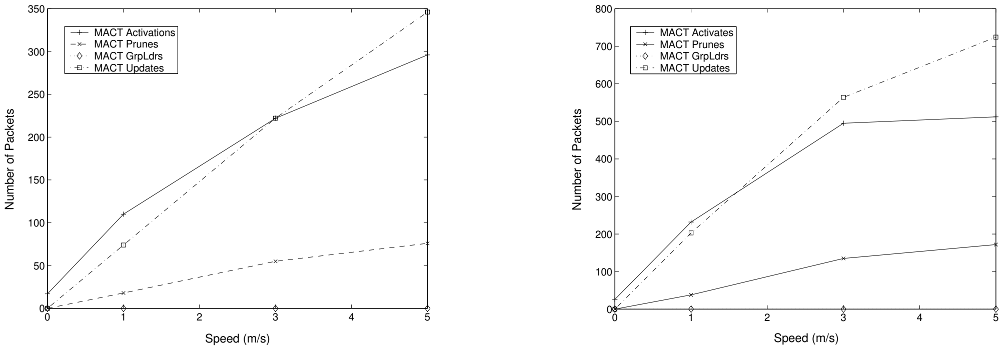
Figure 3.11 illustrates the achieved packet delivery ratio for the simulations. For zero mobility, the number of received data packets is between 95% and 100%. 100% packet delivery is not achieved for zero mobility because of collisions. As the speed of the nodes increases, the packet delivery ratio decreases because it becomes more difficult to maintain the multicast tree links. For slow node speeds, however, the protocol performs well, achieving between 87% and 97% packet delivery.

Before examining the specific characteristics of the different networks, the amount of control overhead generated is studied. Figure 3.12 represents the total number of control packets generated per simulation. This number is found by summing the RREQ, RREP, MACT, GRPH, and Hello messages transmitted during the simulation. As stated in Section 2.3.5, Hello messages are needed to monitor the status of multicast tree links. The two



(a) 50 Node Network. (b) 100 Node Network.

Figure 3.13: Control Overhead.



(a) 50 Node Network. (b) 100 Node Network.

Figure 3.14: Multicast Activation Breakdown.

networks have similar amounts of control traffic for the stationary networks. As mobility increases, however, the number of control messages needed to maintain the 100-node network increases at a much greater rate than that of the 50-node network. The reason is that there are more links in the multicast tree for the 100-node network, due to the larger network area. Again, as node mobility increases, the number of control messages increases because of the more frequent need to repair multicast tree branches.

Figure 3.13 gives a breakdown of the number of each control message generated for the 50 and 100-node networks, respectively. For scaling reasons, the number of Hello messages is not shown in these figures. The graphs show that the number of MACT messages clearly dominates the control traffic, particularly for higher mobilities. The number of MACT messages is found by summing the number of transmitted *Join*, *Prune*, *Update*, and *Grpldr* versions of the message. The GRPHs are sent periodically, and are not affected by the speed of the nodes. The number of RREQs and RREPs transmitted increases as the node movement increases, but by a much smaller percentage than the number of MACTs. A breakdown of the MACT message overhead is shown in Figure 3.14. The MACT link

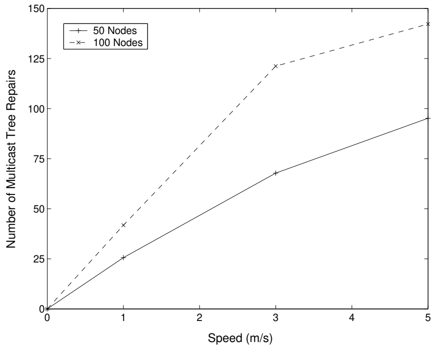


Figure 3.15: Number of Multicast Tree Repairs.

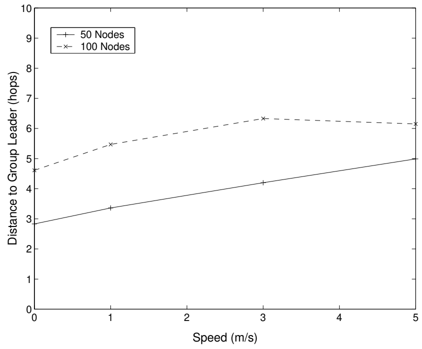


Figure 3.16: Average Distance to Group Leader.

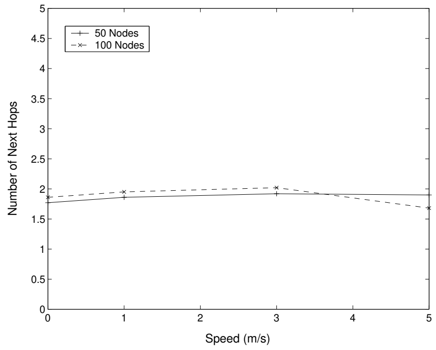


Figure 3.17: Average Number of Next Hops on Multicast Tree.

activations and updates are the dominant forms of MACT traffic sent. The number of prune messages is minimal, and there are virtually no grpldr messages sent, due to the infrequent partitioning of the multicast tree in the simulations.

To understand why node speeds have such an effect on the packet delivery ratio and the amount of control overhead, it is necessary to examine some other results of the simulations. Figure 3.15 shows the number of repairs to the multicast tree needed during the simulations. As the node speed increases, increasingly more repairs are needed to keep the multicast tree intact. The 100-node network requires more repairs than the 50-node network due to the larger size of the simulation area. Figure 3.16 shows that the average distance from a group member to the group leader in the 100-node network is a hop or two greater than that in the 50-node network. Because of the additional length of the tree branches in this network, tree link breaks are more likely to occur, and hence more repairs are needed. The increase in the number of repairs results in a lower packet delivery ratio and greater amount of control overhead, as shown in Figures 3.11 and 3.12.

Finally, Figure 3.17 illustrates the average number of next hops for multicast group members. This graph gives an idea of the structure of the multicast trees. The number of next hops in both networks is slightly under two, indicating that most group members are either leaf nodes or have only one downstream neighbor.

3.5 Conclusion

The operation of AODV can be observed and validated through simulation. Simulation is a powerful tool for evaluating the performance of a protocol in repeatable scenarios. Small variances can be made between simulation runs that allow the effects of certain parameters, such as mobility, to be studied. Simulations of AODV's unicast capabilities have been performed in networks ranging from 50 to 500 nodes. The simulations show that mobility has a negative effect on the performance of AODV. As the speed of the nodes increases, performance degrades due to the difficulty in maintaining routes. However, AODV is still able to find routes when the nodes are rapidly moving and delivers a large percentage of the transmitted data packets.

The multicast capabilities of AODV have been evaluated through the simulation of 50 and 100-node networks. These simulations show that AODV successfully creates multicast trees connecting group members and delivers data packets to those group members. Control overhead is minimal in both networks and increases only slightly in the larger network. This increase is due to the greater number of hops needed to connect the group members. Longer path lengths result in more link breaks, which require more repairs to keep the tree intact. However, despite the increase in the frequency of link breaks, AODV maintains the multicast tree throughout the lifetime of the multicast group and achieves high message delivery rates.

Chapter 4

Clustering Protocols

4.1 Introduction

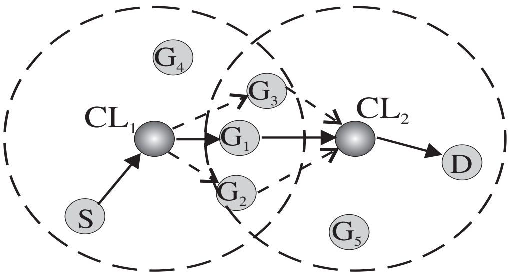


Figure 4.1: Hierarchical Addressing.

A clustering protocol can be run as an interface between the IP layer and the routing protocol to increase robustness of routes in dynamic networks. In a network with flat addressing, routes are recorded hop-by-hop. When a link between two nodes in an active route breaks, some action must be taken to repair the link break. This action can be in the form of notifying the source of the route, repairing the link break locally, etc. On the other hand, in a clustered network, addressing can be hierarchical and routes can instead be recorded cluster-leader-by-cluster-leader. For instance, in Figure 4.1, a hop-by-hop route between the source (S) and the destination (D) would be recorded as

$$S \rightarrow CL_1 \rightarrow G_1 \rightarrow CL_2 \rightarrow D,$$

whereas utilizing hierarchical addressing the route would be

$$S \rightarrow CL_1 \rightarrow CL_2 \rightarrow D.$$

The primary difference lies in the link between CL_1 and CL_2. If the link between CL_1 and G_1 breaks, the flat addressing scheme requires the link to be repaired. However, with hierarchical addressing, *any* of the gateway nodes between leaders CL_1 and CL_2 can be used to route between the two clusters. Hence, if gateway G_1 moves out of reach, G_2 or G_3 could be used instead. Furthermore, as gateways move out of range of CL_1 and CL_2, it is

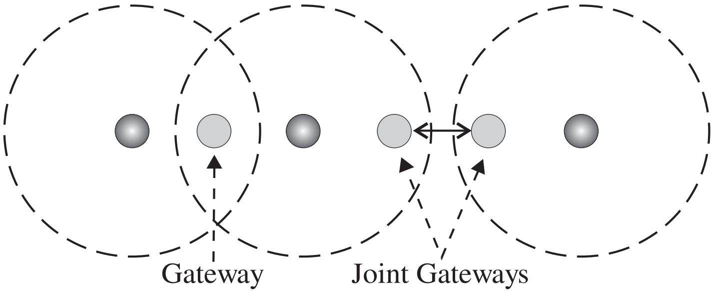


Figure 4.2: Cluster Gateways.

likely that other nodes, such as G_4 or G_5 will become gateways for the two clusters, thereby maintaining the robustness of the route. In routes with longer path lengths, the increase in robustness is much greater.

This chapter describes the Adaptive Routing using Clusters (ARC) and Adaptive Routing using Cluster Hierarchies (ARCH) algorithms, both of which can be used to initialize and maintain a hierarchical topology in a network. ARC creates a one-level hierarchy on the set of network nodes, while ARCH is able to dynamically adjust the depth of the hierarchy in response to the changing topology of the network.

4.2 Adaptive Routing using Clusters

The Adaptive Routing using Clusters (ARC) algorithm creates a one-level cluster topology across a network of nodes. Nodes are grouped into clusters based on their proximity to one another. Each cluster has a cluster leader. Nodes within a cluster are one-hop neighbors of the cluster leader. A node that is only one hop from more than one cluster leader is called a *gateway*, and can be used by the cluster leaders to route between the clusters. In a densely connected network, it is likely that two clusters will have multiple gateways between them. It is also possible for a pair of nodes to serve as *joint gateways* between two clusters. In such a pairing, each node is a member of a different cluster, and the two nodes are able to communicate directly. Figure 4.2 illustrates the difference between gateways and joint gateways.

The ARC algorithm uses a novel method for determining when a cluster leader should revoke its leader status, which results in minimal cluster leader changes during the lifetime of the network. Furthermore, it is able to utilize the Limited Broadcast mechanism, whereby the impact of network-wide broadcasts is reduced by allowing the flooding of only cluster leaders, as opposed to each network node.

4.2.1 Clustering Tables

A node maintains either one or two clustering tables, depending on its status level. Nodes that are cluster leaders maintain a Node Table, in which all nodes that are members of that cluster are recorded, and a Neighbor Table, in which all neighboring cluster leaders and the gateways used to reach them are stored. The Node Table has the following fields:

- Node IP Address
- Cluster Leaders

The Cluster Leaders field is a linked list containing each cluster leader IP address for which the node is a gateway. If the node is not a gateway, the Cluster Leaders list is empty.

The Neighbor Table has the following fields:

- Cluster Leader IP Address
- Gateways
- IP Gateway
- Flags

The Gateways field is a list of nodes that are gateways for the cluster leader indicated in the Cluster Leader IP Address field. The IP Gateway is the node currently recorded in the IP routing table as the next hop for that neighboring cluster leader. If a cluster leader is within another leader's communication radius, the IP Gateway is the neighboring cluster leader itself. Otherwise, it is one of the listed gateways. The only defined flag is *Monitor*. This flag indicates that the cluster leader is in direct communication with the neighboring cluster leader. If the *Monitor* flag of any neighbor table entry is set, the leader must monitor its cluster to determine whether it becomes a subset of a neighboring cluster (see Section 4.2.4).

Nodes that are not cluster leaders maintain only one table, the Cluster Leader Table. This table has the following fields:

- Cluster Leader IP Address
- Joint Gateways
- IP Gateway
- Flags

Nodes have an entry for each cluster leader of whose cluster they are a member, as well as those cluster leaders that can be reached through a joint gateway. If the node is a member of the cluster leader's cluster, the *oneHop* flag is set; otherwise, it is left unset. The Joint Gateways field is a list of neighboring nodes that are members of the cluster leader's cluster, and that can be used to reach the neighboring cluster leader. If the *oneHop* flag of the entry is not set, then there must necessarily be at least one node listed in the Joint Gateways field. This node serves as a joint gateway to reach the cluster leader. If the *oneHop* flag is set, the Joint Gateways field may still have one or more entries. These nodes can be utilized as joint gateways in the event that the node loses its direct connection to the cluster leader. The IP Gateway field contains the IP address of the node currently recorded in the IP routing table as the next hop for that cluster leader. If the node is a member of the leader's cluster, the IP Gateway is the cluster leader itself. Otherwise, it is one of the joint gateways.

4.2.2 Node Initialization

The cluster topology is initialized and maintained through the periodic beaconing of each node. When a node is first initialized, it broadcasts a *Hello* message to announce its presence and to look for neighboring cluster leaders. The Hello message has the following fields:

$$< Flags,\ Source\_Addr,\ Source\_Status,\ CLdr1\_List,\ CLdr2\_List >$$

The *Source_Status* field is set to the sending node's current status, which is undefined (`undef`) at initialization. The *CLdr1_List* field contains a list of those cluster leaders with which the node can communicate directly, while the *Cldr2_List* field contains a list of those cluster leaders with which the node can communicate only through the use of a joint gateway. At initialization, both of these lists are empty. Additionally, both of these lists are empty in Hellos initiated by cluster leaders.

After sending the initial Hello, the node sets a timer to wait for the reception of Hellos from cluster leaders. If it receives any such Hellos, the node enters the cluster leader information into its cluster leader table. At the end of the initialization period, if the node has heard from exactly one cluster leader, it becomes an "ordinary node" and is a member of that cluster. If it has heard from more than one cluster leader, it becomes a gateway for the different clusters. However, if it did not hear from any cluster leader, then it becomes a cluster leader itself. The node sets its status level depending upon which of the previous conditions is satisfied. It then sends a Hello indicating its new status, and reporting those cluster leaders with which it is able to communicate, if any.

4.2.3 Discovering the Topology

Nodes learn of the presence of their neighbors through Hello messages. There are four different cases of node status and status of the source of the Hello:

(i) Cluster leader receives a Hello from a node
(ii) Cluster leader receives a Hello from another cluster leader
(iii) Node receives a Hello from a cluster leader
(iv) Node receives a Hello from another node

The processing that takes place at the receiver is dependent upon the status of both the receiver and the source of the Hello.

Cluster Leader Receives Hello from a Node

When a cluster leader receives a Hello from a non-cluster leader, it first ensures that it has an entry for that node recorded in its node table. If not, it creates one. It then verifies that its record of with which cluster leaders the node can communicate is consistent with that recorded in the CLdr Lists of the Hello message. If the CLdr Lists differ from the cluster leader's recorded information for that node, the cluster leader makes the appropriate updates.

The CLdr Lists in the Hello message indicate the other cluster leaders for which the node is a gateway. Hence, the cluster leader goes through the reverse process with the neighboring cluster leader information by correlating the leader information included in the Hello message with that maintained in the neighbor table. The cluster leader ensures that there is an entry in the neighbor table for each leader listed in the node's Hello message. It then verifies that, for each neighboring cluster leader included in the Hello message, the Gateway list for that leader includes the node that sent the Hello. Similarly, if any Gateway lists in the neighbor table include the node as a gateway, and if the node no longer lists that cluster leader in its Hello message, the node is deleted from that list.

If the node is deleted from the Gateway list for a neighbor, the cluster leader determines whether there are other gateways for the neighbor, or whether it can communicate with the leader directly (i.e. the *Monitor* flag is set). If neither of these conditions holds, the cluster leader removes the neighbor from its neighbor table.

Before the leader finishes processing the packet, it ensures that the node is not listed as a cluster leader in its neighbor table. Similarly, it ensures that the node is not listed as a cluster leader with which a node in the node table can communicate. These situations can arise if the node was previously a cluster leader and has recently given up its leader status, as described in Section 4.2.4.

Cluster Leader Receives Hello from Another Cluster Leader

When a cluster leader receives a Hello from another cluster leader, it first ensures that it has an entry for the other cluster leader in its neighbor table. If no such entry exists, it creates one. It then sets the *Monitor* flag for this entry, to indicate that the status of its cluster needs to be monitored to determine whether it becomes a subset of the other cluster. The determination of this occurrence and the resulting actions are further described in Section 4.2.4. Finally, the leader ensures that the source of the Hello is not listed as a node in its node table, or as a gateway for any cluster leaders in its neighbor table.

Node Receives Hello from A Cluster Leader

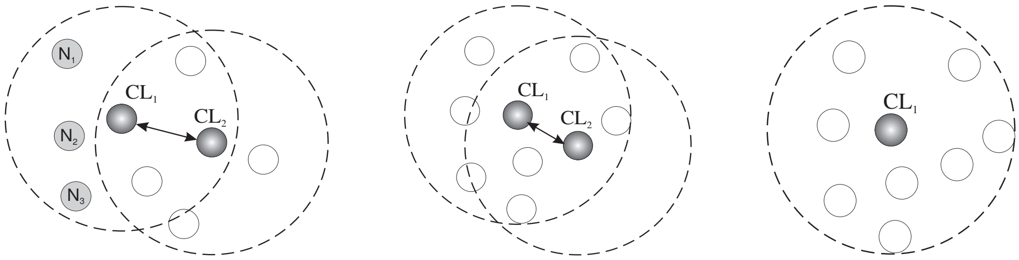
A node that receives a Hello message from a cluster leader checks its cluster leader table to verify that it has an entry for that cluster leader. If it does not, it creates one, and it sets the *oneHop* flag to indicate that it can reach this leader without the need of a joint gateway. If an entry for the cluster leader already exists in the cluster leader table, it verifies that the *oneHop* flag for that entry is set. The node then verifies its own status. If it can now communicate with more than one cluster leader, it updates its status to `gateway`. Otherwise, it remains an ordinary `node`. As a final step, the node ensures that the cluster leader is not listed as a joint gateway to any other cluster leaders. This situation could occur if the leader has recently become a leader.

Node Receives Hello from Another Node

When a node receives a Hello from another node, it verifies its record of the cluster leaders for which this node is a joint gateway with those cluster leaders recorded in *CLdr1_List* of the Hello message. It ensures that it has the node as a joint gateway for each cluster leader listed in *CLdr1_List*, and similarly verifies that the node is not listed as a joint gateway for any cluster leaders that are not included in *CLdr1_List*.

If the node removes the source node as a joint gateway for any cluster leaders, it verifies that it can reach the cluster leader directly in one hop, or that it has other joint gateways through which it can reach the cluster leader. If neither of these conditions holds, it deletes the cluster leader entry from its cluster leader table. It then verifies that it can still communicate with more than one cluster leader. If not, it updates its status to an ordinary `node`.

Finally, it checks whether the node from which it received the Hello is listed as a cluster leader in the Cluster Leader Table. If so, it removes the entry.



(a) Cluster Leaders come into contact

(b) CL_2's cluster becomes a subset of CL_1's

(c) CL_2 revokes leader status; Clusters merge

Figure 4.3: Merge of Two Clusters.

4.2.4 Revoking Cluster Leader Status

When a cluster leader receives a Hello from another cluster leader, it learns that it is in the direct transmission range of the other leader. It sets the *Monitor* flag in the neighbor table entry for the cluster leader, indicating the new communication status. Unlike other clustering algorithms [11, 21, 39], one cluster leader does not necessarily have to give up its leader status if two cluster leaders come within transmission range. Instead, the two leaders in range of each other monitor the status of their respective clusters to determine whether their cluster has become a subset of the other cluster. A cluster leader knows that its cluster has become a subset of another cluster because each node in its cluster reports its status as a one hop gateway for the other cluster leader, indicating that it can communicate directly with that cluster leader. Only when one cluster has become a subset of another cluster does the leader of the subset cluster have to give up its leader status. By ensuring that the cluster is a subset of the larger cluster, no further cluster changes are propagated throughout the network, as is the case with the other clustering protocols, and the *rippling effect* is prevented. An example of this effect is shown in Figure 4.3. CL_1 and CL_2 come within range of each other. For this example, let the cluster leader with the higher IP address retain its leader status while the leader with the lower IP address becomes an ordinary node. Using this method, CL_1 becomes an ordinary node and joins CL_2's cluster, while CL_2 remains a leader. Then, consider what happens to nodes N_1, N_2 and N_3. Because CL_1 is no longer a cluster leader, each of these three nodes is now not a member of any cluster. Hence, they each broadcast Hellos, receive no response, and one will become the new leader. In this way, the change of CL_1's status results in further status changes within the network. This rippling effect can cause many such changes in a large network.

ARC prevents this occurrence by requiring one cluster to be a subset of another in order for one leader to revoke its leader status. If one or more *Monitor* flags are set in a cluster leader's neighbor table, then each time there is a change in the connectivity information of one of its member nodes, it must determine whether its cluster is now a subset of another cluster. Similarly, when a cluster leader gains or loses a member node, it must perform the subset check. Figure 4.3 illustrates the progression of how one cluster can become a subset of another. In Figure 4.3(a), the two cluster leaders have come into contact, but each cluster has members that are not also a part of the other cluster. Figure 4.3(b) shows that the

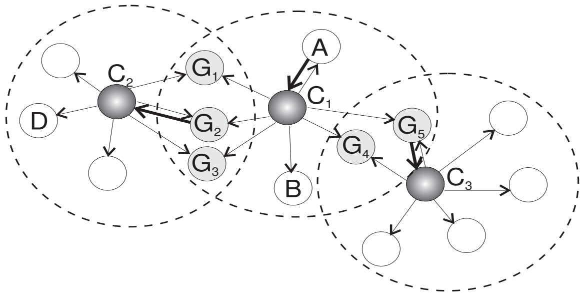


Figure 4.4: Limited Broadcast.

movement of some of the nodes has resulted in CL_2's cluster becoming a subset of CL_1's cluster. Finally, Figure 4.3(c) illustrates the resulting cluster after CL_2 revokes its leader status and joins CL_1's cluster. Note that no further cluster reconfigurations are necessary because none of the nodes in CL_2's cluster is left without a leader after CL_2's status change. Furthermore, if two leaders are only momentarily within transmission range of each other, no cluster reconfigurations are necessary.

4.2.5 The Limited Broadcast Mechanism

In addition to increased robustness of routes, one of the primary benefits of clustering is that the flooding of control messages (such as AODV's Route Request, Section 2.2.2) across the network can be reduced so that only the cluster leaders need to be flooded. This reduced broadcast mechanism is called *limited broadcast*. When a cluster leader broadcasts a packet with TTL greater than one (implying it should be rebroadcast), the gateways only rebroadcast the packet if they are able to reach a cluster leader to which the packet has not yet been transmitted. The following example illustrates the limited broadcast algorithm in detail.

Suppose node A in Figure 4.4 wishes to find a route to some destination D. Node A broadcasts its route request as specified in Section 2.2.2, except that a header is added to the packet. This header lists the cluster leaders that node A is able to reach directly, in this case, CL_1. When cluster leader CL_1 receives the packet, it processes it and rebroadcasts it. When a non-cluster leader receives the packet from either node A or cluster leader CL_1, it checks the list of cluster leaders that it can reach directly against the list contained in the packet header. In the case of the rebroadcast by the cluster leader, this list is empty. If the node receiving the packet can reach at least one cluster leader that is not listed in the packet header, it buffers the packet, along with a list of cluster leaders it can reach, and sets a random timer. During this time, it listens for rebroadcasts of the packet by other nodes. If it receives a rebroadcast by another node, it checks the packet's list of cluster leaders against its list that it has buffered with the packet. If any of the cluster leaders in its own list are listed in the packet header, it marks those cluster leaders as having received the rebroadcast. When the node's timer expires, it checks its retransmission list to determine whether each cluster leader listed has already received the packet. If any of the cluster leaders have not yet received the packet, the node retransmits the packet, and includes in the packet header the identifier of *every* cluster leader the node is capable of reaching, including those that have already received the packet. Otherwise, if every cluster leader has already received the packet, the node deletes the packet. Note that the gateway does not process the packet; the gateway determines only whether it needs to retransmit it.

In the example in Figure 4.4, suppose G_2's timer expires first. It has not heard the packet be retransmitted to CL_2, so it proceeds to retransmit the packet and lists CL_2 in the packet header. Gateway G_1 receives G_2's transmission. It finds the packet in its buffer and checks the retransmission list associated with it. Because CL_2 is listed in its buffer, G_1 marks this leader as having received the packet. When its retransmission timer expires, it notes that the only cluster leader list in the retransmission list has already been transmitted the packet, and so it does not rebroadcast the packet. Gateway G_3 proceeds in a similar manner. Similarly, suppose gateway G_5's timer expires before G_4's. G_5 rebroadcasts the packet and lists CL_3 in the packet header. When G_4 receives the packet, it marks CL_3 as having received the packet, and hence does not rebroadcast the packet when its timer expires. Note that nodes such as B that are not gateways do not buffer the packet for retransmission because there are no additional cluster leaders that it is able to reach. In this manner redundant broadcasts are suppressed, so that the impact of flooding is greatly reduced. In a network without limited broadcast (such as in AODV without clustering), node A would broadcast the original route request, and every node within its transmission radius would rebroadcast it, and so forth, until every node in the network had broadcast the packet. Hence, limited broadcast greatly reduces the number of redundant broadcasts caused by a single original broadcast. The measured reduction of packet transmission is discussed in Section 5.4.

4.2.6 Combining ARC with AODV

The ARC clustering protocol can be run underneath virtually any on-demand routing protocol for ad hoc networks. In such a scenario, ARC serves as an interface between the routing protocol and the IP layer. For the purposes of the simulations described in Chapter 5, the Ad hoc On-Demand Distance Vector routing protocol (AODV) (Chapter 2) has been selected to provide routing capability.

A cluster leader serves as a representative for the nodes in its cluster and processes routing packets on the node's behalf. To accomplish this, the ARC protocol intercepts all AODV control packets before they reach the AODV protocol. If the node is a cluster leader, the packet is passed up to the AODV protocol for processing. Otherwise, the packet is either forwarded or buffered, depending on the type of packet. If it is a unicast packet, such as a route reply, the packet is forwarded by the gateway to the next cluster leader on the route to the destination. Otherwise, if the packet is a broadcast packet such as a route request, then the gateway either rebroadcasts the packet or buffers it, depending on whether or not the limited broadcast algorithm is being utilized.

Because non-cluster leaders do not process AODV control packets, they do not learn of routes through route requests and replies. It is the responsibility of the cluster leader to inform gateway nodes of routes of which they should be aware. This is accomplished through a new packet type called the *Route Activation* packet (RTAct). This packet has the following fields:

$$< Hop\_Count, Source\_Addr, Dest\_Addr, CLdr\_Addr >$$

When a cluster leader receives a route reply, the next hop indicated is the next cluster leader towards the destination. Unless the cluster leader is within transmission range of this next hop cluster leader, it must select a gateway to reach that neighbor. It chooses one of the gateways listed in the neighbor table entry for the neighboring cluster leader, and then unicasts this node a RTAct message. The *Dest_Addr* field of the RTAct contains the destination node's address, while the *CLdr_Addr* field contains the next hop cluster leader used to reach the destination. The *Source_Addr* field contains the sending cluster leader's address, and the *Hop_Count* is the number of hops to the destination.

When the gateway receives this packet, it creates a route table entry for the destination. If this node cannot reach the next hop cluster leader directly, but instead requires a joint gateway, then it in turn sends the joint gateway the RTAct message. In this way, when the gateway nodes receive data packets addressed to the destination, they know the next hop to which to forward the packet.

In the event that a gateway selected as the next hop along an active route moves out of reach of either of its cluster leaders along that route, a new gateway must be chosen to connect the two cluster leaders. The cluster leader upstream of the break selects a new gateway and unicasts to this node a RTAct packet that contains all of the necessary information to continue routing to the destination. If there are no other gateways between the two cluster leaders, the upstream cluster leader times out the route, notifies AODV of the broken link, and then AODV sends a route error message to the source of the route, as described in Section 2.2.5. Because of the likelihood of multiple gateways between cluster leaders, many link breaks can be repaired locally.

4.2.7 Related Work

One of the first clustering algorithms designed for packet radio networks was developed by Baker, Ephremides, and Flynn [9]. They describe two algorithms, the Linked Cluster Algorithm (LCA) for cluster formation, and the Link Activation Algorithm (LAA) for link activation (i.e. allocating transmission slots and resolving scheduling conflicts). The algorithms utilize a TDMA scheduling algorithm for transmission; each time a link is created or deleted the scheduling algorithm must be recomputed at each node.

Chiang, Wu, Liu, and Gerla have developed a clustering system known as Clusterhead Gateway Switch Routing (CGSR) [21]. In this protocol, packets are routed alternately between cluster leaders and gateways. The authors define various extensions that can be added to CGSR, such as priority token scheduling and gateway code scheduling, in order to control channel access. Additionally, they define a Least Cluster Change algorithm (LCC), which is designed to reduce the number of cluster leader changes, because such changes can result in a substantial amount of overhead. According to the LCC algorithm, a cluster leader change only occurs when two cluster leaders come within range of each other, or when a node is out of range of all other cluster leaders. When two cluster leaders come into contact, the leader that gives up its leader status is chosen by some means such as lowest ID or highest connectivity. Because some of the nodes in one cluster may not be members of the other cluster leader's cluster, the cluster leader change may cause some nodes to be left without a cluster leader. In this case, one of those nodes must become a cluster leader. Such changes can propagate across the network, causing a *rippling effect* of cluster leader changes. If nodes are moving rapidly, cluster leader changes may continually affect the topology of the network and generate a significant amount of control overhead. The cluster leader change mechanism utilized by ARC (Section 4.2.4) does not suffer from this problem, as it ensures that each cluster member node has at least two cluster leaders before one of them revokes its leader status.

Lin and Gerla [57] take a different approach to clustering by requiring that clusters are *non-overlapping*, and by adopting a fully distributed approach to clustering which eliminates the need for a cluster leader. Their rationale for the elimination of the cluster leader is that such a leader must do extra work on behalf of the cluster, which may become a bottleneck for the cluster. One of the assumptions needed for their algorithm is that the network topology does not change during the algorithm execution. This assumption may be unrealistic, particularly in networks with rapidly moving nodes. The ARC algorithm does not require such an assumption. Additionally, nodes within a given cluster must be at most two hops apart, using Lin and Gerla's algorithm. When a node in a cluster moves such that it becomes more than two hops away from some member of the cluster, the cluster must be reconfigured. To accomplish the reconfiguration, the highest connectivity node and its neighbors stay in the original cluster, and all other nodes are removed. This requires message overhead and coordination on the part of the cluster members, and, like the algorithm described in [21], it can also result in a rippling effect of cluster reconfigurations.

A different approach to clustering is taken by Basagni in [11], which presents two clustering algorithms. The first of these is the Distributed Clustering Algorithm (DCA), which is intended for "quasi-static" networks in which nodes are slow moving, if moving at all. The other algorithm is designed for higher mobility and is called the Distributed and Mobility-Adaptive Clustering algorithm (DMAC). Both DCA and DMAC assign nodes different *weights* and assume that each node is aware of its respective weight. The weights are used in the determination of the cluster leaders. In the DMAC protocol, if two cluster leaders come into contact, the one with the smaller weight must revoke its leader status. Like the other clustering protocols, this requirement may cause other topological changes in the clusters, since each node that just lost its cluster leader must determine whether it is still within the communication range of any other leaders. In a network of rapidly moving nodes, it is likely that frequent cluster leader changes will be required. The ARC algorithm prevents such frequent cluster changes by requiring a cluster leader to revoke its leader status only if its cluster becomes a subset of another cluster.

Like CGSR and DMAC, the Cluster Based Routing Protocol (CBRP) [39] also requires a cluster leader to revoke its leader status if two leaders come into contact. However, CBRP is able to take advantage of uni-directional links by the discovery of these links through hello messages. CBRP also has a mechanism for reducing the impact of *route-finding* broadcasts that is similar to the limited broadcast mechanism. A route request carries a list of gateway nodes that should forward the route request, together with the cluster leader to which they should forward the packet. Cluster leaders that have received the packet add their own addresses to the route request. The requirement of listing this information in each route request, and updating it each time the request is rebroadcast, results in a larger byte overhead for each request packet than is required by the limited broadcast mechanism. Additionally, by listing each gateway in the request, the algorithm is less flexible in the event of routing information that has recently become stale. By allowing each gateway to make local forwarding decisions, the limited broadcast algorithm is able to utilize the most up-to-date routing information. Finally, CBRP utilizes source routing, which requires the route reply and all data packets to carry the full source route between source and destination, and consequently adds to the byte overhead generated by the protocol.

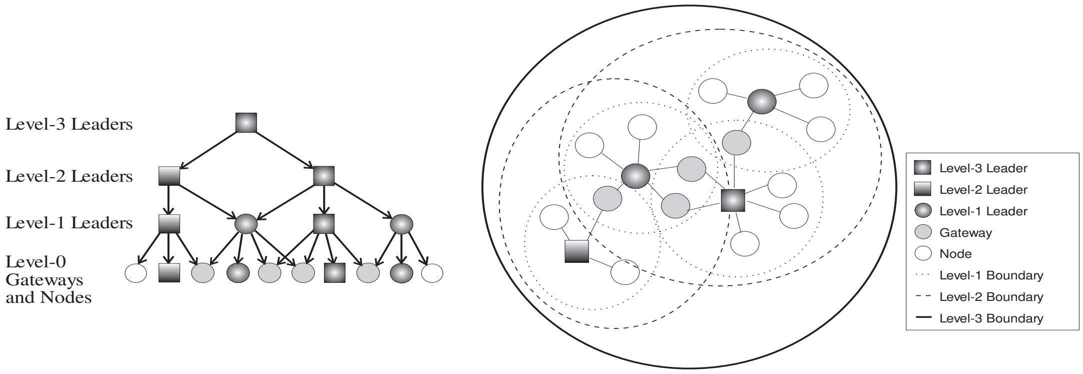


Figure 4.5: Logical Structure of Cluster Hierarchy.

One final approach worth noting is that taken by the Core Extraction Distributed Ad Hoc Routing (CEDAR) algorithm [92]. Instead of creating a cluster topology, CEDAR builds a set of nodes (i.e., a *core*) to perform route computation. Using the local state information, a minimum dominating set of the network is approximated to form the core. CEDAR establishes QoS routes that satisfy bandwidth requirements using the directionality of the core path. Link state and bandwidth availability is exchanged to maintain important information for computing QoS routes. Although CEDAR builds a core infrastructure that yields low overhead, the protocol is fairly complex and difficult to implement. The problem of calculating the minimum dominating set and the core is known to be NP-hard [36].

4.3 Adaptive Routing using Cluster Hierarchies

The clustering algorithm described in the previous section can be extended so that the cluster hierarchy is able to dynamically adapt to the topology of the network to create a hierarchical structure. A one-level clustering system increases scalability and robustness of routes, but its benefit is limited when networks grow to thousands or tens of thousands of nodes. Creating a hierarchy allows the network to dynamically adapt to the number of nodes in the network, thereby further increasing the scaling ability of the system. In such a hierarchy, the cluster leaders become the nodes of the next level, and hence are placed into level-2 clusters, and so on. Figure 4.5 illustrates an example of a cluster hierarchy. Nodes in the figure are at level-0. Nodes are grouped into clusters, where each cluster has a cluster leader. Cluster leaders are at level-1. The cluster leaders are in turn placed into level-2 groups, which has a group leader per group. Finally, the group leaders themselves are placed into a level-3 supergroup, and there is an overall supergroup leader of the entire network. The tree diagram of the figure shows how a level-2 leader is also a level-1 cluster leader, which is also a node at level-0.

Routes in a hierarchically clustered network can be recorded logically across the hierarchical tree. By recording the routes hierarchically, the robustness gained is greater than with the one-level clustering of ARC. For instance, consider the hierarchical tree given in Figure 4.6. Suppose node A wishes to send a message to node I. The physical path of the route could be $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow H \rightarrow I$. However, recording the route hierarchically, A's packet is routed to node B, which in turn sends the packet to node D.

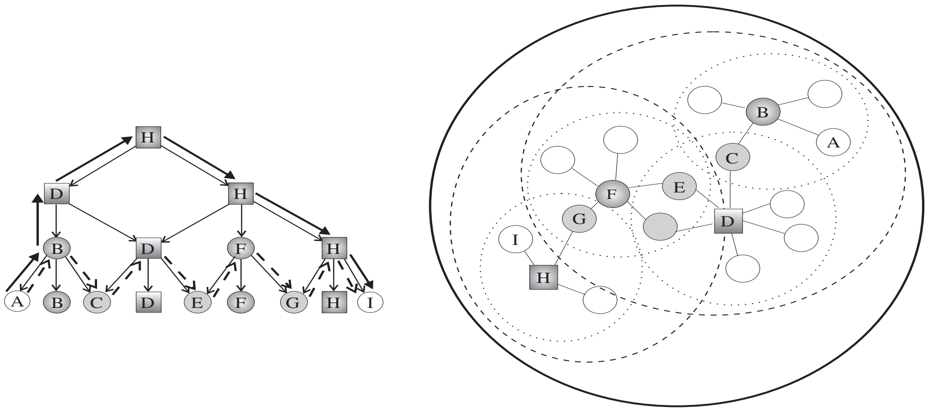


Figure 4.6: Hierarchical Routing.

Node D then sends the packet to node H, which then delivers the packet to its destination I. Hence, the hierarchical route would be recorded as $A \to B \to D \to H \to I$.

Hierarchical routing has existed for many years, having first appeared in [60]. Since then, it has seen many variations. Clusters may be disjoint, or they may overlap at multiple hierarchical levels. References [33, 51, 90, 91] discuss the benefits and drawbacks of cluster overlap for routing in mobile networks. Hierarchical clustering allows nodes to be addressed based on their position in the hierarchy. For instance, a node whose base address is z which is a member of level-1 cluster y and level-2 cluster x could have address $x.y.z$. If the node is a member of multiple clusters, it could feasibly have multiple addresses. The multiplicity of addresses allows for ease in locating nodes as they move through the network, since it is likely that one of the addresses is valid at any given time. However, this addressing scheme has the drawback that, if a node is a member of n different hierarchies, the amount of network resources required to distribute and store routing information increases by n times over that in a single hierarchy [90].

The following sections describe a new hierarchical clustering protocol called the Adaptive Routing using Cluster Hierarchies (ARCH) protocol. ARCH is a natural extension of ARC in that the cluster formation occurs in a similar manner. ARCH builds the cluster hierarchy through the periodic exchange of *Hello* messages between neighboring nodes; no other message types are needed. The hierarchy built and maintained by ARCH is able to adapt dynamically in response to the changing conditions of the network.

In the ARCH protocol, each node maintains its own sequence number, which it increments each time it transmits a Hello message. The sequence number indicates the recentness of routing information. ARCH requires this sequence number while ARC does not because of the difference in the way information in Hello messages is exchanged. In ARC, ordinary nodes and gateways record cluster leader information in their Hello messages. When other non-leaders receive these Hellos, they propagate cluster leader information if they are a joint gateway for those leaders at most one more time (Section 4.2.3). Because there is a propagation limit on this information, and because the cluster leaders do not transmit information about other cluster leaders, there are no consistency problems with the information.

On the other hand, such consistency problems do arise for the hierarchical routing information in the ARCH protocol. Information about the hierarchical status level of nodes that is transmitted by one node is retransmitted by all other nodes that are within the hierarchical boundaries. When a node transmits an update in status information for its hierarchical parents to its neighbors, those neighbors make the corresponding updates to their routing tables. However, if other nodes which have not yet received the status update transmit a Hello with the stale information, recipients of that message will have no way to determine whether this information is accurate, since it conflicts with what they had recently received. Hence, sequence numbers are needed to indicate the freshness of status information for the nodes.

4.3.1 Status Levels

The status levels of the nodes and the leaders are monotonically increasing numbers. Nodes which do not have a status level are assigned the status value `undef`. Non-leader nodes (`gateway` and `node`) have status level zero. Level-1 leaders (`cldr`) have status level one, and level-2 leaders have status level two. The status assignments continue in this manner, whereby the numerical value of the status level is equal to the hierarchical level of the leader.

4.3.2 Cluster Tables

The tables maintained for the ARCH protocol are similar to those maintained for ARC (Section 4.2.1). A node maintains one or two tables, depending on its status. The possible tables to maintain are the Node Table, Neighbor Table, and Leader Table. The fields of these tables differ only slightly from their counterparts for ARC.

The Node Table is maintained by any node with status `cldr` or greater. This table has the following fields:

- Node IP Address
- Sequence Number
- Leaders

The Sequence Number field is used to store the node's current sequence number, while the Leaders field is a linked list of neighboring leader IP addresses for which the node is a gateway. If the node is not a gateway, the Leaders list is empty.

Nodes with status greater than or equal to `cldr` also maintain a Neighbor Table. The neighbor table has the following fields:

- Leader IP Address
- Status
- Sequence Number
- HopCount
- Gateways
- IP Gateway
- Flags

The Status field indicates the status level of the neighboring leader, while the Sequence Number field indicates the leader's current sequence number. The leaders listed in the neighbor table entries need not be neighboring cluster leaders; they can be neighboring leaders on other hierarchical levels. For this reason, the hopcount to the neighboring leader is included in the entry for that neighbor. For instance, two level-2 leaders could be directly connected through a gateway, or logically connected through another cluster. Hence, the HopCount field indicates how far the leaders are from each other, and is given in *cluster hops*, whereby each cluster traversed to reach the leader is considered one hop. Individual node-to-node hops are not used because of the inconsistency in distances between clusters. Cluster leaders can be in direct contact with one another, can be connected through a single gateway, or can be connected through a pair of joint gateways. Calculating the distance by the number of clusters provides a uniform method of calculation. Note that the number of clusters traversed is incremented each time a leader appears in the path. Finally, the Gateways field is a list of the IP addresses of nodes that can be used to reach the neighboring leader.

Nodes that are not leaders maintain a Leader Table. The fields of this table are as follows:

- Leader IP Address
- Status
- Sequence Number
- HopCount
- Joint Gateways
- IP Gateway
- Flags

The Leader IP Address field is the IP address of the leader that is either a cluster leader for this node (or a cluster leader that the node can reach through a joint gateway), or that is some higher level leader for the node. The Status field indicates the status level of the leader, while the Sequence Number is the last known sequence number of that leader. The HopCount field indicates the number of hops needed to reach the leader. As in ARC, the Joint Gateways field is a list of joint gateways that can be used to reach the leader. The flags for this table are the *oneHop* and *twoHop* flags. The *oneHop* flag is set if the node is a member of this leaders' cluster. The *twoHop* flag is set if the node can reach the other leader through a directly connected joint gateway.

4.3.3 Hierarchy Initialization

The hierarchical structure of the network is created through the use of *Hello* messages. At initialization, a node sets its status level to `undef` and broadcasts a Hello message in search of neighboring cluster leaders so that it may join a cluster. The Hello message has the following fields:

$$< Flags,\ Source\_Addr,\ Source\_Status,\ Source\_Seqno,\ Hier\_List,\ CLdr1\_List,\ CLdr2\_List >$$

The node places its IP address, status level, and current sequence number in the *Source_Addr*, *Source_Status*, and *Source_Seqno* fields, respectively; it leaves the *Hier_List*, *CLdr1_List*, and *Cldr2_List* fields empty. After transmitting the Hello, the node sets a timer to wait for the reception of Hello messages. The timer expires after `cluster_discovery` milliseconds.

During the discovery period, the node keeps track of any Hello messages it receives. If the node receives a Hello message from a cluster leader, it creates an entry for that cluster leader in its leader table, and it sets the *oneHop* flag associated with that entry. Similarly, if the node receives a Hello message from a non-cluster leader, it creates entries for the cluster leaders listed in that node's Hello message, and lists the sending node as a joint gateway for those cluster leaders. If the node has not heard from that cluster leader directly, it sets the *twoHop* flag associated with that entry.

At the end of the discovery period, the node determines whether it has received a Hello message directly from one or more leaders. If it has heard from exactly one leader, it initializes its status level to `node` and joins that leader's cluster. Otherwise, if it has heard directly from more than one leader, it initializes its status level to `gateway` and becomes a gateway for those leaders from which it has heard. On the other hand, if the node did not directly receive a Hello from any leaders, it must become a cluster leader itself. In any case, it broadcasts a Hello message announcing its new status level, and those leaders from which it has heard, if any.

4.3.4 Discovering the Topology

Nodes learn of their neighbors, both physical and logical, and their hierarchical parents through the periodic transmission of Hello messages. Physical neighbors are those neighbors with which a node can communicate directly because they are within transmission range of each other. Logical neighbors are those nodes at the same status level, and contained within the same status level+1 cluster. These nodes are not necessarily within transmission range of each other. Hierarchical parents of a node are those nodes having a greater status level than the node, and within whose logical cluster the node is contained.

Each time a node transmits a Hello, it increments its sequence number. The *Hier_List* of the Hello is a linked list of structs, each of which contains the following fields:

- Leader IP Address
- Leader Status Level
- Leader Sequence Number
- Distance to Leader

The Distance to Leader is given in cluster hops.

When a non-leader transmits a Hello, it records in the *Hier_List* the identifier of each leader contained in its leader table for which it is not a member of that leader's level-1 cluster, and cannot reach that leader's level-1 cluster through the use of a joint gateway. Similarly, when a leader transmits a Hello, the *Hier_List* contained in that Hello contains a list of all the hierarchical parents of that leader.

As in ARC, there are four scenarios for the reception of a Hello message:

(i) Leader receives a Hello from non-leader
(ii) Leader receives a Hello from another leader
(iii) Non-leader receives a Hello from leader
(iv) Non-leader receives a Hello from another non-leader

The method by which the Hello is processed depends upon which of these scenarios is satisfied.

Leader Receives a Hello from Non-Leader

When a leader receives a Hello from a non-leader, it first ensures that it has an entry in its node table for that node. If it does not, it creates one. It then checks whether it has

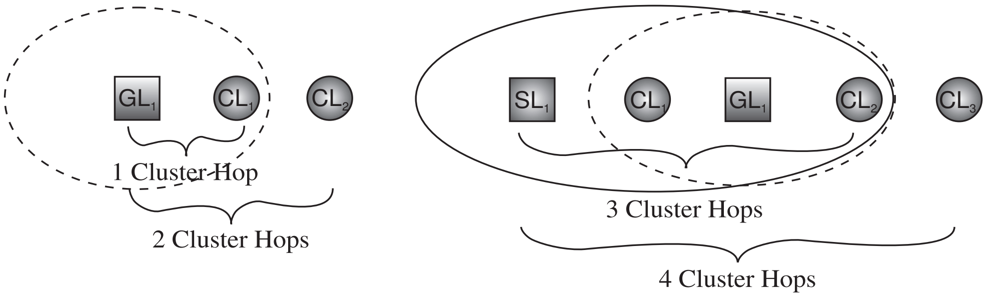


Figure 4.7: Maximum Distance Between Hierarchical Levels.

the correct listing of neighboring cluster leaders with which this node can communicate. It ensures that every cluster leader listed in *CLdr1_List* and *Cldr2_List* is contained in the Leaders list for that node. Conversely, it ensures that there are no entries in the Leaders list that are not contained in one of these lists. It then executes a similar process with the neighbor table, whereby it ensures that there is a neighbor table entry for each cluster leader contained in these lists, and that the source of the Hello message is listed as a gateway to each of those leaders. It also ensures that the node is not listed as a gateway for any leaders not contained in those lists.

The leader then examines the hierarchy information contained in the *Hier_List* field. The leader ensures that it has a neighbor table entry for each node contained in the *Hier_List*. If it has an entry for a leader in the list, it updates the status and hopcount information for that leader only if one of the following conditions are true:

(i) The sequence number for the leader contained in the *Hier_List* is greater than it's own record of that neighboring leader's sequence number in the corresponding neighbor table entry.

(ii) The sequence number for the leader contained in the *Hier_List* is equal to that in its corresponding neighbor table entry, and the hopcount indicated in the *Hier_List* entry is less than that in the corresponding entry in the leader table.

If either of these conditions is satisfied, it updates the status and hopcount information for that leader. Otherwise, it does not update this information to aviod polluting its table entries with stale status information.

After updating the neighbor table, the *Hier_List* information is used to determine whether the leader should remain at its present status, or whether it should increase its status level by one. The first step in making this decision is to determine the maximum status level of the three lists. The leader then uses the *Hier_List* to determine whether it has any parents of which it is not already aware. It traverses the list, examining the status level and hopcount values for each entry. For a leader in this list to be a parent of the leader receiving the Hello, the following formula must hold true:

$$2^{(SL-1)} - 1 \geq hc$$

where SL is the status level of the parent node, and hc is the distance, in cluster hops, of the leader from the possible parent. Hence, $2^{SL-1} - 1$ represents a maximum bound on the distance from a leader to one of its children.

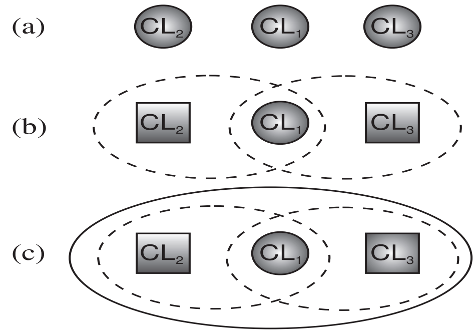


Figure 4.8: Creation of Cluster Hierarchy.

Figure 4.7 illustrates an example of calculating the hierarchical information. In the three node network, CL_1 is only one cluster away from GL_1, and so it can be a member of that node's group. However, CL_2 is two cluster hops away from GL_1, and so it is too far to be a part of that group. Hence, it needs to become a level-2 leader itself. Similarly, the five-node network is an example of a level-3 group (supergroup). Leaders CL_1, GL_1, and CL_2 are within three cluster hops from SL_1. The status level of SL_1 is three, and since $2(3-1)-1 \geq 3$, each of those nodes are a part of SL_1's supergroup. However, CL_3 is four cluster hops away, and so it is too far to be a member of that supergroup.

If this formula holds, then the leader creates an entry for this parent in its neighbor table. As a final step, the leader must determine whether it should increase its status by one level. It does this by traversing the *Hier_List*, *CLdr1_List* and Cldr2_List fields. The following conditions must hold true for a leader to increase its status level:

(i) The leader does not already have a parent.
(ii) The leader is the same status level as another leader in one of the lists.
(iii) The maximum status level of the network is greater than the status level of this leader, OR the maximum status level is the same as the status of this leader AND this leader's IP address is greater than that of the other leader with the same status level.

If each of these conditions holds, the node increases its status level by one. Otherwise, it remains at the same status level. Figure 4.8 shows an example of how the status levels are calculated. In Figure 4.8(a), leaders CL_2 and CL_1 learn of each other through the Hello messages of the nodes in their clusters. Leaders CL_1 and CL_3 also learn of each other. Because each of the leaders is only a level-1 leader, and because they each have a neighbor at that level and no hierarchical parents, one or more of these nodes must increase their status level. Between nodes CL_2 and CL_1, CL_2 becomes a level-2 leader since it has the larger IP address. Similarly, CL_3 also becomes a level-2 leader, since its IP address is greater than that of CL_1. Now the network looks like Figure 4.8(b). As Hellos continue to be exchanged, leaders CL_2 and CL_3 learn of the existence of each other. These two nodes are level-2 neighbors, and so one must increase its status. CL_3 becomes the level-3 leader, because its IP address is greater than that of CL_2. The resulting network is given in Figure 4.8(c).

As a final step after receiving the Hello, the leader must determine whether it should decrease its status level. This process is described in Section 4.3.5.

Leader Receives a Hello from Another Leader

When a leader receives a Hello message from another leader, it notes that it is now within direct communication of that leader by setting the *Monitor* flag associated with that leader's entry in its neighbor table. It then determines whether its cluster is a subset of the neighbor's cluster. The method by which this is determined, and the resulting actions taken if it is true, are described in Section 4.3.5.

If the leader's cluster is not a subset of the other leader's cluster, then the leader goes through the same process as described above with the *Hier_List* whereby it updates the neighbor table entry of the leaders contained in the *Hier_List* and then determines whether it should increase its status level.

Non-Leader Receives a Hello from Leader

A non-leader receiving a Hello from a leader first ensures that it has an entry for this leader in its leader table. If it does not, it creates one, and then updates its status to `gateway` if it was not already. Otherwise, if an entry for the leader already exists in the table, the node ensures that the *oneHop* flag is set, since it is now in direct contact with this leader. The node then traverses the *Hier_List* and ensures that there is an entry for each leader in this list in the node's leader table. It updates the status level and hopcount information for any leader for which either the sequence number in the *Hier_List* is greater than that in the leader table entry for that leader, or the two sequence numbers are equal and the hopcount for the *Hier_List* entry is smaller than that already recorded in the leader table.

Non-Leader Receives a Hello from Another Non-Leader

When a non-leader receives a Hello from another non-leader, it traverses the list of leaders in the *CLdr1_List* and ensures that the source of the Hello is listed as a joint gateway for each of those cluster leaders. It then goes through the reverse process, whereby it verifies that the source of the Hello is not listed as a joint gateway for any leaders not in *CLdr1_List*.

Finally, the node iterates through the *Hier_List*. For each entry in that list, it first determines whether that leader is already in its leader table. If so, it determines whether it should update the status information for that leader by checking the sequence number. If the sequence number for the leader in the *Hier_List* is greater than that for the leader in the node's leader table, or if the two sequence numbers are equal and the hopcount to that leader is smaller in the *Hier_List* than in the leader table entry, the node updates its information for that leader. Otherwise, it leaves the entry as is so that it does not pollute the table with stale information. If there is no entry for the leader in its leader table, it creates one.

4.3.5 Maintaining the Hierarchy

A leader decrements its status level in one of two situations: (i) when its cluster becomes a subset of another cluster, or (ii) when it determines that it no longer needs to be at as high a level.

When two leaders come within direct transmission range of each other, it is possible that one of them will have to revoke its leader status. The only time a leader revokes its status is when its cluster becomes a subset of another cluster, as described for the ARC protocol in Section 4.2.4. For the ARCH protocol, an additional constraint is imposed. A leader does

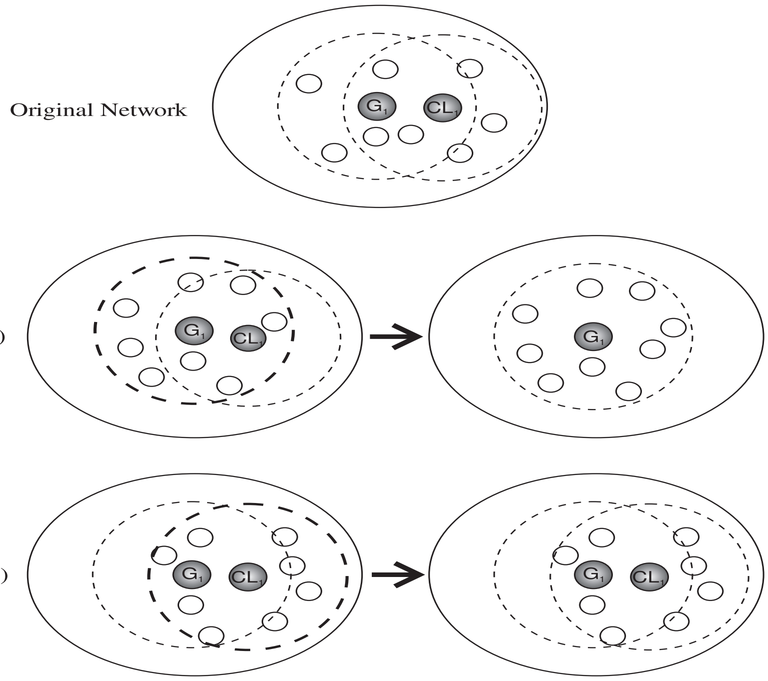


Figure 4.9: Possibilities for Decreasing Hierarchical Levels.

not give up its leader status unless its cluster is a subset of another leader's cluster, *and* its status level is less than that of the other leader's. If this condition does not hold, then the leader does not give up its status.

Figure 4.9 illustrates two possible scenarios for a given network configuration. In the original network, CL_1 and G_1 come within transmission range of each other. However, each still has members of its level-1 cluster that are not in the other leader's cluster. As the nodes move, one possible scenario is that all the nodes in CL_1's cluster will also be a part of G_1's level-1 cluster. In this case, since the status level of C_1 is one, while that of G_1 is two, C_1 can give up its leader status. This scenario is shown in Figure 4.9(i). The other possibility is that all the nodes in G_1's cluster move within the boundaries of CL_1's cluster. The subset property then holds, but because G_1's status level is greater than that of CL_1, it does not reduce its hierarchical level. Figure 4.9(ii) illustrates this scenario.

Alternatively, a leader may decrement its status level whenever its status level is greater than `cldr`, and it determines that it no longer needs to be at that high a level. A leader performs this check whenever it receives a Hello from a non-leader node, as indicated in Section 4.3.4. The leader must determine whether it has any neighboring leaders at the status level of one less than its current status level. As it checks each entry of its neighbor table, if it does not find a neighbor at a status level one less than its status level, or if it does find such a neighbor and

$$2^{(SL-1)} - 2 < hc$$

where SL is the status level of the neighboring leader and hc is the distance, in cluster hops, between the two leaders, then the leader must decrement its status level by one. Otherwise, it retains its current status level. If the distance of the leader at the lower status level satisfies the above formula, then that leader is likely to have another leader at a higher level, and so this node does not need to retain its status. However, if the neighboring leader is closer than $2^{SL-1} - 2$, then this node must continue to be its hierarchical leader.

4.3.6 Related Work

Kleinrock was an early pioneer of hierarchical routing schemes for static networks. In [45], Kleinrock and Kamoun investigate a hierarchical routing scheme with the goal of reducing routing table size. The authors determined that the length of the routing table is a strict function of the clustering structure. Clustering generally has the unwanted result of an increase in path length, and so the goal was to find an optimal clustering scheme which minimized this path length. It was determined that the number of entries in a node's forwarding table is minimized when the number of level-i clusters in each level-$(i+1)$ cluster is e, and the number of levels in the clustering hierarchy equals $\ln N$. In this case, the forwarding table contains $e \ln N$ entries. The authors also found that, under certain ideal assumptions, as the number of network nodes tends towards ∞, the difference in lengths between the optimal route and the route determined by this clustering scheme tends to zero [45]. These ideal assumptions include all level-i clusters containing the same number of level-$(i-1)$ clusters, and each level-i cluster containing the minimum-hop path between any two nodes within that cluster.

The Landmark Routing technique [98] is a novel approach to building a hierarchy as it is based on landmarks, as opposed to areas. A landmark is a router whose location is known by its neighboring routers up to some radius. All routers within that radius know how to reach the router. A hierarchy of such landmarks is built by increasing the radius of some of the routers. Nodes have hierarchical addresses based on the landmarks with which they are associated. A source node routes to a destination by sending the packet to the lowest level landmark of which both nodes are associated. As the packet approaches the destination, the granularity of routing knowledge about that destination improves, and so the packet can be accurately routed to the destination. Reference [98] does not specify an algorithm for creating the landmark hierarchy or for discovering routes to nearby landmarks, but it does note that link-state routing will not work in such a network; the distance-vector approach must be taken.

A more recent approach to hierarchical clustering is the Multihop Mobile Wireless Networks (MMWN) system [78]. The system divides nodes into switches and endpoints, where only switches can serve as packet routers. An ad hoc network is a special case where all nodes are switches. Each endpoint is associated with exactly one switch, and the switches are grouped together to form clusters. The clusters are in turn placed into hierarchical clusters, and so on. The MMWN approach requires all clusters to be disjoint. Clusters are connected through *virtual gateways*, which are similar to the joint gateways of ARCH. Cluster merges occur when the number of switches in a cluster falls below some merging threshold, N_{merge}. Similarly, cluster splits occur when the number of switches becomes greater than the splitting threshold, N_{split}.

The hierarchical clustering scheme defined by [68] distinguishes between physical and logical clustering. Physical clusters are based on geographical relationships, while logical clusters are collections of nodes with common characteristics, such as a single search team. Nodes receive both a hierarchical address based on their physical location and a logical address of the form $< subnet, host >$. A source wishing to send data to some destination sends the data to the home agent on the destination's logical cluster, using that home agent's hierarchical address. The home agent then delivers the packet to its destination using the host ID in the logical address. This Hierarchical State Routing protocol (HSR) is link-state based, where nodes store routing table entries only for nodes in their logical clusters.

4.4 Conclusion

This chapter has presented the Adaptive Routing using Clusters (ARC) algorithm and the Adaptive Routing using Cluster Hierarchies (ARCH) algorithm for establishing a hierarchical topology on network nodes. The ARC algorithm groups nodes into clusters and designates one node per cluster to serve as the cluster leader. ARCH builds upon the basic design of ARC by enabling the creation of a multi-level hierarchy which dynamically adjusts the depth of the routing hierarchy based on the topology of the network. Both ARC and ARCH record routes hierarchically, as opposed to the utilization of flat addressing, which results in an increase in the robustness of routes. Robust routes are longer-lived and hence are able to deliver more data packets to their destinations. Additionally, the algorithms define a new metric for merging clusters whenever one cluster becomes a subset of another. This scheme prevents the *rippling effect* whereby, when one cluster leader gives up its leader status, another node is left without a cluster leader and is forced to become one. Furthermore, through the *limited broadcast* mechanism, ARC and ARCH are able to achieve a reduction in the number of redundant routing control messages broadcast throughout the network.

Because hierarchical routes tend to be longer lived, routes discovered by the clustering protocols should be able to deliver more data packets than a flat hierarchical scheme. This is verified in the next chapter, where the advantages and disadvantages of hierarchical routing are discovered through the use of simulation.

Chapter 5

Clustering Performance

5.1 Introduction

Clustering protocols create a hierarchical topology on a network of nodes. This hierarchical topology has the advantages that nodes can be addressed using hierarchical addressing, and routes can be recorded across logical hierarchical links, as opposed to across per-node physical links. Recording routes hierarchically enables them to be longer lived, as logical links break less frequently than do physical links.

To evaluate the performance of the ARC protocol, two sets of simulations have been performed - ARC alone, and ARC combined with AODV. The first set of simulation results are used to evaluate the cluster topology ARC creates. The stability of the clusters is determined by examining the number of cluster leader changes and the overall cluster leader to non-cluster leader ratio. In the second set of simulations, the performance of the ARC/AODV combination is compared with that of AODV by itself to evaluate the benefit that clustering can provide.

Then, the performance of the ARCH protocol is evaluated through the examination of the hierarchical structure created by the protocol. ARCH is run in networks of differing sizes and node speeds, and the effect of these parameters is determined.

5.2 Simulation Environment

The simulations described in this chapter use the GloMoSim simulator, as described in Chapter 3. They also use the IEEE 802.11 DCF MAC layer protocol [27] for channel access.

Number of Nodes	Room Size
50	1000m×1000m
100	1500m×1500m
250	2400m×2400m
500	3450m×3450m
750	4300m×4300m

Table 5.1: Room Sizes for Network Simulations

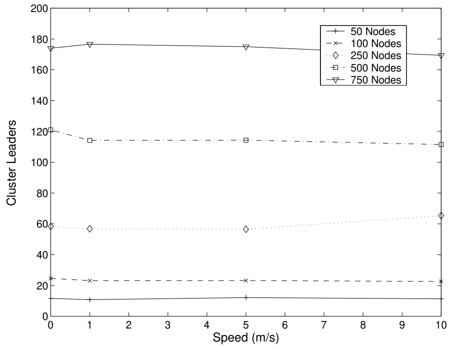


Figure 5.1: Mean Number of Cluster Leaders at Simulation End.

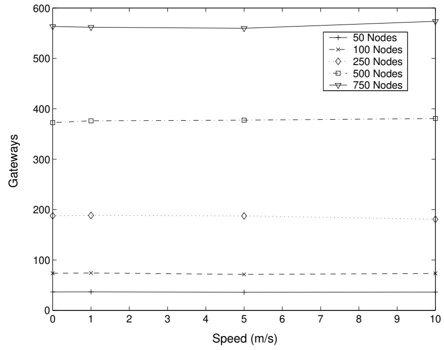


Figure 5.2: Mean Number of Gateways at Simulation End.

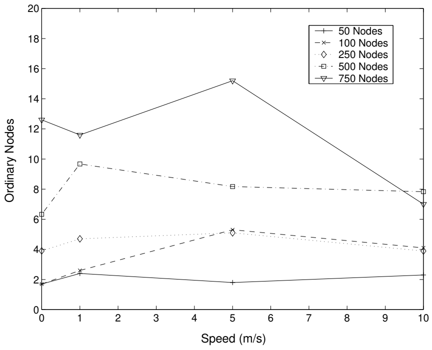


Figure 5.3: Mean Number of Ordinary Nodes at Simulation End.

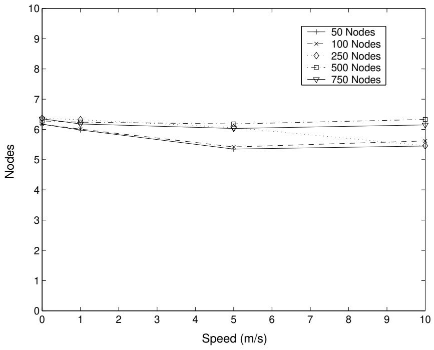


Figure 5.4: Mean Number of Nodes per Cluster.

The mobility model used is the random direction model [82], described in Section 3.2.1. The data rate in the simulations is 2 Mb/sec, and the data packet size is 64 bytes. The propagation model used is the free space model [79] with threshold cutoff included in the GloMoSim simulation package.

Networks with between 50 and 750 nodes are evaluated. The room sizes for the different numbers of nodes are shown in Table 5.1. These sizes were chosen so that the node density is relatively constant in the different simulations. There are approximately seven neighbors per node in each simulation. For each simulation, the node mobility ranges from 0 m/s to 10 m/s and the pause time is 30 seconds. Ten simulation runs were completed for each protocol/network size/node speed combination, where each run had a different initial network configuration. The results of these simulations were averaged together to produce the resulting graphs. Each simulation models 300 seconds of real time.

5.3 Basic Clustering

It is important to analyze the performance characteristics of ARC to determine the stablility of the clusters, as well as the inter-cluster connectivity level. To this end, the composition of the clusters at the simulation end is examined, as well as the inter-cluster characteristics.

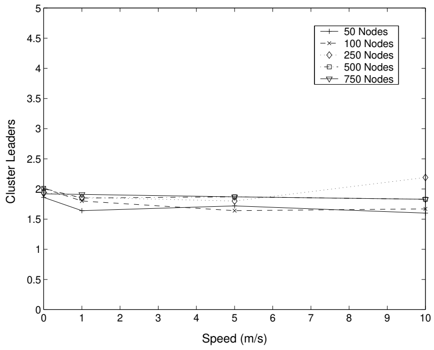


Figure 5.5: Mean Number of One Hop Cluster Leaders per Node.

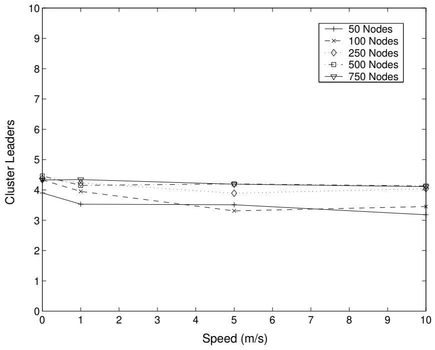


Figure 5.6: Mean Number of Reachable Cluster Leaders per Node.

Figures 5.1, 5.2, and 5.3 show the average number of cluster leaders, gateways, and ordinary nodes, respectively, at the simulation end for each of the simulated scenarios. The graphs show that the number of nodes at each status level is not particularly affected by the mobility of the nodes. Given that the node density is held constant, the primary determinant of the number of nodes at each status level is the number of nodes in the network. The number of cluster leaders in each simulation is approximately 25% of the number of nodes in the network. The number of cluster leaders in the networks is directly related to the transmission radius of the network nodes. If the nodes had a larger transmission radius, there would be fewer cluster leaders, more nodes per cluster, and more gateways between clusters. The reverse is true for a smaller transmission radius. However, whatever transmission radius for the network nodes is selected, it is important that the percentage of cluster leaders to network nodes remains relatively constant over the different network sizes, as is shown here. A constant ratio indicates the scalability of the protocol.

One other observation to note from Figures 5.2 and 5.3 is the number of gateways in the network relative to the number of ordinary nodes. In each simulation, there is only a handful of ordinary nodes. The vast majority of the nodes is able to communicate with two or more cluster leaders. This leads to high inter-cluster connectivity and increased robustness of routes, since there are multiple gateways between each cluster.

Figure 5.4 illustrates the average number of nodes per cluster in the different simulations. The figure shows that for a 250m transmission radius, the average number of nodes per cluster remains nearly constant at six, regardless of the number of nodes in the network or mobility speed of the nodes.

Figure 5.5 shows the average number of cluster leaders that are directly reachable from a given gateway or ordinary node. Similarly, Figure 5.6 shows the total number of cluster leaders that are reachable from gateways or ordinary nodes, either directly or through a joint gateway. The figures show that nodes are in direct communication with approximately two cluster leaders, or are members of approximately two clusters, but are able to communicate with, on average, two additional cluster leaders through the use of joint gateways. As in Figure 5.4, the values represented in these figures remain approximately constant over the varying number of network nodes and node mobilities. Figures 5.5 and 5.6, together with Figures 5.1, 5.2, 5.3 and 5.4, illustrate the stability of ARC across varying network sizes and mobility levels.

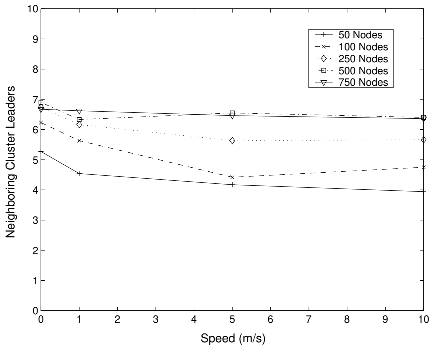


Figure 5.7: Mean Number of Neighbors per Cluster Leader.

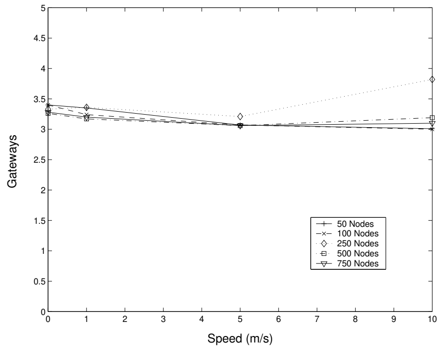


Figure 5.8: Mean Number of Gateways per Neighboring Cluster Leader.

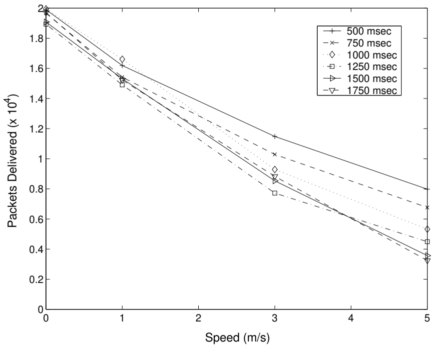


Figure 5.9: Number of Packets Delivered for Varying Hello Frequency (Without Limited Broadcast).

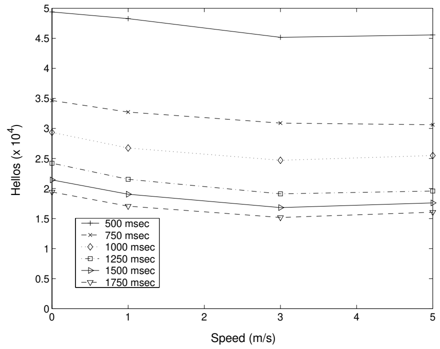


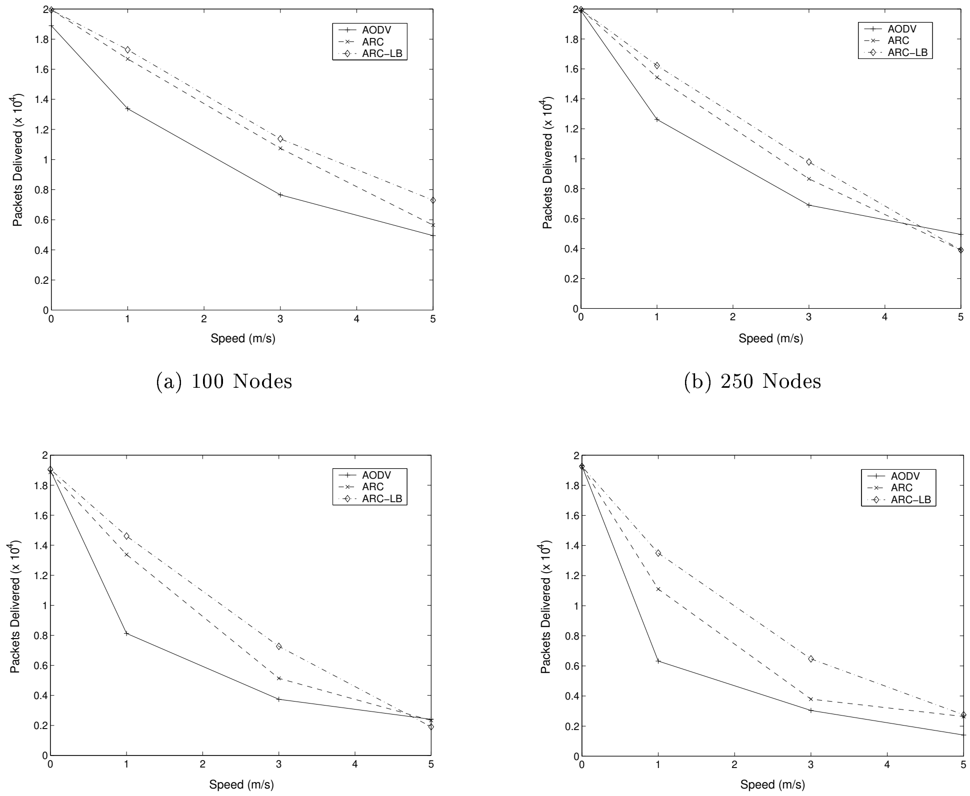
Figure 5.10: Number of Hellos Transmitted for Varying Hello Frequency.

Figures 5.7 and 5.8 illustrate the high inter-cluster connectivity levels achieved by the protocol. Figure 5.7 indicates the average number of cluster leaders that are reachable by a given cluster leader through its gateways. The number of neighboring cluster leaders remains relatively constant over the varying mobilities. The number of neighboring cluster leaders increases slightly for the increasing network sizes, due to the larger number of cluster leaders in these networks. The figure indicates a high connectivity level for all networks, with the average cluster leader being able to reach between four and seven neighboring cluster leaders through its gateways.

The number of gateways between clusters is indicated by Figure 5.8. With an average of three to four gateways between clusters, the robustness of routes is greatly increased, because if one gateway moves out of range, there are likely to be many others capable of maintaining the connection. Again, this figure also indicates the scalability of the protocol to larger networks.

5.4 ARC and AODV

The sending frequency of Hellos represents a tradeoff between bandwidth utilization and power consumption, versus the most up-to-date routing information. Figure 5.9 represents



(a) 100 Nodes

(b) 250 Nodes

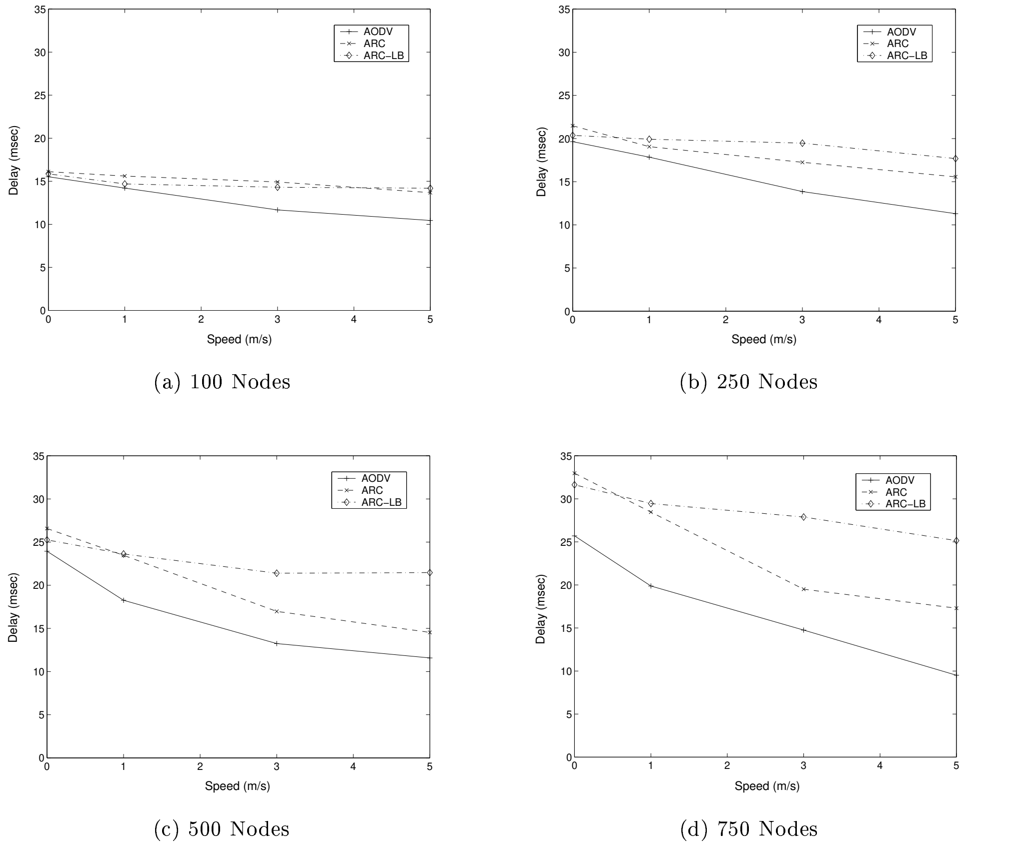
(c) 500 Nodes

(d) 750 Nodes

Figure 5.11: Number of Packets Delivered

the number of packets that can be delivered by ARC running over AODV without the limited broadcast mechanism in a network of 100 nodes. The Hello frequency of cluster leaders is held constant at 1 Hello per second, while that of the non-cluster leaders is varied. As was shown in Figure 5.1, cluster leaders comprise approximately 25% of the network nodes. Figure 5.9 shows that the difference in the number of delivered data packets increases as the rate of movement increases. For low rates of movement, a sending rate of 1000 msec produces the greatest number of delivered packets, while for faster mobility the highest sending frequency produces the best results. The drawback to increasing the sending rate, however, is shown in Figure 5.10 as the total number of Hello messages transmitted by all network nodes during the simulation. To balance the number of delivered data packets with the number of Hello messages produced, a sending rate of 1,000 msec was used for the remainder of the simulations.

Figure 5.11 shows the number of data packets delivered to destinations in each of the different sized networks. In each scenario, ARC with the Limited Broadcast (ARC-LB) mechanism outperforms ARC without Limited Broadcast. For networks with moderate movement, both ARC versions show superior performance to AODV. The difference is largest



(a) 100 Nodes

(b) 250 Nodes

(c) 500 Nodes

(d) 750 Nodes

Figure 5.12: Mean Delay of Data Packet Delivery

with slower moving nodes, and diminishes somewhat as the mobility increases. This is due in part to the selection of 1,000 msec as the Hello sending rate for non-cluster leader nodes. If a more frequent sending rate had been chosen, the performance would have improved for the higher mobilites, at the expense of more Hello messages.

The average delay witnessed by data packets is shown in Figure 5.12. The delay is the total average time for data packets to reach their destination once they are generated by the source node. The overall trend of the graphs shows that the delay increases as the size of the network grows. This is due to the increase in path length as the number hops needed to connect a source and destination grows with the size of the network. Additionally, the clustering protocols consistently have greater delay than does AODV. The path lengths determined by the clustering protocols is generally a hop or two greater than those found by AODV. This path length disparity is examined in Figure 5.13. Also, because there is more overhead in the clustered networks due to the periodic messaging requirement, nodes experience longer channel acquisition delays than they do in the AODV network. This delay contributes to the overall delay for the data packets to reach their destinations.

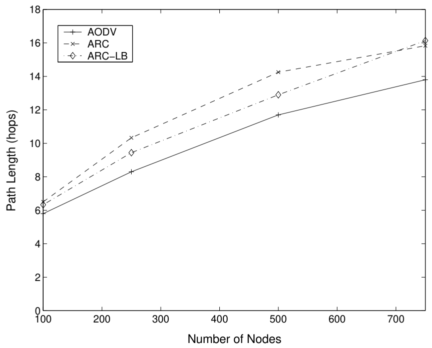


Figure 5.13: Mean Path Length in Networks.

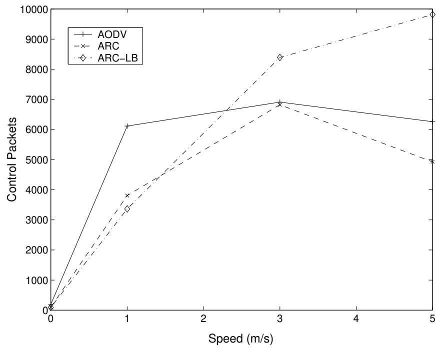


Figure 5.14: Number of Control Packets Generated in the 250 Node Network.

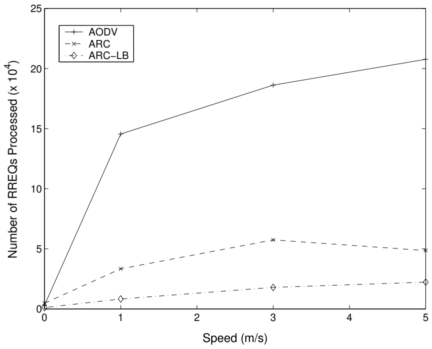


Figure 5.15: Number of Route Requests Processed in the 250 Node Network.

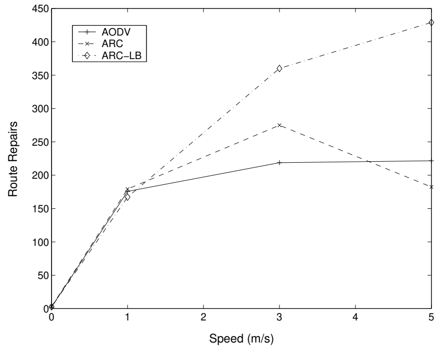


Figure 5.16: Number of Route Repairs in the 250 Node Network.

The average path length for each network is shown in Figure 5.13. One of the complaints against clustering protocols is that they often produce longer path lengths, due to the cluster leader-gateway-cluster leader routing requirement. As the figure shows, however, the average path length in ARC is only a few hops longer than that discovered by AODV.

Figure 5.14 shows the number of AODV control messages (RREQ, RREP, and RERR) messages generated in the 250-node network. For moderate node speeds, the ARC protocols generate less AODV control traffic than AODV alone. As the node speed increases, however, the amount of AODV control traffic increases in ARC-LB, relative to that of the other protocols. Fewer sessions are aborted in ARC-LB due to its superior ability to maintain routes. Because there are more sessions being maintained in the ARC-LB simulation, there is additional control traffic needed to maintain them, particularly for higher mobilities since routes are more likely to break. In AODV and ARC without Limited Broadcast, many of the sessions are forced to abort due to the inability to maintain routes in networks with rapidly changing topologies. In the basic ARC protocol, there are many more control messages rebroadcast, resulting in an increased number of collisions. Consequently, this protocol has greater difficulty maintaining routes.

A further benefit of clustering and Limited Broadcast is shown in Figure 5.15. This figure represents the total number of AODV Route Requests that were processed by network nodes during the simulation. With the clustering protocol, only cluster leaders process RREQs;

non-leaders simply rebroadcast them. Hence, while the control processing is centralized on the cluster leaders, in the AODV network each network node does the same amount of AODV control packet processing as each cluster leader does in the ARC network. The figure also shows the benefit of Limited Broadcast by illustrating the further reduction in number of Route Requests.

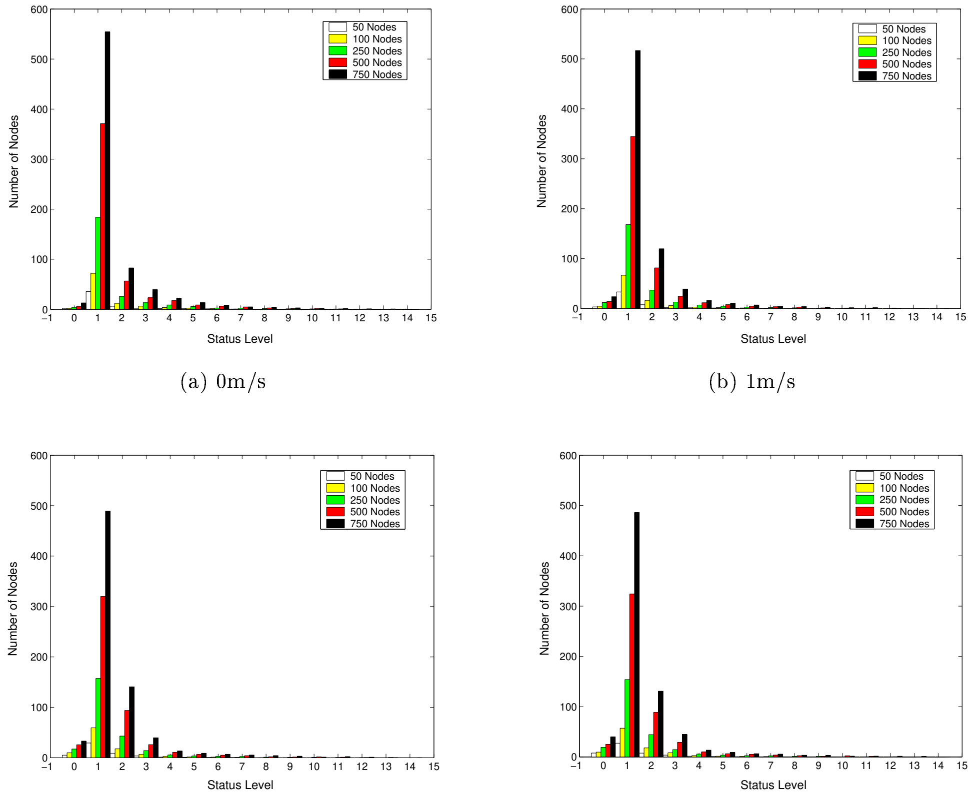
Finally, the number of repairs to active routes is illustrated in Figure 5.16. Because ARC is able to "patch" routes as gateways move out of range through the selection of a new gateway to reach the next hop cluster leader, it would seem that there should be fewer route repairs needed to maintain these routes. Only route repairs that generate Route Error messages are counted in this figure. The figure shows a unique phenomenon of cluster networks. In many clustered networks, the routes are centralized through the cluster leaders. The cluster leaders are often able to patch route breaks through the selection of new gateway nodes. However, in the event that two neighboring cluster leaders lose contact with each other (i.e., there is no longer a gateway that can be used to connect them), the number of broken routes is much greater than it would be if the routes were not as centralized. Hence, there is not a reduction in the total number of route repairs. The number of route repairs needed in ARC-LB is greater than that in the other networks because ARC-LB is able to maintain more routes. As the mobility of the nodes increases, the number of required route repairs similarly increases.

5.5 ARCH Simulations

The ARCH protocol has also been run in networks ranging from 50 to 750 nodes. Table 5.1 shows the room size corresponding to each of the different networks simulated. The node speed in each network was varied between 0 m/s and 10 m/s. The 0 m/s simulations show the operation of the protocol in a stationary network, while the other speeds allow the protocol operation to be evaluated as the nodes move more rapidly.

To evaluate the performance of the ARCH protocol, the maximum hierarchical level for each network size, as well as the number of nodes at each hierarchical level, was investigated. Figure 5.17 represents the number of nodes at each hierarchical level for the different network sizes and varying node speeds. For the purposes of the figures, status level zero corresponds to an ordinary node, while level one indicates a gateway. Cluster leaders have status level two; the hierachical levels build upwards of level two. The figure shows that, for each of the different network size and speed combinations, the vast majority of nodes are gateways. This concurs with the results presented in Section 5.3. The majority of leaders in each of the networks occurs at the cluster leader level. The number of leaders in the levels above this level gradually tapers off. The largest networks have leaders at the highest levels, as would be expected. Larger networks have more clusters, and hence require greater hierarchical levels in order to create the hierarchy.

The figure also shows that the hierarchy initially created in the static network does not change significantly as the speed of the nodes increases. There is a slight increase in the number of nodes that become cluster leaders as the speed of movement increases. This corresponds to a decrease in the number of gateways. The increase in the number of cluster leaders occurs due to the additional constraint for a cluster leader to revoke its leader status. As described in Section 4.3.5, a cluster must become a subset of another cluster, as in the case of ARC. However, the additional constraint of the leader having a lower hierarchical level than the superset leader is also needed for the leader to give up its leader status.



(a) 0m/s

(b) 1m/s

(c) 5m/s

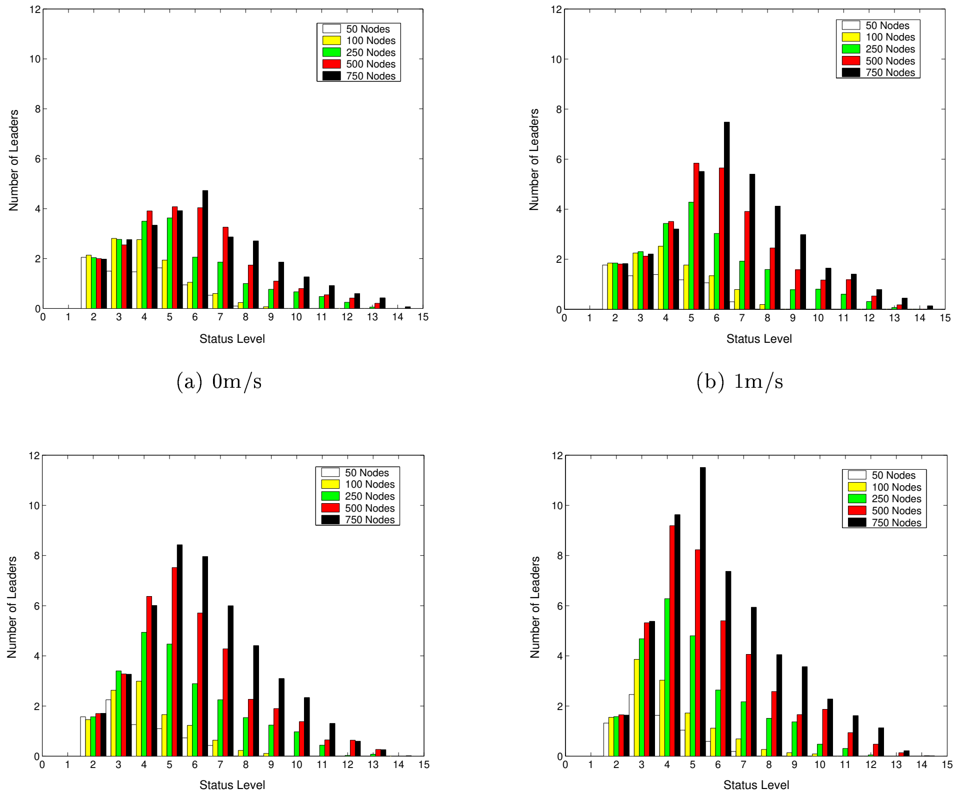
(d) 10m/s

Figure 5.17: Number of Nodes at Each Status Level

This additional constaint results in fewer leaders becoming non-leaders, and hence there is a slight growth in the number of leaders.

Figure 5.18 represents the number of leaders per node at each hierarchical level. Because the cluster leaders at level two are at the lowest leader level, the data begins at this level. The values are calculated by summing the numbers of leaders at a particular level for each of the nodes at lower hierarchical levels. For instance, the number of leaders per node at level three is found by adding the number of level three leaders for the ordinary nodes, gateways, and level two cluster leaders.

The graphs resemble bell curves due to the increasing coverage areas of the greater hierarchical levels. A node must be a member of a particular cluster, or must be able to reach that cluster leader through a joint gateway, in order to call that leader its leader. Because the coverage area grows for each hierarchical status level increase, increasingly more nodes are members of each hierarchical grouping. There are few leaders at the highest status levels, and so the number of leaders per node decreases at these levels. The number of leaders in the middle status levels grows as the node speed increases due to the increase in availability of hierarchical information. When the nodes are moving at 10 m/s, they



(a) 0m/s (b) 1m/s

(c) 5m/s (d) 10m/s

Figure 5.18: Number of Leaders per Node at Each Status Level

come into contact with more hierarchical clusters because these clusters cover a larger area. Hence, the nodes gain greater knowledge of the hierarchical structure, and are members of more hierarchical clusters. They may not actually be members of each hierarchical cluster of which they are aware, but they move in and out of the hierarchical areas so quickly compared to the Hello frequency that they do not have exact knowledge of which hierarchical areas they are a member at any given instant. Increasing the Hello frequency in the networks of more rapidly moving nodes would increase the accuracy of hierarchical routing information, at the expense of an increase in control traffic overhead, as was shown in Section 5.4.

5.6 Conclusion

The utilization of a clustering protocol for the establishment of a hierarchical network topology results in a greater number of packets delivered to destinations due to the achievement of longer-lived routes. When routes are recorded hierarchically (i.e., cluster leader to cluster leader), *any* gateway that lies between the two cluster leaders can be used to route between them. When one gateway moves out of range of either cluster leader, it is likely that there are one or more gateways that are able to take its place. This routing flexibility results in increased robustness of routes, thereby enabling more data packets to be delivered to the destinations.

Further benefits of clustering include the reduction in the number of routing protocol control packets that network nodes must process. Because control packet processing is centralized at the cluster leaders, non-cluster leaders are relieved of the responsibility of processing these packets. This does not result in an increase in packet processing by the cluster leaders, because they would still be required to process the packets in a network with flat addressing. Additionally, the limited broadcast mechanism provides a further reduction in the flooding overhead generated by defining a method for the suppression of superfluous control messages.

Clustering is not without its drawbacks, however. Because of the necessity to maintain continuously the cluster topology, periodic Hello messages must be transmitted by each node to signify its presence (or lack thereof). While the sending frequency of the Hello messages can be adjusted, the rate is directly related to the freshness of topology information. The higher the sending frequency, the more accurate the cluster information and, consequently, the more accurate the routing information. It would be beneficial to develop a scheme whereby the Hello frequency could be varied based on the average mobility of the network nodes. For slower, or even stagnant, networks, Hello messages could be sent relatively infrequently. On the other hand, for more rapidly moving networks, the sending rate could be increased. This would result in the optimum connectivity information, but would require some method for nodes to determine their mobility rate.

Another disadvantage of some clustering protocols is the centralization of routes through the cluster leaders. This centralization results in the extra taxation of cluster leaders' power supply. Additionally, because routes are centralized, when all possible routes between two gateways are finally exhausted, a large number of broken routes results. It would be beneficial to investigate the migration of established routes from the cluster leader to gateway nodes, where the cluster leader could "oversee" the routing, while not directly participating in it. Such a scheme, however, would require coordination on the part of the cluster leaders, as they would need to have information about which gateways are within communication radius of which other gateways. In a chapter on cluster-based networks, Steenstrup discusses some methods for distributed routing [94].

Chapter 6

OSI Layer Interaction Studies

6.1 Introduction

In addition to the validation of protocol operation, simulation is a useful tool for studying other properties of networks and network layer interactions. The interaction between network layers and the effect of external network conditions can have a significant impact on the performance of an ad hoc routing protocol. For instance, it is often possible for network nodes to adjust the power level of their transmissions so as to conserve battery power while still maintaining network connectivity. However, how should the nodes optimally configure themselves so that bandwidth is efficiently utilized and battery power is conserved, yet network connectivity is still maintained? Section 6.2 investigates the optimum density of nodes in mobile networks by exploring the tradeoff between reducing the number of hops to reach a destination and utilizing the bandwidth most efficiently. Using AODV as the routing protocol, the network is saturated with data packets and various results are examined. Different average network densities and mobility speeds are studied, and conclusions are drawn as to the optimum network density.

A second look is taken at the effects of transmission range in Section 6.3. Here, the effects of differing transmission ranges on AODV's multicast communication are studied. The tradeoff is now between throughput and power utilization/likelihood of collisions. In each simulation, the transmission range of the nodes is modified, and various resulting network characteristics, such as throughput, control overhead, and number of collisions, are examined.

Another network characteristic of interest is the interaction between the network layer protocol and the MAC layer protocol. It has previously been found that a protocol with low overhead at the network layer may result in a large amount of MAC overhead, due to the characteristics of the network layer transmissions [25]. Furthermore, researchers often wonder which MAC layer protocol to use for their simulations, and whether the choice of MAC layer protocol qualitatively affects the results of their simulations. Section 6.4 investigates this problem by simulating three ad hoc routing protocols over different MAC layer protocols. The results are then studied to determine whether the selection of MAC layer protocol significantly affects the relative performance of the routing protocols.

Finally, scalability is another topic of interest for ad hoc routing protocols. There is some skepticism as to the scaling potential of ad hoc routing. Many on-demand routing protocols operate through the use of a route discovery flood, where the Route Request is broadcast across the entire network. Because of the flooding nature of this broadcast, this method is likely not to scale well. In Section 6.5, AODV's operation in networks as large as 10,000 nodes is studied. Additionally, modifications to it's operation are suggested and the performance of these modifications is compared to determine mechanisms for improving the performance of on-demand protocols in large networks.

6.2 Optimum Node Density

In 1978 Kleinrock and Silvester published their well-known paper "Optimum Transmission Radii for Packet Radio Networks [46]." The paper provides an analysis that explores the tradeoffs between increased transmission radius, resulting in fewer hops to reach a destination, and the effective bandwidth lost at each node as a result of the increase in transmission power. The paper shows that the optimum number of neighbors for a given node is 5.89 (rounded to 6), and concludes that a node's transmission radius should be adjusted so that it has 6 neighbors.

While this result may be valid for stationary networks, it does not consider the ramifications of node movement on the optimum transmission power. As mobile networking becomes more popular, it is important to understand the characteristics of this type of communication so that users can communicate in an optimal manner without wasting battery life or bandwidth.

This section examines the effects of transmission power on mobile network performance. Specifically, it explores what, if any, effect mobility has on the optimum transmission radius determined for stationary networks. While increasing the transmission radius does reduce the effective bandwidth seen at individual nodes, increasing the connectivity of the network may be more important as node mobility increases. To investigate this topic, the AODV routing protocol is used for route establishment. While it is likely that different routing protocols will have different saturation levels and route characteristics, the results obtained with AODV can be generalized to most on-demand ad hoc routing protocols.

6.2.1 Simulation Environment

To investigate this topic, the average number of neighbors in each simulation is varied by adjusting the transmission range. Each simulation simulates 240 seconds and models a network of 100 nodes in a 1000m$\times$1000m area. The mobility model used is the random direction model [82]. Nodes move at a constant speed between 0 and 10 m/s. Once they reach their randomly selected destination, they rest for 30 seconds, and then resume movement towards a new destination.

The MAC layer protocol used is the IEEE 802.11 Distributed Coordination Function [27]. Because network saturation must be achieved to determine the maximum throughput of the network, the number of sources is set at 40, and each source sends twelve 512-byte data packets per second. This combination of send rate and number of sources results in network saturation. For the simulations with nodes speeds of 0 m/s and 1.0 m/s simulations, the sessions are randomly started at the beginning of the simulation and continued until the end of the simulation. For the 5.0 m/s and 10.0 m/s simulations, however, the nodes move for 200 seconds in order to stabilize the number of neighbors (see Section 3.2.1). The data sessions then begin and the simulation continues for 240 additional seconds.

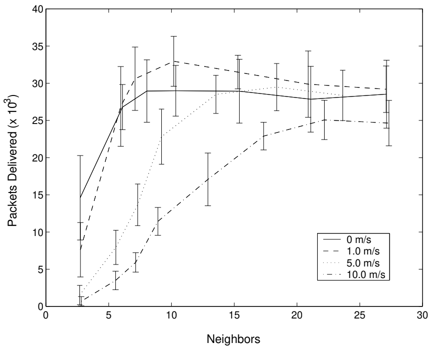


Figure 6.1: Number of Packets Delivered.

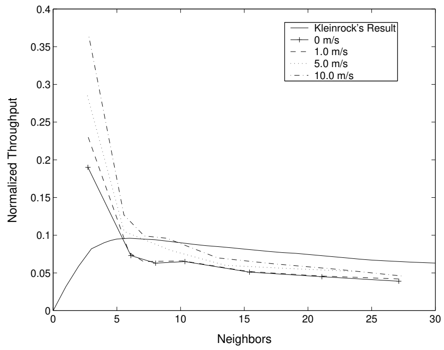


Figure 6.2: Normalized Throughput.

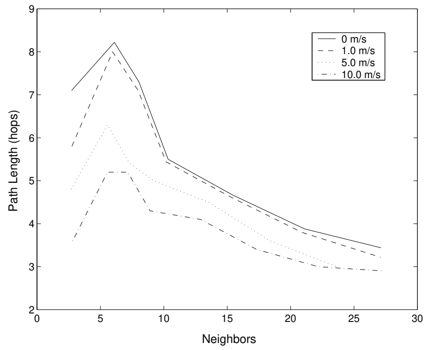


Figure 6.3: Mean Path Length.

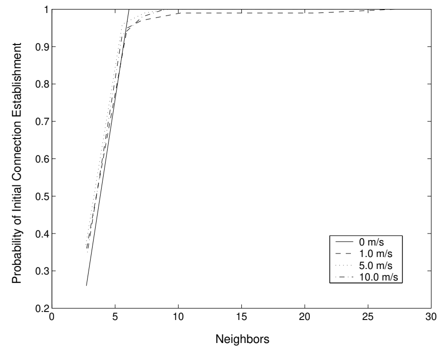


Figure 6.4: Probability of Being Able to Establish Initial Route.

6.2.2 Results

The number of packets received during the simulation is illustrated in Figure 6.1. The figure shows that for small transmission radius and low connectivity, few data packets are delivered. This is due to the frequent lack of a route between a given source and destination. As the connectivity increases, however, the number of packets delivered rapidly increases, until the curves level off. As the transmission power continues to increase, there is a gradual decline in the number of data packets delivered, due to saturation of the network. Network saturation results in an increased number of collisions, as well as reduced channel access which leads to buffer overflow. There does not appear to be a global optimal number of neighbors for all mobilities. For 0 m/s mobility, the results show this optimum to be around 7 or 8 neighbors per node, which differs only slightly from what Kleinrock proved for a stationary network [46]. As mobility increases, however, the optimum shifts to higher connectivity. The faster nodes move, the more frequently link breaks occur. Hence, even though the effective bandwidth seen at individual nodes suffers due to the increased transmission power and collisions, the number of packets delivered still increases, relative to shorter transmission ranges. This is because link breaks are less frequent and routes are maintained for longer periods of time.

Figure 6.2 represents Kleinrock's normalized throughput achieved by simulation, and compares these values to an approximation derived by Kleinrock in Figure 3 of his paper [46].

The network throughput is given by the number of successful transmissions divided by the average path length,

$$\gamma = \frac{n}{Ne\overline{h}}$$

where n is the total number of nodes, N is the average node degree, and $\overline{h}$ is the average path length. To normalize the result, the value $\frac{\gamma}{\sqrt{n}}$ is plotted.

The normalized throughput seen by the individual nodes in the simulation differs greatly from the theoretical value, particularly for a small number of neighbors. In Kleinrock's paper, it is assumed that as the transmission power decreases and the number of neighbors similarly decreases, connectivity in the network is still maintained. The throughput suffers as a result of the increased number of hops to reach the destination. However, the simulations show that for very low node densities, the network does not, in fact, remain connected. Numerous nodes or groups of nodes become disconnected from the node majority. The result of the disconnected operation is that many of the sessions abort because routes to the destination are unavailable. The only sessions that complete are those with a small path length. Hence, the normalized throughput appears artificially high. Because of this effect, the utility of this metric for mobile networks is doubtful.

Figure 6.3 illustrates the average path length of active routes. It substantiates that the lower transmission powers do have shorter path lengths, because those are the only routes that are able to be completed. Once the network is fully connected, the path length increases, and then decreases as the transmission power increases and fewer hops are needed to connect the source and the destination.

The probability of being able to establish an initial route between a source and destination is shown in Figure 6.4. For a sparsely connected network, the probability is fairly low as the network suffers from partitioning. As the node density increases, however, the probability rapidly increases for all mobilities, until it eventually stabilizes at one.

6.2.3 Conclusion

As mobile networking gains popularity, it is important to understand the characteristics of these networks so that they can be tuned to achieve optimum performance. The transmission power of wireless nodes is a key component in determining the interconnection pattern of the network. For wireless transmission, a tradeoff exists between decreasing the number of hops between sources and destinations and decreasing the effective bandwidth available to individual network nodes.

It has been shown that, as the mobility speed of nodes increases, it is desirable to increase transmission power to achieve delivery of data packets to their destinations. Moreover, there does not exist a global optimum connectivity level for varying node mobilities. For a stationary network, the optimum connectivity is 7 or 8 neighbors per node, which is similar to Kleinrock's conclusion of 6 neighbors per node. As the mobility of the nodes increases, however, an increase in the transmission power of the nodes results in a larger percentage of data packets reaching their destination and a higher network throughput. While it is likely that different ad hoc routing algorithms have different optimal connectivity levels, the results presented here can be generalized for any such algorithm.

6.3 The Effect of Transmission Range

The recent surge in wireless communications has resulted in a wide variety of available power ranges offered by wireless modems for fairly similar bandwidth. Different modems offer between 90m and 600m transmission range for 2Mb/s data rates. It would seem desirable to have the largest possible transmission radius because this would provide the greatest amount of connectivity. However, high density networks suffer from channel access delays and an increased number of collisions. Moreover, applications sometimes have to run in constrained environments. For instance, in a conference scenario, attendees are likely to be confined within a fixed area, possibly in a single large room. With that area, it is not necessarily the case that the largest transmission radius is the best solution for connectivity. A large transmission radius necessarily implies that more users are affected by each transmission, thereby limiting the effective bandwidth of neighboring users. The larger the transmission radius, the more users affected by the transmissions.

In the case of unicast data, the effects of a large transmission radius in a confined area can be somewhat mitigated by channel access schemes such as the IEEE 802.11 Distributed Coordination Function (DCF) [27]. For unicast communication, DCF provides short control packets for acquiring the channel and provides an acknowledgment to ensure data reception. Thus, although the number of collisions of the control packets might increase with larger transmission radii, thereby increasing channel access time, data throughput might not be notably affected. Because the number of hops to reach a destination is smaller for a larger transmission radius, the total delay for a data packet between source and destination might even decrease.

However, for multicast data, the situation is quite different. Multicast data packets are in some ways similar to broadcast traffic. In the case of AODV, when a node receives a data packet with a multicast destination address, it must send the packet up the protocol stack at least as far as the IP layer to determine whether to accept or forward the packet. For short transmission radii, the number of nodes affected by a single transmission is small, and therefore a network node that does not belong to any multicast groups is less likely to waste significant processing power discarding useless packets. However, as the transmission range increases, the number of nodes that receive multicast data transmissions also increases, and, assuming that the membership of a multicast group is a relatively small percentage of the total network population, the number of nodes adversely affected by these multicast transmissions similarly rises. Hence there is a tradeoff between being able to reach multicast group members in a smaller number of hops, and keeping the set of nodes affected by multicast data transmissions to a minimum.

This section investigates the nature of that tradeoff, examining the effects of varying transmission range within different confined network areas. A variety of results are examined, including the packet delivery ratio for multicast data packet delivery. This ratio is a key indicator of how well the protocol performs under the given conditions. The packet delivery ratio is found by taking the number of data packets received, divided by the number of data packets transmitted. Because each data packet must be received by every member of the multicast group, the packet delivery ratio is then divided by the number of group members to yield the normalized overall packet delivery ratio. To understand the packet delivery ratio results, various other results are examined. The average distance from a multicast group member to the group leader is directly affected by the transmission range. In networks with relatively short transmission range, the path length to the multicast group leader is much longer. Longer path lengths lead to higher probability of packet collision, and also higher probability of tree link breaks. Both of these events result in a lower packet delivery ratio. Additionally, the amount of control message overhead is directly related to the number of breaks in the multicast tree. Hence, a shorter transmission radius is likely to result in a greater amount of control traffic.

A large transmission radius increases the average number of neighbors per node. This leads to shorter path lengths to reach multicast group members, but a greater number of nodes receive each data packet broadcast. Consequently, there is an increase in battery utilization at individual nodes, as well as an increase in the likelihood of data packet collisions.

6.3.1 Simulation Environment

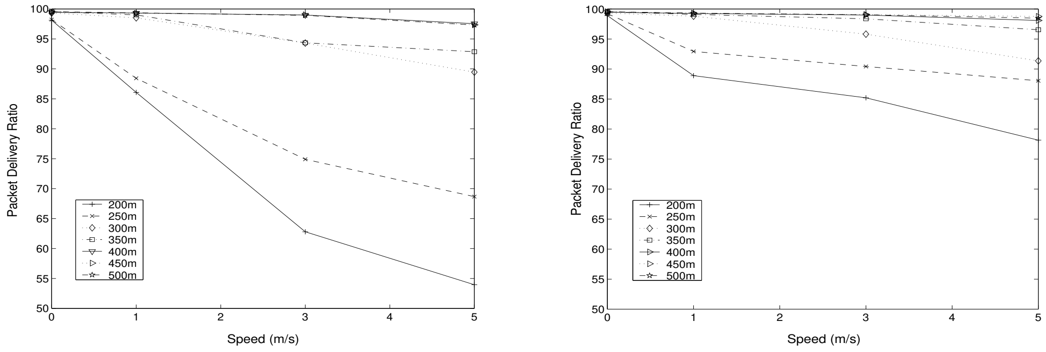
To study the effects of transmission range, two different network sizes are modeled: 1000m$\times$1000m, and 1500m$\times$300m. These two area sizes have been used in several other ad hoc network simulations [16, 25, 40]. The transmission radius is varied between 200m and 500m. Below 200m, the network nodes lose connectivity and the multicast tree cannot be maintained. The results of the 450m and 500m simulations indicate there is no significant difference in the performance of the protocol at greater transmission ranges.

In the following simulations, node movement is defined by the random direction movement model [82]. Nodes move between 0 m/s and 5 m/s and have a pause time of 30 seconds. Each simulation models a network of 50 nodes for 300 seconds. During each simulation, there is one multicast group that contains ten members. Nodes join the multicast group at the beginning of the simulation. Once all the nodes have joined the group and the tree is formed, data transmission begins. Data packets are sent at a constant rate of eight packets per second throughout the duration of the simulation. Each data packet is 64 bytes. The propagation model is the freespace model [79] with threshold cutoff. Channel access is acquired using the IEEE 802.11 Distributed Coordination Function [27].

Data packets destined for the multicast group are sent by broadcast. When a node receives a multicast data packet, it checks whether it is a part of the multicast tree for that multicast group. If not, it discards the packet. If it is a member of the multicast tree, it must process the packet. It begins by recording the unique id (sequence number) of the data packet, together with the multicast group destination address. In this way, if the node later receives the same data packet rebroadcast by another neighbor, it knows not to reprocess the packet. Data packet sequence numbers are buffered in this manner for `pkt_id_save` seconds. This process is similar to that described in Section 2.4 for broadcast.

If a node receiving the data packet is a member of the multicast group, it sends the packet up to the application layer for processing and increments its count of the number of data packets it has received. It then rebroadcasts the packet. If the node is not a member of the multicast group, it simply rebroadcasts the packet to allow reception of the packet by its next hops. To achieve 100% packet delivery, every member of the multicast group must receive the data packet. No layer 2 support for multicast is assumed.

AODV does not guarantee packet delivery; however, it does find good routes for IP's best-effort delivery. Because data packets are not buffered for retransmission, losses can occur. If a collision involving a data packet occurs at a node and the packet cannot be captured, the packet is lost. Typically, if a link break occurs on the multicast tree, data packets are lost before that break is noticed and while the link is being repaired. Hence, it is essential to take steps to monitor multicast tree links and provide for immediate repair



(a) 1000m×1000m Network (b) 1500m×300m Network

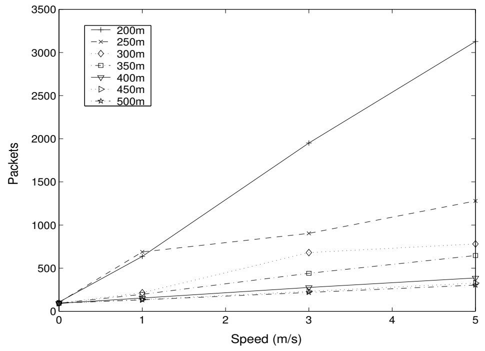
Figure 6.5: Packet Delivery Ratio.

so that link breaks result in minimum packet loss. Unfortunately, because multicast data packets are broadcast, MAC layer feedback provides no notification of broken links. For this reason, AODV provides a method of monitoring these active links, as described in Section 2.3.5.

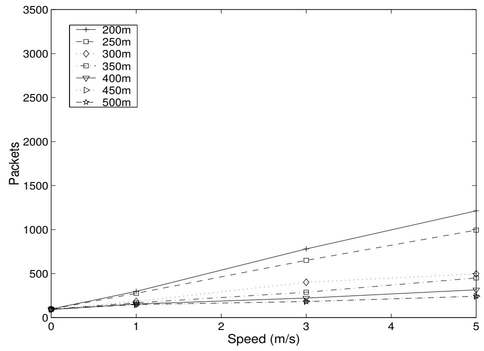
6.3.2 Results

Figure 6.5(a) shows the achieved packet delivery ratio results in the 1000m×1000m area for the seven different transmission ranges modeled, and illustrates how the packet delivery ratio is affected by the changing speed of the network nodes. Figure 6.5(b) shows the same results for the 1500m×300m simulations. For both network configurations, an increase in range yields an increase in the packet delivery ratio, or the number of data packets received by multicast group members. For the 1000m×1000m network, the increase in speed results in a decrease in the packet delivery ratio for the more sparsely connected networks. In the 1500m×300m network, the increase in speed has a smaller effect on the overall packet delivery.

Figure 6.6 shows the number of control packets produced during each of the simulations. The number of control packets represented here is found by summing the number of RREQ, RREP, MACT, and GRPH packets initiated. The figure shows a reduction in the number of control messages as the transmission range is increased. For zero mobility, there are no repairs needed to the multicast tree. Figure 6.6 indicates that the number of control messages needed to initialize the multicast tree is approximately constant at the different transmission ranges. The variation in control overhead becomes more significant once the nodes begin moving. As nodes travel more quickly, there are more breaks and repairs to the multicast tree, and hence there are more control packets generated. Because there are fewer link breaks for the higher transmission ranges, there are subsequently fewer control packets generated in these simulations as well. The 200m transmission range in the 1000m×1000m network particularly suffers from the increase in mobility. The network topology and low connectivity in this network configuration often results in multiple attempts per link break to re-establish tree link connections.

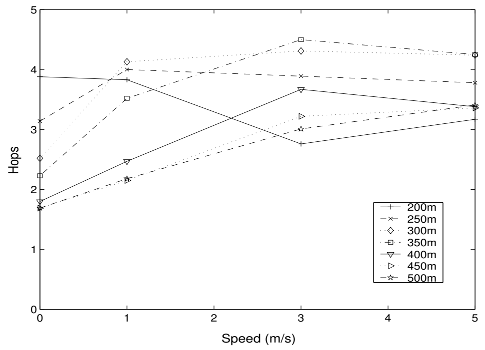


(a) 1000m×1000m Network

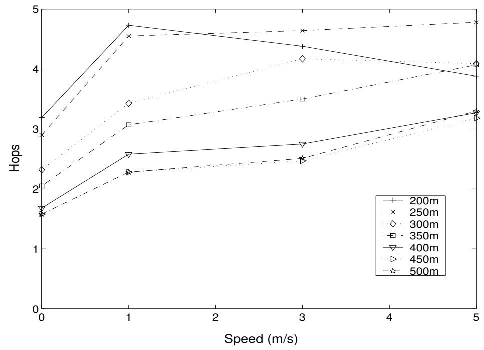


(b) 1500m×300m Network

Figure 6.6: Mean Number of Control Packets Generated.



(a) 1000m×1000m Network

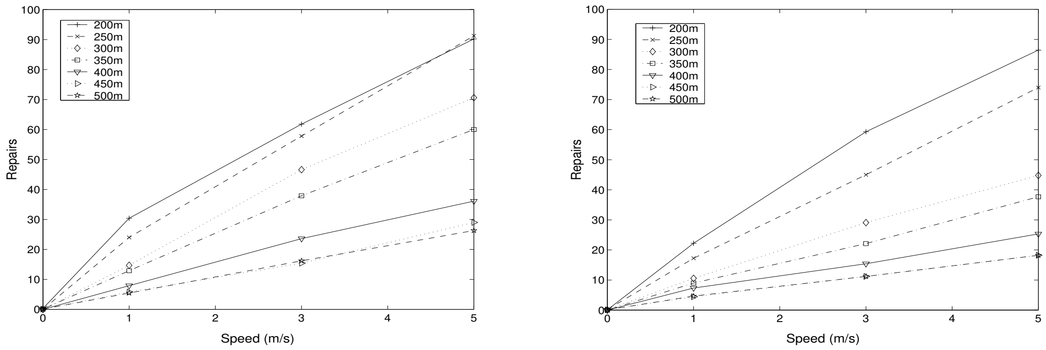


(b) 1500m×300m Network

Figure 6.7: Mean Number of Hops to Multicast Group Leader.

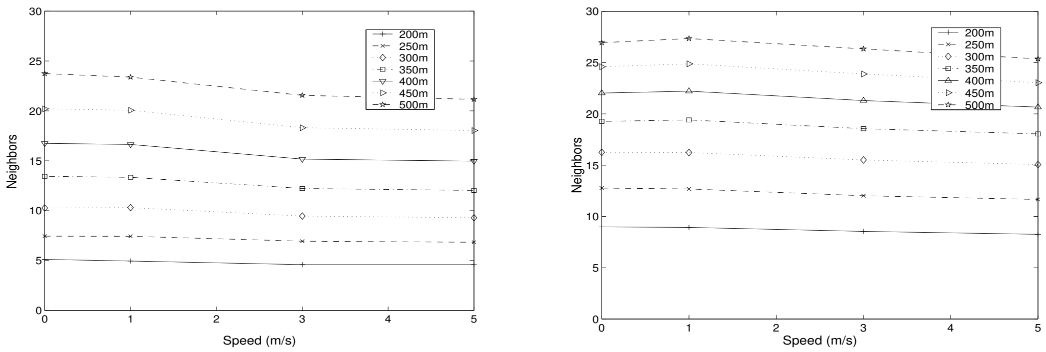
To understand the variations in packet delivery ratio and control overhead, it is necessary to investigate how the transmission range affects other aspects of the network. Figure 6.7 illustrates the effect of the transmission range on the average distance of a multicast group member to its group leader. Though the multicast tree is bi-directional, the distance to the group leader gives an indication of the size of the tree and the number of hops data packets must traverse between destinations. The graphs indicate that for smaller transmission range, the average number of hops to the group leader is larger. A larger distance to the group leader results in greater potential for packet collisions, as well as a higher chance of link breaks. Thus, a greater transmission range results in fewer hops on the multicast tree, which produces a better packet delivery ratio. It is interesting to note that there is not a notable difference in results between the two network sizes.

Because the average path length to the group leader is inversely proportional to the transmission radius, the number of repairs to the multicast tree is also likely to be inversely proportional. This is verified in Figure 6.8. These graphs show the average number of repairs needed to fix broken multicast tree links during the simulation. As expected, the number of repairs increases for increasing speed. Also, as expected, the largest transmission range



(a) 1000m×1000m Network (b) 1500m×300m Network

Figure 6.8: Mean Number of Multicast Tree Repairs.

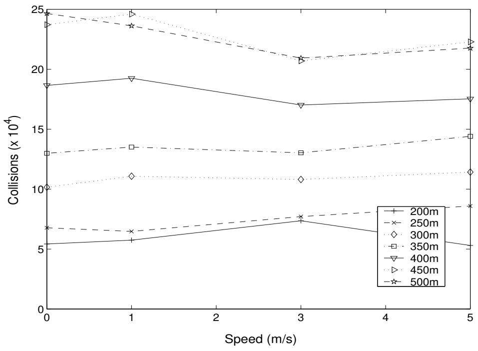


(a) 1000m×1000m Network (b) 1500m×300m Network

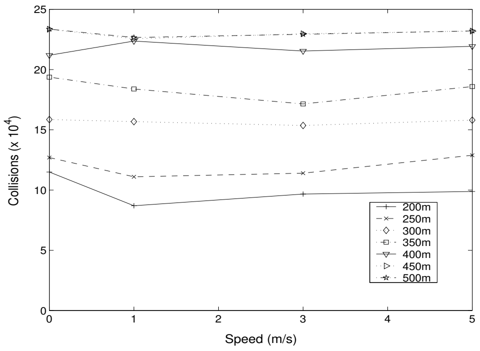
Figure 6.9: Mean Number of Neighbors.

resulted in the fewest number of repairs, again resulting in a better packet delivery ratio for these networks. Because the amount of control overhead generated during the simulation directly corresponds to the number of repairs to the multicast tree, the increase in repairs for the networks with smaller transmission ranges results in an increase in control overhead, as Figure 6.6 shows.

Having examined only these results, it appears that increasing the transmission radius has a uniformly positive effect on the network. A larger transmission radius results in better packet delivery ratio, fewer link breaks in the multicast tree, and less control overhead. However, it is also necessary to examine the impact that increasing the neighborhood size has on the network. Increasing the transmission range also increases the number of neighboring nodes affected by each transmission. Figure 6.9 indicates the number of neighbors of nodes at the varying transmission ranges. For the purpose of the figure, two nodes are considered to be neighbors if the distance between them is less than or equal to the given transmission radius. As would be expected, the number of neighbors per node increases with increasing transmission range.

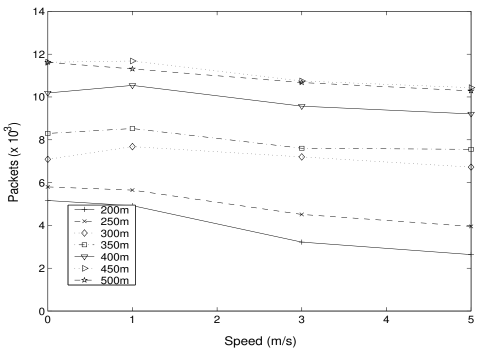


(a) 1000m×1000m Network

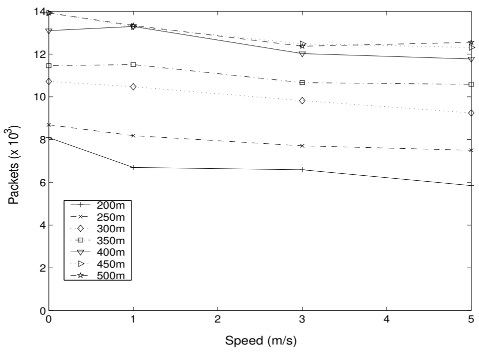


(b) 1500m×300m Network

Figure 6.10: Mean Number of Collisions.



(a) 1000m×1000m Network

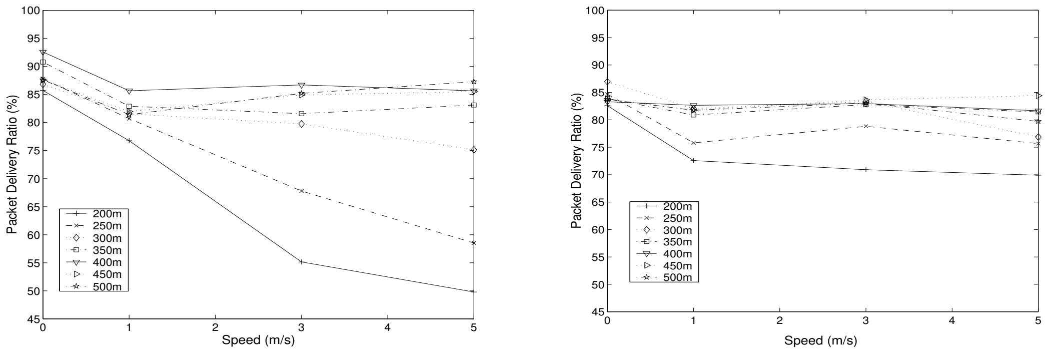


(b) 1500m×300m Network

Figure 6.11: Mean Number of Multicast Data Packets Received By Non-Group Members.

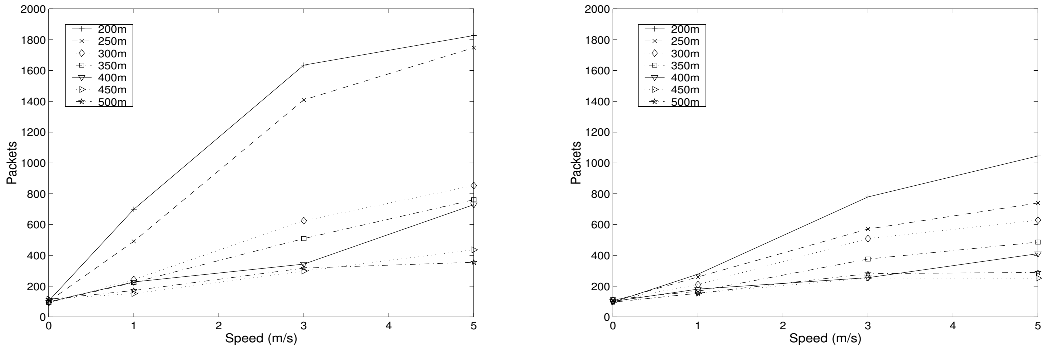
One of the primary effects of increasing the size of the neighborhood is the increase in the number of packet collisions. Figure 6.10 illustrates the total number of packet collisions in the networks. The graphs show that the number of collisions rapidly increases as the transmission range grows. The number of collisions with 450m and 500m transmission ranges is nearly five times that of the 200m transmission range network.

To explore further the effect of the different neighborhood sizes, the number of multicast data packets received at nodes that are not group members is examined in Figure 6.11. At a packet transmission frequency of eight packets per second and simulation length of 300 seconds, just under 2200 data packets are initiated during the simulation. The figures indicate that, for a transmission radius of only 200m, non-group member nodes receive each multicast data packet approximately twice, on average. However, for the greatest transmission radius, nodes receive each data packet approximately six times. Such a redundancy in packet reception is likely to have a negative effect on a node's battery lifetime, as the node will spend a large percentage of its battery power processing unnecessary packets. The higher nodal density engenders additional contention for slotted MAC schemes. This causes more collisions during the contention period, resulting in increased



(a) 1000m×1000m Network (b) 1500m×300m Network

Figure 6.12: Packet Delivery Ratio for Increased Data Rate.



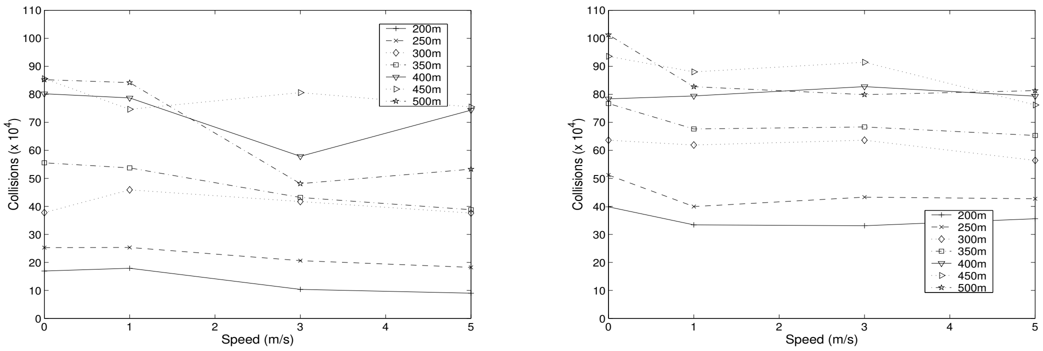
(a) 1000m×1000m Network (b) 1500m×300m Network

Figure 6.13: Control Packet Overhead for Increased Data Rate.

queuing delays as nodes are forced to wait longer periods of time between packet transmissions.

The results presented so far are based on one source with a moderate sending rate (eight packets per second). To determine the interaction between transmission range and network traffic, a second set of experiments was performed with the number of sources increased to two and with each source sending 20 packets per second. The data packet size in these simulations is 512 bytes, as opposed to the 64 byte packets in the previous experiments.

The packet delivery ratio for this set of experiments is shown in Figure 6.12. The results here differ from the packet delivery ratio for the lower sending rate, shown in Figure 6.5. Here, it is no longer the case that the longest transmission range produces the greatest packet delivery ratio. The nodes in these networks are not able to deliver as many data packets due to the increased contention for channel access and the increased likelihood of collisions. The transmission range of 400m in the 1000m×1000m networks and 400-450m in the 1500m×300m network results in more delivered data packets than does the 500m transmission range. The ranges of 200m and 250m still produce the lowest packet delivery ratio due to the lower connectivity in these networks.



(a) 1000m×1000m Network (b) 1500m×300m Network

Figure 6.14: Number of Collisions for Increased Data Rate.

The control packet overhead for the increased data rate is given in Figure 6.13. These graphs do not vary significantly from the results shown in Figure 6.6. The control traffic in the 200 and 250m networks still dominates in both network sizes.

Finally, the number of collisions is shown in Figure 6.14. Here, the results vary slightly from those in Figure 6.10 for the large transmission range. Except for the static networks, the number of collisions is greatest in the 450m transmission range scenarios.

6.3.3 Conclusion

The transmission range and network size are key determinants of AODV's multicast performance. Increasing the transmission range has many benefits. The number of links on the multicast tree is reduced, resulting in fewer tree links which need to be maintained. Each multicast tree link repair requires control message overhead. Reducing the number of repairs has the advantage of also decreasing the amount of control overhead. For unloaded networks, the packet delivery ratio increases for longer transmission ranges due to the reduction in the number of hops between group members and the longer-lived tree links.

Despite these advantages, a large transmission range also causes more network nodes to be affected by multicast data transmissions, even when the nodes do not need to receive these packets. A large transmission radius therefore drains the battery not only of the transmitting node, but also of neighboring nodes within the source's transmission range. Worse, a large transmission radius reduces the effective bandwidth available to the individual nodes and increases the number of collisions seen throughout the network because more nodes are competing for and utilizing the same network bandwidth. This increase in the number of collisions causes a reduction in the packet delivery ratio for traffic patterns that significantly load the wireless medium. Perhaps most importantly, increasing the transmission range places disproportionately greater demands on the power requirements of the (typically battery-powered) mobile nodes. Thus, it is crucial for the consumer market to find ways to minimize power consumption.

6.4 The Effect of the MAC Layer Protocol

With the increase of interest in ad hoc routing protocols, there have recently been investigations comparing the performance of these protocols under various conditions and constraints to determine the relative merits of the protocols [16, 25, 40, 55]. There has been some discussion as to the correct Medium Access Control (MAC), or link layer (level-2 of the OSI reference model), protocol to use for channel access. Many early protocol simulations utilized the Carrier Sense Multiple Access (CSMA) protocol [47]. Since the advent of the IEEE 802.11 protocol [27], however, most protocol evaluations have elected to run over this channel access protocol, because it provides both prevention and detection of the hidden terminal problem [96].

This section combines three ad hoc routing protocols with each of four MAC layer protocols and compares the performance of each combination to determine whether the selection of the MAC layer affects the relative performance of ad hoc routing protocols. It is likely that the performance of the protocols will be best when run over IEEE 802.11, due to its channel acquisition characteristics. However, the question is whether the protocols degrade proportionately to each other when run over the other MAC layer protocols.

6.4.1 Routing Protocols

To analyze the effects of MAC protocols, three ad hoc routing protocols were selected for study. The first is the Wireless Routing Protocol (WRP) [62], which is a distance vector table-driven protocol [59]. Table-driven protocols periodically exchange routing table information in an attempt to maintain an up-to-date route from each node to every other node in the network at all times.

The second protocol studied is the Fisheye State Routing protocol [67]. This protocol is a variation on the basic link state table-driven algorithm [61], whereby update message entries are exchanged between nodes at different frequencies, depending on their distance from each other. Routing information for a node's immediate neighborhood is kept the most up-to-date, while that for nodes farther away is less accurate. This method helps reduce the table size in routing table exchanges, while still maintaining routes to each network node.

Finally, the Ad hoc On-Demand Distance Vector Routing protocol [72, 74] is included as an example of an on-demand protocol. On-demand protocols establish routes only when they are needed by a source node, and maintain these routes only as long as the source node requires them.

The following sections provide overviews of the WRP and Fisheye protocols. See Section 2.2 for a description of unicast AODV.

Wireless Routing Protocol

The Wireless Routing Protocol (WRP) [62] maintains routing information through the exchange of triggered and periodic updates. When a node notices a link break with one of its neighbors, it broadcasts an update message containing the distance and second-to-last hop information for each destination for which the routing information has changed. The second-to-last hop information is used to reduce routing loops. A neighboring node receiving an update message modifies its distance table entries and checks for new paths through other nodes. Any new paths are relayed back to the original node so that routing consistency is maintained throughout the network. Furthermore, a node successfully receiving an

Protocol	Mechanism
CSMA	CSMA
MACA	PSMA/RTS/CTS
FAMA	CSMA/RTS/CTS
IEEE 802.11 DCF	CSMA/CA/RTS/CTS/ACK

Table 6.1: Summary of MAC Protocols

update message transmits an acknowledgment back to the sender, indicating that the link is still viable.

In the event that a node has not transmitted anything within a specified period of time, it must transmit a *Hello* message (instead of exchanging the entire route table) to ensure connectivity. Otherwise, the lack of messages from a node indicates the failure of that link. When a node receives a Hello message from a new node, it sends that neighbor a copy of its routing table information.

Fisheye State Routing

Fisheye State Routing (FSR) [67] is a variation of link state table-driven routing that maintains a topology map at each node. To reduce the overhead incurred by control packets, FSR modifies the link state algorithm in three ways. First, link state packets are not flooded; only neighboring nodes exchange link state information. Second, the link state exchange is time-triggered, not event-triggered. Finally, instead of transmitting all routing table information at each iteration, FSR uses different exchange intervals for different entries in the table. More precisely, entries corresponding to nodes that are nearby (within a predefined *scope*) are propagated to neighbors more frequently than entries of nodes that are far away. These modifications reduce the control packet size and the frequency of transmissions. As a result, FSR scales well to large networks because link state exchange overhead is kept low. As mobility increases, routing information for remote destinations can become less accurate; however, as a packet travels nearer to its destination, it is forwarded by nodes with increasingly more accurate routing information.

6.4.2 MAC Protocols

The MAC protocols selected for this study represent a progression in protocol development. Each one builds upon the previous one through the addition of either control overhead or carrier sensing to mitigate the effects of the hidden terminal problem and to achieve better network throughput. Table 6.1 summarizes the mechanism of each MAC protocol included in the study. Packet sensing (PSMA) implies that carrier sensing is not performed before packet transmissions. The following sections describe each of the MAC protocols considered in this evaluation.

Carrier Sense Multiple Access

The Carrier Sense Multiple Access (CSMA) [47] protocol is the most primitive of the MAC protocols in this study. The CSMA version used is non-persistent CSMA. In this protocol, a node senses the channel for ongoing transmissions before sending a packet. If the channel

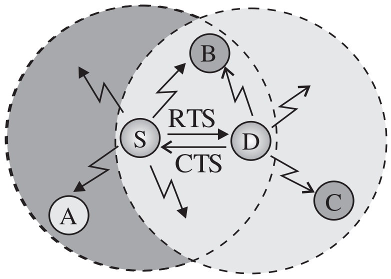


Figure 6.15: Effect of RTS/CTS Control Messages.

is already in use, the node sets a random timer and then waits this period of time before re-attempting the transmission. On the other hand, if the channel is not currently in use, the node begins transmission.

Multiple Access with Collision Avoidance

The Multiple Access with Collision Avoidance (MACA) [44] protocol improves upon CSMA by taking steps towards the avoidance of the hidden terminal problem. The protocol defines Request-To-Send (RTS) and Clear-To-Send (CTS) control packets to announce an upcoming transmission. A node wishing to send a unicast transmission broadcasts a RTS message containing the length of the data frame that will follow. Upon receiving the RTS, the receiver responds by broadcasting a CTS packet that also contains the length of the upcoming data frame. Any node hearing either of these two control packets must be silent long enough for the data packet to be transmitted. In this way, neighboring nodes do not transmit during the data transmission, and the number of collisions is reduced. Figure 6.15 illustrates the operation of the RTS/CTS control messages. When S broadcasts the RTS message, both nodes A and B receive it and delay their transmission attempts. Similarly, when node D responds with a CTS, nodes B and C also receive the CTS and are silent during the data transmission.

In the event that two nodes send simultaneous RTS frames to the same node, the RTS transmissions collide and are lost. If this occurs, the nodes that sent the unsuccessful RTS packets set a random timer utilizing the binary exponential backoff algorithm for the next transmission attempt.

Floor Acquisition Multiple Access

The Floor Acquisition Multiple Access (FAMA) variant utilized in this study is FAMA-NTR (Non-persistent Transmit Request) [32]. FAMA-NTR builds upon the MACA protocol by adding non-persistent carrier sensing to the RTS/CTS exchange. Before transmitting a RTS frame, a node first senses the channel to determine if it is already in use. If the channel is busy, the node calculates a random backoff period to wait before sensing the channel again. The addition of this carrier sense to the control packet exchange aids in the prevention of control packet collisions.

IEEE 802.11 DCF

The IEEE 802.11 MAC protocol specifies a Distributed Coordination Function (DCF) [27] that is based on the same RTS/CTS message exchange for unicast data transmissions as the previous MAC protocols. Where 802.11 differs, however, is in its use of collision avoidance before RTS transmission, and its requirement of an acknowledgment (ACK) transmission by the receiver after the successful reception of the data packet. The inclusion of the ACK allows immediate retransmission, if necessary, by determining whether the data packet was successfully received. In the case of node mobility, the ACK can aid in the detection of hidden-terminal interference that was not detectable when the CTS message was sent.

6.4.3 Simulations

Simulation Environment

The simulations were performed using the GloMoSim Network Simulator developed at UCLA [8]. Node movement is modeled by the random waypoint mobility model [16]. Nodes move at a speed between 0 and 10 m/s. When a node arrives at its randomly chosen destination, it rests for some pause time. It then chooses a new destination within the simulation area and resumes movement. The pause times are varied between 0 and 300 seconds. Each MAC protocol/routing protocol/pause time combination is run for five different initial network configurations.

Each run is executed for 300 seconds of simulation time and models a network of 100 nodes in a 1500m$\times$1500m area. Each node has a transmission radius of 250m. The propagation model is the free space model [79] with threshold cutoff. This model has a power signal attenuation of $1/d^2$, where d is the distance between nodes. The radio model also has capture capability, whereby a node can receive a packet even in the presence of noise. There are 20 data sessions between randomly selected sources and destinations. The bandwidth is 2 Mb/s, the data packet size is 512 bytes, and packets are sent at a rate of four per second by each source.

Table 6.2 shows the parameter values used for the routing protocols in the experiments. The Hello Interval for each protocol is the frequency at which the protocol sends Hello messages, and the Max Allowed Missed Hellos value is the number of Hello messages a node is allowed to miss from a neighbor before it must assume that neighbor is no longer within transmission range.

The majority of the parameter values for WRP were taken from those suggested by the designers of the protocol and specified in [62]; however, a few of the values were modified to maximize WRP's performance in the simulation environment. The timer values were set so as to send more frequent connectivity updates but less frequent retransmissions than suggested. The former modification is needed because of the high mobility speed in the experiments, and the latter is due to the fact that with the MAC protocols selected, retransmitting at twice the round trip time would flood the MAC buffer, in addition to causing unnecessary collisions with cross traffic in the channel.

Using the FSR protocol, a node includes in its route update message entries for nodes outside its scope every interscope update interval. Entries for nodes inside the scope are included in every update message transmission. Note that the interscope update interval is much larger than that of intrascope update.

	Parameter	Value
WRP	HELLO Interval	1 sec
	Max Allowed Missed HELLOS	4
	Update ACK Timeout Interval	1 sec
	Retransmission Timer	1 sec
	Retransmission Counter	4
FSR	Scope	2 hops
	HELLO interval	5 sec
	Max Allowed Missed HELLOS	3
	INTRASCOPE UPDATE interval	5 sec
	INTERSCOPE UPDATE interval	15 sec
AODV	HELLO Interval	1 sec
	Max Allowed Missed HELLOS	3
	RETRANSMIT TIME	750 msec

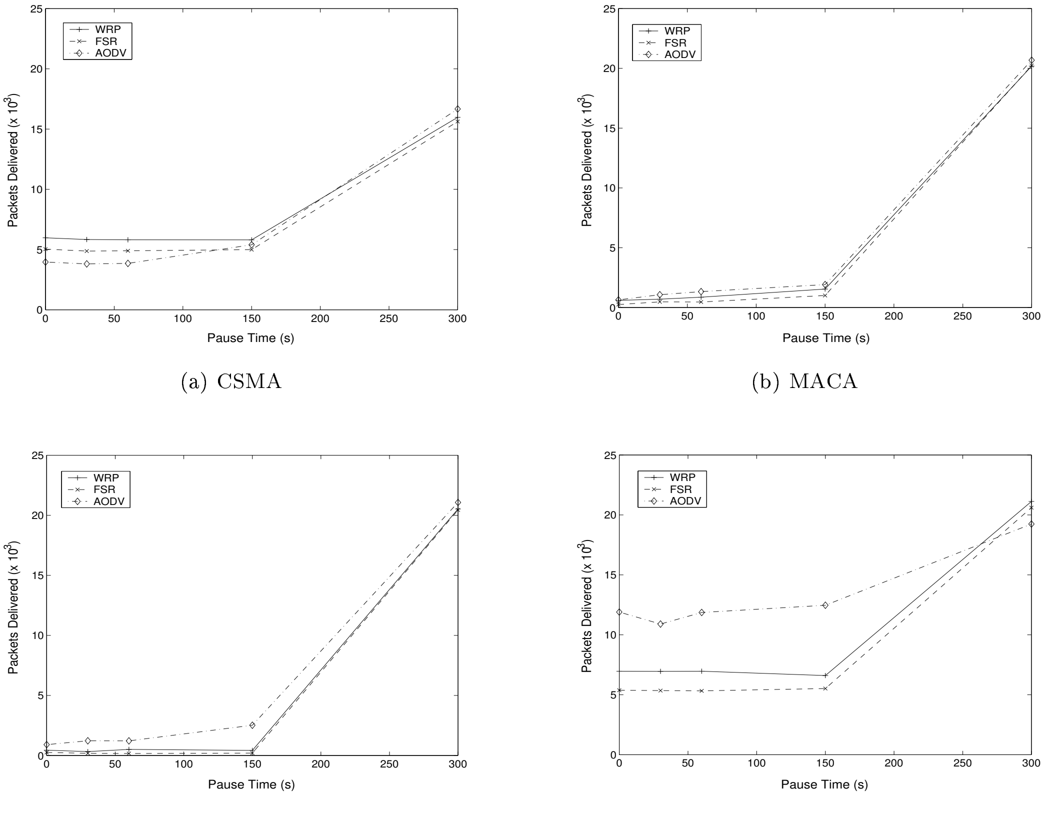
Table 6.2: Parameter Values

When AODV is run over IEEE 802.11, Hello messages do not need to be used due to the MAC layer feedback of unreachable next hops. When combined with the other MAC protocols, however, Hello messages are needed because such feedback is not available. When Hello messages are used, a node transmits a Hello once each second as long as the node has not broadcast any other control message during the previous second. Additionally, promiscuous listening mode is enabled for AODV whenever Hello messages are utilized. In wireless transmissions, all the neighbors of a given node receive that node's transmissions. The nodes then filter these transmissions based on the MAC address included in the packet header; if the packet is not destined for a particular neighbor, that neighbor does not process or forward the packet. However, when promiscuous listening mode is enabled, the neighboring nodes receiving these packets may choose to process the received packets in order to, for instance, obtain connectivity information. This promiscuous reception of packets allows AODV to determine more quickly when link breaks have occurred. The RETRANSMIT TIME value in Table 6.2 is the maximum allowable time between receptions of data packets from neighbors on active paths.

Results

To determine whether the selection of MAC protocols affects the relative performance of the protocols, three results are examined: the number of data packets received by their destinations, the control packet overhead, and the normalized routing load. The control packet overhead is computed by counting the number of hop-wise control packet transmissions. The normalized routing load is calculated by taking the total number of per-hop control packet transmissions, and dividing this by the number of data packets successfully delivered to their destinations.

Figure 6.16 illustrates the number of data packets delivered to destinations in each of the networks. The relative performance of WRP and FSR is fairly constant while that of AODV tends to vary with the MAC protocol used. When run over CSMA, WRP performs best for the higher mobility scenarios; however, while using IEEE 802.11, AODV outperforms the



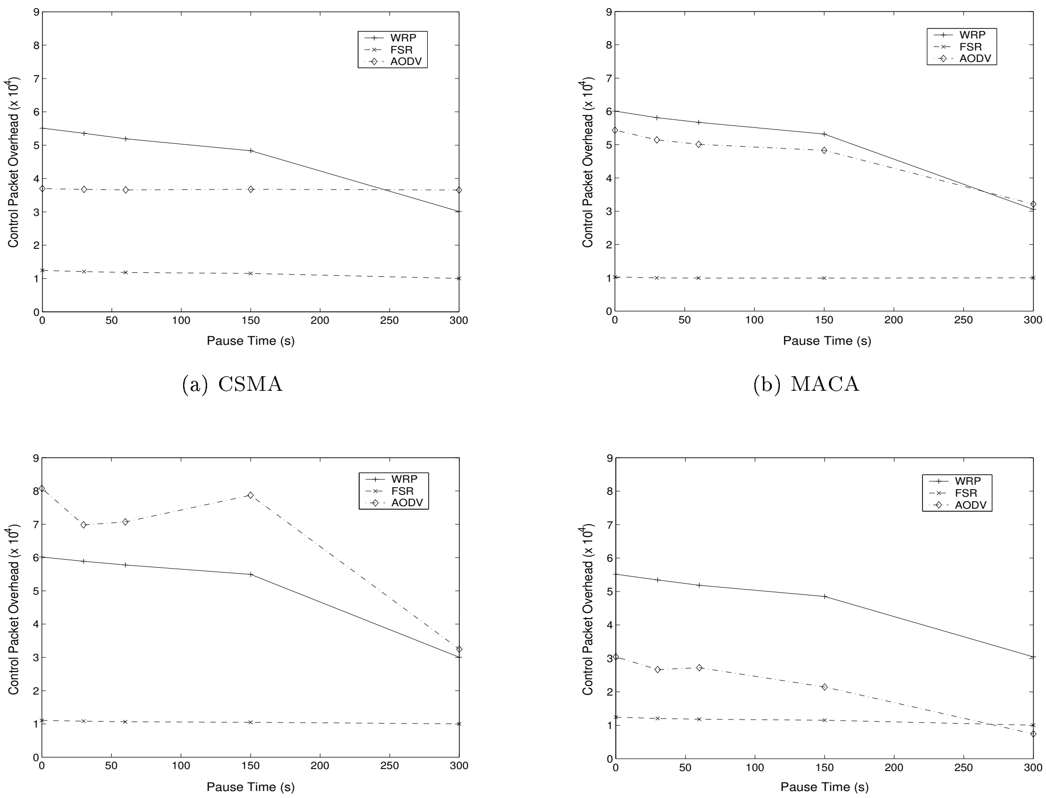
(a) CSMA (b) MACA

(c) FAMA (d) IEEE 802.11 DCF

Figure 6.16: Data Packets Delivered.

other protocols. The protocols achieve nearly the same number of delivered data packets when combined with the MACA and FAMA protocols, with AODV performing slightly better using the FAMA MAC protocol. The protocols have better overall performance using CSMA than using MACA or FAMA because of the RTS/CTS messages. MACA sources transmit RTS packets whenever they have a data packet to send without first sensing the channel. This results in an increase in packet collisions and hence decreased throughput. The collision avoidance mechanism incorporated into IEEE 802.11 for the transmission of RTS packets aids in the reduction of the number of collisions. Consequently, more data packets reach their destinations. Further analysis of the MAC protocols under UDP can be found in [22].

The number of hop-wise control packet transmissions during each simulation is shown in Figure 6.17. Because FSR uses periodic messaging regardless of the underlying MAC protocol, the amount of control overhead generated by this protocol remains constant over the different simulations. WRP has both triggered and periodic updates, and hence the amount of control overhead increases as mobility increases (i.e., as the pause time becomes



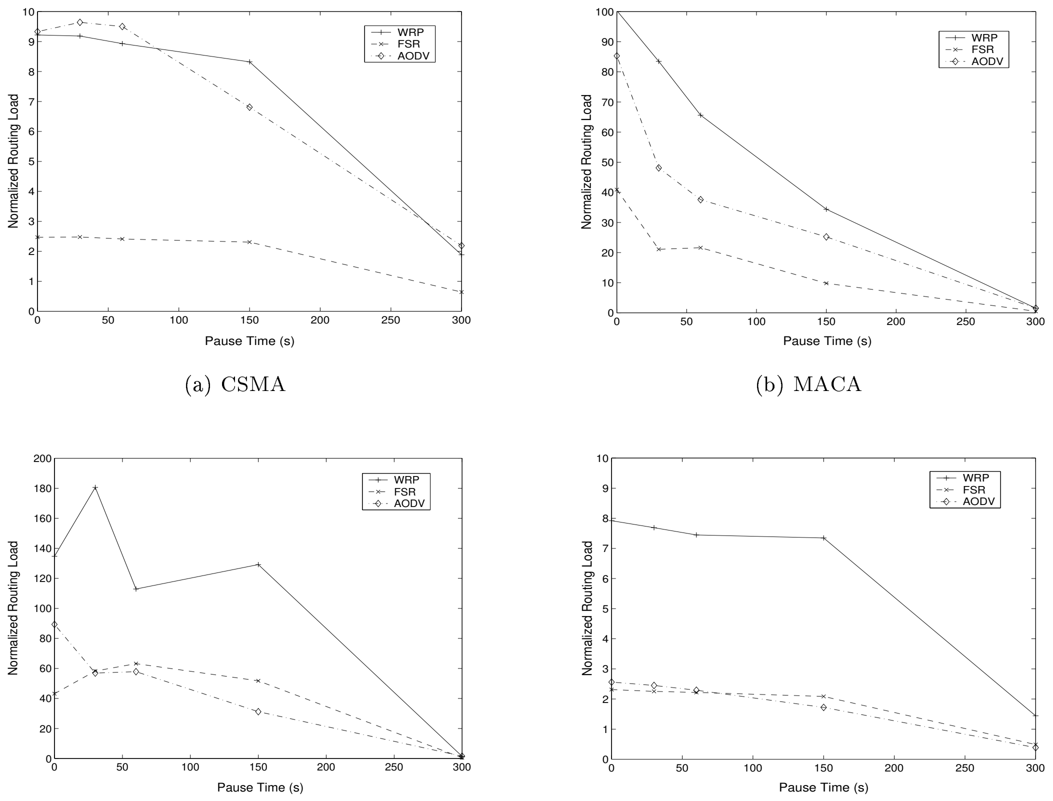
(a) CSMA (b) MACA

(c) FAMA (d) IEEE 802.11 DCF

Figure 6.17: Control Packet Overhead.

shorter). AODV is the only protocol significantly affected by the MAC layer. When run over CSMA, MACA or FAMA, AODV must utilize Hello messages to maintain connectivity. Hence, it is expected that the number of control messages in these simulations is greater than that in the IEEE 802.11 simulation. Additionally, the amount of control overhead generated by AODV is directly related to the number of routes it maintains. Because there are so many packet collisions when utilizing the CSMA MAC layer protocol, AODV is not able to maintain as many routes. Hence, the control overhead is lower for this simulation. As the number of routes AODV attempts to maintain increases, however, the amount of control traffic generated similarly increases.

The normalized routing load (NRL) is a measure of a protocol's efficiency. This measure is important because link layer protocols in ad hoc networks are contention-based. This result is shown in Figure 6.18. WRP consistently has a greater NRL than FSR, and has greater NRL than AODV in all but a few cases of CSMA. The ratio of control messages generated by WRP and FSR remains approximately constant regardless of the underlying MAC protocol. Note the variation in y-axis scaling. The NRL quantitative measure varies



(a) CSMA

(b) MACA

(c) FAMA

(d) IEEE 802.11 DCF

Figure 6.18: Normalized Routing Load.

because the throughput of WRP and FSR depends upon the MAC protocols used. Hence, this metric aids in the analysis of how efficiently the routing protocols utilize routing packets to deliver data packets. AODV is most efficient when used with IEEE 802.11. This result is expected because AODV does not need Hello packet transmissions when combined with IEEE 802.11.

6.4.4 Conclusion

This section has presented a performance comparison of the WRP, FSR, and AODV routing protocols when combined with varying MAC protocols. The relative performance of the WRP and FSR protocols does not show notable variation when run over the different MAC protocols. Neither routing protocol requires operational changes depend on the underlying MAC protocol, and the results show that their relative performance remains approximately constant. This leads to the conclusion that table-driven protocols act similarly with different MAC protocols, although further study of additional table-driven protocols is needed to completely validate this conclusion.

Because AODV requires periodic Hello messaging when run over link layer protocols that do not provide feedback when the next hop is unreachable, the amount of control traffic generated with these MAC protocols is considerably greater than when AODV is run over IEEE 802.11 DCF. AODV proves to be sensitive to the functionality of the MAC protocol, and hence its relative performance varies depending upon which MAC layer is used.

Table-driven and on-demand protocols may react differently depending upon the MAC protocol used; however, the question of whether two different on-demand ad hoc routing protocols would exhibit the same variation due to MAC layer effects remains open. The results show that the MAC protocol selected for simulation study is a key component of the performance of a routing protocol, and this aspect must be taken into consideration when comparing the performance of routing protocols.

6.5 Improving Scalability

Recent advances in the portability, power, and capabilities of wireless devices and applications have resulted in the proliferation and increased popularity of these devices. As the number of users continues to grow, wireless routing protocols will be required to scale to increasingly larger populations of nodes. Conference networking scenarios can require the formation of networks on the order of tens to hundreds of nodes, while many military applications can involve thousands to tens of thousands of nodes. Furthermore, as the deployment of wireless networks becomes more widespread, new applications will encourage the formation of large ad hoc networks. For instance, sensor networks can include thousands of sensors that must be able to self-configure and establish routes. Similarly, military battlefield operations often require the formation of ad hoc networks containing hundreds to thousands of soldiers and personnel.

There have been many recent proposals of unicast routing protocols for ad hoc mobile networks [4, 41, 65, 66, 72, 92, 97]. Many of these papers include simulations of the protocols they describe, illustrating the performance of their protocol. To determine the relative merits and strengths of the various protocols, simulations of the protocols under various input conditions [16, 25, 40, 55] have been conducted. While these simulations and studies are informative in evaluating the performance of the protocols for relatively small numbers of nodes (i.e., 50 nodes), they do not show how the protocols scale to larger node populations. The simulations presented in [72] and [66] evaluate the Ad hoc On-Demand Distance Vector (AODV) Routing protocol and the Zone Routing Protocol (ZRP), respectively, for networks as large as 1,000 nodes. These are simulations of some of the largest network sizes to date. Because ad hoc routing protocols could well be used in networks containing a large number of nodes, it is important to know how these protocols scale and perform in these scenarios.

Many of the proposed protocols for ad hoc networks [4, 41, 65, 72, 97] use a broadcast route discovery mechanism whereby a route request is flooded across the entire network. While the impact of such route discovery floods is limited in small networks, the impact is significantly greater for larger networks. When a link break in an active route occurs, many of these protocols [41, 65, 72] require that an error notification be sent to nodes that were using that link. Again, for small networks with limited network diameters, this route error message can be propagated back to a source node relatively quickly, and some repair action can be taken. However, as the network diameter and average path length increases, the error messages might have to propagate across tens of hops to reach the source node. For such large networks, or even smaller networks with rapidly moving nodes, it is likely that the source node will be unable to make a repair before another link in the route breaks. Hence, this mechanism may prove ineffective for more stressful scenarios.

There are other approaches to unicast routing in ad hoc mobile networks than those previously described which might prove more scalable; however, each of those methods also has its limitations. Clustering and hierarchical addressing methods have long been known for attempting to increase protocol scalability [11, 21, 39, 50, 83, 91]. Clustering protocols group nodes into *clusters* based on their proximity to each other. Each cluster usually has a cluster leader, which is the representative of the nodes in its cluster. The cluster leader typically participates in the network routing protocol, freeing the other network nodes of this burden. Routes in clustered networks can be recorded hierarchically between clusters, as in [21, 83]. These logical hierarchical paths are generally longer lived than routes that utilize flat addressing, because any gateway between two clusters can be used to route between the clusters. This may result in fewer route reconstructions, and hence also reduce the number of on-demand control messages required to maintain the routes. Cluster-based protocols, however, have their drawbacks. They require periodic messaging from each network node to maintain the clusters. This periodic messaging results in higher processing and control packet overhead, as well as increased bandwidth utilization and longer delays. Moreover, if the protocol constrains routes to traverse cluster leaders, longer path lengths may be required, which is another contributor to increased bandwidth utilization. Finally, complications can arise when the cluster leaders fail or give up their cluster leader status.

Instead of performing routing on-demand, other protocols have been based on modified versions of either the distance vector [59] or the link state [61] routing algorithms. Because both distance vector and link state algorithms not only use periodic updates, but also triggered updates in the event of a change in link status, they are not well-suited for most mobile networks. A network composed of moderately moving nodes results in a high number of triggered updates, consuming bandwidth and making route convergence difficult, if not impossible. The protocols described in [12, 42, 67, 95] present modified versions of those protocols. For instance, [42], [67] and [95] each utilize a prioritized connectivity information exchange algorithm, whereby information about parts of the network more distant from the sending node is sent less frequently than information about neighboring nodes. [42] and [95] apply this technique to the distance vector algorithm, while [67] modifies the link state protocol. A similar approach [12] sends update information to only those nodes that actually need the information. These protocols have the benefit of a reduction in routing update overhead as compared with the basic link state and distance vector algorithms. However, they still have the drawback that they require updates based on node movement, which can result in a large amount of control overhead and bandwidth consumption in a mobile network.

A different approach to route finding is taken by the Core Extraction Distributed Ad hoc Routing (CEDAR) algorithm [92]. CEDAR is an algorithm that builds a set of nodes (i.e., a *core*) to perform route computation. Using the local state information, a minimum dominating set of the network is approximated to form the core. CEDAR establishes QoS routes that satisfy bandwidth requirements using the directionality of the core path. Link state and bandwidth availability is exchanged to maintain important information for computing QoS routes. Although CEDAR builds a core infrastructure that yields low overhead, the protocol is fairly complex and difficult to implement. The problem of calculating the minimum dominating set and the core is known to be NP-hard [36].

Finally, [4] and [66] are variations of on-demand routing protocols which attempt to increase scalability through other methods. The Relative Distance Micro-discovery Ad hoc Routing (RDMAR) protocol allows for local repair of link breaks in active routes [4], and the ZRP protocol maintains route information to *all* nodes within a “zone” [66]. ZRP is a hybrid protocol that maintains the route information for the zone via a link state or distance vector protocol and then applies the on-demand technique communication for nodes outside the zone. These protocols can reduce the number of route discovery floods required by a source node by either repairing link breaks locally where they occur (RDMAR) or by maintaining routes to some destinations before they are actually needed (ZRP). Nevertheless, the protocols still suffer from the same disadvantage as the class of on-demand protocols whereby efficiency drops as the number of source-destination pairs increases, due to the likely requirement of a route discovery flood.

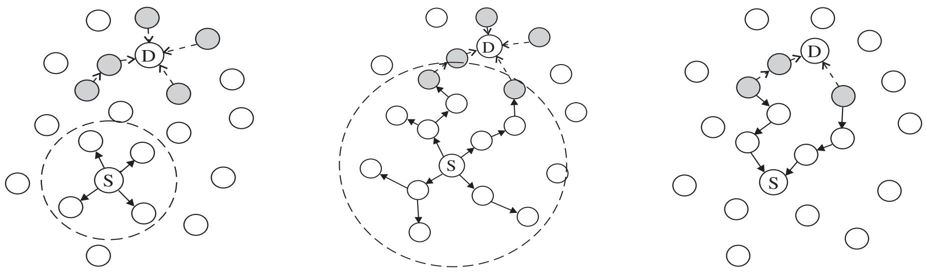
This section evaluates the scaling potential of on-demand ad hoc routing protocols by comparing a base routing protocol with the performance of that protocol combined with various enhancements. The Ad hoc On-Demand Distance Vector (AODV) Routing protocol [72, 74] is used as a representative of on-demand routing protocols. AODV was chosen because it is currently one of the leading protocols for routing in ad hoc mobile networks. The scalability of AODV is investigated by evaluating its performance in networks as large as 10,000 nodes. Three methods for improving the scalability of ad hoc routing protocols are described and integrated into the AODV protocol for their evaluation. The enhancements include an expanding ring search for route discoveries initiated by a source node, a query localization protocol (proposed in [19]) that also attempts to prevent the flooding of route requests, and the local repair of link breaks in active routes. The methods for preventing discovery floods are each in turn combined with the local repair mechanism, to yield a total of five possible improvement algorithms. Each of these enhancement combinations is evaluated, through simulation, in networks of up to 10,000 nodes, and compared with the results achieved by the unmodified AODV protocol.

6.5.1 Modifications

The scalability of many on-demand routing protocols is limited due to several important factors. The first is the need for flooding each RREQ. In small networks, flooding the RREQ across the network has a limited impact due to the small number of nodes in the network. However, as networks grow to thousands and tens of thousands of nodes, flooding the entire network each time a route needs to be discovered consumes significant processing power at each network node, as well as excessive bandwidth during the flooding.

As path lengths increase and node mobility speeds rise, breaks in active routes occur with increasing frequency. Requiring an error message to be sent to the source node for each link break can result in an overwhelming number of route repairs by the source node. Particularly for high mobility and/or long path lengths, the source node might barely have enough time to rediscover a route before that route suffers from another link break.

Because of these characteristics, on-demand routing protocols might not scale well to networks of large numbers of nodes and high mobility. To improve their scalability, the following modifications are offered. The expanding ring search and query localization can be used to reduce the area searched during a route discovery, and hence prevent flooding of the network. The current Internet draft specification of AODV [74] recommends that such an expanding ring search be used for route discoveries. Local repair can also be used to



(a) RREQ: TTL=1 (b) RREQ: TTL=3 (c) Path of RREPs to Source

Figure 6.19: Example of an Expanding Ring Search.

provide immediate patching of breaks in active routes. Finally, the expanding ring search and query localization can be combined with local repair to provide increased scalability in both of these domains.

Expanding Ring Search

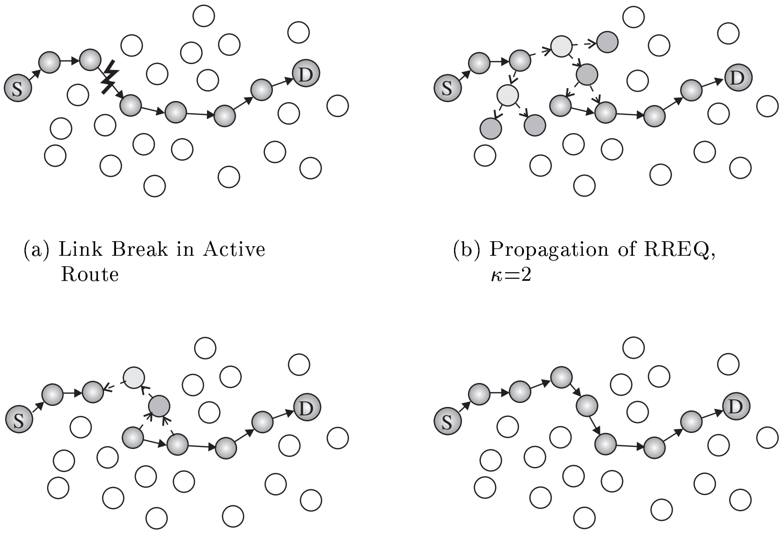
An expanding ring search works by searching successively larger areas, centered around the source node, until a node with a route to the destination is located. The basic premise behind the expanding ring search is to find some local node with a route to the destination, and thereby avoid flooding the entire network in search of such a route. Using an expanding ring search, the initial RREQ has a small time to live (TTL), i.e., two hops. Each time the RREQ is rebroadcast, the sending node decrements the TTL. Once the TTL reaches zero, the RREQ is no longer forwarded. The source node waits the discovery period for a RREP to be returned. If it does not receive a RREP by the end of the discovery time, it initiates a new RREQ with the TTL increased by an increment. This process continues until a threshold TTL value is reached. After this point, if no route has been located, the RREQ is flooded across the network. Figure 6.19 illustrates an example of an expanding ring search. In the figure, the shaded nodes indicate nodes that have a route to the destination. In a larger network with more nodes than that illustrated, the number of nodes undisturbed by the route query would likewise be greater.

To optimize the expanding ring search, the discovery time can be calculated so that it is proportional to the size of the area being searched. For instance,

$$\texttt{rte\_disc\_tmo} = 2 \times \text{TTL} \times \texttt{node\_traversal\_time}$$

results in the route discovery timeout being directly proportional to the TTL used for that discovery. Here the node traversal time is an approximation of the time required by the node to process and transmit a packet.

When re-discovering a route after a link break, the source places the last known hop count to the destination in the TTL field of the RREQ. If no route is found in this attempt, the TTL is increased by the TTL increment value. The TTL is increased on each subsequent route discovery attempt until the TTL threshold is reached. After this point, the RREQ is simply flooded across the entire network.



(a) Link Break in Active Route

(b) Propagation of RREQ, κ=2

(c) Path of RREPs

(d) Repaired Route

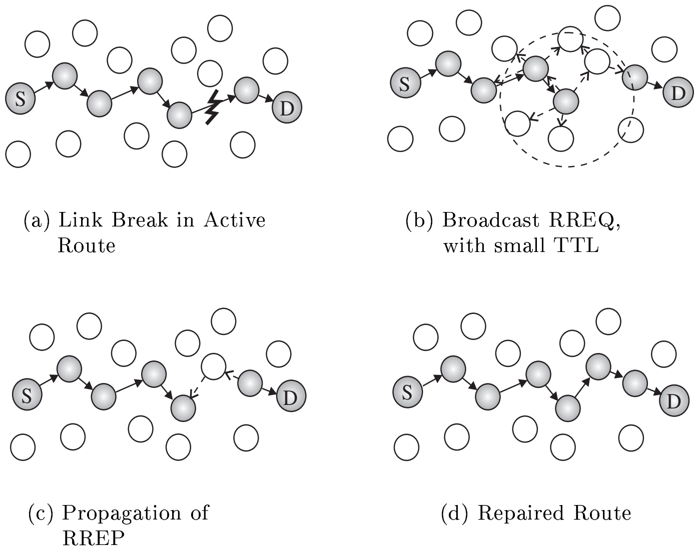
Figure 6.20: Example of Query Localization.

Utilizing the expanding ring search, a tradeoff exists between both the latency in finding the route (if it is not located on the first attempt) and the number of times local nodes receive the RREQ, and the drawback of flooding the entire network.

Query Localization

The query localization technique was developed by Castaneda and Das [19]. Query localization is a method by which the flooding of the RREQ is restricted to some area that is based on the previously known route to the destination node. Hence, the RREQ is not actually flooded at all, but instead is limited to a specific region of the network. Castaneda and Das present two different techniques for performing query localization. For the purposes of this study, method 2 (*exploiting node locality*) is selected. This method assumes that the destination has traveled a bounded distance from its previous location, and hence can be found within some small number of hops from the most recently used route to it. To enable query localization, a counter is placed in the RREQ packet. Whenever a node that was not on the previous route to the destination receives the RREQ, it increments the counter. Conversely, when a node that was previously on the route to the destination receives the RREQ, it resets the counter to zero. Once the counter exceeds the threshold value κ, the RREQ is dropped.

On the initial route discovery for a destination, κ is set to the network diameter, so that the RREQ traverses the entire network. For a route repair, however, κ is initialized to a small value, i.e., to two. If a route to the destination is not located on the first attempt, the value of κ may be increased until some maximum value is reached. Figure 6.20 illustrates an example of query localization. In the figure the last known route between the source and destination is highlighted. On the repair route discovery, κ is set to two. The shading of the nodes indicates their distance off of the previously known route to the destination. As



(a) Link Break in Active Route

(b) Broadcast RREQ, with small TTL

(c) Propagation of RREP

(d) Repaired Route

Figure 6.21: Example of Local Repair.

is evident from the figure, many of the network nodes do not need to receive the RREQ, and the query is able to be contained.

As with the expanding ring search, the drawback of the query localization protocol occurs when a route to the destination is not located on the first attempt. This results in certain nodes being repeatedly queried, as well as an increase in route acquisition latency.

Local Repair

Local repair of link breaks in active routes is another approach to increasing scalability. In the current AODV specification, when a link break in an active route occurs, the node *upstream* of the break creates a Route Error (RERR) message listing all of the destinations that have become unreachable due to the break. The node then sends this message to its upstream neighbors, as described in Section 2.2.5. If, instead of sending an error message to the source node, the upstream node attempts to repair the broken link itself, fewer data packets are lost and the link can be repaired without the source node (and other upstream nodes) being disturbed. For short routes, local repair might not have any significant performance advantages. But for large networks with increasingly long routes (i.e., 10+ hops), it is likely that link breaks will occur so frequently that it will be nearly impossible for the source node to keep up with all the necessary repairs.

A node upstream of a link break that attempts to repair the route does so by broadcasting a RREQ with a TTL set to the last known distance to the destination, plus an increment value. This TTL value is used so that only the most recent whereabouts of the destination will be searched, which prevents flooding the entire network. The upstream node places the sequence number of the destination, incremented by one, into the RREQ. This prevents nodes further upstream on the route from replying to the RREQ, which would form a loop. Figure 6.21 illustrates an example of a local repair.

# of Nodes	Room Size (m^2)	Avg. # of Neighbors
50	1000 * 1000	7.32
100	1500 * 1500	7.46
500	3500 * 3500	7.33
1000	5000 * 5000	7.69
5000	11500 * 11500	7.22
10000	16000 * 16000	7.50

Table 6.3: Summary of Room Sizes.

If a route to the destination is not located on the first attempt, a RERR message is sent back to the source node, and route re-discovery continues as described in Section 2.2.5.

Combining the Enhancements

The above enhancements can be combined in various ways for increased protocol scalability. Specifically, the expanding ring search and local repair can work together, as can query localization and local repair. The expanding ring search and query localization are used to optimize route discoveries initiated by a source node, while local repair is used to decrease the number of route discoveries which a source node must initiate. Local repair, when combined with those two enhancements, operates in the same fashion as described previously. One attempt at the repair is made locally. If this attempt is unsuccessful, a RERR message is sent back to the source.

6.5.2 Simulations

Environment

The simulations used to evaluate the scalability of AODV and its enhancements were implemented within the GloMoSim library [8]. The GloMoSim library is a scalable simulation environment for wireless network systems using the parallel discrete-event simulation capability provided by PARSEC [7]. The simulations model networks between 50 and 10,000 mobile hosts placed randomly within the simulation area. The simulation boundary and average connectivity for each simulated number of nodes is shown in Table 6.3. The room size for each simulation was chosen so as to keep the node density approximately constant in the different size networks.

The radio propagation range for each node is 250 meters and channel capacity is 2 Mbits/sec. Each simulation is executed for 300 seconds of simulation time. Multiple runs with different seed numbers were conducted for each scenario and collected data was averaged over those runs.

Channel and Radio Model

A free space propagation model [79] with a threshold cutoff was used in the experiments. In the free space model, the power of a signal attenuates as $1/d^2$, where d is the distance between radios. In the radio model, radio capture is assumed, whereby a radio has the ability to lock onto a sufficiently strong signal in the presence of interfering signals. If the capture ratio (the minimum ratio of an arriving packet's signal strength relative to those

	Parameter	Value
General	`net_diameter`	35, 70
	`node_traversal_time`	40 ms
Expanding Ring Search	`ttl_start`	1
	`ttl_increment`	2
	`ttl_threshold`	7
Query Localization	κ	2
	κ_2	$\kappa \times 2$
	τ	10 sec
Local Repair	`local_add_ttl`	2

Table 6.4: Parameter Values

of other colliding packets) [79] is greater than the predefined threshold value, the arriving packet is received while other interfering packets are dropped.

Medium Access Control Protocol

The IEEE 802.11 MAC protocol with Distributed Coordination Function (DCF) [27] is used as the MAC layer in the experiments. DCF is the basic access method used by mobiles to share the wireless channel under independent ad hoc configuration. The access scheme is Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) with acknowledgments. Optionally, the nodes can make use of Request-To-Send/Clear-To-Send (RTS/CTS) channel reservation control frames for unicast, virtual carrier sense, and fragmentation of packets larger than a given threshold. By setting timers based on the reservations in RTS/CTS packets, the virtual carrier sense augments the physical carrier sense in determining when mobile nodes perceive that the medium is busy. Fragmentation is useful in the presence of high bit error and loss rates, as it reduces the size of the data units that need to be retransmitted.

In the experiments presented here, RTS/CTS and virtual carrier sense is employed. This configuration was chosen to minimize the frequency and deleterious effects of collisions over the wireless medium. Fragmentation was not employed because the data packets are small enough that the additional overhead would reduce overall network throughput.

Traffic Pattern

The size of data payload is 512 bytes. Twenty data sessions with randomly selected sources and destinations are simulated. Each source transmits data packets at a rate of four packets/sec.

Mobility Pattern

The random waypoint model [41] is utilized in the experiments. In this model, a node selects a random destination within the terrain range and moves towards that destination at a speed between the pre-defined minimum and maximum speed. Once the node arrives at the destination, it stays at its current position for a pause time. After being stationary for the pause time, it randomly selects another destination and speed and then resumes

Protocol Combination	Abbreviation
AODV	AODV
AODV and Expanding Ring Search	AODV-ERS
AODV and Query Localization	AODV-QL
AODV and Local Recovery	AODV-LR
AODV, Expanding Ring Search and Local Repair	AODV-ERS-LR
AODV, Query Localization and Local Repair	AODV-QL-LR

Table 6.5: Protocol Abbreviations

movement. The minimum speed for the simulations is 0 m/s, while the maximum speed is 10 m/s. The selected pause time is 30 seconds.

Parameter Values

Table 6.4 gives a summary of the chosen parameter values. The network diameter (`net_diameter`) for the simulations represents the approximate diameter of the network, and is used for setting the TTL value of broadcast control packets. It is also a factor in the calculation of how long a node should wait to receive a RREP after sending another RREQ. If the RREQ is broadcast across the network, the reception of the RREP will take longer for large networks than for small. The setting of the `net_diameter` variable to 35 for small networks (50, 100, 500, and 1,000 nodes) and 70 for the larger networks (5,000 and 10,000 nodes) provides an upper bound of the actual network diameter for these networks.

The `node_traversal_time` represents an estimation of the processing time of a packet at a given node. It is also used for estimating the period of time a source node should wait to receive a RREP after broadcasting a RREQ.

The values selected for the modification parameters represent a tradeoff between minimizing the number of searches required to locate a given destination, and reducing the number of nodes that must receive and process the RREQ packet. For the expanding ring search, the initial TTL `ttl_start` of the RREQ is set to one. Each time a reply is not received, the TTL is incremented by `ttl_increment`, until the threshold value (`ttl_threshold`) is reached. After that point, the RREQ is broadcast across the network. When rediscovering routes, the initial TTL is the last known hop count to the destination by the source.

The value of κ for query localization represents the number of hops the RREQ is allowed to travel off the previously known path to the destination. The initial value of κ is set to two. If no reply is received, the value of κ is doubled for the second attempt. The value of τ is ten seconds. If a node has been part of the most recent route to the destination within the past τ time units and receives the RREQ, it resets the value of κ to zero.

Finally, the `local_add_ttl` parameter is used for a local repair. It represents the value added to the previously known distance to the destination. This sum is used as the TTL of the RREQ for the local repair.

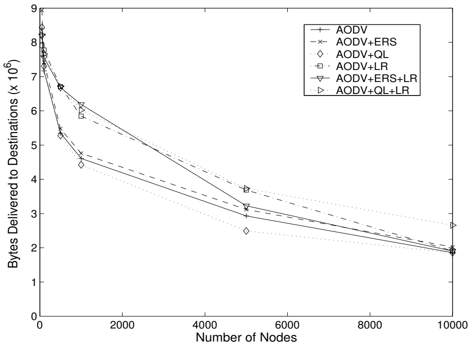


Figure 6.22: Throughput.

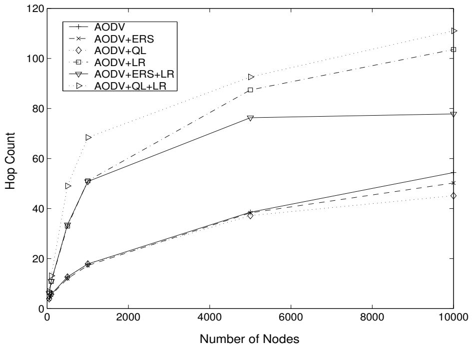


Figure 6.23: Path Length.

Results

The following sections present the results achieved by the different protocol combinations. Table 6.5 indicates the abbreviation associated with each modified protocol in the following figures.

Figure 6.22 shows the number of bytes of data each scheme is able to deliver to the destinations. The figure shows that the ability of the protocols to deliver packets to their destination degrades as the network size becomes larger. The path length is greater in larger networks because the simulation terrain range and the number of nodes increases while the average number of neighbors is kept relatively constant (see Table 6.3). Routes are more prone to disconnections in mobile networks when path lengths are longer. Because a single link failure results in the inability of the source to reach the destination, longer routes have a greater probability of route disconnection than shorter routes. An increased route length in larger multihop networks is a characteristic not only of AODV (or on-demand routing protocols), but also of other routing protocols such as table driven (i.e., distance vector and link state) algorithms and hierarchical clustered routing protocols. It is observed that performing route recovery locally improves throughput. Because nodes closer to the destination than the source initiate route rediscovery, new routes are repaired more quickly and fewer data packets are dropped. It is interesting to note that AODV-QL has the poorest data delivery result. The main purpose of query localization is to exploit node locality and to reduce the number of routing message transmissions. Localizing the query, however, has the risk of not being able to establish the route.

The path length of each scheme is presented in Figure 6.23. The route length is measured by calculating the distance between the source and destination when the route is constructed. The measure includes the first discovered route for both the construction of new routes, and the repair of broken routes. It is observed that schemes that utilize the local recovery technique yield longer paths. For protocols that do not use local recovery, only the source node can reconstruct routes. When a source rediscovers routes with a request/reply cycle, a new route is obtained based on current network information such as hop count, route freshness, node location, network topology, etc. On the other hand, in local recovery schemes, the node immediately upstream of the disconnected link initiates a route reconstruction. Because of the possibility that the destination has actually moved closer to the source node, but the distance between the node reconstructing the route and destination has increased, path lengths tend to grow as intermediate nodes repair routes.

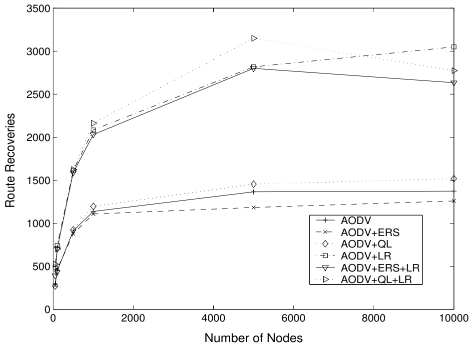


Figure 6.24: Number of Route Recovery Attempts.

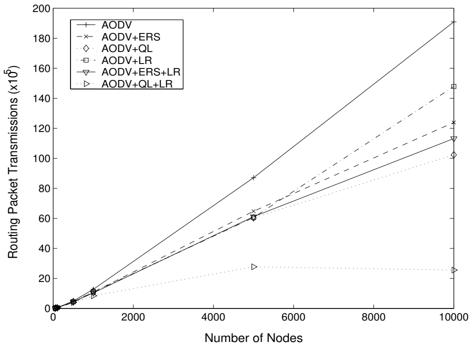


Figure 6.25: Routing Message Overhead.

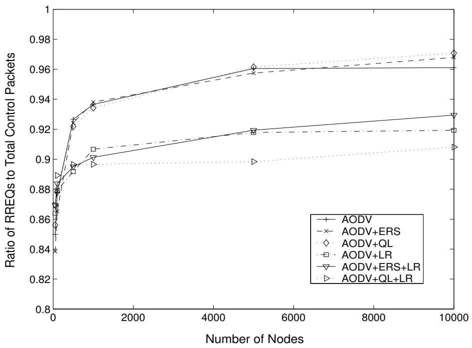


Figure 6.26: Percentage of RREQ Transmissions Among Control Packet Transmissions.

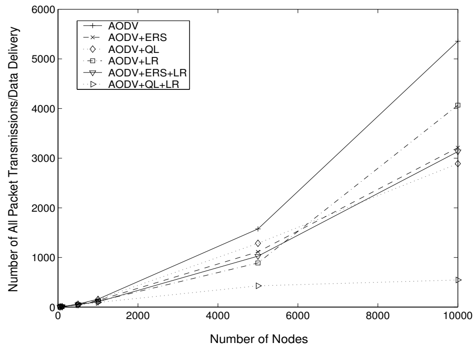


Figure 6.27: Number of All Packet Transmissions per Data Delivery.

Longer path lengths naturally result in more route breaks and more route recoveries, as shown in Figure 6.24. Local recovery schemes have more route reconstruction attempts for the following two reasons. First, longer routes can fail more easily than shorter routes. Second, no RERR message is sent upstream to the source in local recovery. If a link upstream of the previously broken link becomes disconnected while a new route is being discovered, another local recovery procedure is initiated. This results in more route reconstruction attempts by local recovery schemes. It is interesting to note that the number of recovery attempts does not significantly increase between 5,000 and 10,000 nodes. This is due to the difficultly of all of the protocol variations in maintaining routes with such a large path length. In these scenarios, many sessions are forced to abort due to the inability to maintain a route. Hence, with fewer sessions being maintained, fewer route discoveries are necessary.

In Figure 6.25, the routing message overhead is presented as the number of control message transmissions. Each hop-wise transmission of a control message by a node is counted as one transmission. As expected, AODV without enhancements has the most control packet transmissions. Local recovery schemes have less control overhead compared with schemes that perform route recovery by sources. AODV-QL-LR has the least control overhead among all protocols. Local recovery schemes reduce the number of RREQ transmissions. As shown in Figure 6.26, the percentage of RREQ transmissions among all routing packet transmis-

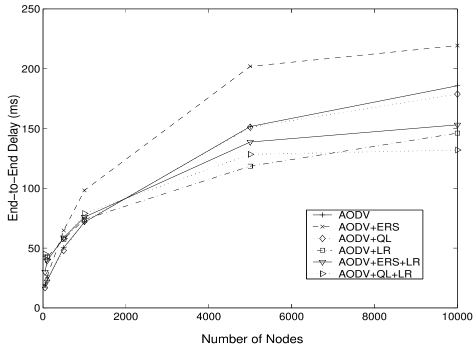


Figure 6.28: End-to-End Delay.

sions (RREQ, RREP, and RERR) by local recovery schemes is lower than source recovery schemes.

To evaluate protocol efficiency, the number of control packet (i.e., data, RREQ, RREP, and RERR) transmissions per data delivery is investigated. Because link layer protocols for ad hoc networks are contention-based, this measure is very important for protocol analysis. The measure is presented in Figure 6.27. The scope and the ranking of protocols are similar to those of Figure 6.25. Note the huge number of packet transmissions per successful delivery at high node populations shown in Figure 6.27. This ratio, which can grow as large as 5,000, indicates the drastic need for work in a crucial area affecting the scalability of AODV, and probably all on-demand ad hoc routing protocols, to large network populations. The ratio should be brought down by three orders of magnitude; such a reduction would probably also be accompanied by a proportional increase in the packet delivery fraction, which is sometimes as low as 15%. Work towards developing techniques for quickly re-establishing valid routes is likely to be of the highest importance for improving the scalability of ad hoc networks.

The end-to-end delay of each protocol is reported in Figure 6.28. Schemes that utilize the local recovery technique have shorter delays. Protocols in which sources initiate route recovery have longer end-to-end delays because of longer route re-establishment latency. To recover a broken route, a RERR packet must first be delivered from the node upstream of the broken link to the source of the route. The RREQ must then be broadcast from the source to the destination, and a RREP consequently has to be transmitted back to the source. Data packets are buffered at the source node during this process and the duration of their buffering adds to the end-to-end delay. In local recovery schemes, on the other hand, the node upstream of the disconnected link initiates an immediate route reconstruction. Because route rediscovery is done locally, less time is needed to search for and obtain a new route. Local recovery schemes can, therefore, yield shorter delays. Note that AODV-ERS has the longest delay because a route might not be built in the initial attempt (i.e., `ttl` = 1 or last known hop count of the route). Among local recovery schemes, AODV-ERS-LR has the longest delay for the same reason.

6.5.3 Observations

In the previous sections, we have studied the scalability characteristics of on-demand routing protocols. We learned that routing in ad hoc networks of tens of thousands of nodes is extremely difficult. In large networks, path lengths are longer compared with those in small networks (i.e., 50 or 100 nodes). Because network hosts are capable of mobility, longer routes are more prone to disconnection since a single link failure results in a route break. Each route invalidation invokes a route recovery process and clogs the network with control messages. Because there are generally multiple hops between a source and destination, and because nodes are mobile, many route discoveries are unsuccessful. Although the flooded RREQ packet reaches the destination or intermediate nodes with routing information to the destination, the unicast RREP packet might not reach the source due to link breaks during route discovery. Even when the RREP packet survives to reach the source, the route can break shortly after and the source will need to initiate another route discovery. Therefore, maintaining routes with many hops in mobile ad hoc networks is a difficult challenge.

This study introduces three techniques and applies five different modification combinations to improve AODV scalability. The expanding ring search reduces the routing message overhead, but yields longer delays because of initial route discovery failures. Query localization also decreases control overhead but has poor throughput performance due to low route recovery rate. This is especially true when routes have many hops. Local repair proves to be effective in enhancing AODV's performance in large networks. Because route recovery is localized, new routes are found more quickly than source initiated route discoveries. Consequently, packet drops are minimized. Local repair works efficiently with expanding ring search and query localization to reduce control message overhead. The drawback of local repair is that multiple repairs for the same route can be present at the same time.

Local repair can benefit from some mechanism to reduce the growth in path lengths that result from this method. One possible solution is to combine local repair with a RERR unicast back to the source. If a link breaks in an active route, the node upstream of that break could repair the route using local repair, and then send a RERR message back to the source. In this way, as the upstream node continues to receive data packets while the RERR is traveling to the source node, the data packets can still be forwarded to the destination. When the source receives the RERR, it can decide whether to reinitiate route discovery to look for a better route in order to reduce the length of the route if it has increased significantly. This method will result in fewer dropped packets than not using local repair, while also reducing the increase in path lengths that result from local repair.

6.5.4 Conclusion

This section has evaluated the scalability of on-demand ad hoc routing protocols by selecting a representative from this set of protocols and simulating it in networks of up to 10,000 nodes. To improve the performance of on-demand protocols in large networks, five combinations of modifications have been separately incorporated into an on-demand protocol, and their respective performance has been studied. It has been shown that the use of local repair is beneficial in increasing the number of data packets that reach their destinations. The expanding ring search and query localization techniques further reduce the amount of control overhead generated by the protocol by limiting the number of nodes affected by route discoveries.

The results of this work are not specific to the AODV protocol. The expanding ring search, query localization and local repair modifications are protocol independent and can be incorporated into virtually any on-demand protocol to improve that protocol's scaling potential. Scalability in ad hoc mobile networks is inherently difficult due to the mobility of the nodes and the transience of network links. Work on large-scale ad hoc networks is likely to uncover techniques that would be valuable for stabilizing routing protocols in the Internet at large, leading to faster route convergence and reduced route flaps. Creating ad hoc routing protocols that experience minimal performance degradation when used in increasingly large networks is a challenge, and a significant amount of work remains to reach this goal.

6.6 Conclusion

Fully understanding the characteristics of ad hoc networks and being able to predict their behavior is extremely challenging, due to their dynamic nature. Simulation is an invaluable tool for studying these characteristics and for attaining a better general understanding of these networks. Various aspects of ad hoc mobile networks have been examined in this section to make further steps towards this goal. The effects of transmission range have been investigated, both in terms of determining the optimum node density and in determining its effects on AODV multicast communication. Both studies show that there exists a tradeoff between throughput and efficient utilization of bandwidth and battery power. The study of the interaction between ad hoc routing protocols and MAC layer protocols illustrates that the choice of MAC layer protocol is not inconsequential in outcome of the simulations. These three studies show the importance of the conditions external to the routing protocol itself in determining the overall performance of the system.

Finally, while it is important to consider external conditions when evaluating a protocol's performance, it is also necessary to optimize the protocol so that it can perform at its best. The scalability of AODV was examined to determine how its performance degrades in large networks. Modifications to the basic protocol were proposed, and the performance improvements from these modifications were examined.

An ad hoc mobile network is a complex system with numerous interacting factors. While it is essential to optimize a routing protocol to achieve the best possible performance, there are many factors that affect the end performance of the system. The routing protocol, the MAC layer protocol, the transmission power, and numerous other factors must be coordinated so as to achieve the best overall network performance.

Chapter 7

Conclusion

An ad hoc mobile network is a collection of nodes, each of which is capable of and is likely to be moving, resulting in continual changes in the topology of the network. These nodes communicate through wireless transmission, and each of them serves as a router for the other network nodes. Ad hoc networks have many unique characteristics that make network communication challenging. The mobility of the nodes introduces the problem of discovering and maintaining paths over a dynamic network topology. Additionally, due to their mobile nature they often run on battery and are hence power constrained. Furthermore, because of the wireless transmission, the network has limited bandwidth, and there are often high error rates. Because of these limitations, protocols designed for providing communication in wired networks are often not suitable for wireless networks. Ad hoc routing protocols must be designed with these limitations in mind, and hence must aim to minimize processing and transmission overhead, in addition to being able to find and maintain routes over a dynamic topology.

One strategy for providing connectivity in ad hoc networks is known as the pro-active, or table-driven, approach. Protocols of this nature are typically extensions of the basic link-state or distance-vector protocols designed for communication in wired networks. These protocols generally require the periodic exchange of routing tables. In addition, some utilize triggered updates, whereby changes in routing information are exchanged whenever they occur. These protocols are well-suited for relatively small networks (on the order of tens of nodes) and/or networks with large numbers of data sessions (i.e., multiple sessions per node). In periods of very high mobility, however, these protocols often do not perform well. This is especially true for protocols that use triggered updates. These protocols also do not scale to large networks.

To this end, the Ad hoc On-Demand Distance Vector Routing (AODV) protocol has been designed. Instead of being pro-active, AODV provides routing in an on-demand, or reactive, manner. Routes are established only when they are needed by a source node. As long as the source node requires the route, AODV maintains it by repairing any link breaks that occur. Once the route is no longer needed, it may be expired from the system. Hence, only links that are part of active routes must be maintained. Links that are not used for the transmission of data packets do not require any protocol action if they break.

In addition to providing unicast communication capability, AODV also provides multicast communication through the formation of multicast trees. Multicast trees connect multicast group members through the use of tree router nodes. Each group has associated with it a group leader, whose primary responsibility is the maintenance and dissemination of the multicast group sequence number. The multicast tree is maintained throughout the lifetime of the multicast group.

Both the unicast and multicast routes that AODV discovers are loop-free through the use of sequence numbers. Each node in the network maintains its own monotonically increasing sequence number. Similarly, each multicast group leader maintains a sequence number for the multicast group. When given a choice between two routes to some destination, a node selects the route with the greatest sequence number. The sequence number indicates the freshness of the route.

The operation of the AODV protocol has been validated through extensive simulations. Simulations of both AODV's unicast and multicast communication capability have been completed in a variety of network sizes and mobility rates. AODV efficiently establishes routes with a minimum of control overhead. When link breaks in active routes occur, AODV quickly notices the disconnection and repairs the link.

AODV provides some rudimentary support for QoS. One area for future work is in extending the QoS functionality that AODV is able to offer. Currently, AODV is able to determine routes with maximum delay and minimum bandwidth constraints. In the future, these should be extended to include other QoS measures. In addition, the protocol operation when a link fails to meet its previously determined QoS requirements should be better defined.

One additional area for future research is in defining the operation of AODV with wired or MobileIP enabled networks. If one or more nodes are within range of a wireless access point, either a network router or a foreign agent, the cloud of ad hoc nodes should be able to take advantage of this connection to reach the Internet. This requires both the nodes being able to discover the existence of the access point, and the determination of an addressing scheme whereby the nodes are reachable by the rest of the Internet.

The class of on-demand protocols encounters scaling difficulties when run on networks of thousands or tens of thousands of nodes. The basic route discovery flood is not scalable, nor is the current route repair method utilized by AODV. One method for increasingly the scalability of both AODV and on-demand protocols in general is to localize certain operations so that the entire network is not effected. The modification of the route discovery flood was investigated so that the flooding is restricted to the area where the destination was last seen, or that progressively larger areas of the network are searched, with the end goal of not flooding the entire network. Additionally, a local repair mechanism was introduced so that the source node is not responsible for repairing all link breaks along its route to a destination. It was shown that each of these methods serves to improve the performance of AODV in large networks. The results of this study can be applied to virtually any on-demand routing protocol.

An alternative to modifying the on-demand protocol is to create a hierarchy for routing within the network. The hierarchy should be dynamic in that it can adjust the depth of the hierarchy to accommodate the number of nodes or the topology of the network. In such a structure, routes can be recorded logically, across the hierarchy, as opposed to physically, on a per-link basis. The advantage of hierarchical protocols is that breaks in the logical links occur less frequently than do those in the physical links. Hence, routes tend to be longer lived, and fewer route reconstructions are necessary. Two such hierarchical routing protocols were presented. The first of these, Adaptive Routing using Clusters (ARC), creates a one-level hierarchy of clusters. Each cluster has a cluster leader, and routes are recorded between these leaders. Gateway nodes are used to route between clusters. A new method for merging clusters was designed, such that the *rippling effect* resulting from cluster leader changes is avoided. This results in a more stable cluster topology, which is beneficial due to the high cost of cluster leader changes. Additionally, a new *limited broadcast* mechanism was developed. Limited broadcast reduces the number of nodes retransmitting broadcast packets, and hence utilizes bandwidth more efficiently. The benefits of clustering in terms of increasing network throughput and reducing the number of redundant broadcasts were illustrated.

In addition to ARC, a hierarchical routing protocol, the Adaptive Routing using Cluster Hierarchies (ARCH) protocol, was designed. Instead of being restricted to a one-level hierarchy, ARCH has the ability of dynamically adjusting the depth of the routing hierarchy in response to the changing network conditions. In a network of a thousand nodes, a one-level hierarchy can significantly improve the routing over a flat-addressing scheme. However, when scaling to networks of thousands or tens of thousands of nodes, the scaling improvements of a one-level scheme are limited. A protocol that is able to adjust dynamically witnesses more performance improvements.

One area for improvement in the clustering protocols is in the decentralization of routes from the cluster leaders. In the current proposed schemes, routes are established between cluster leaders. While this increases the robustness of the routes, it places an unfair burden on the cluster leaders in that the traffic patterns become centralized through them. Additionally, in the event that a route between two cluster leaders becomes unattainable, the number of routes that must be rebuilt is often much greater than it would be otherwise. A method for migrating the data routes off of the cluster leaders would be extremely beneficial in terms of conserving the batteries of those leaders and further increasing route robustness.

In addition, clustering requires the periodic beaconing of network nodes to maintain the cluster topology. The more frequent the beaconing, the more accurate the cluster topologies. This requirement causes more rapid drain of nodal batteries. In addition, it creates more traffic in the network, and hence reduces the potential overall network throughput. The development of a beaconing algorithm that could adapt to the speed of the network nodes would be beneficial. In periods of relatively slow node movement, the period between beacons could be rather large. In contrast, when nodes are moving more rapidly, the beaconing interval should be shorter, so that more accurate topology information is obtained.

The interaction between a routing protocol and other network parameters plays an important role in determining the performance of the protocol. It was shown that the node density and average nodal transmission range are key factors affecting network throughput and control overhead generated by the routing protocol. In addition, the interaction between the MAC layer protocol and the network layer should not be ignored. A routing protocol that produces a small amount of overhead at the network layer might require many MAC layer overhead packets due to the nature of the network layer control packet transmissions.

There are numerous research problems remaining in the area of mobile networking. One of these is security. Wireless transmission is inherently insecure. Packets are received by anyone within transmission range, and, if they are not encrypted, they can be read by anyone. A method for the encryption of packet transmissions is needed so that users can have secure remote access to their files and transmit data across the channels. Security is also needed for the authentication of users of access points. It is not always the case that companies or administrators desire anyone to be able to communicate with their wireless access points. A mechanism for defining who is allowed to use the system and also for authenticating each is user is needed. Additionally, ad hoc routing protocols are particularly susceptible to denial of service attacks and the introduction of false routing information by malicious nodes. The prevention of each of these occurrences is crucial to preserving the integrity of the network.

Additionally, power management is a major issue for mobile computing. Mobile users generally operate their devices on batteries. Battery power improvements have traditionally not followed Moore's law. Processors are increasingly requiring more power, and wireless transmission similarly drains node batteries. Routing protocols that take into consideration node battery life are going to be in high demand. To this end, AODV can be modified to take into consideration node battery lifetimes when selecting a route. Route selection might be as simple as a route with the maximum of the minimum of all battery lifetimes. Or, it might be more complicated by considering which nodes are already participating in other data paths. There is also the question as to whether users should be allowed to elect not to route other user's traffic. A user with limited battery lifetime will not want to spend his/her remaining battery power forwarding traffic for other users. However, if numerous users exercise the right of not forwarding data packets, the connectivity of the network will break down and the network can become disconnected.

Ad hoc mobile networking is currently one of the most rapidly growing research areas. This dissertation has made the contribution of routing protocols for providing connectivity within ad hoc networks of mobile nodes. While the foundation for providing this connectivity has been laid, more research is required to define a complete solution for ad hoc mobile networking.

Bibliography

[1] Internet Engineering Task Force (IETF). http://www.ietf.org.

[2] Internet Engineering Task Force (IETF) IP Routing for Wireless/Mobile Hosts (MobileIP) Working Group Charter. http://www.ietf.org/html.charters/mobileip-charter.html.

[3] OPNET Modeler. http://www.mil3.com/products/modeler/home.html.

[4] G. Aggelou and R. Tafazolli. RDMAR: A Bandwidth-efficient Routing Protocol for Mobile Ad Hoc Networks. In *Proceedings of the Second Annual ACM International Workshop on Wireless Mobile Multimedia (WoWMoM)*, pages 26–33, Seattle, WA, August 1999.

[5] G. Aggelou and R. Tafazolli. Relative Distance Micro-discovery Ad Hoc Routing (RDMAR) Protocol. *IETF Internet Draft, draft-ietf-manet-rdmar-00.txt*, September 1999. (Work in Progress).

[6] G.-S. Ahn, A. T. Campbell, S.-B. Lee, and X. Zhang. INSIGNIA. *IETF Internet Draft, draft-ietf-manet-insignia-01.txt*, October 1999. (Work in Progress).

[7] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, and H. Song. PARSEC: A Parallel Simulation Environment for Complex Systems. *IEEE Computer*, 31(10):77–85, October 1998.

[8] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GlomoSim: A Scalable Network Simulation Environment. Technical Report CSD Technical Report, #990027, UCLA, 1997.

[9] D. J. Baker, A. Ephremides, and J. A. Flynn. The Design and Simulation of a Mobile Radio Network with Distributed Control. *IEEE Journal on Selected Areas in Communications*, 2(1):226–237, January 1984.

[10] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT) - An Architecture for Scalable Inter-Domain Multicast Routing. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 85–95, August 1993.

[11] S. Basagni. Distributed Clustering for Ad Hoc Networks. In *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 310–315, Australia, June 1999.

[12] D. Beyer, M. Frankel, J. Hight, D. Lee, M. Lewis, R. McKenney, J. Naar, R. Ogier, N. Shacham, and W. Zaumen. Packet Radio Network Research, Development and Application. In *Proceedings of the SHAPE Packet Radio Symposium*, 1989.

[13] K. Bhargavan, C. A. Gunter, and D. Obradovic. Fault Origin Adjudication. In *Proceedings of the Workshop on Formal Methods in Software Practice*, Portland, OR, August 2000.

[14] E. Bommaiah, A. McAuley, R. Talpade, and M.-K. Liu. AMRoute: Adhoc Multicast Routing Protocol. *IETF Internet Draft, draft-talpade-manet-amroute-00.txt*, August 1998. (Work in Progress).

[15] J. Broch, D. B. Johnson, and D. A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. *IETF Internet Draft, draft-ietf-manet-dsr-03.txt*, October 1999. (Work in Progress).

[16] J. Broch, D. A. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the 4^{th} Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, Dallas, Texas, October 1998.

[17] R. Caceres and V. N. Padmanabhan. Fast and Scalable Handoffs for Wireless Internetworks. In *Proceedings of the 2^{nd} ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 56–66, November 1996.

[18] R. Caceres and V. N. Padmanabhan. Fast and Scalable Wireless Handoffs in Support of Mobile Internet Audio. *ACM/Baltzer Mobile Networks and Applications*, 3(4):351–364, December 1998.

[19] R. Castaneda and S. R. Das. Query Localization Techniques for On-demand Routing Protocols in Ad Hoc Networks. In *Proceedings of the 5^{th} Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 186–194, Seattle, WA, August 1999.

[20] C.-C. Chiang, M. Gerla, and L. Zhang. Adaptive Shared Tree Multicast in Mobile Wireless Networks. In *Proceedings of IEEE GLOBECOM*, pages 1817–1822, Sydney, Australia, December 1998.

[21] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. In *Proceedings of IEEE Singapore International Conference on Networks (SICON)*, pages 197–211, April 1997.

[22] M. Correa, K. Tang, and M. Gerla. Isolation of Wireless Ad hoc Medium Access Mechanisms Under UDP. Technical Report 990035, University of California, Los Angeles, Computer Science Department, June 1999.

[23] M. S. Corson and A. Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *ACM/Baltzer Wireless Networks*, 1(1):61–81, February 1995.

[24] S. R. Das, R. Castaneda, and J. Yan. Comparative Performance Evaluation of Routing Protocols for Mobile, Ad hoc Networks. In *Proceedings of the 7^{th} International Conference on Computer Communications and Networks*, pages 153–161, Lafayette, LA, October 1998.

[25] S. R. Das, C. E. Perkins, and E. M. Royer. Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 3–12, Tel Aviv, Israel, March 2000.

[26] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. The PIM Architecture for Wide-Area Multicast Routing. *IEEE/ACM Transactions on Networking*, 4(2):153–162, April 1996.

[27] I. S. Department. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Standard 802.11–1997*, 1994.

[28] H. Eriksson. MBONE: The Multicast Backbone. *Communications of the ACM*, 37(8):54–60, August 1994.

[29] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of the* 5^{th} *Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 263–270, Seattle, WA, August 1999.

[30] K. Fall and E. Kannan Varadhan. *ns* notes and documentation. *http://www-mash.cs.berkeley.edu/ns/*, 1999.

[31] L. Feeney. An Energy Consumption Model of Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks. http://www.ietf.org/proceedings/99jul/slides/manet-feeney-99jul.pdf, July 1999.

[32] C. L. Fullmer and J. Garcia-Luna-Aceves. Floor Acquisition Multiple Access (FAMA) for Packet-Radio Networks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 262–273, 1995.

[33] J. Garcia-Luna-Aceves and N. Shacham. Analysis of Routing Strategies for Packet Radio Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 292–302, March 1985.

[34] J. Garcia-Luna-Aceves, M. Spohn, and D. Beyer. Source Tree Adaptive Routing (STAR) Protocol. *IETF Internet Draft, draft-ietf-manet-star-00.txt*, October 1999. (Work in Progress).

[35] J. J. Garcia-Luna-Aceves and E. L. Madruga. A Multicast Routing Protocol for Ad-Hoc Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 784–792, New York, NY, March 1999.

[36] S. Guha and S. Khuller. Approximation Algorithms for Connected Dominating Sets. Technical Report 3660, Institute for Advanced Computer Studies, Department of Computer Science, University of Maryland, College Park, June 1996.

[37] P. Jacquet, P. Muhlethaler, and A. Qayyum. Optimized Link State Routing Protocol. *IETF Internet Draft, draft-ietf-manet-olsr-01.txt*, February 2000. (Work in Progress).

[38] L. Ji and M. S. Corson. A Lightweight Adaptive Multicast Algorithm. In *Proceedings of IEEE GLOBECOM*, pages 1036–1042, Sydney, Australia, December 1998.

[39] M. Jiang, J. Li, and Y. C. Tay. Cluster Based Routing Protocol (CBRP) Functional Specification. *IETF Internet Draft, draft-ietf-manet-cbrp-spec-00.txt*, August 1998. (Work in Progress).

[40] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the* 5^{th} *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 195–206, Seattle, WA, August 1999.

[41] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[42] J. Jubin. Uphill tier routing, less frequent tier data updating, and larger networks. SURAN Program Technical Note (SRNTN) 2, Richardson, TX, Rockwell Inc., August 1983.

[43] J. Jubin and J. Tornow. The DARPA Packet Radio Network Protocols. *Proceedings of the IEEE*, 75(1):21–32, 1987.

[44] P. Karn. MACA - A New Channel Access Protocol for Packet Radio. In *Proceedings of the ARRL/CRRL Amateur Radio Ninth Computer Networking Conference*, pages 134–140, 1990.

[45] L. Kleinrock and F. Kamoun. Hierarchical Routing for Large Networks. *Computer Networks*, 1(1):155–174, 1977.

[46] L. Kleinrock and J. Silvester. Optimum Transmission Radii for Packet Radio Networks or Why Six is a Magic Number. In *Proceedings of the IEEE National Telecommunications Conference*, pages 4.3.1–4.3.5, Birmingham, Alabama, December 1978.

[47] L. Kleinrock and F. A. Tobagi. Packet Switching in Radio Channels: Part I-Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics. *IEEE Transactions on Communications*, 23(12):1400–1416, December 1975.

[48] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the* 4^{th} *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 66–75, Dallas, Texas, October 1998.

[49] Y.-B. Ko and N. H. Vaidya. Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms. In *Proceedings of* 2^{nd} *IEEE Workshop on Mobile Computing Systems and Applications*, pages 101–110, New Orleans, LA, February 1999.

[50] G. S. Lauer. Hierarchical Routing Design for SURAN. In *Proceedings of the IEEE International Conference on Communications ICC*, pages 443–451, June 1988.

[51] G. S. Lauer. Packet-Radio Routing. In M. Steenstrup, editor, *Routing in Communications Networks*. Prentice Hall, 1995.

[52] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-Demand Multicast Routing Protocol. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1298–1304, New Orleans, LA, September 1999.

[53] S.-J. Lee, E. M. Royer, and C. E. Perkins. Ad hoc Routing Protocol Scalability. Submitted for publication.

[54] S.-J. Lee, W. Su, and M. Gerla. On-Demand Multicast Routing Protocol (ODMRP) for Ad-Hoc Networks. *IETF Internet Draft, draft-ietf-manet-odmrp-02.txt*, January 2000. (Work in Progress).

[55] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 565–574, Tel Aviv, Israel, March 2000.

[56] S.-J. Lee, C.-K. Toh, and M. Gerla. A Simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad-Hoc Networks. *IEEE Network*, 13(4):48–54, July/August 1999.

[57] C. R. Lin and M. Gerla. Adaptive Clustering for Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, September 1997.

[58] J. Macker and M. S. Corson. Internet Engineering Task Force (IETF) Mobile Ad Hoc Networks (MANET) Working Group Charter. http://www.ietf.org/html.charters/manet-charter.html.

[59] G. S. Malkin and M. E. Steenstrup. Distance-Vector Routing. In M. Steenstrup, editor, *Routing in Communications Networks*, pages 83–98. Prentice Hall, 1995.

[60] J. McQuillan. Adaptive Routing Algorithms for Distributed Computer Networks. Technical Report Technical Report, #2831, BBN, May 1974.

[61] J. Moy. Link-State Routing. In M. Steenstrup, editor, *Routing in Communications Networks*, pages 135–157. Prentice Hall, 1995.

[62] S. Murthy and J. J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM/Baltzer Mobile Networks and Applications*, 1(2):183–197, October 1996.

[63] M. Nemzow. *Implementing Wireless Networks*, chapter 3. McGraw-Hill, Inc, 1995.

[64] V. Park and S. Corson. Temporally-Ordered Routing Algorithm (TORA) Version 1. *IETF Internet Draft, draft-ietf-manet-tora-02.txt*, October 1999. (Work in Progress).

[65] V. D. Park and M. S. Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1405–1413, Kobe, Japan, April 1997.

[66] M. R. Pearlman and Z. J. Haas. Determining the Optimal Configuration for the Zone Routing Protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1395–1414, August 1999.

[67] G. Pei, M. Gerla, and T. W. Chen. Fisheye State Routing: A Routing Scheme for Ad hoc Wireless Networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 70–74, New Orleans, LA, June 2000.

[68] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang. A Wireless Hierarchical Routing Protocol with Group Mobility. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1538–1542, New Orleans, LA, September 1999.

[69] C. E. Perkins. IP Mobility Support. *RFC 2002*, October 1996.

[70] C. E. Perkins. Mobile IP. *IEEE Communications Magazine*, 3(5):84–99, May 1997.

[71] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM '94: Computer Communications Review*, 24(4):234–244, October 1994.

[72] C. E. Perkins and E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proceedings of the* 2^{nd} *IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, February 1999.

[73] C. E. Perkins, E. M. Royer, and S. R. Das. Ad hoc Address Autoconfiguration. *IETF Internet Draft, draft-ietf-manet-autoconf-00.txt*, July 2000. (Work in Progress).

[74] C. E. Perkins, E. M. Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. *IETF Internet Draft, draft-ietf-manet-aodv-07.txt*, November 2000. (Work in Progress).

[75] C. E. Perkins, E. M. Royer, and S. R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing for IP version 6. *IETF Internet Draft, draft-ietf-perkins-aodv6-00.txt*, November 2000. (Work in Progress).

[76] C. E. Perkins, E. M. Royer, and S. R. Das. IP Broadcast in Ad hoc Mobile Networks. *IETF Internet Draft, draft-ietf-manet-bcast-00.txt*, July 2000. (Work in Progress).

[77] C. E. Perkins, E. M. Royer, and S. R. Das. Quality of Service in Ad hoc On-Demand Distance Vector Routing. *IETF Internet Draft, draft-ietf-manet-aodvqos-00.txt*, July 2000. (Work in Progress).

[78] R. Ramanathan and M. Steenstrup. Hierarchically-organized, Multihop Mobile Wireless Networks for Quality-of-Service Support. *ACM/Baltzer Mobile Networks and Applications*, 3(1):101–118, 1998.

[79] T. S. Rappaport. *Wireless Communications, Principles & Practices*, chapter 3, pages 70–74. Prentice Hall, 1996.

[80] M. J. Riezenman. Technology 2000: Communications. *IEEE Spectrum*, 37(1):33–39, January 2000.

[81] E. M. Royer, S.-J. Lee, and C. E. Perkins. The Effects of MAC Protocols on Ad hoc Network Communication. In *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, Chicago, IL, September 2000.

[82] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. Submitted for publication.

[83] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser. Hierarchical Routing in Ad hoc Mobile Networks. Submitted for publication.

[84] E. M. Royer, L. E. Moser, and P. M. Melliar-Smith. Wireless Mobile Networking. In *Encyclopedia of Distributed Computing*. Kluwer Academic Publishers, 2000.

[85] E. M. Royer and C. E. Perkins. Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol. In *Proceedings of the* 5^{th} *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 207–218, Seattle, WA, August 1999.

[86] E. M. Royer and C. E. Perkins. An Implementation Study of the AODV Routing Protocol. In *Proceedings of the Wireless Communications and Networking Conference (WCNC)*, Chicago, IL, September 2000.

[87] E. M. Royer and C. E. Perkins. Multicast Ad hoc On-Demand Distance Vector (MAODV) Routing. *IETF Internet Draft, draft-ietf-manet-maodv-00.txt*, July 2000. (Work in Progress).

[88] E. M. Royer and C. E. Perkins. Transmission Range Effects on AODV Multicast Communication. *To appear in ACM/Baltzer Mobile Networks and Applications*, 2001.

[89] E. M. Royer and C.-K. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Networks. *IEEE Personal Communications*, 6(2):46–55, April 1999.

[90] N. Shacham. Organization of Dynamic Radio Networks by Overlapping Clusters: Architecture Considerations and Optimization. In *Performance '84*, pages 435–447, December 1984.

[91] N. Shacham. Hierarchical Routing in Large, Dynamic Ground Radio Networks. In *Proceedings of the* 18^{th} *Hawaii International Conference on System Sciences*, pages 292–301, 1985.

[92] P. Sinha, R. Sivakumar, and V. Bharghavan. CEDAR: a Core-Extraction Distributed Ad hoc Routing algorithm. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 202–209, New York, NY, March 1999.

[93] K. Sohrabi and G. J. Pottie. Performance of a Novel Self-Organization Protocol for Wireless Ad hoc Sensor Networks. In *IEEE VTS* 50^{th} *Vehicular Technology Conference*, pages 1222–1226, Piscataway, NJ, September 1979.

[94] M. Steenstrup. Cluster-Based Networks. In C. E. Perkins, editor, *To appear in Ad Hoc Networking*, chapter 4. Addison-Wesley Publishers, 2000.

[95] J. Stevens. Spreading Connectivity Information out over Multiple PROP Periods, and Timeliness of Information. SURAN Program Technical Note (SRNTN) 21, Richardson, TX, Rockwell Inc., May 1985.

[96] F. A. Tobagi and L. Kleinrock. Packet Switching in Radio Channels: Part-II - The Hidden Terminal Problem in Carrier Sense MultipleAccess Models and the BusyTone Solution. *IEEE Transactions on Communications*, 23(12):1417–1433–20, December 1975.

[97] C.-K. Toh. A Novel Distributed Routing Protocol to Support Ad-Hoc Mobile Computing. In *Proceedings of the* 15^{th} *Annual IEEE International Phoenix Conference on Computers and Communications*, pages 480–486, Scottsdale, AZ, March 1996.

[98] P. F. Tsuchiya. The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks. In *Computer Communication Review*, pages 35–42, Stanford, CA, August 1988.

Appendix A

Proof of the Loop-free Property

Verifying that AODV establishes only loop-free routes is easy because of the effect of the destination sequence number on the maintenance of routes. Suppose that there is a loop in a route to a destination Z, and that nodes X_i are the nodes in the loop for i=1,2,...,n. As a matter of terminology, we say that X_i "points to" X_j (symbolically, $X_i \to X_j$) if the routing table entry of X_i for destination Z shows node X_j as the next hop to Z. Then, for each X_i, $X_i \to X_{i+1}$ for i=1,2,...,n, and $X_n \to X_1$.

Let T_i be the destination sequence number for the route entry at X_i for destination Z. Then, $T_i \leq T_{i+1}$ whenever $X_i \to X_{i+1}$, because of the processing of the RREP specified in Section 2.2.3. Since $T_1 \leq T_2 \leq \ldots \leq T_n \leq T_1$, the destination sequence numbers are the same for every node X_i in the routing loop. Moreover, if it were possible to create a routing loop, this equality would hold the instant the loop is created. Furthermore, because the destination sequence numbers are all the same, the next hop information must have been derived at every node X_i from the same RREP transmitted by the destination Z.

Consider now the distance m_i of the node X_i to the destination Z. Since $X_i \to X_{i+1}$ only if $m_i = m_{i+1} + 1$, it follows that $m_1 = m_n + (n-1)$. But, since $X_n \to X_1$, $m_n = m_1 + 1$ and $n = 0$, this is a contradiction. $\square$

An inductive argument is also possible. Letting k be the minimum length of a routing loop, one can construct another routing loop of length $k-1$ by modifying the routing table at the node X_k to point to X_2 instead of of X_1. This works as long as $k \geq 3$, showing that it is only possible to construct a routing loop of length 2. A RREP handling shows that routing loops cannot have length 2, so that routing loops cannot contain more than one node. But this means that there cannot be any routing loops.

Appendix B

AODV Message Formats

B.1 Route Request (RREQ) Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |J|R|G|       Reserved          |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Broadcast ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Destination Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Source Sequence Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Route Request message is illustrated above, and contains the following fields:

Type	1
J	Join flag; set when source node wants to join a multicast group.
R	Repair flag; set when a node wants to initiate a repair to connect two previously disconnected portions of the multicast tree.
G	Gratuitous RREP flag; indicates whether a gratuitous RREP should be unicast to the node specified in the Destination IP Address field.
Reserved	Sent as 0; ignored on reception.
Hop Count	The number of hops from the Source IP Address to the node handling the request.

Broadcast ID — A sequence number uniquely identifying the particular RREQ when taken in conjunction with the source node's IP address.

Destination IP Address — The IP address of destination for which a route is desired.

Destination Sequence Number — The last sequence number received in the past by the source for any route towards the destination.

Source IP Address — The IP address of the node which originated the Route Request.

Source Sequence Number — The current sequence number to be used for route entries pointing to (and generated by) the source of the route request.

B.2 Route Reply (RREP) Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |R|A|    Reserved     |Prefix Sz|   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Destination IP address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Destination Sequence Number                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Lifetime                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Route Reply message is illustrated above, and contains the following fields:

Type — 2

R — Repair flag; set when a node is responding to a repair request to connect two previously disconnected portions of the multicast tree.

A — Acknowledgment flag; indicates an acknowledgment is required from the next hop for the RREP.

Reserved — Sent as 0; ignored on reception.

Prefix Size — If nonzero, the 5-bit Prefix Size specifies that the indicated next hop may be used for any nodes with the same routing prefix (as defined by the Prefix Size) as the requested destination.

Hop Count
: The number of hops from the Source IP Address to the Destination IP Address. For multicast route requests this indicates the number of hops to the multicast tree member sending the RREP.

Destination IP Address
: The IP address of the destination for which a route is supplied.

Destination Sequence Number
: The destination sequence number associated to the route.

Source IP Address
: The IP address of the source node which issued the RREQ for which the route is supplied.

Lifetime
: The time for which nodes receiving the RREP consider the route to be valid.

B.3 Route Error (RERR) Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |N|          Reserved           |   DestCount   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Unreachable Destination IP Address (1)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Unreachable Destination Sequence Number (1)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|  Additional Unreachable Destination IP Addresses (if needed)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Additional Unreachable Destination Sequence Numbers (if needed)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Route Error message is illustrated above, and contains the following fields:

Type
: 3

N
: No delete flag; set when a node has performed a local repair of a link, and upstream nodes should not delete the route.

Reserved
: Sent as 0; ignored on reception.

DestCount
: The number of unreachable destinations included in the message; must be at least 1.

Unreachable Destination IP Address
: The IP address of the destination which has become unreachable due to a link break.

Unreachable Destination Sequence Number
The last known sequence number, incremented by one, of the destination listed in the previous Unreachable Destination IP Address field.

B.4 Route Reply Acknowledgment (RREP-ACK) Message Format

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type 4

Reserved Sent as 0; ignored on reception.

B.5 Multicast Activation (MACT) Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |J|P|G|U|R|    Reserved         |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Multicast Group IP address                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Source Sequence Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Multicast Activation message is illustrated above, and contains the following fields:

Type 5

J Join flag; set when a node is joining the multicast group, as opposed to finding a route to the group for the transmission of data messages.

P Prune flag; set when a node wishes to prune itself from the tree, unset when the node is activating a tree link.

G — Group Leader flag; set by a multicast tree member that fails to repair a multicast tree link breakage, and indicates to the group member receiving the message that it should become the new multicast group leader.

U — Update flag; set when a multicast tree member has repaired a broken tree link and is now a new distance from the group leader.

R — Reboot flag; set when a node has just rebooted (see Section 2.6).

Reserved — Sent as 0; ignored on reception.

Hop Count — The distance of the sending node from the multicast group leader. Used only when the 'U' flag is set; otherwise sent as 0.

Multicast Group IP Address — The IP address of the Multicast Group for which a route is supplied.

Source IP Address — The IP address of the sending node.

Source Sequence Number — The current sequence number for route information generated by the source of the route request.

B.6 Group Hello (GRPH) Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |U|M|        Reserved           |   Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Group Leader IP address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Multicast Group IP address                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Multicast Group Sequence Number                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Group Hello message is illustrated above, and contains the following fields:

Type — 6

U — Update flag; set when there has been a change in group leader information.

M — Off_Mtree flag; set by a node receiving the group hello that is not on the multicast tree.

Reserved
: Sent as 0; ignored on reception.

Hop Count
: The number of hops the packet has traveled. Used by multicast tree nodes to update their distance from the group leader when the M flag is not set.

Group Leader IP Address
: The IP address of the group leader.

Multicast Group IP Address
: The IP address of the Multicast Group for which the sequence number is supplied.

Multicast Group Sequence Number
: The current sequence number of the multicast group.

Appendix C

AODV Message Extension Formats

AODV control messages have extensions defined in the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |     type-specific data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

Type	x
Length	The length of the type-specific data, not including the Type and Length fields of the extension.

C.1 Multicast Group Leader Extension Format

This extension is appended to a RREQ by a node wishing to repair a multicast tree.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     | Multicast Group Leader IP ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   ... Address (continued)     | Previous Hop IP Address ...   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   ... (continued)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type 3

Length 8

Multicast Group Leader IP Address
The IP Address of the Multicast Group Leader.

Previous Hop IP Address
The IP Address of the node which previously received the RREQ. This field is used when the RREQ is unicast to the group leader when a node wishes to join a multicast group.

C.2 Multicast Group Rebuild Extension Format

This extension is appended to a RREQ by a node wishing to repair a multicast tree.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |  Multicast Group Hop Count    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type 4

Length 2

Multicast Group Hop Count
The distance in hops of the node sending the RREQ from the Multicast Group Leader.

C.3 Multicast Group Information Extension Format

The following extension is used to carry additional information for the RREP message (see Section 2.3.3) when sent to establish a route to a multicast destination.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |  Multicast Group Hop Count    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Multicast Group Leader IP Address               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type 5

Length 6

Multicast Group Hop Count
The distance of the node from the Multicast Group Leader.

Multicast Group Leader IP Address
The IP Address of the current Multicast Group Leader.

C.4 Maximum Delay Extension Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |             Delay             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type 6

Length 2

Delay This field has different meanings for RREQ and RREP messages. In a RREQ message it indicates the maximum number of seconds allowed for a transmission from the source (or from an intermediate node forwarding the RREQ) to the destination. In a RREP message, it indicates the current estimate of cumulative delay from the intermediate node forwarding the RREP to the destination.

The Maximum Delay Extension can be appended to a RREQ by a node requesting a QoS route in order to place a maximum bound on the acceptable time delay experienced on any acceptable path from the source to the destination.

C.5 Minimum Bandwidth Extension Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |         Bandwidth ...         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ... Bandwidth            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type 7

Length 4

Bandwidth In a RREQ message this field indicates the minimum bandwidth (in Kbits/sec) that must be available along an acceptable path from source to destination. In a RREP message this indicates the minimum bandwidth available in the route from the node forwarding the RREP to the destination.

The Minimum Bandwidth Extension can be appended to a RREQ by a requesting node in order to specify the minimum amount of bandwidth that must be made available along an acceptable path from the source to the destination.

C.6 ICMP QOS LOST Message

An ICMP QOS_LOST message is generated when an intermediate node experiences an increase in NODE_TRAVERSAL_TIME or a decrease in link capacity. This may affect the QoS guarantees previously provided to some sources. The format of this message is as follows.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |           Destination IP Address ...          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    ...        |
+-+-+-+-+-+-+-+-+
```

Type 8

Destination IP address
: IP address of the destination node using the link for which there has been a change in a QoS parameter.

This message is extended using either the Minimum Bandwidth or the Maximum Delay Extensions as described previously. The Minimum Bandwidth Extension is used when there is a drop in link capacity and the change in bandwidth is indicated in the Bandwidth field. The Maximum Delay Extension is used when there is a change in delay as indicated in the Delay field.

The QOS_LOST message is forwarded to all sources potentially affected by the change in the QoS parameter. These are those sources to which a RREP with a QoS extension was previously forwarded.

Appendix D

ARC Message Formats

D.1 Hello Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |N|   Reserved  |  CLdr1 Length | CLdr2 Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source IP Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Source Status                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Source Sequence Number                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      CLdr1_List Entry (1)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Additional CLdr1_List Entries (if needed) ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      CLdr2_List Entry (1)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Additional CLdr2_List Entries (if needed) ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Hello message is illustrated above, and contains the following fields:

Type	1
N	Hello_Now flag; set when a node is looking for cluster leaders. It indicates they should reply immediately.
Reserved	Sent as 0; ignored on reception.
CLdr1 Length	Number of Entries if CLdr1_List.
CLdr2 Length	Number of Entries in CLdr2_List.

Source IP Address
: The IP address of the node sending the packet.

Source Status
: The status level of the node sending the packet.

Source Sequence Number
: The sequence number of the node sending the packet.

CLdr1_List Entry
: IP Address of a cluster leader with which the node is in contact.

CLdr2_List Entry
: IP Address of a cluster leader which the node can reach through a joint gateway.

D.2 Route Activation (RTAct) Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |            Reserved             |  Hop Count   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Source Status                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Cluster Leader IP Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Route Activation message is illustrated above, and contains the following fields:

Type
: 2

Reserved
: Sent as 0; ignored on reception.

Hop Count
: Distance, in hops, of the destination from the node.

Source IP Address
: The IP address of the node sending the packet.

Source Status
: The status level of the node sending the packet.

Destination IP Address
: IP Address of the destination node.

Cluster Leader IP Address
: IP Address of the cluster leader that is the next hop to the destination.

Appendix E

ARCH Message Formats

E.1 Hello Message Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |N|   Reserved  |  CLdr1 Length | CLdr2 Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Hier_List Length                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Source Status                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Source Sequence Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Hier_List Entry (1)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Additional Hier_List Entries (if needed) ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      CLdr1_List Entry (1)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Additional CLdr1_List Entries (if needed) ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      CLdr2_List Entry (1)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Additional CLdr2_List Entries (if needed) ...        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The format of the Hello message is illustrated above, and contains the following fields:

Type	1
N	Hello_Now flag; set when a node is looking for cluster leaders. It indicates they should reply immediately.

Reserved — Sent as 0; ignored on reception.

CLdr1 Length — Number of Entries if CLdr1_List.

CLdr2 Length — Number of Entries in CLdr2_List.

Hier_List Length — Number of Entries in Hier_List.

Source IP Address — The IP address of the node sending the packet.

Source Status — The status level of the node sending the packet.

Source Sequence Number — The sequence number of the node sending the packet.

Hier_List Entry — IP Address of a hierarchical leader within whose hierarchical group the node is a member.

CLdr1_List Entry — IP Address of a cluster leader with which the node is in contact.

CLdr2_List Entry — IP Address of a cluster leader which the node can reach through a joint gateway.

Appendix F

Configuration Parameters

Parameter	Value
`active_route_timeout`	3,000 msec
`allowed_hello_loss`	2
`broadcast_record_time`	2 * `net_traversal_time`
`delete_period`	see note below
`grouph_hello_interval`	5,000 msec
`hello_interval`	1,000 msec
`local_add_ttl`	2
`max_repair_ttl`	0.3 * `net_diameter`
`mtree_build`	2 * `rev_route_life`
`my_route_timeout`	2 * `active_route_timeout`
`net_diameter`	35
`next_hop_wait`	`node_traversal_time` + 10
`node_traversal_time`	40
`prune_timeout`	`active_route_timeout`
`retransmit_time`	750 msec
`rev_route_life`	`net_traversal_time`
`rreq_retries`	2
`ttl_start`	1
`ttl_increment`	2
`ttl_threshold`	7

Table F.1: Suggested Parameter Values

The `delete_period` is an upper bound on the time for which an upstream node A can have a neighbor B to be an active next hop for destination D, while B has invalidated the route to D. Beyond this time B can delete the route to D. The determination of the upper bound somewhat depends on the characteristics of the underlying link layer. For example, if the link layer feedback is used to detect the loss of a link, `delete_period` must be at least `active_route_timeout`. If there is no feedback and Hello messages must be used, `delete_period` must be at least the maximum of `active_route_timeout` and `allowed_hello_loss` * `hello_interval`. If Hello messages are received from a neighbor but data packets to that neighbor are lost (i.e. due to temporary link asymmetry), more concrete assumptions about the underlying link layer have to be made. It is assumed that such asymmetry cannot persist beyond a certain certain time, say, a multiple K of `allowed_hello_loss` * `hello_interval`. In other words, it cannot not be the case that a node receives K subsequent Hello messages from a neighbor, while that same neighbor fails to receive any data packet from the node in this period. This is a reasonable assumption as AODV currently works only with symmetric links. Covering all possibilities,

`delete_period` = K * max (`active_route_timeout`,
`allowed_hello_loss` * `hello_interval`) (K = 5 is recommended).

The `net_diameter` parameter measures the maximum possible number of hops between two nodes in the network. The `node_traversal_time` is a conservative estimate of the average one hop traversal time for packets and should include queueing delays, interrupt processing times and transfer times. The `ttl_start` value should be at least two if Hello messages are used for local connectivity information. Performance of the AODV protocol is sensitive to the chosen values of these constants, which often depend on the characteristics of the underlying link layer protocol, radio technologies, etc.