Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks

Jorjeta G. Jetcheva

CMU-CS-04-126
May 6, 2004

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee

David B. Johnson, Chair
Jim Kurose
Srinivasan Seshan
Daniel P. Siewiorek

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

This work was supported in part by NASA under grant NAG3-2534 at Rice University, by NSF under grant CCR-0209204 at Rice University, by a gift from Schlumberger to Rice University, and by the Air Force Materiel Command (AFMC) under DARPA contract number F19628-96-C-0061. The views and conclusions contained here are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NASA, NSF, Schlumberger, AFMC, DARPA, Rice University, Carnegie Mellon University, or the U.S. Government or any of its agencies.

Keywords: wireless, ad hoc networks, routing, multicast, unidirectional links, mesh networks.

Abstract

An ad hoc network does not require any pre-existing infrastructure or configuration but is formed spontaneously by (possibly mobile) nodes that wish to communicate. Each node in the ad hoc network acts as a router and forwards packets on behalf of other nodes, allowing nodes that are not within wireless range of each other to communicate over “multi-hop” paths. Example ad hoc network applications include disaster relief scenarios, conference attendees who want to form a network in order to exchange documents, friends involved in a distributed outdoors game, surveillance teams composed of persons or robots exploring a dangerous area, or another planet.

Previous efforts to design general-purpose on-demand multicast routing protocols for ad hoc networks have utilized periodic (non-on-demand) mechanisms within some portions of the protocol. The overall on-demand nature of such protocols derives from the fact that significant portions of their operation are active only for active multicast groups. However, the periodic mechanisms within the protocol are responsible for core routing functionality and significantly affect overall protocol performance.

My thesis in this dissertation is that on-demand multicast that does not rely on periodic techniques is more efficient and performs better than multicast that utilizes such techniques.

To support my thesis statement, in this dissertation I present the design and evaluation of a new multicast protocol, the Adaptive Demand-Driven Multicast Routing protocol (ADMR) for multi-hop wireless ad hoc networks. ADMR uses no periodic control packet network-wide floods, periodic neighbor sensing, or periodic routing table exchanges, and adapts its behavior based on network conditions and application sending pattern, allowing efficient detection of broken links and expiration of routing state that is no longer needed. I conduct an extensive simulation of ADMR and show that it compares well against protocols that utilize proactive mechanisms, and typically generates three to five times less packet overhead.

In addition, in this dissertation, I study the impact of unidirectional links on the routing characteristics of ad hoc networks, and use this study to explore the effect of unidirectional links on multicast routing performance. Using the lessons learned from this work, I extend ADMR with mechanisms that enable it to route over unidirectional links, and show that the unidirectional extensions improve the performance of the protocol by increasing packet delivery ratio by up to 45% and decreasing overhead by up to 68%.

Acknowledgments

There are so many people who have contributed in various ways to my life in graduate school. Alas, I can only make a few brief remarks here.

I would like to thank my advisor, Dave Johnson, for all his help, advice, humor and patience, both in the past seven years, and in the future. I have been very lucky to work with him.

My thesis committee members, Srini Seshan, Jim Kurose and Dan Siewiorek, have been great resources in the dissertation process and during my job hunt.

I am indebted to my parents, Marlena Krasteva and Gueorgui Jetchev, and to my sisters, Petia Ovcharova and Aleksandra Taskova, for their continued support and caring, for being there for me, through thick and thin.

I would like to thank my friends, Desislava Rangelova, Amanda Abbey, Michelle Ellis, Tedy Shamandoura, Helene Bulte, Olga Zemlyanaya, Kip Walker, Melissa Duarte, and Nu Lu, now spread all over the world, who nonetheless manage to be deeply involved in my life, and are always there to brighten my mood, and fuel my optimism.

My colleagues and office mates at both CMU and Rice have been invaluable resources both professionally and personally, and have helped me laugh and procrastinate on many occasions. In particular, I would like to thank my friends and colleagues, Brigitte Pientka, Laurie Hiyakumoto, and Orna Raz, fellow monarchists, Yih-Chun Hu, Dave Maltz, Josh Broch, Santa PalChaudhuri, Amit Saha, Shu Du, and Khoa To, and my officemates, Raj Bandyopadhyay, Umur Akar and Lars Birkedal.

I would also like to thank Jeannette Wing, Catherine Copetas, and the whole Computer Science department faculty and staff for the amazing care they take of all of us, and especially Sharon Burks for always being so kind and helpful.

Contents

Illustrations

List of Tables

Chapter 1

Introduction

The emergence and wide-spread deployment of high-bandwidth communication networks has revolutionized contemporary lifestyle. Millions of people world-wide use communication technology every day to access, post, and exchange information, or to interact with each other. While wired networks enable users to be connected while at home or in the office, wireless networks have the potential to enable *ubiquitous connectivity*.

In this dissertation, I focus on wireless *ad hoc networks*. An ad hoc network does not require any pre-existing infrastructure or configuration but is formed spontaneously by (possibly mobile) nodes that wish to communicate. Each node in the ad hoc network acts as a router and forwards packets on behalf of other nodes, allowing nodes that are not within wireless range of each other to communicate over "multi-hop" paths (Figure 1.1). The ad hoc network routing protocol is responsible for finding such paths and for detecting broken links. These may be caused by wireless interference, propagation phenomena, or node motion, as a result of which nodes that were previously within wireless transmission range of each other may no longer be in range.

An ad hoc network can be used to extend the coverage area of another network such as the Internet or a cellular network, or can be used independently. An example of an ad hoc network application is a disaster relief scenario, in which a team is deployed to an area affected by an earthquake, where the networking infrastructure may have been destroyed or disabled. Team members with wireless-enabled devices may self-organize into an ad hoc network and exchange information with each other over that network even as they move around as required by the tasks they need to perform. Other applications of ad hoc networks include conference attendees who want to form a network in order to exchange documents, friends involved in a distributed outdoors game, or surveillance teams composed of persons or robots exploring a dangerous area, or another planet.

1.1. Routing in Ad Hoc Networks

The design of routing protocols for ad hoc networks is subject to constraints that are different from those relevant to the Internet environment. Ad hoc networks are characterized by low bandwidth, high packet loss, finite energy resources, and dynamic, potentially frequent topology changes.

The wireless environment is broadcast in nature and nodes within transmission range of each other share the use of the available bandwidth. Thus, the effective bandwidth along a path in an ad hoc network is heavily dependent on the number of actively transmitting nodes within transmission range of the nodes along that path. In addition, each node along a multi-hop path, shares the available bandwidth with the node before and after it in the path. Overall, the bandwidth in an

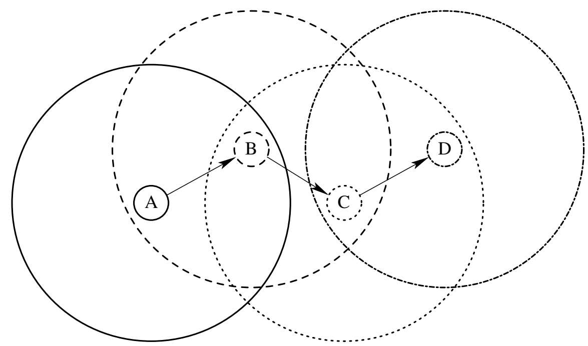

Figure 1.1: Node **A** communicates with node **C** over a multi-hop path. The large circles denote each node's transmission range.

ad hoc network may be significantly lower than the nominal bandwidth the network hardware can support.

Ad hoc networks are much more prone to packet loss than wired networks. The reception of a packet at a node depends on the signal-to-noise ratio at which it is received, which in turn is affected by concurrent transmissions in the vicinity of the node, as well as by propagation conditions. Loss in the wireless environment may also be caused by interference sources that emit signal at a frequency overlapping the one used by the ad hoc network.

Since nodes in ad hoc networks are mobile and wireless, they typically need to rely on a finite energy supply.

Ad hoc networks are characterized by a potentially highly dynamic topology. Topology variations can happen on a variety of time-scales and can be caused by node motion, propagation effects, depletion of battery resources on a node, node shutdown or failure.

Given the constraints typical of the ad hoc network environment, the main requirements for ad hoc network routing protocol design can be summarized as follows:

- *Low overhead.* The routing protocol needs to minimize the number and size of control messages it transmits, as well as the number of times it forwards each data packet, in order to conserve bandwidth and battery resources.

- *Adaptiveness.* In order to operate efficiently in a wide range of network conditions, the routing protocol needs to be able to adapt to a highly dynamic environment in which topology and propagation conditions may vary significantly on a variety of time scales.

- *Resilience to loss.* The routing protocol needs to operate correctly and efficiently in the presence of (control) packet loss. The likelihood of packet loss in the ad hoc network environment may be high, especially for multicast and broadcast packets, which standard link layers deliver with limited media access control and without acknowledgments.

There are two approaches to designing routing protocols for ad hoc networks. Protocols based on the *proactive* approach distribute and maintain routing information periodically, regardless of whether this information is currently needed for communication. Proactive mechanisms include network-wide control packet floods, periodic neighbor sensing (beaconing), and periodic routing table exchanges. Protocols based on the *on-demand* approach only discover paths between nodes that wish to communicate, and maintain the connectivity between them only during the communication.

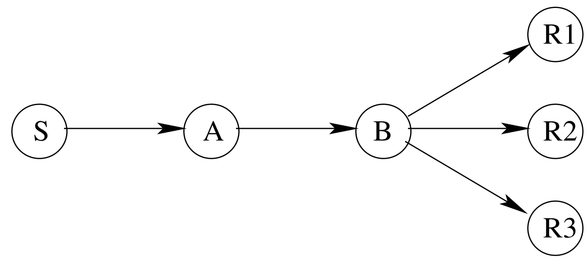

Figure 1.2: Packets unicast by **S** to **R1**, **R2**, and **R3**, traverse the links (**S**,**A**) and (**A**, **B**) multiple times.

The benefits of the proactive approach typically include simplicity of design, and lower latency to begin communicating because communication state is usually already set up before the communication begins. However, proactive routing mechanisms can generate significant overhead, a large portion of which is typically unnecessary as it is not generated in response to any communication needs. On-demand protocols may incur higher latency to establish connectivity than proactive protocols, but they typically generate less overhead, and operate efficiently by scaling this overhead according to the level of mobility in the network as well as the connectivity that needs to be established and maintained.

1.2. Multicast in Ad Hoc Networks

While *unicast* routing involves a pair of communicating nodes, in *multicast*, multiple nodes can be involved in a communication session simultaneously. In the multicast service model [6], any machine (or node) in the network can join or leave a multicast group at any time, and any node in the network can send data to any multicast group. The multicast sources do not need to know who the receivers are but need only know the multicast group address.

Multicast is usually a separate service from unicast for efficiency reasons. For example, the path from a source to two or more receivers may share one or more links (Figure 1.2) and as a result, sending a separate unicast packet to each receiver would mean that the same packet would traverse each shared link multiple times. The goal of an efficient multicast routing protocol is to deliver a copy of each packet to each multicast receiver by duplicating the packet in the network as few times as possible (e.g., only at node **B** in Figure 1.2).

Multicast applications include audio and video conferencing, distributed gaming, as well as data distribution such as news or stock quotes, in which one or more servers continuously send data to a multicast group. Nodes interested in receiving this data may subscribe to the relevant group, and when they are no longer interested in receiving the content can unsubscribe from the group.

In a disaster relief scenario, a surveillance team can explore the area first and send video and audio data to request the aid of specialized team members needed at a particular location. It may be necessary to dispatch medical personnel to certain areas, and debris clearing professionals to others. The coordinators of the various subteams such as medical, debris clearance, and fire fighters, can decide how many team members to dispatch and what equipment they should take along based on the data received from the surveillance team. Each team comprised of professionals with a given specialty can form a multicast group. Sending data to all medical personnel, for example, can be done efficiently by only addressing it to the “medical personnel” multicast group, rather than using broadcast and sending the data to all nodes in the network. In addition, medical personnel that are currently not occupied can join an "available medical personnel" multicast group, and information that only needs to reach available medical personnel will only be sent to this group instead of being distributed to all medical personnel.

Most prior work in ad hoc network routing has focused on routing of unicast packets, but a number of protocols for multicast routing have been proposed over the past few years as well [2, 5, 9, 14, 16, 17, 37–39, 41, 42], using a variety of basic routing algorithms and techniques. Of these multicast routing protocols, a few attempt to operate in an *on-demand* fashion [14, 16, 17, 37, 38, 41]. However, designing an ad hoc network multicast routing protocol that operates *entirely* on-demand has been difficult. Several such multicast protocols have been proposed [14, 17] that perform well in scenarios with small groups [17] or in networks in which mobility is very high and flooding is the only way to deliver packets successfully [14]. However, these protocols do not scale well, and are not efficient to use as general-purpose multicast protocols.

Previous efforts to design general-purpose on-demand multicast protocols for ad hoc networks have utilized periodic (non-on-demand) mechanisms within some portions of the protocol. The overall on-demand nature of such protocols derives from the fact that significant portions of the protocol operation are active only for active multicast groups. However, the periodic mechanisms within the protocol are responsible for core routing functionality and significantly affect overall protocol performance. For example, the On-Demand Multicast Routing Protocol (ODMRP) [38] builds multicast meshes through periodic network-wide control packet floods. The protocol also relies on these floods to repair broken links in the mesh that occur between floods. The Multicast Ad Hoc On-Demand Distance Vector (MAODV) protocol [37] requires continuous periodic neighbor sensing for broken link detection, and periodic group hello packets for multicast forwarding state creation. The group hello messages are sent regardless of whether or not there are any senders for the multicast group in the network, as long as there is at least one receiver. Similarly to MAODV, the Associativity-Based Multicast protocol (ABAM) [41] requires continuous periodic neighbor sensing for broken link detection and distribution of link characteristics. The Lightweight Adaptive Multicast protocol (LAM) [16] requires similar functionality from the underlying unicast protocol on top of which it operates.

Current proposals for general-purpose on-demand multicast protocols do not meet the requirements for routing protocol design outlined in Section 1.1. The protocols do not adapt their behavior to network conditions; instead they use fixed-interval continuous periodic control overhead, which imposes a high load on the network and reduces the available bandwidth and battery power.

1.3. Thesis Statement

My thesis in this dissertation is that on-demand multicast that does not rely on periodic techniques is more efficient and performs better than multicast that utilizes such techniques.

1.4. Dissertation Overview

To support my thesis statement, in this dissertation I present the design and evaluation of a new multicast protocol, the Adaptive Demand-Driven Multicast Routing protocol (ADMR) for multi-hop wireless ad hoc networks. ADMR uses no periodic control packet network-wide floods, periodic neighbor sensing, or periodic routing table exchanges, and adapts its behavior based on network conditions and application sending pattern, allowing efficient detection of broken links and expiration of routing state that is no longer needed. I show that ADMR works well in a variety of simulation scenarios, and compares well against protocols that utilize proactive mechanisms and generates significantly lower packet overhead. The protocols that I have chosen to compare ADMR against are the On-Demand Multicast Routing Protocol (ODMRP) [38] and the Multicast Ad Hoc On-Demand Distance Vector (MAODV) [37] protocol. These protocols represent two different design points in the multicast protocol design space, are well documented, and have been shown to perform well in previous studies.

In addition, in this dissertation, I study the impact of unidirectional links on the routing characteristics of ad hoc networks, and use this study to explore the effect of unidirectional links on multicast routing performance. Using the lessons learned from this work, I extend ADMR with mechanisms that enable it to route over unidirectional links and show that the unidirectional extensions improve the performance of the protocol by increasing packet delivery ratio by up to 45% and decreasing overhead by up to 68%.

1.5. Research Contributions

The main research contributions of this dissertation are as follows:

- The design of ADMR, the first general purpose-multicast routing protocol for ad hoc networks that does not utilize any periodic control packet transmissions.
- A detailed comparative performance evaluation of ADMR, ODMRP and MAODV which explores the effectiveness of different multicast mechanisms in a wide range of ad hoc network simulation scenarios.
- A detailed study of the impact of unidirectional links on the routing characteristics of ad hoc networks.
- The first study of the impact of unidirectional links on multicast protocol performance in ad hoc networks.
- The design and evaluation of the first unidirectional routing extensions specifically for multicast, which are the first unidirectional extensions that operate entirely on-demand without at the same time requiring GPS information.

Chapter 2

The Adaptive Demand-Driven Multicast Routing protocol (ADMR)

ADMR supports the traditional IP multicast service model of allowing receivers to receive multicast packets sent by any sender [6], as well as the newer source-specific multicast service model in which receivers may join a multicast group only for specific senders [12]. As in both multicast service models, a node need not be a receiver for the group to be able to send to the group, senders need not declare their intention to send multicast packets to the group before doing so, and senders need not explicitly declare their intention to stop being multicast senders. In addition, multicast sources do not need to know who the receivers are or where the network they are located, and receivers do not need to know who the sources are or where they are located.

ADMR was designed under the assumption that nodes in the network may move at any time, and that any packet may be lost due to factors such as packet collision, wireless interference, or signal attenuation due to distance. ADMR is intended for networks of devices with omni-directional antennas where a transmission by one node may be overheard by all nodes within its wireless transmission range (Figure 2.1).

ADMR features can be summarized as follows:

- ADMR is completely distributed and does not rely on any centralized coordination or control.
- If there are no sources or receivers for a multicast group, ADMR does not send any control packets.
- If there are no receivers, ADMR sources only flood infrequent data (to heal partitions) and do not transmit other data or control packets.
- ADMR uses no periodic network-wide floods of control packets, periodic neighbor sensing, or periodic routing table exchanges (and does not expect neighbor sensing to be performed at the MAC layer), and requires no core. No other *general-purpose* multicast protocol for ad hoc networks proposed prior to ADMR has all these properties in one protocol. (A general-purpose protocol is one which was not intended to be used in a limited set of network environments such as networks with a small number of nodes, or with a small diameter.)
- ADMR builds an "extended" source-rooted tree, called a *source mesh*, for each multicast source. Multicast packets are forwarded along this mesh from the source to the multicast receivers along the shortest-delay paths within the mesh.
- ADMR adapts its behavior based on application sending pattern, allowing efficient detection of broken links and expiration of routing state that is no longer needed.
- To handle bursty sources, ADMR sends a limited number of keep-alives along the multicast mesh, in order to distinguish lack of data from disconnection.

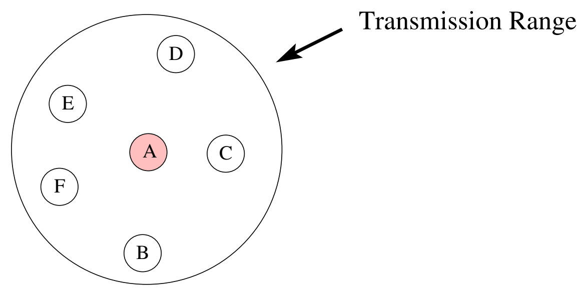

Figure 2.1: Omnidirectional Range: Nodes within **A**'s transmission range may receive its packets.

- ADMR uses passive acknowledgments for efficient automatic mesh pruning.
- ADMR does not expect reliable or in-order delivery of its control or data packets for correct operation.
- ADMR does not require or use GPS, or other position information.
- ADMR detects high mobility without the use of GPS information, or additional control traffic, and switches to flooding for some period of time before reverting back to normal operation.
- ADMR is designed to work independently of the unicast protocol used in the ad hoc network and can thus work with any unicast protocol or even without a unicast protocol.

In this chapter, I describe the operation of ADMR in bidirectional networks. The mechanisms I developed to allow ADMR to operate in networks where unidirectional links may be present are described in Chapter 5.

For clarity, all figures pertaining to the protocol description depict transmission of packets as arrows. In addition, reception of a packet is only shown for nodes that will forward the packet further, i.e., each transmission is not depicted as a broadcast received by all nodes within the transmitting node's wireless range, even though the wireless medium under consideration is a broadcast medium.

2.1. Overview

Multicast senders and receivers using ADMR cooperate to establish forwarding state in the network to allow multicast communication (*multicast state setup*). In addition, ADMR adaptively monitors the correct operation of the multicast forwarding state and incrementally repairs it when one or more receivers or forwarding nodes become disconnected from a multicast sender (*multicast state maintenance*).

2.1.1. Multicast State Setup

When the routing layer at a node **S** receives a multicast packet for a group **G** from the application layer, if **S** is not currently a source for **G**, ADMR attaches an ADMR header to the packet and floods the packet in the network. When a node **R** who is a receiver for **G** receives this packet, it unicasts a RECEIVER JOIN packet back to **S**. This packet sets up forwarding state for **S** and **G** in each node through which it is forwarded.

When the application on a node **R** first issues a join request for group **G**, ADMR floods a MULTICAST SOLICITATION packet in the network. Each source for **G** responds to the solicitation by unicasting a UNICAST KEEPALIVE packet back to **R**, to which **R** then responds with a RECEIVER JOIN, which sets up forwarding state as described above.

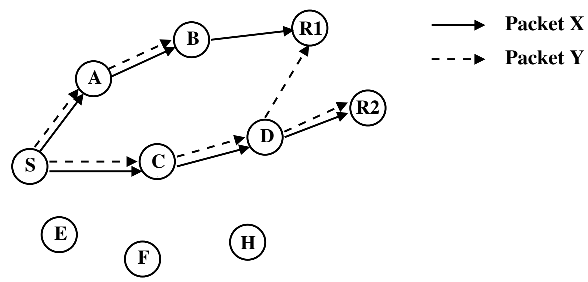

Figure 2.2: Multicast Data Packet Forwarding within the ADMR Mesh.

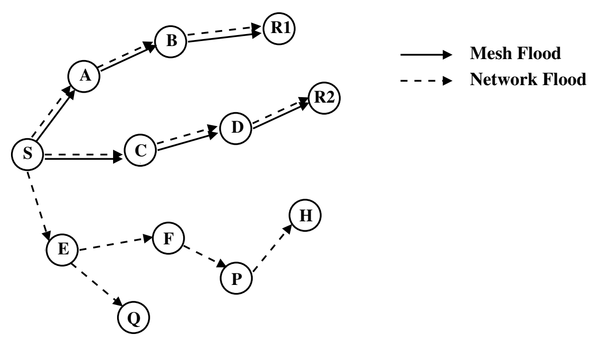

Figure 2.3: Mesh Flood vs. Network Flood.

The multicast forwarding state for a given multicast group **G** and sender **S** in ADMR forms a source-based multicast forwarding mesh. In some cases the nodes which are part of the source-based mesh represent a tree rooted at **S**, while in others, as a result of multicast state maintenance, additional nodes may become multicast forwarders, and thus the resulting set of nodes with forwarding state may no longer be a tree. In general, ADMR does not aim to create redundant state; such state may result as a side-effect of its normal operation.

Even when the set of nodes with multicast forwarding state represents a source-rooted tree, packets in ADMR are not constrained to follow any particular branches or parent/child links but instead are *flooded* through all nodes with multicast forwarding state; such a node forwards each received packet regardless of which node transmitted the packet to it. Each packet is thus dynamically forwarded from **S** to the multicast receivers along the shortest-delay paths within the mesh. For example, in Figure 2.2, packet **X** reaches receiver **R1** through node **B**, whereas packet **Y** reaches receiver **R1** through node **D**. This can happen when node **D** acquires the wireless medium before **B** and forwards the packet first, or when **B** does not receive the packet correctly due to wireless interference and is therefore unable to forward it.

I refer to the flood of a packet constrained to the nodes with multicast forwarding state as a *mesh flood*, and to the more general type of flood of a packet to all nodes in the network as a *network flood* (Figure 2.3).

This use of flooding within the multicast forwarding mesh is similar to the “forwarding group” concept introduced in the FGMP protocol [5] and also used in ODMRP [38], except that in ADMR forwarding state is specific to each sender rather than being shared by the entire group. As a result, when a sender using ADMR sends a multicast packet, it floods within the multicast forwarding mesh *only* towards the group’s receivers, whereas when using FGMP or ODMRP, the packet also floods back towards any other senders even though they may not be multicast receivers for the group. Although the source-specific forwarding requires ADMR to maintain source-specific state in forwarding nodes, such state is required anyway in order to support the source-specific multicast service model [12]. In addition, even FGMP and ODMRP require source-specific state at each node, since they must detect duplicate packets during a flood within the forwarding group, and any type of packet identifiers used for this duplicate detection when there may be multiple group senders must be source-specific.

2.1.2. Multicast State Maintenance

Multicast state maintenance in ADMR is performed on-demand; ADMR initiates mesh repair only when one or more receivers are disconnected from the mesh. The repair is only aimed at reconnecting the mesh rather than duplicating the previously existing mesh structure through a different set of nodes.

To enable efficient multicast mesh reconnection, multicast mesh repair is performed with respect to the source-rooted tree "traced" by the last forwarded multicast packet. Each node in the forwarding mesh records the node from which it received the last non-duplicate multicast packet for **S** and **G** treats this node as its *parent node* in a tree rooted at the source during mesh repair.

Each multicast packet originated by some node **S** contains a small ADMR header (Section 2.3), including a number of fields used by the protocol in forwarding the packet and for maintaining the multicast forwarding mesh for **S** and **G**. One such field is the *expected packet inter-arrival time* at which new packets should be expected from **S** for **G**. This field in the ADMR header, initialized by **S** for each originated packet, is based on dynamically tracking the inter-arrival times of packets sent by the multicast application. If the application layer at node **S** originates no new multicast packets for **G** within some multiple (e.g., 1.5) of the current expected packet inter-arrival time for **G**, the routing layer at **S** begins generating *keep-alive* packets for **G**; each keep-alive packet is multicast to the group (not flooded through the network) and is used to maintain the existing forwarding state for the multicast forwarding mesh for **S** and **G**. The inter-arrival time of the keep-alive packets is multiplied by some factor (e.g., 2) with each successive keep-alive, until reaching a maximum interval; after some further multiple of this interval, **S** is assumed to no longer be an active sender for **G**, the keep-alives are stopped, and all forwarding state for **S** and **G** is allowed to expire. The ADMR header includes the multiplicative factor increasing the time between successive keep-alives and a count of keep-alives left to send, allowing all nodes receiving any of these keep-alive packets to know when the mesh is scheduled to expire. Expiration will be canceled if the sender application begins to send new multicast data packets before the scheduled expiration time.

Absence of data packets and keep-alives within a multiple of the expected packet inter-arrival time is an indication of forwarding mesh disconnection (unless the forwarding state has been scheduled to expire within this many packets). When a forwarding node **F**, for source **S** and group **G**, does not receive data packets or keep-alives from **S** within a multiple of the expected packet inter-arrival time, it performs *local repair* to reconnect to the mesh. If the local repair procedure fails, receivers who got their packets through **F** and are now disconnected from **S**, perform a global reconnect procedure by flooding a MULTICAST SOLICITATION packet as in a new group join (Section 2.1.1).

Forwarding nodes, which initiate a local repair, do not react to failure of the local repair as they may have moved and thus may no longer be on the paths between the sender and the receivers within or near the forwarding mesh.

ADMR performs automatic pruning of branches of the multicast mesh that are no longer needed for forwarding. Pruning decisions are based on lack of passive acknowledgments from downstream, instead of relying on the receipt of an explicit prune message. Node **B** considers a transmission of a packet by **A** that was received by**A** from **B** as a (passive) acknowledgment that **A** has received the packet. Passive acknowledgments are enabled by including in each packet's header, the address of the node from which the packet was received by the node currently transmitting it.

2.1.3. Operation in Highly Mobile Networks

A large number of global repair attempts within a short period of time, may signify that the mobility in the network is too high to allow for timely multicast state setup and repair. When a source node receives a large number of global repair requests over some period of time, it may switch to flooding for a while as this may be the only good way to deliver data to the multicast receivers.

2.2. Data Structures

The multicast forwarding state for ADMR is maintained locally by each node in the following three tables:

- *Sender Table*: Logically contains one entry for each multicast group **G** for which this node is an active sender. Each entry in the Sender Table includes the current expected packet inter-arrival time for packets sent to **G** by this node, and a count of consecutive keep-alive packets sent to **G** since the last data packet sent to **G** by this node.
- *Membership Table*: Logically contains one entry for each combination of multicast group address and sender address, **G** and **S**, for which this node is either a receiver member or a forwarder. Each entry in the Membership Table includes a flag bit to indicate if this node is a receiver, a flag bit to indicate if this node is a forwarder, the current expected packet inter-arrival time of packets sent to **G** by **S**, and the current value of the keep-alive count for keep-alives left to send to **G**.
- *Node Table*: Logically contains one entry for each other node in the network from which this node has received a flooded packet. Each entry in the Node Table includes the identification number **I** from the ADMR header of the most recent mesh flooded or network flooded packet received from the node represented by this table entry, e.g., node **X**, plus a bitmap representing a number of previous identification numbers of packets sent by **X**. The identification number and bitmap are used to detect and discard duplicate packets during a flood: if the bit corresponding to some identification number in this bitmap is set, the packet is assumed to be a duplicate; all identification numbers prior to the one corresponding to the first bit in the bitmap are also assumed to be duplicates (or are of no further interest and are discarded); this use of a bitmap is similar to the data structure suggested for anti-replay protection in the IP Security protocols [21]. Each entry for sender node **X** and identification number **I**, includes the previous hop address and hop count contained in the packet with identification number **I**, which had the smallest hop count from **X**. The previous hop address is taken from the MAC-layer transmitting source address of the packet, while the hop count is stored in its ADMR header. To manage space in the Node Table, new entries should be created only as needed, and existing entries should be retained in an LRU fashion.

In addition, each node maintains a *Send Buffer* which is used by multicast source nodes to store packets during multicast state setup (Section 2.5.1).

Option Type	Option Length	Identification Number	
Hop Count	Inter−Arrival Time	Keep−alive Count	Multiplication Factor
Previous Hop Address			

Figure 2.4: ADMR header

2.3. ADMR Header

Each multicast data packet in ADMR contains an ADMR header (Figure 2.4) which includes the following fields:

- The *identification number* in the ADMR header uniquely identifies the packet and is generated as a count of all ADMR packets flooded in any way that originated from the sender of the packet. The identification number enables nodes to perform duplicate detection and forward each packet at most once.
- The *hop count* is initialized to 0 by the source of the packet and is incremented by each node forwarding the packet. This field is used to determine the distance of a node from a multicast source and is used in both multicast state setup and maintenance.
- The *previous hop address* field in a packet that is being forwarded, is the MAC-layer transmitting source address from which this packet was received; it is copied from the MAC-layer header of the packet before it is forwarded; when the packet is originated, this field is initialized to 0. This field is used for passive acknowledgments.
- The *expected packet inter-arrival time* field is computed by the source node based on the expected packet inter-arrival time of the multicast application to which this packet belongs. This information is used by multicast mesh nodes to detect and repair broken links (Section 2.6.2).
- The *keep-alive count* field contains the number of keep-alives that will be sent before the multicast application is considered inactive and the multicast mesh is expired (Section 2.7.2).
- The *multiplication factor* field contains the multiplicative factor used to increase the expected keep-alive inter-arrival time as the multicast mesh is getting ready to expire (Section 2.7.2). It is used by multicast forwarders and receivers to schedule the expiration of their forwarding state for the source and group to which the packet belongs.

2.4. Multicast Packet Forwarding

Any packet with a broadcast destination address containing an ADMR header will be sent as a network flood, i.e., it will be flooded to all nodes in the network (Section 2.1.1). Any packet with a multicast destination address containing an ADMR header will only be sent as a mesh flood, i.e., it will be flooded to all nodes with multicast forwarding state for the source and group indicated in the packet. For either type of flood, the identification number in the packet's ADMR header is recorded in the Node Table of each node that forwards it; this information is used for duplicate detection so that any node that should forward the packet as part of a flood would forward no more than one copy of it.

When a node receives a packet from node **S** with a multicast destination address **G** and an ADMR header in it, it checks its Membership Table entry for **S** and **G** to determine if it should forward the packet. Whether or not it forwards the packet, the receiving node compares the hop count in the received packet's ADMR header to the hop count in this node's Node Table entry for **S**. If the new hop count is less than that already recorded in the Node Table entry, the node may update the entry with the new hop count and set the previous hop address in the entry to the MAC-layer source address from which the packet was received. In addition, if the node forwards the packet, before doing so, it increments the hop count field in the packet's ADMR header and copies the packet's MAC-layer source address into the ADMR header previous hop address field.

Finally, if the packet has a payload following the ADMR header, the node checks its Membership Table to determine if it is a receiver member for **S** and **G**. If so, it passes the packet up within the protocol stack to allow the packet to be processed as a received multicast packet.

Since each node in the multicast source mesh forwards each packet that is mesh flooded without regard to who transmitted the packet to it, each packet will flow within the mesh to the group receivers without being constrained to follow any specific links, and will thus be able to automatically be forwarded around temporarily broken links or failed forwarding nodes in the mesh.

2.5. Multicast State Setup

Multicast state setup for a multicast group and source in ADMR is initiated by both the sources and the receivers for the group, as each such node begins to send packets to the group, or joins the group respectively.

2.5.1. New Multicast Source

When a node **S** originates a multicast packet for some group **G** for which it is not currently an active sender, it will not have a Sender Table entry for **G**. In this case, node **S** creates and initializes a new Sender Table entry for **G**. The expected packet inter-arrival time in this entry may be set to a default value, may be assumed based on the port numbers used in the packet, or may be specified by the sending application if an API is available for this purpose. Node **S** also inserts an ADMR header in the packet, includes a multicast group address option with the address of the multicast group in it, and sets the IP destination address of the packet to the network broadcast address, thus causing the packet to be sent as a network flood (Section 2.4).

After flooding the first data packet, node **S** buffers for a short time (e.g.,INITIAL_PKT_BUFFER_TIME) subsequent multicast data packets that it might originate for group **G**, rather than sending them immediately as they are generated. This initial delay allows forwarding state to be set up and ensures that network resources will not be wasted if there are no receivers for **G** in the network.

Once **S** receives at least one RECEIVER JOIN packet from a receiver for the multicast group (Sections 2.5.2 and 2.5.3), and INITIAL_PKT_BUFFER_TIME has elapsed, **S** sends all packets for **G** that are in its Send Buffer as normal multicast packets. Pacing techniques could optionally be applied in sending these buffered packets, in order to avoid congestion caused by sending them all at once as soon as allowed by the protocol. The packet exchange which takes place when a new source becomes active is depicted in Figure 2.5.

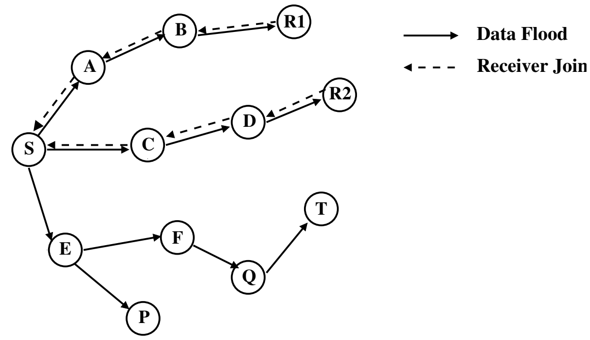

Figure 2.5: New Multicast Source. Node **S** sends a data flood to which multicast receivers reply with a RECEIVER JOIN.

Most subsequent multicast packets for **G** from **S** will be flooded within the multicast mesh for **S** and **G**, rather than being network flooded. However, it is possible that some interested receivers did not receive the initial data flood from **S** due to packet loss or a network partition at the time of the flood. To allow for such occurrences, node **S** uses a network flood rather than a mesh flood for certain of its subsequent multicast packets. The time between each packet selected to be sent as a network flood is increased until reaching a slow background rate, designed to tolerate factors such as intermittent wireless interference, or temporary partition of the ad hoc network. For example, in my simulations of ADMR (Chapter 3), the first data packet sent 5 seconds after the initial network flooded data packet, is sent as a network flood; the first data packet sent 10 additional seconds later is also sent as a network flood, as is one data packet every 30 seconds thereafter. These network floods are sent only when there is data to be sent to **G** and are delayed from their scheduled time until a new multicast data packet is originated from the application layer on **S**. The interval until each subsequent network flooded data packet is relative to the time at which the previous network flooded data was sent, in order to preserve the approximate spacing between the floods.

2.5.2. Multicast Application Join

When an application on some node **R** requests to join a group **G**, the ADMR routing layer on node **R** sends a MULTICAST SOLICITATION packet as a network flood (Figure 2.6). This packet includes the group and source (for source-specific joins) of the group **R** is interested in joining and is intended to announce to sources for **G** that **R** is interested in receiving multicast data from them. If the group is a source-specific multicast group, the specific sender address **S** requested by the application is included as an additional option after the ADMR header.

The MULTICAST SOLICITATION is forwarded by all nodes in the network as described in Section 2.4, except that in the case of source-specific multicast, the specified source **S** does not forward the MULTICAST SOLICITATION packet. Also in this case, if a node receiving the MULTICAST SOLICITATION has a Node Table entry for **S**, and has a Membership Table entry for **S** and **G**, indicating that it is a forwarder, this node will not rebroadcast the packet but will *unicast* it only to the previous hop address indicated in its Node Table entry for **S**; the node which receives this unicast packet will repeat the procedure and the packet will thus reach the multicast source through reverse forwarding within the multicast mesh (Figure 2.7). This mechanism speeds up the receiver join and potentially reduces overhead.

When any source **S** for multicast group **G** receives a MULTICAST SOLICITATION packet, it replies to the MULTICAST SOLICITATION to advertise to **R** its existence as a sender for the group.

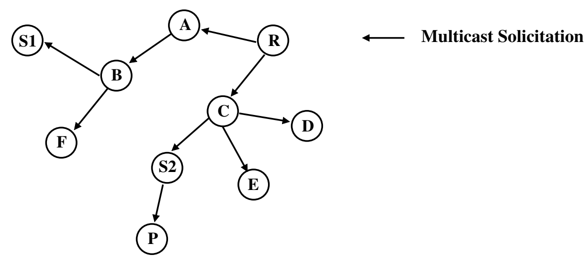

Figure 2.6: Receiver **R** floods a MULTICAST SOLICITATION.

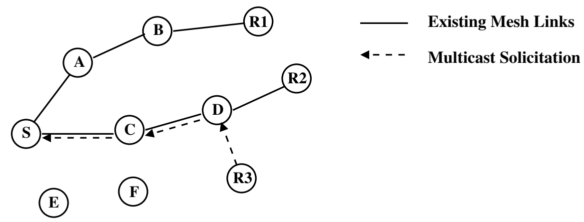

Figure 2.7: Node **D** receives a source-specific MULTICAST SOLICITATION and unicasts it on to the source.

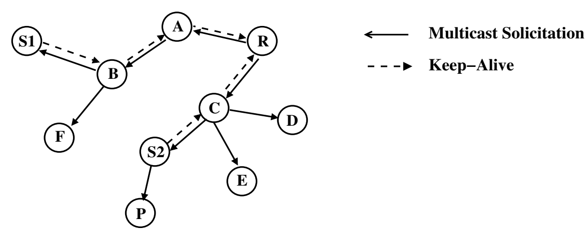

Figure 2.8: Sources **S1** and **S2** respond to receiver **R**'s MULTICAST SOLICITATION.

This reply may take one of two forms. If the next scheduled network flood of a multicast data packet (Section 2.5.1) is to occur soon, **S** may choose to advance the time for this network flood and use it as a reply to **R**'s solicitation. This form of reply is appropriate, for example, when many new receivers attempt to join the group at about the same time, since **S** would then receive a MULTICAST SOLICITATION from each of them, but could use the single existing network flood of the next data packet to reply to all of them. The other form that this reply may take is for **S** to send an ADMR keep-alive packet, unicast to **R**. This keep-alive is a recently sent multicast data packet or a packet with an ADMR header and no payload, and follows the reverse of the path taken by **R**'s MULTICAST SOLICITATION. Each node forwarding this unicast keep-alive packet unicasts it to the address recorded in the previous hop address field in its Node Table entry for **R**, created when the node forwarded **R**'s MULTICAST SOLICITATION flood (Figure 2.8). When forwarding the unicast keep-alive packet toward **R**, each node updates its Node Table entry for **S** in the same way as it would for a flood from **S**, recording the node on the path back to **S** in the entry's previous hop address field.

If node **S** replies to the MULTICAST SOLICITATION from **R** by sending a unicast keep-alive, as described above, then **S** also sets a timer and expects to receive a RECEIVER JOIN from **R** within a short time. If **S** does not receive the RECEIVER JOIN (Figure 2.9) within some time, then the unicast

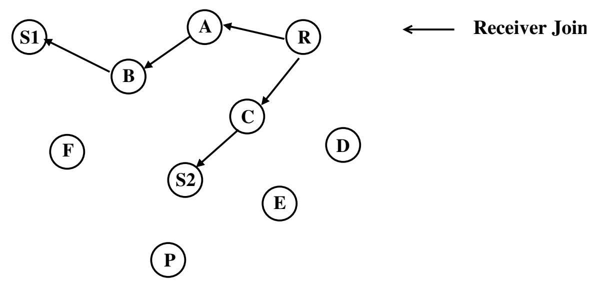

Figure 2.9: Receiver **R** unicasts a RECEIVER JOIN to each source from which it received a keep-alive.

keep-alive was probably lost before reaching **R**, and so **S** will retransmit it. If the timer expires a second time and **S** has not received a RECEIVER JOIN from **R**, then **S** assumes that the path the unicast keep-alive is trying to traverse is broken, and advances its next scheduled network flood of a multicast data packet.

2.5.3. ADMR Packet Processing at Multicast Receivers

When a node **R** receives any ADMR data packet, in addition to forwarding the packet if required by the forwarding procedure described in Section 2.4, node **R** also checks the entry for **S** and **G** in its Membership Table to determine if it is a receiver member for **S** and **G**. If so, then **R** processes the packet as a multicast packet intended for it, passing it up to the next layer within its receiving protocol stack.

In addition, as part of processing a received multicast packet, if the packet was sent as a mesh flood (rather than as a network flood), then this indicates that the receiver node **R** is currently connected to the multicast forwarding mesh for **S** and **G**. The node considers itself to remain connected until detecting that it has become disconnected, as described in Section 2.6.3.

If instead, the received packet was sent as a network flood or the packet is a unicast keep-alive (Section 2.5.2) and if the receiver **R** is not currently connected to the multicast forwarding mesh for **S** and **G**, then **R** replies with a RECEIVER JOIN packet, which causes the necessary nodes along the path back to **S** to become forwarders. Node **R** initializes the expected packet inter-arrival time field in its RECEIVER JOIN packet with the expected packet inter-arrival time from the ADMR header of the received multicast packet. The RECEIVER JOIN then follows the path established by the forwarding of the received data packet flood or unicast keep-alive, as recorded in the previous hop address field in each node's Node Table entry for **S**. Each node that forwards the RECEIVER JOIN creates a Membership Table entry for **S** and **G**, if it does not already have one, and sets the flag in this entry to indicate that it is a forwarder for **S** and **G**. Each node also records in the Membership Table entry the expected packet inter-arrival time from the RECEIVER JOIN packet's header.

If there are multiple new receivers for group **G** near each other in the network, many RECEIVER JOIN packets will traverse the same paths or links on their way to source **S**. However, in order to make each node along these paths a forwarder for **G** and **S**, it is enough for one RECEIVER JOIN packet to be forwarded by each such node. It would thus be possible to filter all but the first of these multiple RECEIVER JOIN packets received by each of these nodes. However, such aggressive filtering would leave the connection of the new receivers to the multicast forwarding mesh susceptible to the loss of the single RECEIVER JOIN packet that was forwarded. To reduce overhead and yet provide resilience to such packet loss, each node will forward at most several RECEIVER JOIN packets, e.g., 3, for the last identification number it has recorded in its Node Table entry for **S** and **G**. To implement this filtering, when sending a RECEIVER JOIN packet, the receiver **R** copies the identification number from the flooded packet received from **S** into its RECEIVER JOIN in order to identify the source data flood in response to which the JOIN is sent, and each node maintains in its Node Table entry a count of RECEIVER JOIN packets forwarded for the identification number in that entry.

To further deal with the possibility of loss of a RECEIVER JOIN packet, each new receiver, after sending its RECEIVER JOIN, sets a timer to a multiple of the expected packet inter-arrival time contained in the ADMR header of the packet that triggered the RECEIVER JOIN; e,g, 3 times the expected packet inter-arrival time. If this timer expires before any new multicast packets (data or keep-alives) have been received from **S**, node **R** resends its RECEIVER JOIN and resets the timer. If this timer expires again with no new multicast packets received from the source, the receiver sends a new MULTICAST SOLICITATION, as described in Section 2.5.2. The first copy of the RECEIVER JOIN packet may have been lost along the path to the source, thus creating forwarding and filtering state along the part of the path that it traversed, and causing retransmitted RECEIVER JOINs to be dropped. To prevent such retransmitted packets from being filtered, ADMR disables the filtering of retransmitted joins by setting a *no filter* flag in the header of retransmitted JOINs.

The 3-way end-to-end exchange of control packets between a source and a receiver that occurs during multicast state setup and involves the MULTICAST SOLICITATION, unicast keep-alive and RECEIVER JOIN packets, is designed to serve as a handshake between them, ensuring that in the event of a control packet loss prior to multicast state establishment, at least one of them will continue to make attempts to reach the other. In general, flooded packets have a higher chance of reception than unicast packets and as a result, only unicast control packets are retransmitted and if multicast state is not established following such retransmissions, ADMR uses flooded packets (e.g., a data flood or MULTICAST SOLICITATION flood). Using this kind of cooperative state setup enables the protocol to increase the reliability of the state setup process, and to increase its aggressiveness only when it determines that a multicast session is possible, i.e., that there are active sources and receivers in the network.

2.6. Multicast State Maintenance

Forwarders or receiver members for source **S** and group **G** may become disconnected from the multicast forwarding mesh for **S** and **G** as nodes in the network move or as wireless propagation conditions change. ADMR performs state maintenance for each source's multicast forwarding mesh by monitoring the flow of traffic from the source to the multicast receivers, and detecting when as a result of a broken link there is no path from the source to a receiver(s). Once such a link is discovered, ADMR performs repair procedures in order to restore the flow of data to the multicast receivers.

2.6.1. Mesh Disconnection Detection

ADMR tracks the inter-arrival time of multicast packets sent by the application on node **S** to group **G** and computes an expected inter-arrival time for the application's packets. The expected packet inter-arrival time may be set to a default value, may be assumed based on the port numbers used in the packet, or may be specified by the sending application if an API is available for this purpose.

If the application layer at node **S** originates no new multicast packets for **G** within some multiple (e.g., 1.5) of this current expected packet inter-arrival time, the routing layer at **S** begins originating "keep-alive" packets for **G**; a keep-alive packet is a recently transmitted data packet or a packet with an ADMR header and no payload. These keep-alives are multicast to **G** just as any regular data packet and are used to maintain the existing forwarding state for the multicast forwarding mesh for **S** and **G**. They are only sent for a limited period of time after which the multicast application is assumed inactive and all multicast state for **S** and **G** in the network is expired (Section 2.7.2).

Each forwarder or receiver for source **S** and group **G** detects that it has become disconnected from the multicast forwarding mesh when it fails to receive a number of successive expected multicast data (or keep-alive) packets (e.g., 3) from **S** for **G**.

Each node maintains a *disconnection timer* for each group and source for which it is either a forwarder or a receiver member, and resets this timer each time it receives a data or keep-alive packet for **G**. The timer value is based on the expected packet inter-arrival time value in the ADMR header of the last received packet, plus a delay proportional to the node's hop count from the source **S**, as recorded in the last packet received from **S** (which updated the node's Node Table entry for **S**). This small delay is intended to generally allow the node directly downstream of the broken link to detect that the link is broken before nodes further downstream from **S** react to the lack of data and keep-alives and initiate repair procedures. Even if multiple nodes initiate repair procedures for a link that is not adjacent to all of them, the protocol would still operate correctly, but would generate more overhead packets than necessary to reconnect the forwarding mesh.

2.6.2. Local Repair

Although each multicast packet is forwarded within the forwarding mesh without regard to how previous packets were forwarded, as described in Section 2.4, the paths actually taken by each packet delivered to a multicast receiver, form a tree rooted at the multicast source. The tree "traced" by the last forwarded multicast packet is the one that ADMR will attempt to repair. Repairing this tree ensures connectivity between the multicast source and the receivers; attempting to restore the forwarding mesh is not necessary, and would generate unnecessary overhead.

When some node **C** detects disconnection from the forwarding mesh for source **S** and group **G**, it initiates local repair. This repair starts by node **C** attempting to ascertain that it is indeed disconnected from the source, and by notifying nodes downstream from it (i.e., closer to the receivers) that it is attempting to reconnect the mesh, so that no redundant local repair attempts would take place. In particular, node **C** sends a REPAIR NOTIFICATION packet, which is forwarded towards the multicast receivers along the paths followed by the last forwarded multicast data or keep-alive packet (Figure 2.10). This packet is intended to reach all nodes that received the last forwarded multicast packet through **C** and who will thus possibly detect the disconnection themselves soon. These nodes will detect disconnection later than **C** does, since their disconnection timeouts are computed based on their higher hop counts from **S**.

To forward the REPAIR NOTIFICATION packet to the nodes in the subtree below **C**, each node processes the packet only if the MAC-layer transmitting source address of the packet matches the previous hop address stored in that node's Node Table entry for the multicast sender **S**.

After sending the REPAIR NOTIFICATION packet, node **C** waits for a START_REPAIR_DELAY period of time before proceeding with its local repair. If, during this delay, node **C** receives a REPAIR NOTIFICATION initiated by an upstream node for this same group and source, then **C** cancels its own local repair, since the repair should be performed by the node that is adjacent to the broken link, and **C** clearly is not.

The REPAIR NOTIFICATION packet serves two purposes. It is a notification to nodes in the subtree below **C** that a local repair is in progress and that they should not initiate their own local repair. It is also a chance to double-check that the link to node **C**'s parent node is indeed the one that is broken. The REPAIR NOTIFICATION will be received by nodes directly below **C** in the forwarding tree traced by the last forwarded multicast packet, and if the link from **C** to its parent **B** in the tree is actually not broken, may also be received by **B**. In the REPAIR NOTIFICATION packet, **C** lists the address of the node that is currently its parent, as represented by the previous hop address in its Node Table entry for **S**. If the REPAIR NOTIFICATION is received by this parent node, it will recognize that one of the nodes directly below it in the tree (node **C**) is performing a local repair and will send a one-hop REPAIR NOTIFICATION to **C**, causing it to cancel its local repair as described above.

When a receiver member for **G** receives a REPAIR NOTIFICATION, or when, it initiates local repair by sending a REPAIR NOTIFICATION, it postpones its disconnection timer for a LOCAL_REPAIR_DURATION interval of time, which is an estimate of the amount of time the local repair is expected to take. If this timer expires and the receiver has not started to receive keep-alives or data packets, it will re-join the group as described in Section 2.6.3.

After START_REPAIR_DELAY time has elapsed, if the repair node **C** has not received a REPAIR NOTIFICATION initiated by an upstream node for **S** and **G**, it will send a hop-limited RECONNECT packet as a limited network flood (Figure 2.11). The hop limit for the RECONNECT packet (e.g., 3) limits this flood to only reaching nodes near **C**. The RECONNECT packet also includes the hop count from **S** to **C** recorded in **C**'s node table entry, so that the RECONNECT can be processed specially by multicast forwarding nodes upstream of the broken link, which are connected to the part of the mesh still connected to **S**. A forwarding node for **S** and **G**, e.g., node **F**, which has a smaller hop count to **S** than the one contained in the RECONNECT packet assumes that it is upstream of the repair node **C** and that it is therefore part of the mesh still connected to source **S**. Rather than forwarding the RECONNECT packet as part of the hop-limited network flood, node **F** reinitializes the packet's hop count limit to the default TTL (time-to-live) value, e.g., 255, and unicasts the packet to the node listed as its parent in the previous hop address field in its Node Table entry for **S** and **G** (node **A** in Figure 2.11). This copy of the RECONNECT is no longer treated as a network flooded packet, but instead is forwarded by each node that receives it to its parent in the mesh in the same way, until reaching **S**. If node **F** is in fact not upstream from the repair node **C** due to inconsistent hop count information due to node mobility, the unicast RECONNECT will reach **C**, which will discard it. Instead, if node **F** is upstream of **C**, the RECONNECT would reach **S**, which will respond with a RECONNECT REPLY packet. This RECONNECT REPLY packet is unicast back to the repair node **C** along the path traversed by the RECONNECT, recorded in the Node Table entries for **C** in the nodes along that path (Figure 2.12). Each node that forwards the RECONNECT REPLY packet joins the mesh for **S** and **G**, recording that it is a forwarder for **S** and **G** in its Membership Table.

The RECONNECT packet is forwarded all the way to the multicast source in order to ensure proper reconnection. Due to packet loss and node movement, it is not possible to always have accurate hop count information at all forwarding nodes. This may lead a disconnected node to reconnect to the multicast mesh through a node that is in the subtree below it and is therefore in the part of the forwarding mesh that is disconnected from the source.

2.6.3. Global Repair

If the local repair procedure described in Section 2.6.2 succeeds, the multicast forwarding mesh will be reconnected and the receiver members will start to receive multicast packets. If LOCAL_REPAIR_DURATION time elapses and the receivers have not received any multicast packets from **S** for **G**, this is an indication that the local repair has probably failed, perhaps because the

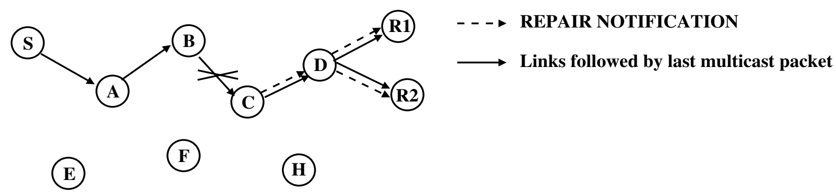

Figure 2.10: Node **C** is downstream of the broken link (between itself and node **B**) and initiates local repair by sending a REPAIR NOTIFICATION to the multicast receivers downstream.

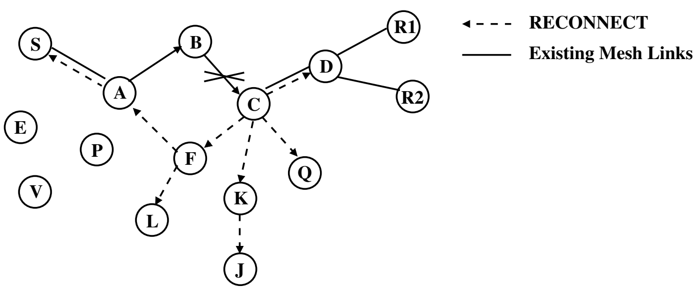

Figure 2.11: Node **C** continues the repair by sending a RECONNECT packet in an attempt to reconnect the submesh downstream to the part of the mesh still connected to the multicast source.

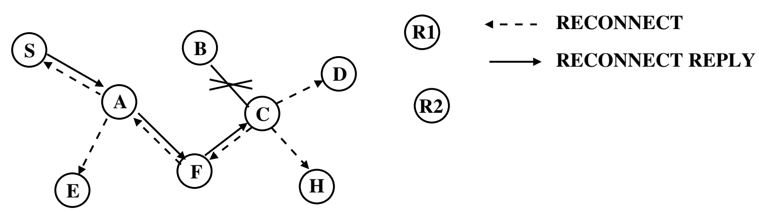

Figure 2.12: Source **S** sends a RECONNECT REPLY to node **C** who initiated the local repair.

amount of mobility in the network has been too high to allow the type of hop-limited repair attempted. In this case, each receiver performs its own individual repair by (re)joining the group and source (Section 2.5.2).

2.7. Multicast State Expiration

There are a number of cases in which multicast state should be expired since not all multicast state that was created would need to be used, or since some multicast state which was previously in use may no longer be needed.

2.7.1. Mesh Pruning

Each forwarding node in the multicast forwarding mesh for group **G** and source **S** automatically expires its own multicast state and leaves the mesh when it determines that it is no longer necessary for multicast forwarding. Similarly, source **S** automatically expires its multicast state and stops transmitting multicast data packets when it determines that there are no downstream receiver members for the group; the sender continues to send certain of its subsequent multicast data packets as infrequent background network floods, but otherwise defers sending other multicast packets for **G** until receiving at least one new RECEIVER JOIN for **G**, as described in Section 2.5.1. This mechanism helps to prune nodes from the forwarding mesh that are no longer needed because a downstream receiver has left or crashed, or because, as a result of a disconnection and an ensuing repair, some forwarding state may no longer be necessary.

The decision to expire multicast forwarding state at a node is based on whether the multicast packets that it originates or forwards are subsequently forwarded by other nodes. In particular, for each multicast packet, a node **C** expects to hear at least one (passive or active) acknowledgment. Active acknowledgments are sent by multicast receivers that are not also forwarders. If node **C** overhears **B** transmitting this packet, with itself listed in the previous hop field, node **C** considers this as confirmation that it should continue forwarding subsequent multicast packets, so that nodes such as **B** can continue to receive them. On the other hand, if **C** fails to receive such confirmation for a number of consecutive multicast packets that it sends, it assumes that it is no longer necessary for it to remain in the multicast forwarding mesh for **S** and **G**. A multicast source stops sending data when it determines that no receiver members or forwarders remain, with the exception of the infrequent data floods aimed at healing possible partitions, sent only as long as the multicast application remains active.

2.7.2. Expiration due to Inactive Multicast Application

A multicast application may become inactive for some period of time. To conserve network resources, ADMR expires multicast state for such applications when the expected time of inactivity becomes sufficiently large.

When a multicast application does not send any data packets for some multiple of its expected packet inter-arrival time, ADMR begins to generate keep-alive packets at increasing inter-arrival times in order to enable maintenance of the multicast forwarding state (Section 2.6). When the application does not send any packets in significant violation of its sending pattern, **S** is assumed to no longer be an active sender for **G**, the keep-alives are stopped, and all forwarding state for **S** and **G** in the network is allowed to expire. The ADMR header includes the multiplicative factor increasing the time between successive keep-alives and a count of keep-alives left to be sent since the last real multicast data packet from the application, allowing all nodes receiving any of these keep-alive packets to know when the mesh is scheduled to expire. Expiration is canceled when a new data packet is sent by the source application.

2.7.3. Expiration of Temporary State

Temporary forwarding state for source **S** and group **G** is created at nodes that forward a RECEIVER JOIN packet during multicast state setup (Section 2.5). This state becomes permanent only when a node forwards a multicast data or keep-alive packet for **S** and **G**. In the event that no multicast packets are received for a while, after the state is created, it is expired, since an alternate forwarding path may have been established. The reason forwarding state may be set up but may not ever get used is that if the RECEIVER JOIN packet is sent along a route that has a broken link it it (which broke after the data flood or unicast keep-alive traversed it), the RECEIVER JOIN will only set up forwarding state along part of that path. However, since the packet cannot be forwarded all the way to the source, after several retransmissions, the multicast receiver that generated it will attempt to find a new path, which would leave unneeded state along the old one.

2.8. Optimizations

Each receiver keeps track of how many times it has had to perform a global repair for a group over some period of time as a result of mesh disconnection (Section 2.6.2). When this number reaches a threshold over some period of time, the receiver sets the HIGH_MOBILITY flag in the ADMR header of the next RECEIVER JOIN packet it sends, to indicate that it has been getting disconnected frequently recently. When the source receives some number of RECEIVER JOINs with this flag set, it may switch to flooding mode in which every multicast packet is flooded. The high number of re-joins indicate that multicast state setup cannot keep up with the high mobility in the network and only flooding can achieve successful packet delivery. After flooding for some period of time, the protocol reverts back to its normal mode of operation as mobility in the network may have decreased.

2.9. Summary

In this chapter, I presented the design of the Adaptive Demand-Driven Multicast Routing protocol (ADMR) for multi-hop mobile ad hoc networks. ADMR supports both the source-specific and traditional multicast models and is the first general-purpose routing protocol for ad hoc networks that does not employ periodic control packet mechanisms of any kind. It scales its overhead based on current communication demands and network conditions and operates efficiently by adapting its operation to multicast application sending patterns. ADMR is completely distributed and does not rely on any centralized coordination or control, and does not expect reliable or in-order delivery of its control or data packets for correct operation. Finally, ADMR is designed to work independently of the unicast protocol used in the ad hoc network and can thus work with any unicast protocol or even without a unicast protocol. ADMR's performance is evaluated in Chapter 3, and compared to that of MAODV and ODMRP.

Chapter 3

Multicast Performance Analysis

In this chapter, I present an extensive performance evaluation of ADMR (Chapter 2) compared against the Multicast Ad Hoc On-Demand Distance-Vector protocol (MAODV) [37] and the On-Demand Multicast Routing Protocol (ODMRP) [38].

3.1. Motivation and Contributions

I conducted an extensive simulation evaluation of ADMR and compared its performance to that of two other routing protocols for ad hoc networks that utilize on-demand mechanisms, namely, MAODV and ODMRP. I chose to compare ADMR against MAODV and ODMRP because they have been well-documented and have been shown to perform well. In addition, all of these protocols contain a significant on-demand (reactive) component, but they differ in how reactive and proactive mechanisms are combined to make the complete protocol: ADMR uses source-based trees and does not utilize any periodic control packet transmissions, MAODV uses a shared group tree and uses periodic Hello messages for link break detection and periodic group leader floods for group information dissemination, and ODMRP uses a group forwarding mesh for packet forwarding and utilizes periodic flood-response cycles for multicast state creation and maintenance.

In this performance evaluation, I use mobile networks composed of 100 and 200 nodes, with both a single active multicast group, and multiple active multicast groups in the network. I study a wide range of multicast scenarios representative of a variety of multicast applications, such as conferencing and single-source vs. multi-source groups. Although some simulation results for these protocols have been published before, the three protocols have not been compared, and prior studies have focused on smaller networks using a small set of simulation scenarios, many with only a single active multicast group. The focus here is on the effects of the protocols' relative degree of on-demand behavior and their performance in different multicast scenarios. My experiments show that ADMR performs well across all simulated scenarios and compared to MAODV and ODMRP, and typically generates 3 to 5 times less packet overhead.

3.2. Related Work

A number of evaluations of ADMR, MAODV and ODMRP have been presented in the literature, all using ad hoc networks composed of 50 or fewer nodes [4, 9, 15, 23–25, 37]. The only exception is the work of Kang et al. [20] who present the Scalable Multi-source Multicast Routing Protocol (SMMRP) and compare its performance to that of ODMRP in a network of 150 nodes. Their performance evaluation does not include ADMR and MAODV, and is only based on simulations of one multicast scenario: 1 group with 10 sources and 10 receivers. In contrast, in this work, I present an extensive evaluation of ADMR, MAODV and ODMRP, in networks with 100 and 200 nodes, in a wide range of multicast scenarios.

Lee et al. [25] compare the performance of ODMRP, CAMP [9], AMRoute [2] and Amris [42]. Their simulations are based on 50-node networks with a variable number of multicast senders and receivers, all part of a single multicast group. Lee et al. [24] propose a new multicast protocol called Neighbor-Supporting Multicast Protocol (NSMP), and compare its performance to that of ODMRP in a 50-node network with a single active multicast group as well. Similarly, Royer et al. [37] present an evaluation of MAODV in a 50-node network with only one multicast group, and Kunz et al. [23] compare the performance of MAODV and ODMRP in a 50-node network also in scenarios with a single active multicast group. Garcia-Luna-Aceves et al. [9] compare the performance of CAMP to that of ODMRP in 30-node networks with 1 or 2 multicast sources, where all nodes in the network are receivers.

Bunchua et al. [4] compare the performance of ABAM [41] to that of ODMRP in a 40-node network, and in my work, I have compared the performance of ADMR [15] to that of ODMRP in a 50-node network. Both of these studies study the protocols' behavior in networks with both a single as well as multiple active multicast groups in the network but in a limited set of multicast scenarios.

In this work, I perform a comparative performance analysis of ADMR, MAODV and ODMRP, in ad hoc networks with 100 and 200 nodes, in a wide range of multicast scenarios, including multiple groups with multiple sources each.

3.3. Protocol Descriptions

In this section, I briefly overview the operation of the Multicast Ad Hoc On-Demand Distance Vector protocol (MAODV) [37] and the On-Demand Multicast Routing Protocol (ODMRP) [38].

3.3.1. MAODV

MAODV builds a group tree, shared by all sources and receivers for a group. This enables it to localize group joins and connection of newly active sources to the multicast tree as well as repairs when the tree becomes disconnected. The use of a shared tree and the localized connection and reconnection to the tree result in longer forwarding paths for data packets. Such paths have a higher likelihood of packet loss due to collisions, and higher end-to-end delay; they are also more likely to break which also leads to packet loss and a more frequent invocation of the route repair mechanisms within the protocol. MAODV requires the use of periodic neighbor detection packets for detection of broken links, and periodic group leader control packet floods (e.g., every 5s) for disseminating a multicast group's sequence number.

MAODV creates a shared tree between the multicast sources and receivers for a multicast group. The root of each group tree is a multicast source or receiver for the group that has been designated as a group leader. When an application on a node **R** issues a join request for a multicast group **G**, the MAODV routing layer at **R** floods the network with a ROUTE REQUEST packet. If no response is received, the flood is repeated. If no response is received to several such floods, node **R** becomes the group leader for **G**. When a new source wants to send packets to a group, it performs the same steps.

The group leader periodically floods the network, e.g., every 5 seconds, with a GROUP HELLO packet to inform network nodes of the existence of group **G** and of **G**'s current sequence number. Nodes that wish to join group **G** or want to send packets to **G**, and have recently received a

GROUP HELLO packet from its leader, unicast a ROUTE REQUEST packet to the group leader, rather than flooding it. Once the leader receives a ROUTE REQUEST, or any node on the shared group tree receives a flooded ROUTE REQUEST, it unicasts a ROUTE REPLY packet back to the originator of the REQUEST, which responds with a MULTICAST ACTIVATION packet. The MULTICAST ACTIVATION packet sets up multicast forwarding state between the newly joined receiver and the shared tree.

Each MAODV multicast tree node keeps a list of its upstream and downstream neighbors in the shared tree. Each data packet is forwarded to all nodes on this list except the node from which it was received. The packet is forwarded as either a unicast to each such neighbor, or as a broadcast, when it needs to be forwarded on to multiple nodes.

Broken links are detected with the help of periodic HELLO packets broadcast by each node in the network. In addition, any packet overheard from a node, can serve as an indication that the link to that node is operational. When a node does not receive any packets from another node within several HELLO interval times, the link between the two nodes is assumed to be broken, and the node uses expanding ring search flooding to reconnect to the shared tree.

MAODV also employs mechanisms for network partition detection and partition healing, which include group leader selection, to ensure that there is only one group leader in each network partition [37].

3.3.2. ODMRP

Unlike MAODV, which attempts to ensure that each multicast receiver is connected to the multicast tree by a single path, ODMRP builds a *group-based forwarding mesh*, in which multiple paths may exist between the sources and the receivers.

Each source performs periodic flood-response cycles, which create multicast forwarding state regardless of existing forwarding state. The frequent state discovery enables the protocol to discover the current shortest paths between each source and the multicast receivers and improves the robustness of the protocol because multiple forwarding paths may be created between the members of the group. This is also why ODMRP's packet delivery ability improves as the number of sources and receivers per multicast group increases and sometimes with increased mobility: the redundant forwarding state improves ODMRP's packet delivery ability because it serves as a form of forward error correction, and makes the protocol less susceptible to mesh disconnection due to broken links. However, the frequent discovery floods and high number of data transmissions significantly increase network load.

Each multicast source for a group **G** in ODMRP periodically floods the network with a JOIN QUERY packet which is forwarded by all nodes in the network. This packet is sent every REFRESH_INTERVAL, e.g., every 3 seconds. Each multicast receiver responds to this flood by sending a JOIN REPLY packet which is forwarded along the shortest path back to the multicast source that originated the QUERY. Before forwarding this packet, each node waits for JOIN_AGGREGATION_TIMEOUT, and combines all JOIN REPLYs for the group received during this time into one JOIN REPLY. Each node that forwards the REPLY packet creates (or refreshes) forwarding state for group **G**.

Each node with forwarding state for **G** forwards each data packet sent by a multicast source for **G** that it receives. A data packet thus follows the shortest paths to the multicast receivers within the forwarding mesh, though is also forwarded towards other sources for the group who may not be group members. Forwarding state is expired after a multiple of the periodic flooding interval to ensure that in the event that some number of forwarding nodes' multicast state is not refreshed due to packet loss, the forwarding state created from a previous flood would still be valid. This mechanism improves the robustness of the protocol, but may cause multiple overlapping trees to be active in the network simultaneously, each created during a subsequent JOIN QUERY flood [15].

3.4. Methodology

In this section, I describe the simulation setup, scenarios, and performance metrics used in the evaluation of ADMR, MAODV and ODMRP.

3.4.1. Simulation Environment

The simulation setup used in this evaluation is consistent with that commonly used in ad hoc network routing protocol performance studies.

All simulations were conducted using the ns-2 [7] discrete event packet-level simulator with Monarch wireless extensions [3], which include implementations of models of signal strength, radio propagation, wireless medium contention, capture effect, and node mobility. The radio model is based on the Lucent Technologies WaveLAN 802.11 product [13], which provides a 2 Mbps transmission rate and a nominal transmission range of 250m.

To simulate the three protocols, I implemented each of them in ns-2 and implemented a framework for supporting ad hoc network multicast into the simulator.

The values of the main parameters of each protocol are listed in tables 3.2, 3.3, and 3.4, and in the case of ODMRP and MAODV, have been set as suggested by their designers in published work, or through personal communication.

3.4.2. Mobility Scenarios

The experiments include networks with 100 nodes placed on a site with dimensions 1200$\times$800 meters, and networks with 200 nodes on a site with dimensions 1720$\times$1120 meters. The node density and ratio of dimensions of each site are nearly identical between the two sites to enable comparison.

Nodes in each scenario move according to the random waypoint model [3], in which each node independently picks a random destination and speed from an interval $(0, Max\_Speed)$ and moves towards the chosen destination at the selected speed. When the node reaches the destination, it stops for *pause time* seconds and then repeats the process. The pause time in all experiments is 0, i.e., nodes move continuously; the maximum speed is 20 m/s or 1 m/s.

Each simulation is run for 900 seconds. Ten randomly generated scenarios are run for each parameter combination, and each point in the graphs is the average of the results of these ten scenarios.

The characteristics of the movement scenarios are shown in Table 3.1, where each value is an average over the 10 scenarios per parameter combination. The *average node degree* is the number of nodes that are within direct transmission range of a node, and is averaged over all nodes and over the duration of the simulation. The *average shortest-path length* is the average path length computed over all shortest paths between all pairs of nodes, over the duration of the simulation. The *maximum shortest-path length* is the length of the longest shortest-path between a pair of nodes encountered during the simulation. The *number of link changes per second* is the number of times a link is formed or breaks divided by the length of the simulation. The *number of shortest-path changes per second* is the average number of times that a shortest route between two nodes breaks or a shorter path becomes available.

The notable differences in the values of the metrics between scenarios with different levels of mobility are in the number of link changes per second and the number of shortest-path changes per

Table 3.1: Scenario Characteristics

Metric	100 nodes, 20 m/s	100 nodes, 1 m/s	200 nodes, 20 m/s
Average Node Degree	23.24	22.08	23.40
Average Shortest-Path Length	2.34	2.41	3.32
Maximum Shortest-Path Length	7.5	7	11
# Link Changes Per Sec.	43.1	3.41	96
# Shortest-Path Changes Per Sec.	496	43	3130

Table 3.2: ADMR Parameter Settings

Parameter Name	Value
MAX_RECEIVER_JOIN_FWDS_PER_FLOOD	3
DEFAULT_EXPECTED_PKT_INTERARRIVAL_TIME	0.2
DEFAULT_KEEPALIVE_COUNT	16
DEFAULT_MULTIPLICATION_FACTOR	1
BEGIN_LOCAL_REPAIR_DELAY	0.2
LOCAL_REPAIR_TTL	2
NUM_MISSING_PKTS_TRIGGER_DISCONNECTION	3
NUM_MISSING_PKTS_TRIGGER_EXPIRATION	10
TEMPORARY_STATE_EXPIRATION	3 sec.

Table 3.3: MAODV Parameter Settings

Parameter Name	Value
HELLO_INTERVAL	1 sec.
ALLOWED_HELLO_LOSS	3
RREQ_RETRIES	2
GROUP_HELLO_INTERVAL	5 sec.
ACTIVE_ROUTE_TIMEOUT	6 sec.
RREP_WAIT_TIME	2 sec.
TTL_INCREMENT	2
TTL_THRESHOLD	7

second. The average node degree is similar at different speeds, because the speed of node movement in the random waypoint model, does not affect node distribution and network density at pause time of 0 as shown in [36]. Higher speeds produce a higher number of link and shortest-path changes with the following relationship: a 20-fold increase in speed of motion leads to a factor of 12.6 increase in the number of link changes per second, and a factor of 11.6 increase in the number of shortest-path changes per second. The increase in number of shortest-path changes at a higher speed is smaller than the increase in the number of link changes because not all link changes lead to a path change.

In the 200-node scenario, the average node degree is similar to that in the 100-node scenario because the network density between the scenarios is nearly the same. The number of link changes per second is 2.2 times higher in the 200-node case as there are twice as many nodes, while the number of shortest path changes is 6.3 times higher, since paths are longer and thus the creation or breaking of one link may lead to the creation or breaking of multiple paths.

Table 3.4: ODMRP Parameter Settings

Parameter Name	**Value**
REFRESH_INTERVAL	3 sec.
JOIN_AGGREGATION_TIMEOUT	0.025 sec.
RREQ_RETRIES	2
FORWARDING_STATE_TIMEOUT	$3x$REFRESH_TIMEOUT
JOIN_REPLY_PASSIVE_ACK_TIMEOUT	2 sec.
MAX_NUM_JOIN_REPLY_RETRIES	7

3.4.3. Communication Scenarios

The set of scenarios used in this evaluation are representative of a variety of possible multicast applications, and reveal the behavior of ADMR, MAODV and ODMRP in different group and network configurations, and under different factors such as mobility, number of nodes, network load, and MAC layer.

The multicast sources begin sending data and the multicast receivers join a multicast group at uniformly randomly chosen times between 0 and 180 seconds from the beginning of the simulation and remain in the multicast session until the end of the simulation unless otherwise noted. In all scenarios, Constant Bit Rate (CBR) traffic generators send 64-byte packets. This packet size was chosen in order to reduce the likelihood of congestion, which would make analyzing the routing protocol behavior meaningless. The packet rates are chosen to continuously probe the routing ability of the protocols rather than to represent any particular application.

In the rest of this chapter, I use the notation $GxSxR$, where G is number of multicast groups, S is number of multicast sources per group and R is number of multicast receivers per group. In addition, in the notation GxN, G is the number of multicast groups, and N is the number of nodes in each group, where each node is both a source and a receiver for the group.

Group Composition Experiments

The group composition experiments were designed to explore the effect on routing performance of varying the number of sources and receivers in a group, and the size of the group relative to the size of the network.

1. *Varying the number of multicast receivers:* This set of experiments was designed to explore the effect on protocol performance of the number of multicast receivers within a multicast group. I used three single-source and three multi-source, multi-group scenarios in which the number of receivers is progressively increased:

   - $1x1x10$, $1x1x50$, $1x1x99$
   - $2x3x10$, $2x3x20$, $2x3x40$

   In all scenarios above, each source sends four 64-byte packets per second.

2. *Group size vs. network size*: I used the scenarios above to analyse protocol behavior when the fraction of network nodes that are part of a multicast session varies. These experiments expose the level of redundant data forwarding in each protocol in terms of the number of nodes in the network that are involved in data forwarding relative to the size of the multicast group they are serving.

3. *Varying the number of multicast sources:* This set of experiments was designed to explore the effect on protocol performance of the number of sources in a multicast group. I used three scenarios with one group and three scenarios with multiple groups, in which the number of sources is progressively increased:

   - $1x1x20$, $1x5x20$, $1x10x20$
   - $2x1x20$, $2x5x20$, $2x10x20$

   The generated traffic is kept the same across all single group scenarios and across all multi-group scenarios. In particular, the load is ten 64-byte packets per second per source, in the 1-source scenarios, two 64-byte packets per second per source in the 5-source scenarios, and one 64-byte packet per second per source in the 10-source scenarios.

Multicast Application Scenarios

The multicast application experiments consist of four sets of experiments, each representative of a possible multicast application configuration.

1. *Single-source groups vs. multi-source groups*: This set of experiments was designed to explore the effect of the distribution of sources among groups. In particular, they expose the performance of shared tree and group mesh-based protocols on one hand, and single-source tree/mesh protocols on the other, serving both single-source applications, and also applications with multiple multicast sources per group. I used 3 sets of scenarios with an increasing number of sources:

   - $1x3x15$, $3x1x5$
   - $1x5x50$, $5x1x10$
   - $1x9x27$, $9x1x3$

   The generated traffic is kept the same across each pair of scenarios and is four 64-byte packets per second per source.

2. *Conferencing Applications*: This set of experiments was designed to explore the performance of the multicast protocols in serving conferencing applications in which all nodes are simultaneously multicast sources and receivers. In particular, I used the following set of scenarios:

   - $1x5$, $1x10$, $1x20$

   The generated traffic is kept the same across all scenarios. In particular, the load is ten 64-byte packets per second per source, in the 5-source scenario, two 64-byte packets per second per source in the 10-source scenario, and one 64-byte packet per second per source in the 20-source scenario.

3. *Short-Term Receiver-Driven Sessions*:

   This set of experiments was designed to explore the behavior of the protocols in the context of repeated short-term receiver joins, e.g., receivers join a multicast session for a short duration, then leave the session and then join it again later, etc. This scenario is representative of applications where content-distribution servers serve content at all times but clients only want to check that content from time to time, e.g., for weather reports, news, stock quotes.

Each receiver in the simulations joins a multicast group for a random period of time uniformly distributed between 60 and 180 seconds and then leaves the group for a random period of time uniformly distributed between 60 and 180 seconds, and then repeats the process until the end of the simulation.

I used the following sets of scenarios:

- $1x1x10$, $1x1x50$
- $2x3x10$, $2x3x20$

Each source sends four 64-byte packets per second.

4. *Short-Term Source-Driven Sessions* :

   This set of experiments was designed to explore the behavior of the protocols in the context of short-term multicast sessions with sources that send data intermittently. This scenario is representative of applications where sources send updates only when new information becomes available, e.g. for weather reports, news, stock quotes.

   Each source in the simulations is active (i.e., is sending data to the respective multicast group) for a random time uniformly distributed between 60 and 180 seconds and then does not send data for a random time uniformly distributed between 60 and 180 seconds, and then repeats the process until the end of the simulation.

   I used the same multicast configurations for this experiment as the ones used in the short-term receiver-driven sessions above.

MAC-layer Effects (null MAC)

In this set of experiments, I turn off the MAC layer in the simulations in order to isolate MAC layer effects and explore the effect that a better (”idealized”) MAC layer would have on protocol performance in anticipation of improved future MAC protocols. With a *null MAC* there are no collision losses and all packet losses are caused by mobility.

I used the following multicast scenarios in this experiment:

- $1x1x10$, $3x3x9$.

Each source sends four 64-byte packets per second.

Effects of Mobility

This set of experiments was designed to explore the differences in the protocols’ performance between scenarios with maximum node speed of 20 m/s and 1 m/s.

I conduct the comparison using the following sets of experiments:

- $1x1x10$, $1x1x50$, $1x1x99$
- $2x3x10$, $2x3x20$, $2x3x40$

and

- $1x1x20$, $1x5x20$, $1x10x20$
- $2x1x20$, $2x5x20$, $2x10x20$

The packet rate is four 64-byte packets per second per source in the first set of scenarios, Set I, while in the second set, Set II, the load is ten 64-byte packets per second per source, in the 1-source scenarios, two 64-byte packets per second per source in the 5-source scenarios, and one 64-byte packet per second per source in the 10-source scenarios; the overall generated traffic load is constant across the scenarios in each set.

Increasing Network Size

This set of experiments was designed to explore the differences in the protocols' performance between scenarios with 100 and 200 nodes, and is based on the multicast scenarios above as well.

3.4.4. Performance Metrics

Protocol performance will be evaluated using the following metrics, which are computed over the whole duration of the simulation:

- *Packet Delivery Ratio:* The fraction of packets sent by the multicast application that are received by the multicast receivers. For example, if there are 2 sources and 5 receivers, and each source sends 10 packets, and each receiver receives 12 multicast packets total, the packet delivery ratio would be $(5 \times 12)/(5 \times 20)$, which is 0.6.

- *Control Packet Overhead*: The total number of control packets originated and forwarded by the protocol.

- *Data Packet Load*: The total number of data packets originated and forwarded by the protocol. This metric reflects the total network load caused by data transmissions.

- *Normalized Data Packet Overhead*: The number of data transmissions performed by the protocol per successfully delivered data packet. This metric evaluates the level of redundancy in forwarding data that the protocol incorporates coupled with the path lengths it utilizes. For example, normalized data packet overhead of 5 means that the protocol makes 5 data packet transmissions on average for each data packet that is delivered to a multicast receiver.

- *Normalized Packet Overhead:* The number of control and data transmissions performed by the protocol per successfully delivered data packet. This metric evaluates the overall effort that the protocol expends for the delivery of a each data packet. For example, normalized packet overhead of 5 means that the protocol makes 5 packet transmissions on average for each data packet that is delivered to a multicast receiver.

- *End-To-End Delay:* The time between the transmission of a data packet from a multicast source and the time of its reception by a multicast receiver, averaged over all packets.

- *Average Path Lenth:* The average number of hops traversed by a data packet from a source to a multicast receiver, averaged over all packets.

I do not discuss control byte overhead in this chapter as the results are very similar to those of control packet overhead. ADMR adds a negligible amount of bytes to each data packet (about 12). Packet overhead is a much more significant determinant of performance because it dominates the time spent by packets at the MAC layer.

3.5. Results

In most of this section, the discussion focuses on 100-node scenarios and a random waypoint model configured with a maximum speed of 20 m/s and 0 pause time. The exceptions are Section 3.5.8 in which I discuss networks with 200 nodes, and Section 3.5.7 in which I discuss networks with 100 nodes with the random waypoint model configured with a maximum speed of 1 m/s. The discussion only addresses statistically significant results.

3.5.1. Varying the Number of Multicast Receivers

Increasing the number of receivers leads to the creation of more multicast forwarding state in the network, and consequently to a higher number of data packet transmissions; this increase leads to a higher packet delivery ratio since nodes have more opportunities to receive a transmission of each packet. MAODV and especially ODMRP create a large amount of redundant forwarding state even for small groups, and so increasing the number of receivers in these protocols has a small effect on their packet delivery ratio.

All three protocols have high packet delivery ratios in these scenarios (Figure 3.1). In the single-source scenarios, the protocols perform comparably and deliver almost all of their packets; in the multi-source scenarios, collision losses caused by the higher network load lead to slightly lower packet delivery ratios. The decrease is most pronounced in MAODV, since it uses longer data forwarding paths, which are more prone to collision losses.

ADMR generates up to 14 times less control packet overhead than MAODV and up to 5 times less overhead than ODMRP (Figure 3.2). The high control overhead in MAODV is due to the periodic group leader floods and the periodic neighbor “Hello” packets. The periodic nature of the overhead is also the reason why it does not change significantly between scenarios. ODMRP’s high overhead is as a result of its periodic source flood and response cycles, with the response part of the cycle growing with the number of receivers. While ADMR’s and ODMRP’s control packet overhead increases when there are more receivers in the network, MAODV’s overhead decreases. In ADMR and ODMRP, the presence of more receivers means a larger number of receiver joins, and in the case of ADMR, more mesh repairs. The decrease in overhead with MAODV is due to the fact that its ability to localize join and repair floods improves when there are more nodes in its group shared tree.

MAODV generates the highest data packet load across all scenarios (Figure 3.3) and its normalized data packet overhead is the worst among the three protocols (Figure 3.4). ADMR forwards data most efficiently generating at most 1.2 transmissions per correctly received data packet, which is about 8 times less than MAODV and 4 times less than ODMRP. Normalized data packet overhead decreases for all protocols as the number of receivers grows because when there are more receivers in the network, the paths to them are more likely to share links, and as a result, the overhead of forwarding a data packet is amortized between them.

ADMR is the most efficient of the three protocols overall, having the lowest normalized packet overhead, while MAODV generates the highest such overhead (Figure 3.5). The normalized packet overhead decreases with an increase in the number of receivers, because it is dominated by the data packet transmissions generated by the protocols, which are better amortized when the number of receivers increases.

End-to-end delay is highest for MAODV due to the longer paths that data packets have to follow within the shared tree and due to the higher network load caused by the high number of control and data packet transmissions (Figure 3.7). Network load translates into a busier wireless medium, which causes nodes to have to wait longer before forwarding each packet. ODMRP’s end-to-end

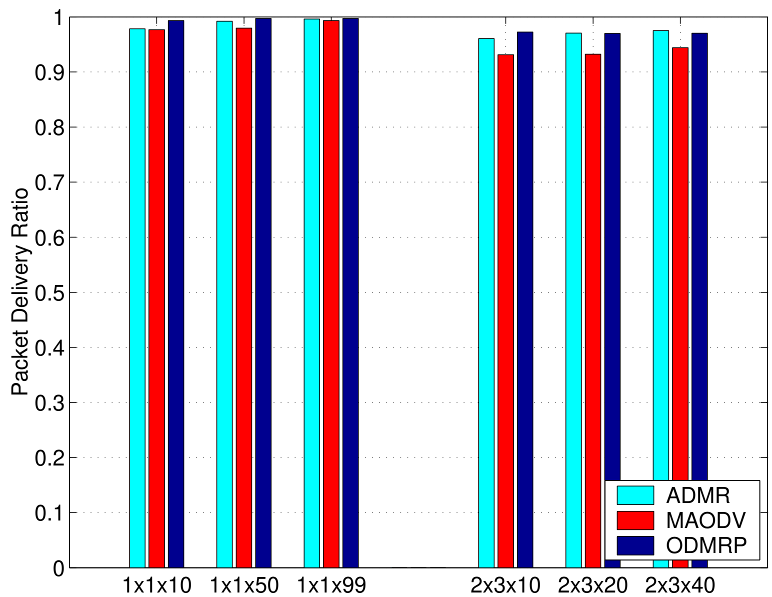

Figure 3.1: Varying the Number of Receivers. Packet delivery ratio for 100 nodes, 0 pause time, 20 m/s maximum speed.

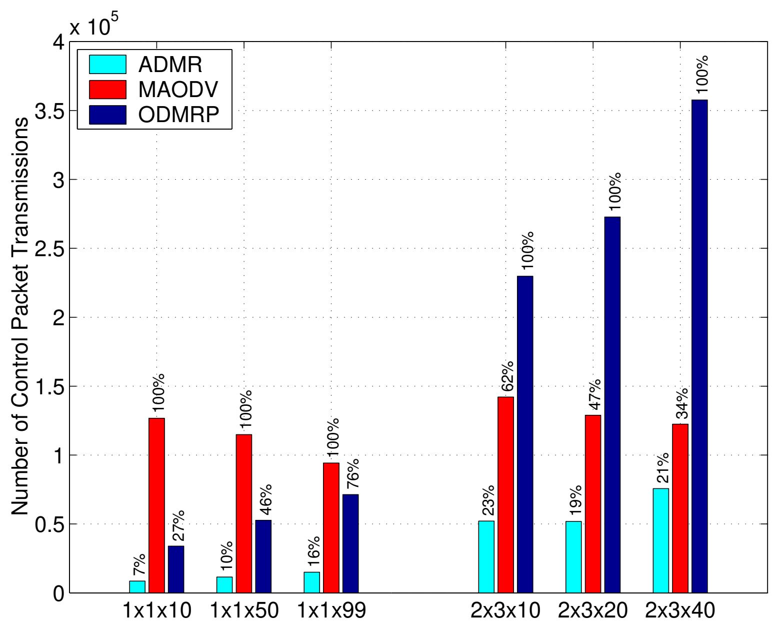

Figure 3.2: Varying the Number of Receivers. Control packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

delay is lowest because, due to its frequent state discovery floods, it uses the shortest forwarding paths among the three protocols (Figure 3.6).

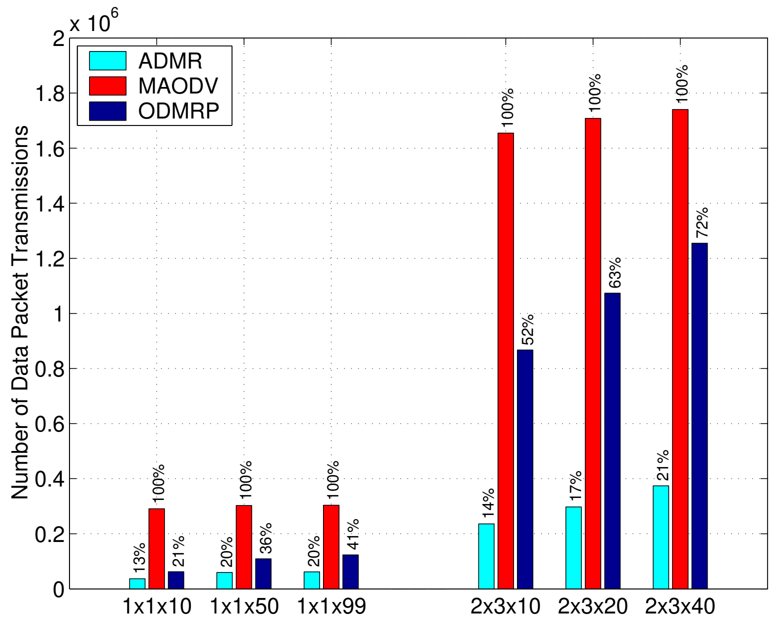

Figure 3.3: Varying the Number of Receivers. Data packet load for 100 nodes, 0 pause time, 20 m/s maximum speed.

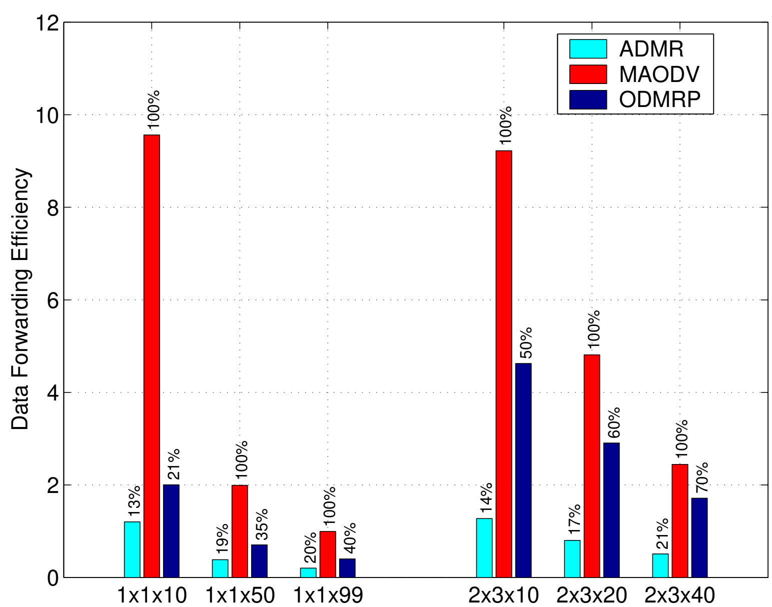

Figure 3.4: Varying the Number of Receivers. Normalized data packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

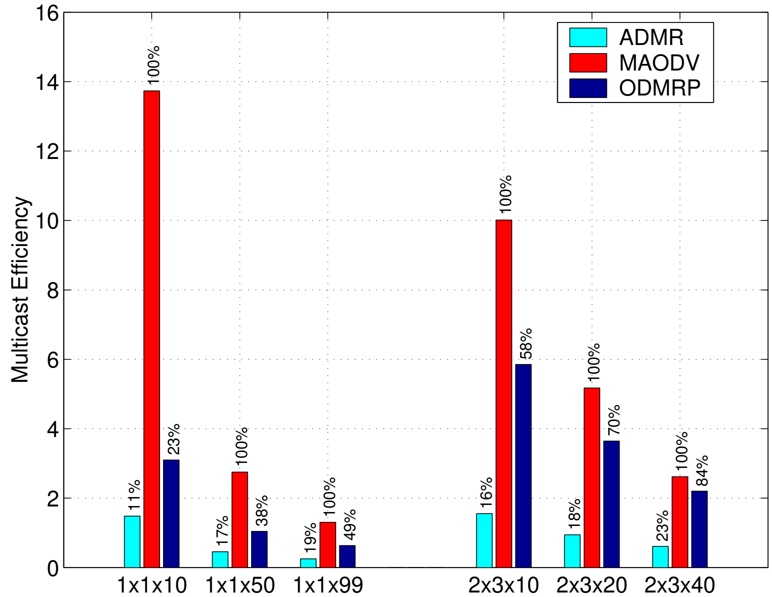

Figure 3.5: Varying the Number of Receivers. Normalized packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

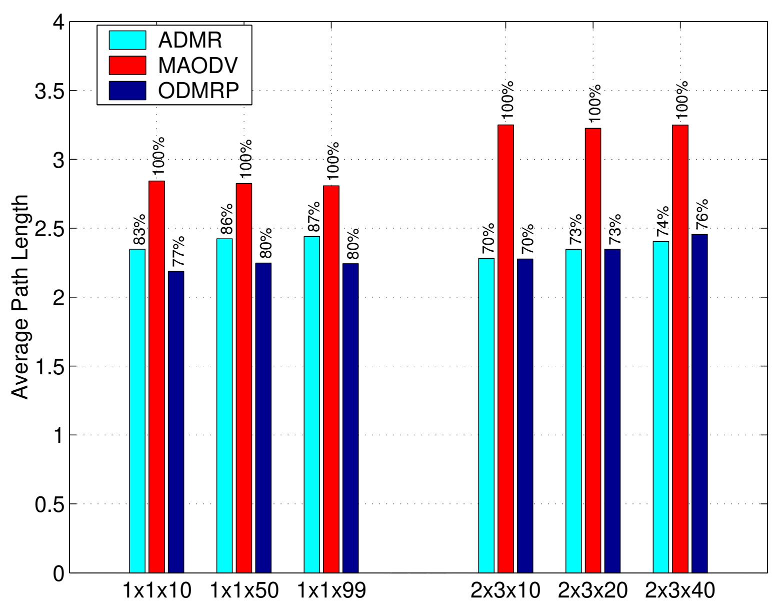

Figure 3.6: Varying the Number of Receivers. Average path length for 100 nodes, 0 pause time, 20 m/s maximum speed.

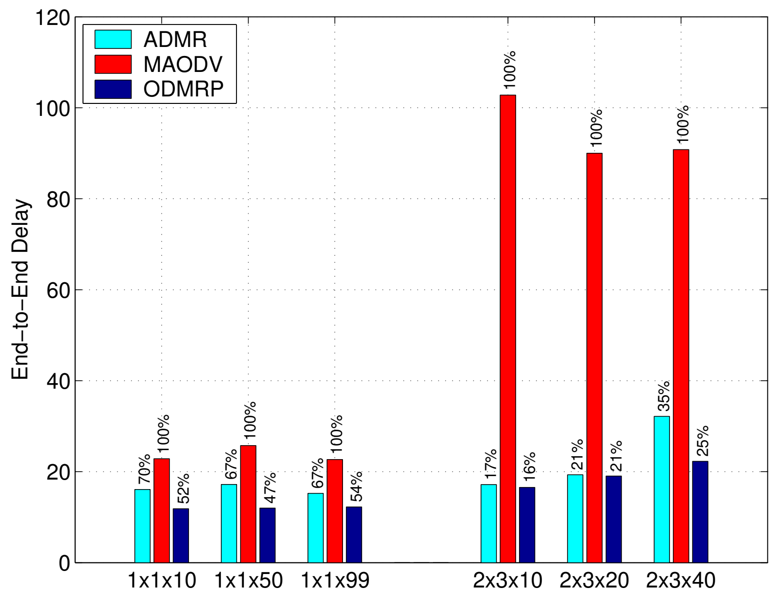

Figure 3.7: Varying the Number of Receivers. End-to-end delay for 100 nodes, 0 pause time, 20 m/s maximum speed.

3.5.2. Varying the Number of Multicast Sources

As the number of sources grows, MAODV and ADMR maintain their packet delivery ratio, while ODMRP's decreases (Figure 3.8). This decrease is more noticeable in the multi-source scenarios and is a result of packet loss due to increased congestion in the network caused by the high control and data packet overhead generated by the protocol (Figure 3.12).

ODMRP's control packet overhead scales approximately linearly with each added multicast source because each ODMRP source initiates periodic control packet request and response cycles and so the per-source control overhead is fixed given a fixed set of receivers. ADMR scales its overhead sublinearly with the number of sources since even though each source creates and operates its own tree, when the receiver members join a multicast group before a given source becomes active, once the source does become active, many RECEIVER JOIN packets sent by the receivers in response to a new source's data flood are filtered. MAODV displays near-constant overhead as additional sources for the same group, only generate a small amount of overhead to join the shared tree through the closest tree node to them.

ADMR generates the lowest data packet load as well, while MAODV generates the highest (Figures 3.10 and 3.11). MAODV generates a large number of data transmissions because of the longer forwarding paths that it uses (Figure 3.13) and because data flows both towards the receivers and the sources. In both ADMR and ODMRP, the number of redundant data packet transmissions grows with the number of sources. In ADMR redundant data transmissions are performed within each multicast tree, while in the case of ODMRP, the state is shared and so each source "stretches out" the mesh by causing additional forwarding state to be created between itself and the multicast receivers. In addition, as in MAODV, in ODMRP, data flows not only towards the receivers but also towards the sources when it is flooded within the group mesh.

ADMR generates the lowest normalized packet overhead in all scenarios (Figure 3.12). Both ADMR and ODMRP's normalized packet overhead increases when there are more multicast sources due to the higher number of state that needs to be created and maintained for each new active source.

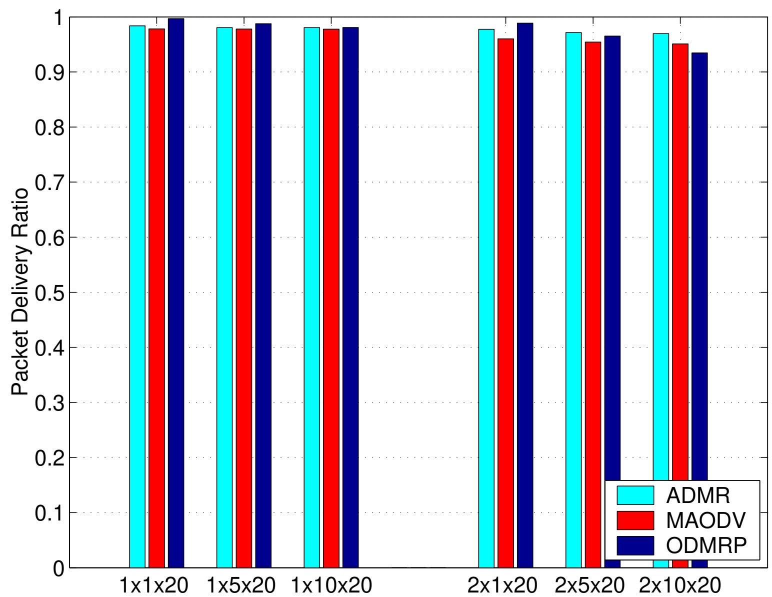

Figure 3.8: Varying the Number of Sources. Packet delivery ratio for 100 nodes, 0 pause time, 20 m/s maximum speed.

MAODV's normalized packet overhead remains nearly the same across all scenarios since the shape of its shared tree does not change significantly when new sources are added to it.

MAODV packets experience the highest end-to-end delay, as a result of the longer paths along which they are forwarded, and as a result of the high network load caused by control and data packet transmissions performed by the protocol. The effect of network load is clear when comparing the path lengths in the single-source and multi-source scenarios and the packet latencies between the two (Figures 3.13 and 3.14): the average path lengths are similar but the latencies much higher in the multi-source scenarios. In ADMR and ODMRP end-to-end delay increases with the increase in network load, and in the case of scenario $2\times10\times20$, ODMRP even begins to route along longer paths than ADMR as the concentration of traffic along the shortest paths is high and leads to congestion and collision losses.

3.5.3. Single-Source vs. Multi-Source groups

Group shared mesh and tree protocols are usually intended to optimize the performance in multi-source groups, while single-source protocols are often assumed to work best for single-source groups. However, in my experiments, these statements frequently do not hold (Figure 3.15). The packet delivery ratio for multi-source groups is higher than the packet delivery ratio for single-source groups for ADMR in all simulated scenarios, and the opposite is true for all but the 3-source ODMRP scenario. In the case of ADMR, the higher packet delivery ratio for multi-source groups is due to a higher data forwarding redundancy resulting from the fact that each multicast mesh contains more receivers, and thus more paths, than in the single-source scenarios. ODMRP's packet delivery ratio decreases in multi-source groups with more than 3 sources because it generates too much forwarding redundancy and control overhead when each group contains multiple sources and this additional network load hurts its performance. MAODV's packet delivery ratio is higher in the multi-source experiments than in the single-source experiments since only one tree is built per group, and so with fewer groups, there is less network overhead. In addition, MAODV also benefits

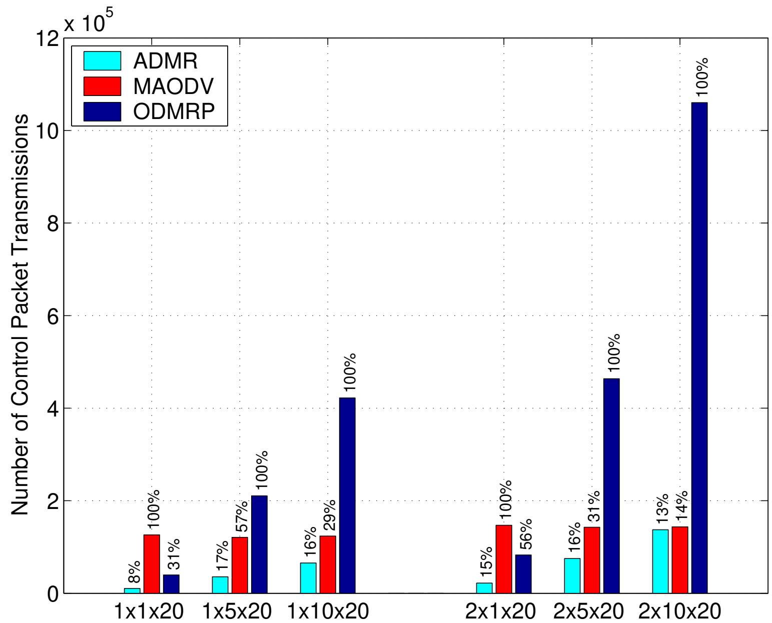

Figure 3.9: Varying the Number of Sources. Control packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

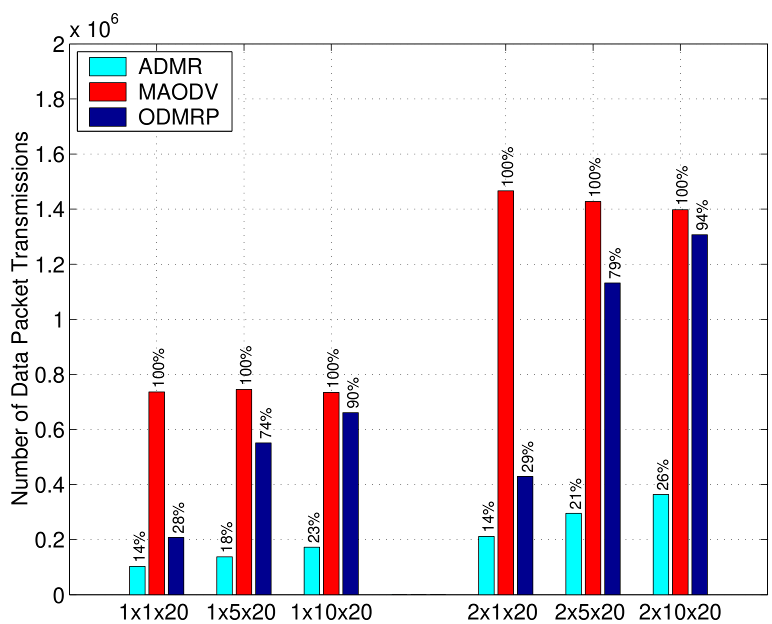

Figure 3.10: Varying the Number of Sources. Data packet load for 100 nodes, 0 pause time, 20 m/s maximum speed.

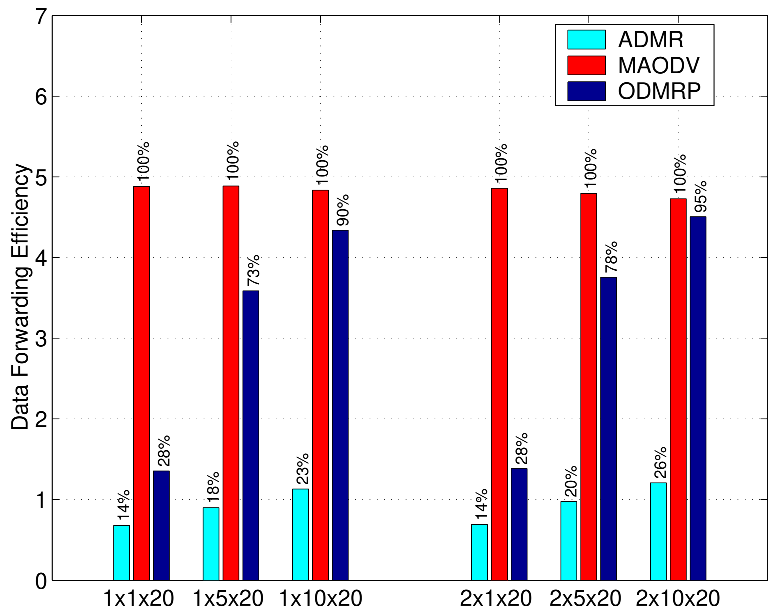

Figure 3.11: Varying the Number of Sources. Normalized data packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

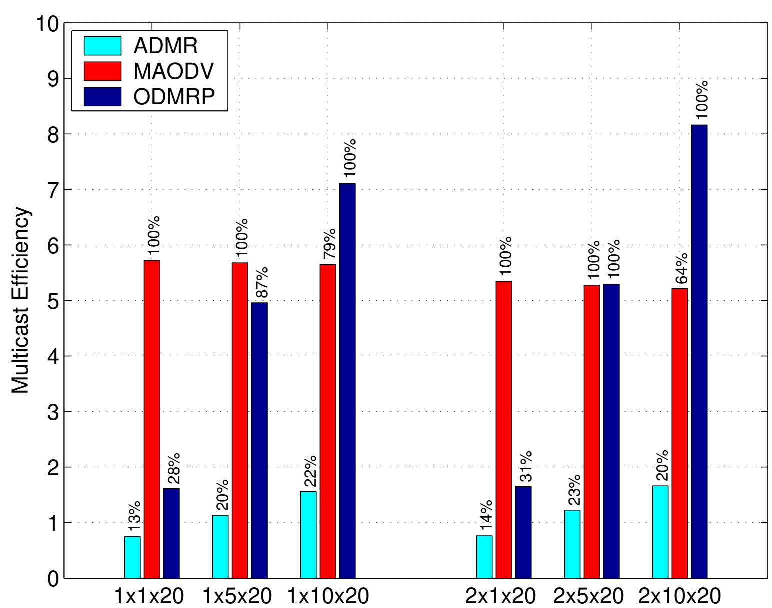

Figure 3.12: Varying the Number of Sources. Normalized packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

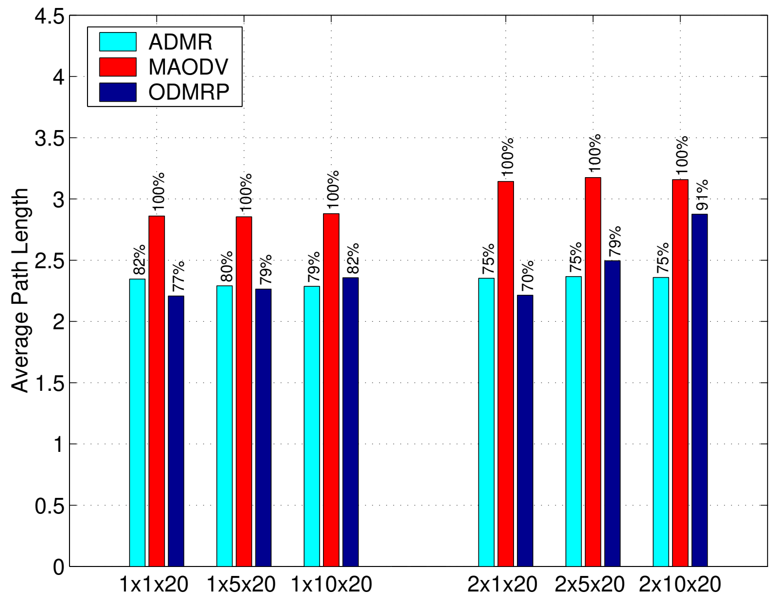

Figure 3.13: Varying the Number of Sources. Average path length for 100 nodes, 0 pause time, 20 m/s maximum speed.

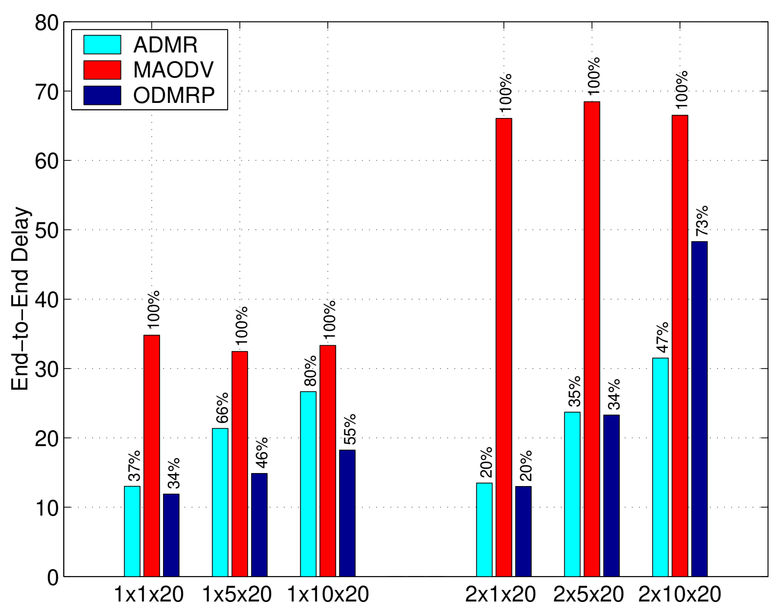

Figure 3.14: Varying the Number of Sources. End-to-end delay for 100 nodes, 0 pause time, 20 m/s maximum speed.

from higher data forwarding redundancy when there are more receivers within a single group, as a packet forwarded by one node may be overheard by more nodes that need to forward or receive it.

ODMRP incurs higher control overhead in multi-source groups (Figure 3.16) than in single-source groups because more JOIN REPLY packets are sent when there are more receivers, and since the mesh is also larger in this case, each REPLY is forwarded more times. MAODV generates more control overhead when more shared trees need to be set up because its overhead increases with the number of groups in the network. As a result, its overhead is higher in the single-source scenarios.

ADMR generates almost the same control packet overhead between the single and multi-source scenarios in all cases except in the heavy (9-source) scenario, where it generates a little more control overhead in the multi-source case. The generated overhead is similar between multi-source and single-source scenarios since ADMR builds a source mesh for each multicast source. The different behavior in the 9-source scenario occurs because the high number of receivers within a single source mesh results in traffic concentration which leads to collision losses which may trigger repairs even though no links are actually broken.

The data packet load is higher in the multi-source scenarios for all protocols because each of them creates more redundant forwarding state when there are more receivers within a group (Figure 3.17).

Normalized data packet overhead is higher in the single-source group scenarios than in the multi-source group scenarios for all protocols (Figure 3.18). This is due to the fact that there are less and more sparsely located receivers in each group and as a result the paths to reach them are longer (Figure 3.20), and are also less likely to share links, leading to a higher number of times each packet is forwarded.

Overall, normalized packet overhead is lower in the multi-source scenarios for all protocols. ADMR generates the lowest normalized packet overhead in all scenarios, and MAODV generates the highest such overhead.

End-to-end delay is lowest for ODMRP due to the shorter paths that it uses (Figure 3.20), followed closely by ADMR, with MAODV having the highest end-to-end delay. In the $1 \times 9 \times 27$ scenario, ODMRP and MAODV are affected by the high control and data packet load they impose on the network, and incur a significantly higher end-to-end delay.

3.5.4. Short-Term Sessions and Session Membership

In this section, I compare the performance of each protocol in the short-term session experiments versus its performance in the same scenario but with long-term multicast sessions, i.e., scenarios in which the receivers and sources stay in the session until the end of the simulation.

The packet delivery ratio is nearly the same for the short-term sessions as for the long-term sessions (Sections 3.5.1 and 3.5.2) for all protocols. The only exception is the $2x3x10$ scenario where the packet delivery ratio is lower for all protocols (Tables 3.8 and 3.9). MAODV is the most significantly affected, achieving a packet delivery ratio in the short-term receiver-driven sessions experiment for the $2x3x10$ scenario that is 13% lower than in the corresponding long-term session scenarios. There are two reasons for this: 1) The control packet overhead is higher in this scenario 2) Unneeded forwarding state cannot be expired instantaneously and continues to operate, and since the receivers are sparsely located, more state is created and maintained, and then would need to be expired. The location of the receivers is similarly sparse in the $1x1x10$ scenario, however, the overall traffic rate is higher in the $2x3x10$ scenario and as a result, it exposes the differences in behavior more clearly.

Control packet overhead varies between the protocols and between the different scenarios. Overall, in the case of ADMR, new receiver joins are not localized and as a result frequent joins

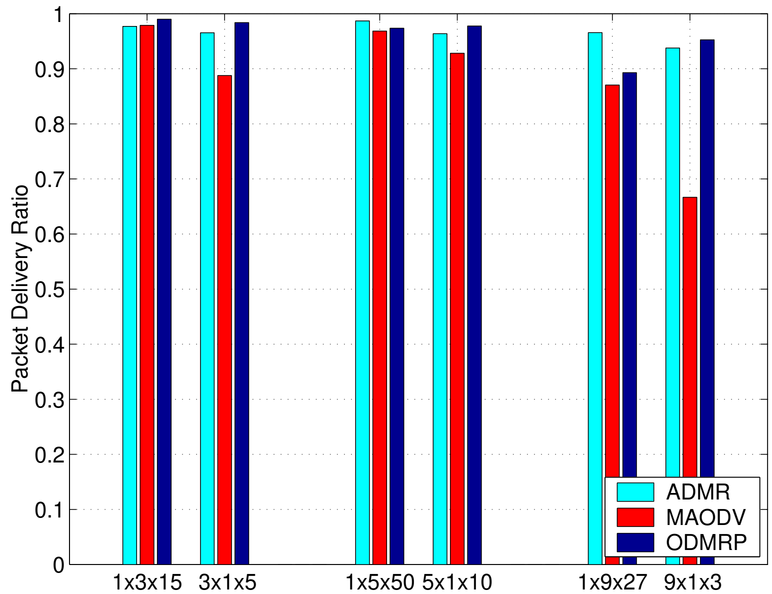

Figure 3.15: Single Source vs. Multi-Source Groups. Packet delivery ratio for 100 nodes, 0 pause time, 20 m/s maximum speed.

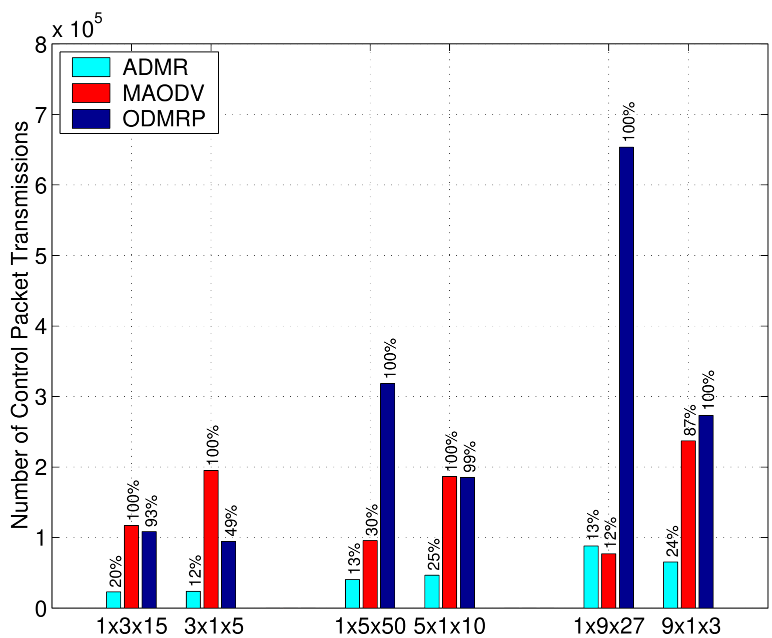

Figure 3.16: Single Source vs. Multi-Source Groups. Control packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

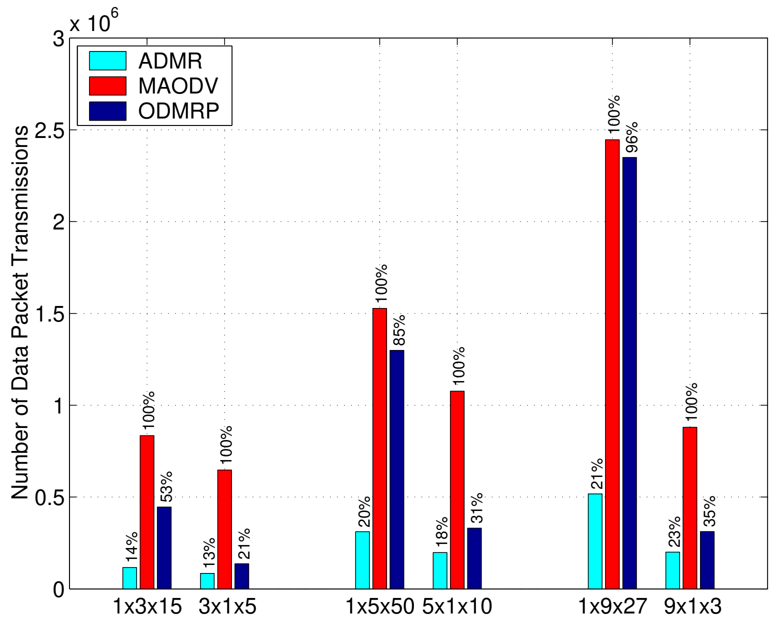

Figure 3.17: Single Source vs. Multi-Source Groups. Data packet load for 100 nodes, 0 pause time, 20 m/s maximum speed.

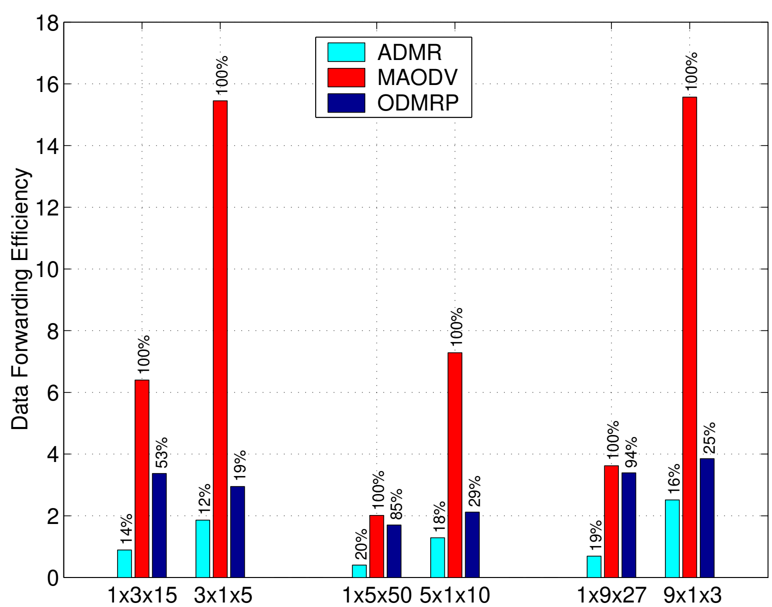

Figure 3.18: Single Source vs. Multi-Source Groups. Normalized data packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

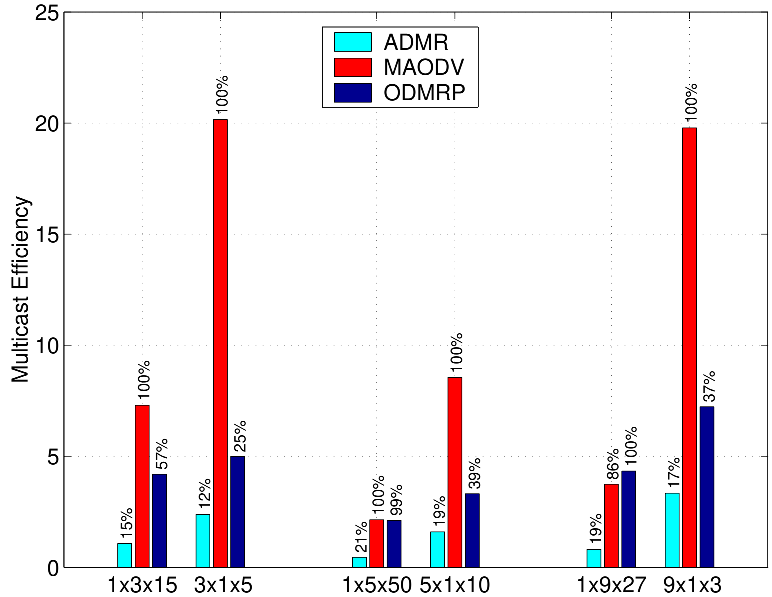

Figure 3.19: Single Source vs. Multi-Source Groups. Normalized packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

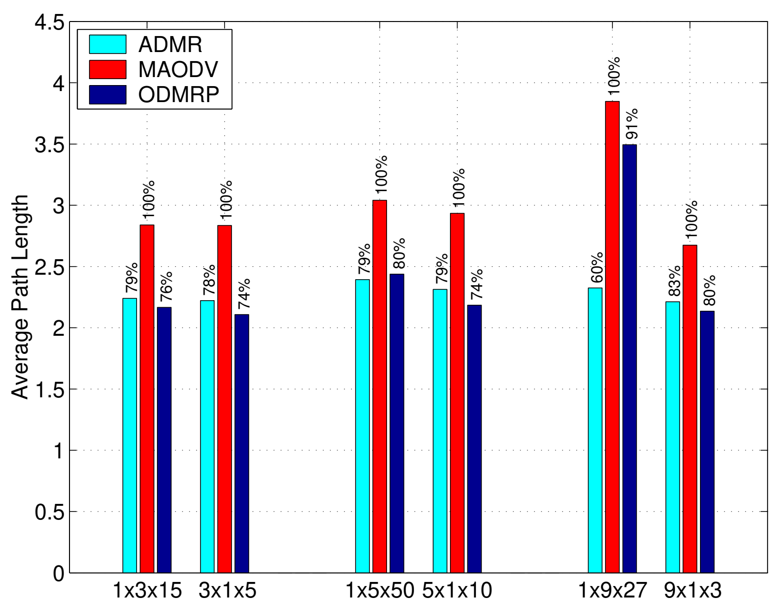

Figure 3.20: Single Source vs. Multi-Source Groups. Average path length for 100 nodes, 0 pause time, 20 m/s maximum speed.

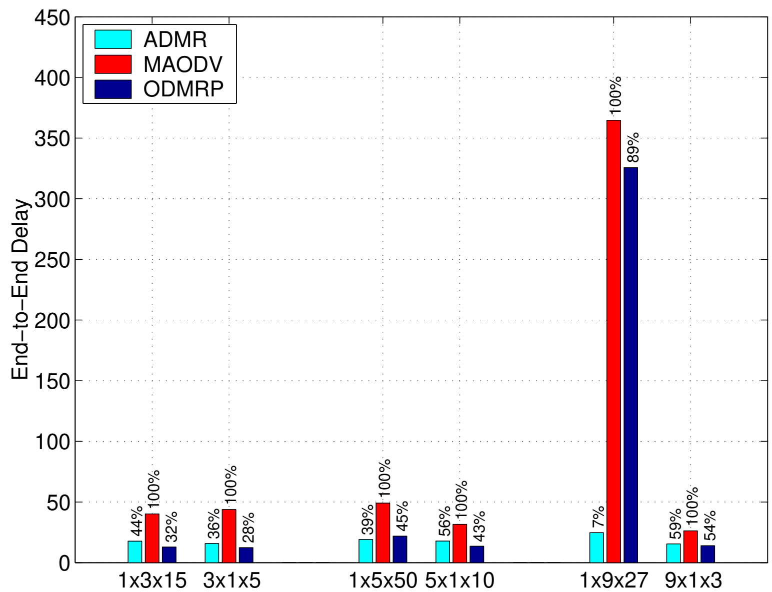

Figure 3.21: Single Source vs. Multi-Source Groups. End-to-end delay for 100 nodes, 0 pause time, 20 m/s maximum speed.

result in a higher amount of overhead and thus higher overall control overhead. In ODMRP, state creation is initiated only by the sources and so the control overhead is higher in the short-term source-driven sessions relative to the corresponding long-term session scenarios. Nodes joining a MAODV multicast session use expanding ring search and are thus able to localize their joins, causing control overhead to only increase by up to a factor of 1.7 and in some cases to even decrease. In contrast, control packet overhead in ADMR goes up by up to a factor of 3.7, and in ODMRP it goes up by up to a factor of 3.1 in the $2x3x10$ scenario, which poses the most difficulty to all protocols as it includes a small number of receivers that are sparsely located in the network and as a result require more control overhead to setup and maintain forwarding state to them. In the $1x1x50$ scenario in the short-term receiver-driven sessions joins experiment, ODMRP's packet control overhead decreases by 13% as a result of aggregation of JOIN REPLY packets which is very effective in the presence of many receivers in the network as is the case in this scenario.

Overall, the relative control packet overhead generated by the protocols does not change, with MAODV generating the most and ADMR generated the least. In ADMR and ODMRP, control overhead can be reduced by more aggressive expiration, e.g., reducing the value of the NUM_MISSING_PKTS_TRIGGER_EXPIRATION parameter in ADMR (Table 3.2) and the value of the FORWARDING_STATE_TIMEOUT parameter in ODMRP (Table 3.4). However, in the case of ODMRP, more aggressive expiration may lead to a noticeable performance degradation, as its high packet delivery ratio is due to redundant forwarding.

Even though the overall time during which receivers and sources are active in these scenarios is less in the short-term experiments, the number of data packet transmissions is not always lower and in fact in the $2x3x10$, it is higher for all protocols, especially in the short-term receiver-driven sessions, with ODMRP generating 5 times more data transmissions and ADMR and MAODV around 3. The reasons for this are the same as those for the higher control overhead – this scenario includes a relative small number of sparsely located receivers and as a result more forwarding state is created and since this state cannot be expired instantaneously, it causes more data transmissions than in the long-term sessions.

Table 3.5: Short-term receiver-driven sessions. Values represent the ratio between each metric for the short-term receiver-driven sessions and the same scenarios for the long-term sessions.

Metric	**Protocol**	$1x1x10$	$1x1x50$	$2x3x10$	$2x3x20$
PDR					
	ADMR	0.99	1.00	0.96	1.00
	MAODV	0.99	1.00	0.87	1.01
	ODMRP	0.99	1.00	0.98	1.00
Control Packet Overhead					
	ADMR	1.44	2.47	3.69	1.33
	MAODV	0.81	1.24	1.70	0.86
	ODMRP	1.01	0.87	2.78	1.01
Data Packet Load					
	ADMR	0.92	1.28	3.01	1.18
	MAODV	0.91	0.79	3.23	0.80
	ODMRP	0.81	0.91	5.11	1.03
Normalized Data Packet Overhead					
	ADMR	1.73	2.53	10.26	1.23
	MAODV	1.69	1.53	12.03	0.83
	ODMRP	1.52	1.78	17.06	1.08
Normalized Packet Overhead					
	ADMR	1.92	2.90	10.71	1.26
	MAODV	1.64	1.77	10.72	0.84
	ODMRP	1.65	1.75	14.21	1.07
End-To-End Delay					
	ADMR	0.83	0.62	0.95	0.85
	MAODV	1.08	0.82	1.55	0.43
	ODMRP	1.01	1.01	1.19	1.04
Average Path Length					
	ADMR	0.96	0.93	0.90	0.98
	MAODV	0.97	1.00	1.03	0.92
	ODMRP	0.99	0.99	0.98	1.00

The normalized data packet overhead is higher in most short-term scenarios, except in the $2x3x20$ short-term source-driven sessions scenarios where the number of data transmissions is lowest and the normalized data packet overhead decreases by up to 50%. Normalized data packet overhead is highest in the $2x3x10$ short-term receiver-driven sessions scenario, going up by a factor of 17 for ODMRP, 12 for MAODV and 10 for ADMR,

ADMR is still the most efficient of the three protocols, generating the least normalized packet overhead. MAODV generates the most such overhead.

End-to-end delay is lower for ADMR and higher for MAODV and ODMRP in most scenarios. Higher end-to-end delay is caused by the higher level of control and data packet load in MAODV and ODMRP. Lower end-to-end delay is due to the fact that the effects of the higher level of overhead are offset by the shorter paths the protocols are able to create due to the more frequent state setup floods caused by sources and receivers re-joining the group.

3.5.5. Conferencing Applications

Both ADMR and MAODV perform well in the conferencing scenarios and deliver around 95% of their packets (Figure 3.22). ODMRP delivers less than 90% of its data in these scenarios because it is not able to create enough redundant forwarding state to compensate for links that break due

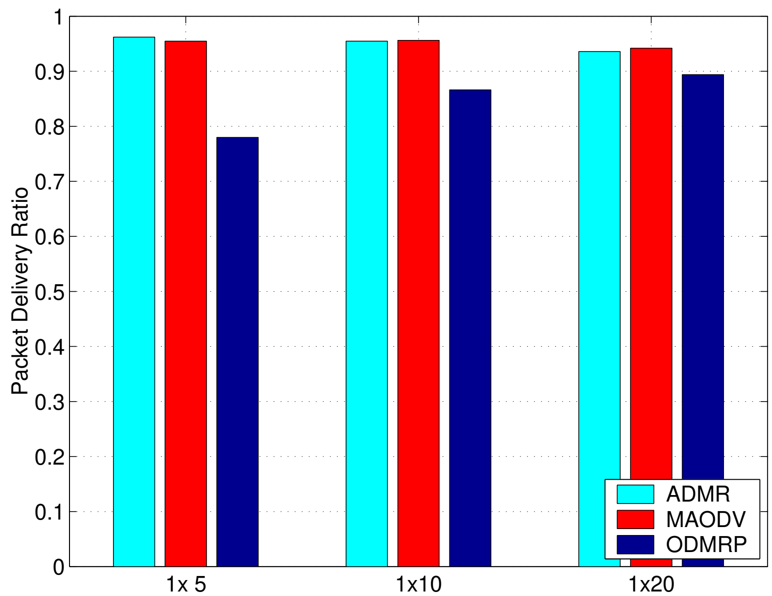

Figure 3.22: Conferencing Applications. Packet delivery ratio for 100 nodes, 0 pause time, 20 m/s maximum speed.

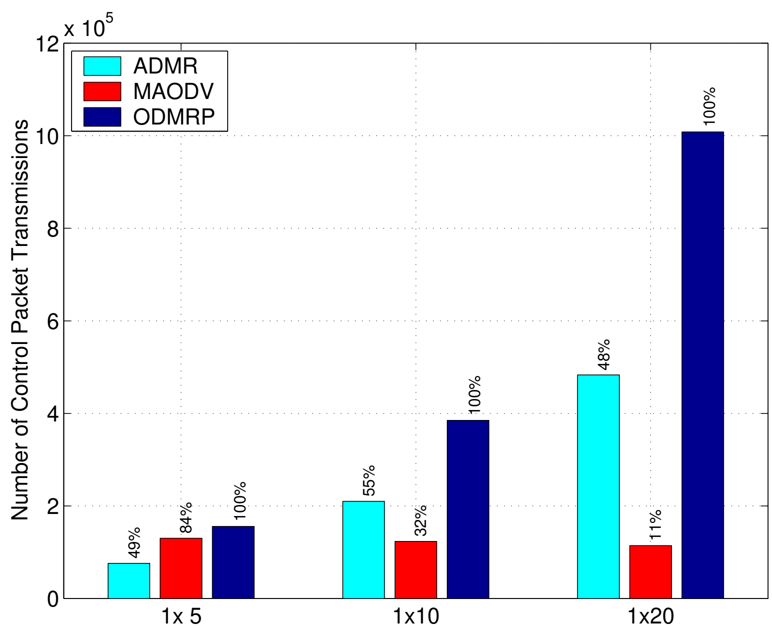

Figure 3.23: Conferencing Applications. Control packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

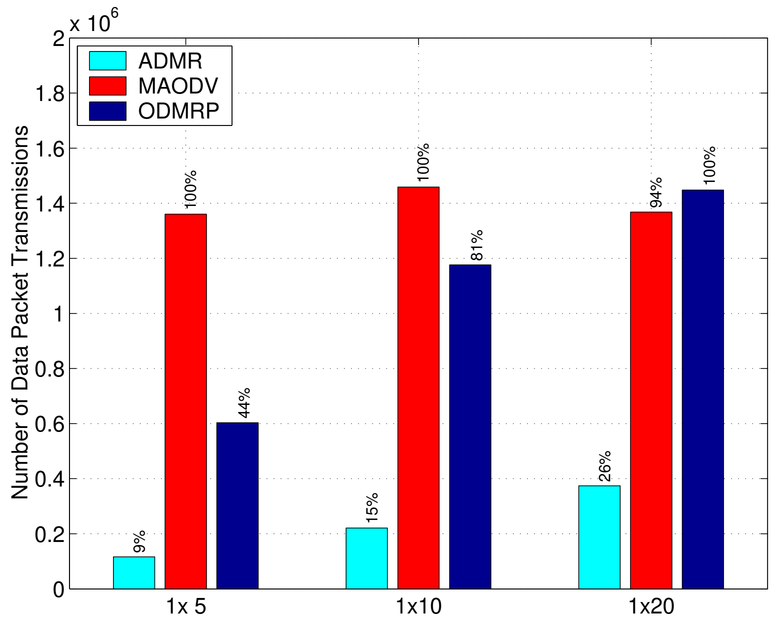

Figure 3.24: Conferencing Applications. Data packet load for 100 nodes, 0 pause time, 20 m/s maximum speed.

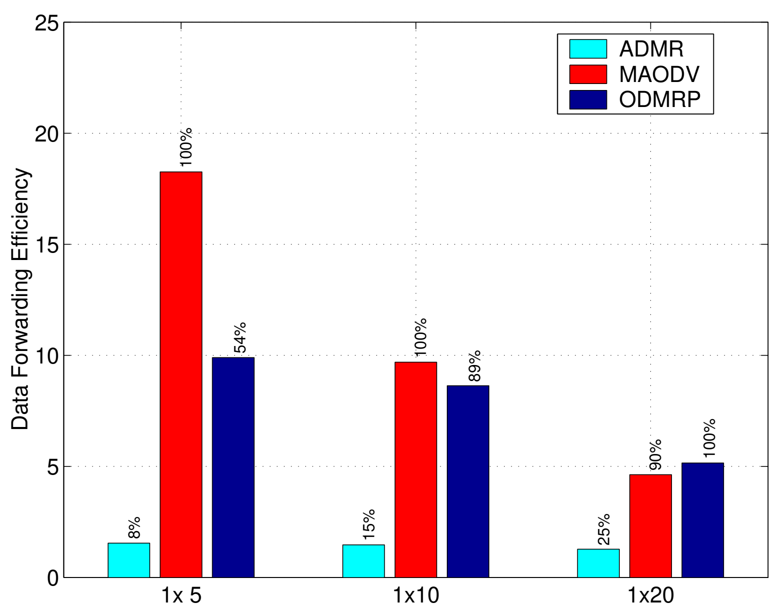

Figure 3.25: Conferencing Applications. Normalized data packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

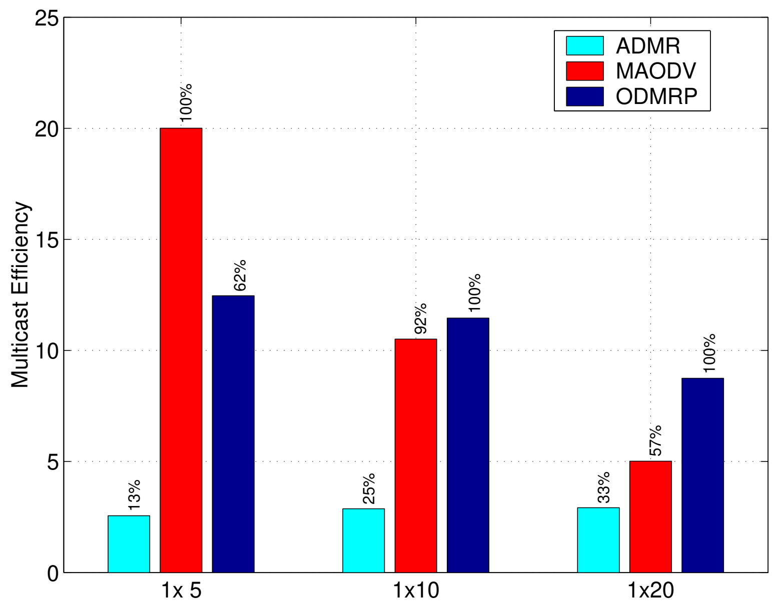

Figure 3.26: Conferencing Applications. Normalized packet overhead for 100 nodes, 0 pause time, 20 m/s maximum speed.

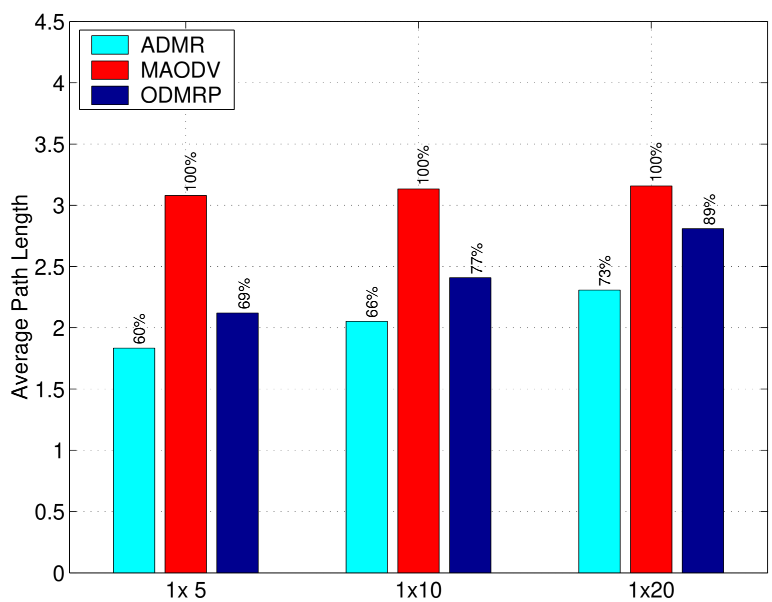

Figure 3.27: Conferencing Applications. Average path length for 100 nodes, 0 pause time, 20 m/s maximum speed.

Table 3.6: Short-term source-driven sessions. Values represent the ratio between each metric for the short-term source-driven sessions and the same scenarios for the long-term sessions.

Metric	**Protocol**	$1x1x10$	$1x1x50$	$2x3x10$	$2x3x20$
PDR					
	ADMR	1.00	1.00	0.97	1.02
	MAODV	0.99	1.01	0.95	1.02
	ODMRP	1.00	1.00	0.99	1.01
Control Packet Overhead					
	ADMR	2.32	1.13	3.14	0.83
	MAODV	1.15	1.17	1.71	1.05
	ODMRP	1.10	1.06	3.08	1.09
Data Packet Load					
	ADMR	0.57	0.75	1.94	0.64
	MAODV	0.52	0.69	2.83	0.54
	ODMRP	0.56	0.70	3.72	0.65
Normalized Data Packet Overhead					
	ADMR	1.08	1.12	5.97	0.62
	MAODV	0.98	1.02	8.90	0.52
	ODMRP	1.05	1.06	11.27	0.62
Normalized Packet Overhead					
	ADMR	1.72	1.22	6.70	0.65
	ODMRP	1.35	1.21	8.07	0.56
	MAODV	1.42	1.23	10.56	0.71
End-To-End Delay					
	ADMR	1.14	0.87	1.17	1.09
	MAODV	1.54	1.44	2.24	0.49
	ODMRP	1.02	1.01	1.15	0.92
Average Path Length					
	ADMR	1.00	1.00	0.93	1.02
	MAODV	1.01	1.00	1.04	0.91
	ODMRP	1.01	1.01	0.99	0.99

to node mobility. As group size increases, it is able to create more forwarding state and its packet delivery ratio improves.

Both ADMR and ODMRP's control packet overhead increases with the increase in group size, while MAODV's control overhead remains the same (Figure 3.23). In MAODV the presence of more shared tree nodes in the network enables better localization of its group joins and repairs. This localization is so successful, that MAODV generates the least control overhead in the $1{\times}10$ and $1{\times}20$ among the three protocols.

The data packet load for ADMR and ODMRP increases with the increase in group size, while MAODV's data load remains largely the same (Figure 3.24). MAODV's behavior is due to the fact that average path length remains nearly the same across all scenarios (Figure 3.27) and since the traffic load is the same, so is the data packet load. Average path length is smaller in the case of the larger conferencing scenarios in ADMR and ODMRP because as they strive to route along the shortest paths between the sources and the receivers, they focus a lot of traffic along the same paths and eventually they are forced to route along slightly longer paths as control and data packets become more prone to loss along the shortest paths. In MAODV on the other hand, the shape of the tree and the resulting path lengths depend on where the sources and receivers are located in the network and on the location of the current group leader. As a result, the traffic is spread out over a larger part of the network and is not as concentrated.

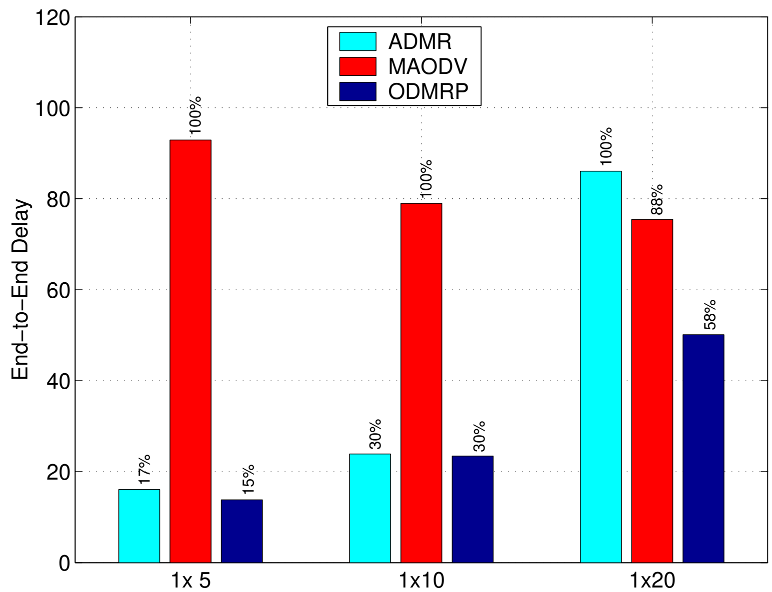

Figure 3.28: Conferencing Applications. End-to-end delay for 100 nodes, 0 pause time, 20 m/s maximum speed.

MAODV and ODMRP's normalized data packet overhead decreases with an increase in group size because due to the already redundant data packet forwarding, data packets are reaching more receivers without a significant addition of forwarding nodes (Figure 3.25). Since MAODV uses long paths, its forwarding state is already spread over a larger part of the network than in ODMRP and ADMR, and as a result, its decrease in normalized data packet overhead is more dramatic than in ODMRP. ADMR also generates slightly less normalized data packet overhead due to the small amount of redundant forwarding state it incorporates.

Overall, ADMR is the most efficient in the conferencing scenarios, generating up to 7 times less normalized packet overhead than MAODV and up to 5 times less normalized packet overhead than ODMRP (Figure 3.26).

End-to-end delay (Figure 3.28) in MAODV is high relative to that of ADMR and ODMRP in the 5 and 10 group scenarios, but decreases as the group size increases and in the 20 source scenario is lower than that of ADMR. The reduction in end-to-end delay is due to the fact that the packet load generated by the protocol is nearly constant and is amortized among an increasing number of nodes. In ADMR and ODMRP, end-to-end delay increases due to the higher number of overhead packets and an increase in path length due to traffic concentration along the shortest paths (Figure 3.27).

3.5.6. Null MAC

When there are no packets lost at the MAC layer, packet delivery ratio is higher as would be expected. ADMR and MAODV still lose a small number of packets during repairs, and ODMRP experiences some losses in sparse groups with few sources also due to mobility (Figure 3.29).

Control overhead is lower for ADMR in both scenarios, though the improvement is more significant in the heavy scenario (Table 3.7). This indicates that collision losses falsely trigger mesh repairs and so when such losses are eliminated (in the null MAC), these repairs no longer occur. Similarly, ODMRP's control overhead is lower in the heavy scenarios, due to the decrease in the

Table 3.7: Null MAC. Values represent the ratio between the value of each metric for the null MAC and for 802.11 MAC for the same scenario.

Metric	**Protocol**	$1x1x10$	$3x3x9$
PDR			
	ADMR	1.00	1.04
	MAODV	1.01	1.16
	ODMRP	1.00	1.05
Control Packet Overhead			
	ADMR	0.99	0.94
	MAODV	1.20	1.50
	ODMRP	1.00	0.88
Data Packet Load			
	ADMR	1.01	1.00
	MAODV	1.00	1.13
	ODMRP	0.98	0.94
Normalized Data Packet Overhead			
	ADMR	1.02	0.96
	MAODV	1.00	0.97
	ODMRP	0.98	0.90
Normalized Packet Overhead			
	ADMR	1.01	0.95
	MAODV	1.05	0.99
	ODMRP	0.99	0.88
End-To-End Delay			
	ADMR	0.87	0.74
	MAODV	0.31	0.02
	ODMRP	0.83	0.35
Average Path Length			
	ADMR	1.01	1.00
	MAODV	0.99	0.73
	ODMRP	1.03	0.93

number of retransmissions of JOIN REPLY packets relative to the 802.11 MAC case; such retransmissions are caused by lack of passive acknowledgments. Since ODMRP generates a lot of control and data overhead, such acknowledgments are much more likely to be lost in heavier scenarios with the 802.11 MAC layer. In MAODV on the other hand, control packet overhead increases by up to 50%. This is an indication that collisions actually help the protocol by eliminating some of its packets, and shows that the protocol is overly aggressive with its overhead.

ADMR does not generate any more data packet load than it does in the 802.11 scenarios, because it is not deterred by excessive control overhead and thus creates nearly the same multicast mesh in both cases. In ODMRP, data load is reduced in the heavy scenario over the 802.11 case, because packets are able to follow shorter paths first and thus short-circuit (preempt) some of the unnecessary data forwards.

Data packet load in MAODV increases in the heavy scenario over the 802.11 scenario, again due to the overly aggressive state setup which causes more forwarding state to be created in the network when the protocol is not slowed down by collision losses.

All protocols have nearly the same or lower normalized data packet overhead when there are no losses at the MAC layer, as the lack of collision losses causes more packets to get through (5%) and due to packets following shorter paths.

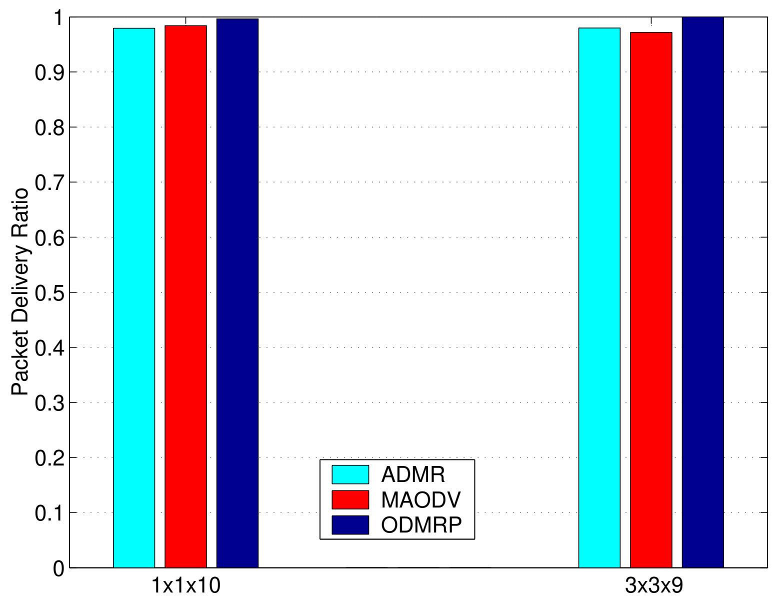

Figure 3.29: Null MAC. Packet delivery ratio for 100 nodes, 0 pause time, 20 m/s maximum speed.

Overall normalized packet overhead decreases for both ADMR and ODMRP in the heavy scenario, and increases in the light scenario for MAODV. The decrease in ODMRP's overhead is highest though the relative order of the protocols is preserved in both scenarios with ADMR generating the least overhead and MAODV the most.

Path lengths are the same in ADMR, for both MAC layers and in both scenarios, while MAODV and ODMRP are able to use shorter paths in the heavy scenario in the null MAC case, due to their improved ability to setup shorter paths when no packets are lost. ADMR on the other hand does not overload the network even in the 802.11 case and as a result, builds a similar enough mesh that paths are of the same length.

End-to-end delay is much lower in the null MAC scenarios as expected, and the largest improvements are for ODMRP and MAODV since they are not hindered by the high network load they cause on the network when using a null MAC.

Overall, improvements at the MAC layer may reduce packet overhead for some protocols and increase it for others. Protocols that operate efficiently will not change their performance significantly. And of course, packet delivery ratio and end-to-end delay improve or remain the same, with a better MAC layer.

3.5.7. Effects of Mobility

In this section, I explore the effects of mobility on multicast protocol performance, by comparing the results of simulating scenarios with the random waypoint model parameterized with maximum speeds of 20 m/s and 1m/s.

In order to highlight the differences in behavior between the 1 m/s and 20 m/s scenarios, rather than showing the absolute values for the performance metrics, Tables 3.8 and 3.9 show the ratio of the values for all performance metrics in the 1 m/s and 20 m/s scenarios.

The packet delivery ratio in the 1 m/s scenarios is typically the same as in the 20 m/s scenarios for ODMRP, and is around 1% better for ADMR. MAODV on the other hand, experiences a decline

Table 3.8: Effects of Mobility (Set I). Values represent the ratio between each metric for a 100-node 1 m/s scenario and a 100-node 20 m/s scenario.

Metric	**Protocol**	$1x1x10$	$1x1x50$	$1x1x99$	$2x3x10$	$2x3x20$	$2x3x40$
PDR							
	ADMR	1.01	1.01	1.00	1.01	1.01	1.01
	MAODV	0.96	0.99	1.00	0.93	0.94	0.91
	ODMRP	1.01	1.00	1.00	1.00	1.00	1.00
Control Packet Overhead							
	ADMR	0.56	0.81	0.94	0.52	0.60	0.51
	MAODV	2.28	2.19	2.22	1.89	2.01	2.24
	ODMRP	1.00	1.00	0.99	1.01	1.01	0.99
Data Packet Load							
	ADMR	0.84	1.02	1.22	0.85	0.95	1.01
	MAODV	0.91	0.97	1.00	0.97	0.97	0.98
	ODMRP	1.03	1.00	1.00	1.02	1.01	1.00
Normalized Data Packet Overhead							
	ADMR	0.83	1.01	1.22	0.84	0.95	1.00
	MAODV	0.95	0.98	1.01	1.04	1.04	1.07
	ODMRP	1.03	1.00	1.00	1.02	1.01	1.00
Normalized Packet Overhead							
	ADMR	0.78	0.97	1.16	0.78	0.89	0.91
	MAODV	1.39	1.31	1.30	1.12	1.12	1.16
	ODMRP	1.02	1.00	1.00	1.02	1.01	1.00
End-To-End Delay							
	ADMR	0.90	0.66	0.69	0.96	0.80	0.49
	MAODV	3.59	3.93	3.11	4.07	5.28	5.75
	ODMRP	0.99	1.00	1.00	1.02	1.00	0.99
Average Path Length							
	ADMR	1.00	0.97	0.95	1.02	0.97	0.97
	MAODV	1.03	1.05	1.00	1.12	1.12	1.13
	ODMRP	1.00	1.00	1.00	1.04	1.01	1.01

of up 9% over its performance in the 20 m/s scenarios. The improved packet delivery ratio in ADMR is due to the smaller number of broken links, which leads to fewer disruptions in packet flow and thus fewer packet losses. In MAODV, the decrease in packet delivery ratio in the 1 m/s scenarios is due to the higher control packet overhead (up to 2.3 times higher) than in the 20 m/s scenarios.

The higher overhead generated by MAODV in the lower mobility scenarios is due to a rise in the number of route repairs that MAODV performs, even though the lower mobility causes less link changes. The reason for this is that higher mobility forces the protocol to find new routes more frequently which results in the protocol finding shorter paths when such paths have become available. On the other hand, lower mobility allows the protocol to use older routes which over time may be longer than the currently available routes. Longer routes are more prone to packet loss, which may trigger route repair and thus disrupt packet flow and increase network load; this in turn causes more packet losses and triggers more repairs. The average path length used by data packets in MAODV in the 1 m/s scenarios is 14% longer than in the 20 m/s scenarios. The increased overhead and higher path lengths are most obvious in scenarios with a smaller number of sources or receivers, as there is less redundant forwarding state and the paths are longer, which increases the likelihood of packet loss and unnecessary repair attempts.

ADMR scales its overhead as needed by current communication needs and network conditions and incurs up to 50% less control overhead in the 1 m/s scenarios.

Table 3.9: Effects of Mobility (Set II). Values represent the ratio between each metric for a 100-node 1 m/s scenario and a 100-node 20 m/s scenario.

Metric	**Protocol**	$1x1x20$	$1x5x20$	$1x10x20$	$2x1x20$	$2x5x20$	$2x10x20$
PDR							
	ADMR	1.01	1.01	1.01	1.01	1.01	1.01
	MAODV	1.00	0.99	0.99	0.91	0.96	0.97
	ODMRP	1.00	1.00	1.00	1.00	1.00	1.00
Control Packet Overhead							
	ADMR	0.66	0.60	0.65	0.64	0.63	0.66
	MAODV	1.84	2.02	1.91	1.90	1.92	1.80
	ODMRP	1.01	1.00	1.00	1.00	1.00	0.99
Data Packet Load							
	ADMR	1.04	0.85	0.78	0.99	0.87	0.80
	MAODV	0.98	0.97	0.95	0.95	0.95	0.95
	ODMRP	1.07	1.01	0.99	1.02	0.99	1.00
Normalized Data Packet Overhead							
	ADMR	1.03	0.85	0.77	0.98	0.87	0.78
	MAODV	0.98	0.98	0.96	1.04	0.99	0.98
	ODMRP	1.06	1.01	1.00	1.02	1.00	0.99
Normalized Packet Overhead							
	ADMR	1.00	0.80	0.74	0.95	0.82	0.75
	MAODV	1.11	1.12	1.10	1.14	1.08	1.06
	ODMRP	1.05	1.01	1.00	1.02	1.00	0.99
End-To-End Delay							
	ADMR	0.92	0.80	0.86	0.92	0.77	0.77
	MAODV	2.08	3.03	2.72	4.72	3.90	3.70
	ODMRP	1.04	1.03	1.01	1.03	1.01	1.00
Average Path Length							
	ADMR	1.03	0.98	0.98	1.00	0.96	0.95
	MAODV	1.09	1.06	1.03	1.14	1.10	1.04
	ODMRP	1.04	1.03	1.01	1.02	1.01	0.98

ODMRP's control packet overhead and all other metrics are the same at both speeds for all scenarios because it operates without regard to network conditions.

MAODV's data packet load is generally slightly lower across all scenarios, while its normalized data packet overhead is slightly better in all scenarios in Set II, and in the single-source scenarios in Set I, where the average path length is higher.

ADMR shows the biggest decrease in normalized data packet overhead (up to 23%)in the 1 m/s scenarios, since there are less route repairs and as a result less leftover forwarding state causing packets to be forwarded unnecessarily.

In terms of overall efficiency, ADMR is still the most efficient, and MAODV the least efficient, with ADMR generating less overhead per packet than in the 20 m/s scenarios and MAODV generating more overhead per packet than in the 20 m/s.

End-to-end delay is lowest for ADMR (by up to 50%), while ODMRP's end-to-end delay remains the same, just like other aspects of its behavior. MAODV incurs 6 times higher end-to-end delay due to the longer paths that it uses and the higher level of network load it generates.

Table 3.10: Increasing Network Size (Set I). Values represent the ratio between each metric for a 200-node scenario and for the 100-node version of that scenario, both at 20 m/s maximum speed.

Metric	**Protocol**	$1x1x10$	$1x1x50$	$1x1x99$	$2x3x10$	$2x3x20$	$2x3x40$
PDR							
	ADMR	0.97	0.98	0.99	0.95	0.96	0.98
	MAODV	1.00	1.00	0.96	0.93	0.93	0.92
	ODMRP	1.00	1.00	1.00	1.00	1.00	1.00
Control Packet Overhead							
	ADMR	2.40	2.38	2.28	2.35	2.43	2.06
	MAODV	2.58	2.07	2.78	2.30	2.36	2.53
	ODMRP	1.88	1.70	1.59	1.94	1.93	1.88
Data Packet Load							
	ADMR	1.59	1.60	1.62	1.52	1.56	1.53
	MAODV	1.97	2.01	1.96	1.88	1.88	1.91
	ODMRP	1.77	1.86	1.94	1.88	1.90	1.91
Normalized Data Packet Overhead							
	ADMR	1.63	1.63	1.63	1.59	1.62	1.57
	MAODV	1.97	2.00	2.04	2.03	2.02	2.07
	ODMRP	1.77	1.86	1.94	1.88	1.90	1.91
Normalized Packet Overhead							
	ADMR	1.79	1.76	1.76	1.75	1.76	1.66
	MAODV	2.16	2.02	2.25	2.06	2.05	2.11
	ODMRP	1.81	1.81	1.81	1.89	1.90	1.91
End-To-End Delay							
	ADMR	2.18	2.38	2.67	2.10	2.15	2.43
	MAODV	3.81	3.64	2.90	4.59	5.59	4.62
	ODMRP	1.44	1.43	1.40	1.57	1.58	1.65
Average Path Length							
	ADMR	1.43	1.51	1.50	1.48	1.50	1.52
	MAODV	1.48	1.47	1.47	1.66	1.69	1.69
	ODMRP	1.44	1.45	1.46	1.54	1.56	1.57

3.5.8. Increasing Network Size

The 200-node scenarios are more challenging than the 100-node scenarios since data forwarding paths are longer, and the number of link and route changes is higher (Section 3.4.2). As a result, the likelihood of packet loss is higher, and the number of falsely triggered repairs increases as well. In addition, in the 200-node scenarios, group members are more sparsely distributed in the network than they would be in a 100-node scenario, which leads to the creation of more forwarding state, though there is less redundant forwarding and as a result packet loss has a stronger impact on the packet delivery ratio.

Tables 3.10 and 3.11 show the ratio of the values of all performance metrics for the 200-node and 100-node scenarios.

ODMRP's packet delivery ratio is the same as in the 100-node scenarios, while ADMR and MAODV's decrease slightly in the multi-source scenarios. This decrease is due to the fact that neither protocol performs as much redundant forwarding as ODMRP and since they also do not perform buffering, more packets are lost during mesh repairs, which are more frequent in the 200-node scenarios due to the longer data forwarding paths. This is also why the decrease in packet delivery ratio is smaller in the experiments in Set II – the data rates are lower, and as a result less packets are

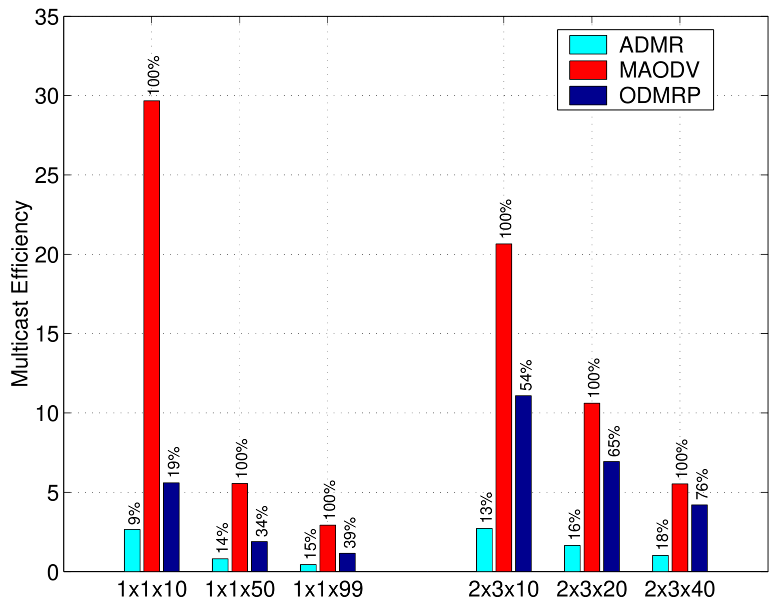

Figure 3.30: Increasing Network Size, Set I. Normalized packet overhead for 200 nodes, 0 pause time, 20 m/s maximum speed.

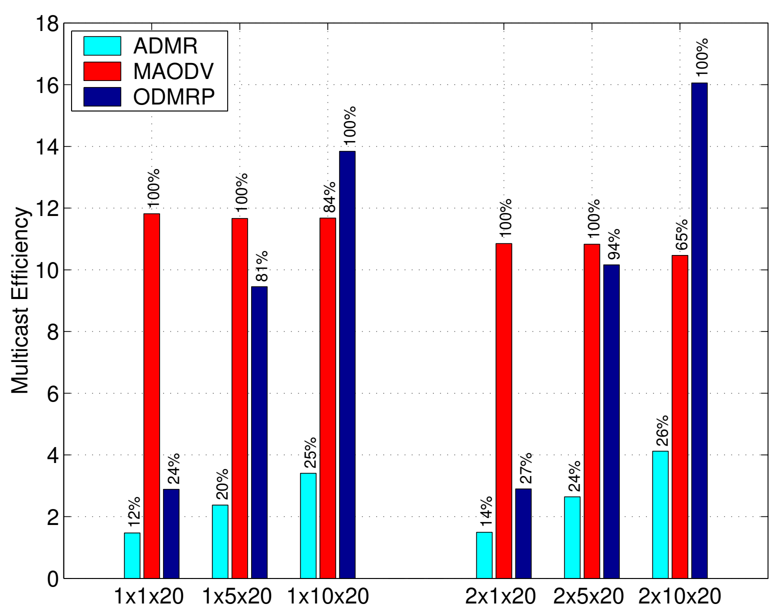

Figure 3.31: Increasing Network Size, Set II. Normalized packet overhead for 200 nodes, 0 pause time, 20 m/s maximum speed.

Table 3.11: Increasing Network Size (Set II). Values represent the ratio between each metric for a 200-node scenario and for the 100-node version of that scenario, both at 20 m/s maximum speed.

Metric	**Protocol**	$1x1x20$	$1x5x20$	$1x10x20$	$2x1x20$	$2x5x20$	$2x10x20$
PDR							
	ADMR	0.99	0.99	0.99	0.98	0.98	0.97
	MAODV	1.00	1.00	1.00	0.95	0.95	0.96
	ODMRP	1.00	1.00	1.00	1.00	1.00	0.99
Control Packet Overhead							
	ADMR	2.34	2.32	2.37	2.40	2.45	3.23
	MAODV	2.27	2.11	2.29	2.11	2.09	2.07
	ODMRP	1.82	1.87	1.90	1.83	1.95	1.85
Data Packet Load							
	ADMR	1.91	2.00	2.07	1.87	2.03	2.09
	MAODV	2.03	2.03	2.01	1.91	1.93	1.91
	ODMRP	1.78	1.92	1.97	1.74	1.90	1.62
Normalized Data Packet Overhead							
	ADMR	1.93	2.03	2.10	1.91	2.07	2.15
	MAODV	2.03	2.04	2.03	2.01	2.04	1.99
	ODMRP	1.78	1.92	1.98	1.74	1.91	1.85
Normalized Packet Overhead							
	ADMR	1.97	2.10	2.18	1.96	2.16	2.48
	MAODV	2.07	2.05	2.07	2.03	2.05	2.01
	ODMRP	1.79	1.91	1.95	1.76	1.92	1.97
End-To-End Delay							
	ADMR	1.50	1.32	1.17	1.46	1.33	2.49
	MAODV	5.07	4.88	3.55	3.63	4.74	4.17
	ODMRP	1.45	1.51	1.54	1.43	1.63	3.11
Average Path Length							
	ADMR	1.48	1.53	1.51	1.40	1.48	1.50
	MAODV	1.54	1.55	1.53	1.60	1.60	1.60
	ODMRP	1.46	1.56	1.56	1.43	1.56	1.69

lost during repair. MAODV's packet delivery ratio is more affected than ADMR's because it uses longer forwarding paths.

Control packet overhead increases by about a factor of 2 for all protocols, with MAODV and ADMR experiencing a slightly higher increase than ODMRP. The increase is due to the higher number of nodes forwarding each flood, and in the case of ADMR and MAODV, due to the higher number of link and route changes, which require more repairs to be performed.

Data packet load is higher for all protocols across all 200-node scenarios, with a larger increase in the scenarios in Set II. Normalized data packet overhead is higher by a factor of 2 for MAODV and by a factor of 1.8 for ODMRP, while ADMR's normalized data packet overhead increases by 1.6 in Set I and to about 2 in the Set II scenarios. Normalized data packet overhead is higher for ADMR in the second set of scenarios since forwarding redundancy in ADMR increases with the number of sources more so than with the number of receivers, unlike ODMRP, where it is the opposite.

Overall, in the 200-node scenarios, ADMR remains the most efficient and generates the least normalized packet overhead, while MAODV generates the most; all protocols generate about twice as much such overhead than in the 100-node scenarios.

End-to-end delay increases by 1.5 to 3 times for ODMRP and ADMR, while MAODV's end-to-end delay increases by 3.5-5 times. The higher increase in MAODV is due to the higher increase in path length and also the traffic concentration due to its use of shared group trees.

Average path length reflects the increase in path length in the network and increases by about 50% for all protocols.

3.6. Summary

In this chapter, I presented an extensive performance comparison of ADMR, MAODV, and ODMRP in a variety of simulation scenarios. I studied mobile networks composed of 100 or 200 nodes, with both a single active multicast group, and multiple active multicast groups in the network, along with a variety of multicast scenarios such as conferencing and single-source vs. multi-source groups. My experiments show that all protocols perform well in terms of packet delivery ratio, and the differences in behavior are in their efficiency. ADMR is the most efficient across all scenarios and typically generates 3 to 5 times less normalized packet overhead than MAODV and ODMRP, because it scales its overhead to meet existing communication demands.

Chapter 4

Routing Characteristics of Networks with Unidirectional Links

In this chapter, I evaluate the routing characteristics and difficulty of ad hoc networks with unidirectional links. In Chapter 5, I introduce unidirectional extensions for multicast routing protocols for ad hoc networks, which I evaluate in the context of ADMR, using the methodology developed in this chapter.

4.1. Motivation and Contributions

Most routing research in wireless ad hoc networks has been based on the simplifying assumption that all links in the network are bidirectional and would therefore work equally well in both directions. However, there exist a variety of circumstances in which this assumption does not hold. *Unidirectional* links can result from factors such as heterogeneity of receiver and transmitter hardware (leading to differing transmission ranges), power control algorithms (in which nodes vary their transmission power based on their current energy reserves), or topology control algorithms (aimed at reducing interference in the network by computing the lowest transmit power that each node needs to stay connected to the network). Another cause of unidirectional links is interference that may be nearer to one node than another, preventing the nearby node from being able to receive some packets sent to it. This interference may be caused by transmissions from other nodes or by other devices (such as jammers) that operate at an overlapping frequency with the one used in the ad hoc network. Although interference is more likely to cause short-lived unidirectional links, a network composed of devices with different wireless ranges may have long-lived or “permanent” unidirectional links for the given positions of the nodes.

One example of a heterogeneous ad hoc network in which unidirectional links may often occur is one in which some wireless nodes are mounted on vehicles, while others are carried by pedestrians. The latter would need to be lighter and use batteries as energy sources, and as a result may not be as sophisticated or powerful as the ones carried on the vehicles and powered by vehicle batteries. The devices carried by the pedestrians may also have different transmit and receive capabilities from each other.

Routing protocols for ad hoc networks may deal with the possibility of unidirectional links in a number of ways:

- Some protocols simply do not consider the problem and thus may create routes that fail to work. For example, a node using DSDV [29] or a conventional distance vector routing protocol assumes that because it has received a routing update from another node advertising routes to certain destinations, it can route packets to those destinations by transmitting them to that node as the next hop; however, if the link between those nodes is unidirectional, all such routes will fail, and any packets sent over those routes will be lost.

- Some protocols handle the presence of unidirectional links by treating all links as if they *might* be unidirectional, in order to avoid the problem above. For example, it is possible to use DSR [19] in this manner. In particular, instead of the regular mechanism of returning a Route Reply using the reverse of the route recorded in the Route Request, which only works over bidirectional links, the protocol can be configured to independently discover the route for returning the Route Reply, which can significantly increase routing overhead. Similarly, Route Maintenance in DSR normally uses direct hop-by-hop acknowledgments, but if all links are assumed to possibly be unidirectional, these acknowledgments must be sent along multi-hop routes that also must be discovered independently, or each acknowledgment may be sent as a multi-hop broadcast [28].
- Some protocols attempt to detect and keep track of unidirectional links as network conditions change, and then either use them for routing [34] or simply ignore them as in AODV [30]. Nodes using AODV attempt to learn which links to neighboring nodes are unidirectional; such a neighbor is remembered in a "blacklist" set, and new routes through such neighbors are ignored and not used for some time. This approach limits the connectivity of the network, and may suffer from either of the two problems outlined above, namely packet losses, and additional and unnecessary overhead, when the mechanism to learn of unidirectional links has not yet learned of a new link or when bidirectional communication becomes possible over a previously unidirectional link.

The examples above illustrate some of the difficulties caused by unidirectional links: the routing protocol may fail, the overhead of using the routing protocol may increase, or both. There is a growing amount of interest in routing techniques that enable efficient routing in networks with unidirectional links, and on the effects of unidirectional links on routing protocol performance (e.g., [1, 26, 28, 34, 40]). There has been little work, however, on evaluating the routing difficulty or *routing characteristics* of ad hoc networks with unidirectional links, for example for characterizing routing scenarios in simulations, analysis, and testbed implementations. For example, routing difficulty is higher when the likelihood that the routing protocol would encounter a unidirectional link is higher, since handling each unidirectional link requires additional mechanisms and control packet transmissions to either ignore or be able to use the link for routing.

In prior work, the difficulty that a unidirectional scenario presents to the routing protocol has frequently been equated to the number or fraction of unidirectional links in the network. However, these metrics present a very limited view of the routing characteristics of the network, and often do not reflect the routing difficulty of a unidirectional network scenario. Consider the following example. AODV [30] is an on-demand routing protocol for ad hoc networks that utilizes ROUTE REQUEST packets to discover routes and to perform route repair when a route breaks. Figure 4.1 shows the number of Route Request transmissions in AODV for a number of scenarios. Since the number of unidirectional links and the fraction of unidirectional links decreases monotonically from left to right in the graph, one would expect that the effort expended by the routing protocol, as expressed by the number of ROUTE REQUEST transmissions, would also decrease monotonically from left to right. However, this is not at all the case.

Routing protocol design and performance analysis require both an understanding of protocol mechanisms and insight into the underlying routing characteristics of the network. In the same way that knowing how many unidirectional links are present in the network is insufficient for understanding complex protocol behavior in a complex network environment, it is also insufficient to

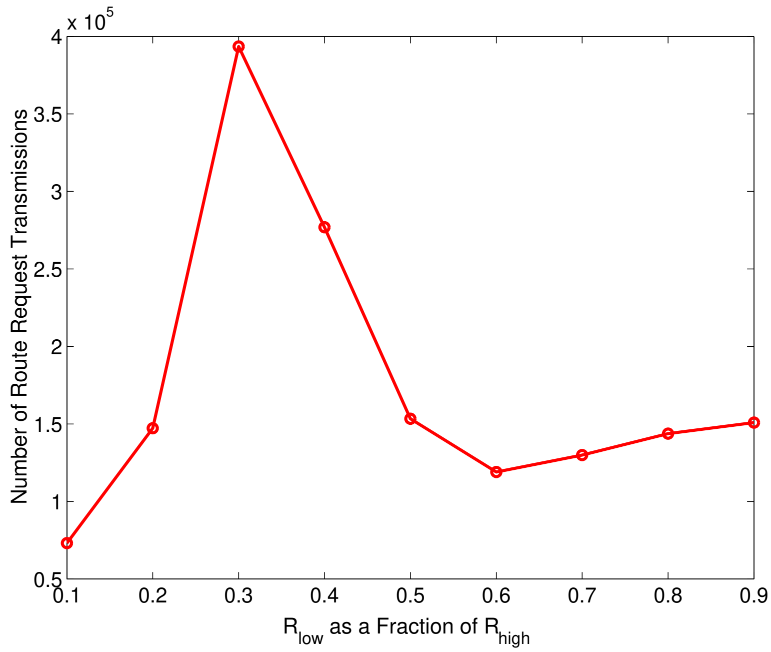

Figure 4.1: Simulation of AODV in mobile ad hoc networks with 10 nodes with transmission range $R_{high} = 250$m and 90 with range R_{low}, all moving according to a random waypoint model with 1 m/s maximum speed and 0 pause time. 35 pairs of nodes exchange one-way CBR traffic at four 64-byte packets per second.

have only a composite metric that would somehow assign a number that would represent the routing difficulty of a unidirectional scenario. To understand protocol behavior in the presence of unidirectional links, we need a detailed characterization of the routing characteristics of the unidirectional network scenarios, expressed by a set of metrics that reflect the multiple aspects of complexity of the network.

In this chapter, I develop a set of metrics to expose the routing characteristics of networks with unidirectional links. Although some of these metrics have been considered separately in prior work, the combined set of metrics is the first to provide a comprehensive view of unidirectional networks from the point of view of routing protocol design and evaluation.

I demonstrate the usefulness of this set of metrics by performing detailed simulation-based analysis of the routing characteristics of networks with unidirectional links generated by the three most common simulation models for generating unidirectional links in ad hoc networks: the *random-power model*, the *two-power model* and the *three-power model*. I use a wide range of parameterizations of each model (e.g., power levels, and number of nodes per power level) and generate both mobile and static unidirectional network scenarios. In the course of the analysis, I show how commonly used sets of metrics may be misleading and may fail to reveal some important characteristics of the network.

The metrics and analysis of the characteristics of the three power models enable protocol designers to better choose a set of network scenarios and parameters that truly explore a wide range of a routing protocol's behaviors in the presence of unidirectional links, and to better understand the subtleties of the interplay between routing mechanisms and network conditions. I analyze the example from Figure 4.1 in Section 4.5.6 with the help of the newly proposed metrics.

4.2. Related Work

Most prior work on routing in the presence of unidirectional links uses unidirectional network scenarios generated by a random-, two- or three-power model. The difficulty or routing characteristics of these scenarios are often not evaluated, and frequently evaluated using only 1 or 2 metrics. In addition, some metrics used in previous work, although useful in characterizing the properties of an ad hoc network were not specifically designed to expose its routing characteristics.

Pearlman et al. [28] present mechanisms for routing over unidirectional links and evaluate these mechanisms in networks with node ranges generated by a two-power model. The authors analyze the routing difficulty of these network scenarios by looking only at the fraction of unidirectional links in each. I build on their work by studying a larger set of parameterizations of the two-power model, and by conducting a comprehensive analysis of the routing characteristics of the scenarios generated by the two-power model.

Sinha et al. [40] propose extensions to the Zone Routing Protocol (ZRP) [10] for handling unidirectional links and evaluate these extensions in networks with unidirectional links produced using the same two-power approach but with less parameterizations and with no characterization of the unidirectional scenarios.

Ramasubramanian et al. [34] present a protocol for a bidirectional abstraction of unidirectional links which handles them below the routing layer. The authors use five unidirectional network scenarios to evaluate the proposed protocol, generated using a two-, a three- and a four-power model. The parameters for generating these scenarios were hand-picked to provide a range of routing difficulty. The authors use only the average number of in-neighbors of a node (neighbors that can reach a node but cannot be reached by it) and average reverse route length to characterize these scenarios.

Marina and Das [26] investigate the potential benefits of using unidirectional links for routing in ad hoc networks, using a random-power model, a two-power model, and a distributed topology control method. They characterize the scenarios generated by each model by analyzing the connectivity of the networks in each scenario in terms of the size and number of strongly connected components in networks with different densities. The authors also measured the difference in path length between a network scenario with unidirectional links and the same scenario but when considering only the bidirectional links. With the exception of the path metric, the rest of the metrics were designed to study overall connectivity rather than the routing characteristics of the network. My analysis is based on a wider range of parameterizations of the random and two-power models, uses a set of metrics to expose the routing characteristics of the network, and also includes the three-power model, not discussed in [26]. Unlike [26], I do not discuss topology control algorithms in this work as each topology control algorithm is unique and thus it is difficult to generalize the results from such analysis.

Ramasubramanian and Mosse [35] also study the connectivity of ad hoc networks with unidirectional links. Their analysis focuses only on static networks with power variations produced by a random-power model, and by a model that determines the directionality of a link according to a uniform random distribution, e.g., each link has a probability P of being unidirectional. The metrics used to characterize network connectivity are reverse route length, the average size and distribution of connected components for reverse route lengths smaller than 1,2,3 or 4 hops, and average path length. In this chapter, I use a broad range of metrics to expose the characteristics of ad hoc networks with unidirectional links from a routing perspective in the context of three different power models.

4.3. Methodology

In this section, I describe the network parameters and power models I used to generate ad hoc networks with unidirectional links.

4.3.1. Network Parameters

I use a network of 100 mobile nodes placed on a rectangular 1200m $\times$ 800m area. The rectangular site was chosen in order to allow for the formation of longer paths (along the longer dimension) than would be possible with a square site having the same area.

The nodes move according to the random waypoint model described in Section 3.4.2 of Chapter 3.

Each simulation was run for 900 seconds of simulated time. I use two maximum speeds, 1 m/s and 20 m/s, and two pause times, 0 seconds (i.e., continuous mobility) and 900 seconds (i.e., a static network scenario).

Power attenuation follows a free-space model up to a reference distance and then a two-ray ground reflection model, as described in [3]. The nominal (omni-directional) range of each device is 250m.

These network parameters are representative of those widely used in ad hoc network performance analysis.

4.3.2. Power Variation Models

The most commonly used methods for generating unidirectional network scenarios for use in ad hoc network simulations are based on models for varying the transmission range of nodes in the network whereby the transmission range of each node (or set of nodes) is set to a fraction of the nominal transmission range.

Two characteristics of these models make them attractive for use in simulations of ad hoc networks: 1) The resulting scenarios are realistic, as they correspond to a potentially common situation in which the ad hoc network consists of heterogeneous devices which may have different transmitter and/or receiver capabilities. 2) The scenarios are straightforward to generate.

For simplicity, in the rest of this chapter, wireless range is discussed in terms of distance instead of transmission power level.

Random-Power Model

In the random-power model, each node is assigned a random transmission power level chosen from a range of transmission powers which correspond to a (R_{min}, R_{max}) range. In this work, I use R_{max} = 250m, the nominal transmission range, and study values of R_{min} ranging from 0m to 225m at 25m increments, thus covering the full set of parameterizations of the random-power model on a 10% discretized scale. This model generates networks that are "completely" heterogeneous – each device may have a different transmission range.

Two-Power Model

In the two-power model, the nodes in the network are divided into two groups, each of which is assigned a different power level, corresponding to a different transmission range, R_{low} and R_{high}, respectively; N_{low} total nodes have range R_{low}, and N_{high} total nodes have range R_{high} (set to the nominal transmission range). R_{low} is varied from 0m to 225m at 25m increments, and N_{low} is varied from 10% to 90% of the total number of nodes, covering the full set of parameterizations of the two-power model on a 10% discretized scale.

Three-Power Model

In the three-power model, the nodes in the network are divided into three groups, each of which is assigned a different power level, corresponding to a different transmission range, R_{low}, R_{medium}, and R_{high} respectively. N_{low} total nodes have range R_{low}, N_{medium} total nodes have range R_{medium} and N_{high} total nodes have range R_{high}(set to the nominal transmission range). In this evaluation, N_{low} and N_{medium} were varied between 10% and 90% at 10% increments, thus covering the whole range of possible combinations of (N_{low},N_{medium},N_{high}) at a 10% discretization. The computational cost of varying R_{low} and R_{medium} to cover all possible combinations even at a 10% discretization is prohibitive. Instead, I used two sets of values for R_{low} and R_{medium}, and only the fraction of nodes in each power group was varied. The values for R_{low} and R_{medium} are 0.2 and 0.4 of the nominal range for the first set of network scenarios (which I call Scenario 1), and 0.6 and 0.7 for the second set of network scenarios (which I call Scenario 2). The differences between the values of any pair in the set R_{low}, R_{medium}, R_{high} in Scenario 1 are twice as large as the corresponding differences in Scenario 1.

4.4. Metrics

In this section, I describe the set of metrics I propose to expose the routing characteristics of unidirectional network scenarios. These metrics cover a wide range of network events and states as expressed by the primitives that directly affect routing protocol performance: links and routes (or paths).

Since most current routing protocols strive to route along the shortest path between a source and a destination, all path-related metrics in the analysis consider only the shortest paths between each pair of nodes that exist in the network, as these are the paths most likely to be used by the routing protocol.

All metrics are computed as averages over all nodes in the network and over the lifetime of the network.

4.4.1. Neighbor-Related Metrics

The number of *bi-neighbors* of a node is the number of neighboring nodes (i.e., nodes within transmission range of the node) with which a node has bidirectional links. The *in-neighbors* of a node are the neighbors of the node who can reach the node but cannot be reached by it. The total number of neighbors of a node is the sum of the in- and bi-neighbors of the node.

The *total number of neighbors* metric reflects the level of connectivity in the network, e.g., the higher the total number of neighbors per node, the higher the number of paths along which nodes can communicate with each other (on average).

The *in-neighbors* metric reflects the average number of unidirectional links that are present in the network.

The *in-neighbors fraction* metric represents the number of in-neighbors as a fraction of the total number of neighbors. It indicates the likelihood that the routing protocol will encounter a unidirectional link rather than a bidirectional link. Each encountered unidirectional link may cause the routing protocol additional overhead to either use the link or to avoid it as described in Section 4.1.

4.4.2. Node Reachability Metrics

The *unreachable nodes* metric represents the average number of nodes to which a node does not have a route and which do not have a route to this node. This metric reflects the extent to which the network is partitioned, and can help protocol designers to differentiate between poor routing protocol performance due to the presence of unidirectional links versus poor routing protocol performance due to the presence of partitions.

The *bidirectionally reachable nodes* metric represents the number of nodes to which the average network node has a shortest path which is bidirectional. This metric reflects the likelihood that the routing protocol would encounter a bidirectional path for routing between a pair of nodes.

A unidirectional path is one that contains at least one unidirectional link. The *mutually unidirectionally reachable nodes* metric represents the number of nodes to which a node's shortest paths are unidirectional, whose shortest paths to it are also unidirectional. This metric reflects the likelihood that the routing protocol would encounter a unidirectional path when trying to route between a pair of nodes.

The *one-way unidirectionally reachable nodes* metric represents the number of nodes to which a node's shortest path is unidirectional, but which do not have a route to this node. Only protocols that do not involve two-way per-hop or end-to-end communication can route to destination nodes that are only reachable in one direction, e.g., a protocol, which uses network-wide broadcast to deliver its data.

4.4.3. Path Characteristics Metrics

The *unidirectional links per path fraction* metric represents the average number of unidirectional links as a fraction of all links in a unidirectional path (over all unidirectional shortest paths in the network). This metric reflects the "unidirectionality" of a unidirectional path, i.e., the level of effort that the routing protocol would need to expend on average when it encounters a unidirectional path. The higher the value for this metric, the higher the routing overhead, as each additional unidirectional link may incur additional overhead in terms of multi-hop acknowledgments, for example.

Path length is an important network routing characteristic because packet latency as well as the likelihood of broken links and packet collisions along a path, are directly proportional to the length of the path. In addition, the discovery and maintenance (monitoring for broken links) of a longer path may incur higher latency, and a higher number of control packet transmissions.

The *average shortest-path length* metric represents the average path length computed over all shortest paths in the network. The *maximum shortest-path length* metric represents the longest path, among all shortest-paths between pairs of nodes, encountered during the lifetime of the network.

The *path length benefit* metric represents the number of hops on average by which the average unidirectional shortest path is shorter than the shortest bidirectional path between a pair of nodes. This metric is computed when the shortest unidirectional path is shorter than any existing bidirectional paths.

4.4.4. Link and Path Change Metrics

The *link changes* metric represents the average number of link changes per second in the network; a link can change from being reachable to unreachable and vice versa, and from unidirectional to bidirectional and vice versa. The number of link changes reflects the level of mobility in the network, which in turn affects routing protocol performance, as it takes time, and often, additional overhead, to react to changes in link directionality and reachability, and to avoid or reduce interruptions in the flow of traffic caused by link changes.

The *unidirectional to bidirectional link changes* metric represents the number of times per second that a unidirectional link becomes bidirectional. Protocols that can detect this condition can automatically reduce their control overhead as they can stop sending acknowledgment packets across multi-hop reverse paths but start sending them directly.

The *bidirectional to unidirectional link changes* metric represents the number of times per second that a bidirectional link becomes unidirectional. This metric indicates how many perceived broken links are actually changes in the directionality of these links. Protocols that can distinguish between a link that has become unidirectional and one that has become disconnected can continue routing along the route containing this link, while ones that cannot make this distinction would initiate route repair procedures and incur more overhead and potentially a disruption in packet delivery.

The *unidirectional to unreachable link changes* metric represents the number of times per second that a unidirectional link becomes unreachable. This metric reflects the frequency of disconnection in the network and can also be used in conjunction with the unidirectional to bidirectional link changes metric to indicate whether unidirectional links in a given scenario are more likely to become bidirectional or unreachable.

Mobility and the resulting link changes affect the paths along which packets are forwarded towards their destinations and may cause the routing protocol to discover alternate routes. The *number of shortest-path changes per second* metric represents the average number of times that a shortest route between two nodes breaks or a shorter path becomes available.

4.5. Results and Analysis

In this section, I analyze the routing characteristics of unidirectional network scenarios produced by the random-, two- and three-power models with the help of the metrics defined in Section 4.4.

The mean values for the metrics characterizing the generated network scenarios for the 1 m/s and 20 m/s cases prior to introducing variations in power (i.e., all nodes have the nominal transmission range) were discussed in Section 3.4.2. The results are reprinted in Table 4.1 in order to make it easier to compare them with those for the static scenario. The total number of neighbors is lower in the static scenario because the random waypoint movement leads to a non-uniform distribution of the nodes on the site such that the density of the network is higher towards the middle of the site and decreases towards the edges of the site [36].

In Sections 4.5.1 through 4.5.4, I consider a random waypoint model with a maximum speed of 1 m/s. Section 4.5.5 discusses the effects of increasing the speed to 20 m/s, as well as the static network case.

Each point (or bar) in the graphs presenting the results is the average of 10 scenarios for the given pause time and speed, generated as described in Section 4.3.1. All metrics are averages computed over all nodes and over the whole duration of the simulated scenario (e.g., 900s).

The axes of the three-dimensional figures presented in this section are not always oriented the same way. This was necessary in order to make visible important features of the graphed data.

In this section, I address the following general questions:

- What is the range of values of each metric for a given model? For example, given a network density, is there a parameterization of the random-power model that will result in scenarios, in which more than 70% of the links in the network are unidirectional? Answering such questions enable protocol designers to pick a model and a set of parameterizations of it that would produce scenarios with the desired routing characteristics and level of difficulty.

Table 4.1: Network Scenarios Without Power Variations

Metric	Static	1 m/s	20 m/s
Total Number of Neighbors	16.12	22.08	23.24
Unreachable Nodes Per Node	0	0.16	0.03
Average Shortest-Path Length	2.92	2.41	2.34
Maximum Shortest-Path Length	7.1	7	7.5
# Link Changes Per Sec.	N/A	3.41	43.1
# Shortest-Path Changes Per Sec.	N/A	42.7	496.2

- What is the relationship between the values of different metrics for a given network density in the context of a given model? For example, is the value of one metric predictive of the values of other metrics?

- Given a unidirectional network scenario, what are its routing characteristics? What are the factors that influence these characteristics? Answering such questions will give researchers insight into the behavior of the network and the effect of this behavior on routing protocol performance. Understanding unidirectional networks would also be useful in designing routing mechanisms and protocols that route over unidirectional links.

4.5.1. Neighbor-Related Metrics

The neighbor-related metrics reveal the level of connectivity and average number of unidirectional links in the network, as well as the likelihood that the routing protocol would encounter a unidirectional link (Section 4.4.1).

Random-Power Model

As the minimum wireless range, R_{min}, increases, the average wireless range in the network increases as well. As a result, more nodes can reach each other (i.e., more links are formed) and the network becomes better connected, as reflected in the monotonic increase in the total number of neighbors per node (Figure 4.2). In addition, for larger values of R_{min}, the differences between the ranges of the nodes become smaller, which leads to a decrease in the likelihood of a link between two nodes being unidirectional. The smaller number of unidirectional links at a larger average range in the network, leads to a decrease in the number of in-neighbors and an increase in the number of bi-neighbors of a node. The increase in the number of bi-neighbors is faster than the decrease in the number of in-neighbors because the formation of one bidirectional link creates one link from the point view of each neighbor, i.e., a total of two one-way links, while the creation or loss of one unidirectional link represents only one one-way link.

In the random-power model, the likelihood of a neighbor being unidirectional reaches a maximum of 50% for $R_{min} = 0$m, and decreases monotonically by about 2 to 5%, down to 3%, as R_{min} increases (Figure 4.3). An approximation of these results can be obtained analytically: in the random-power model, at $R_{min} = 0$m and a maximum range of 250m, half of the nodes have a range smaller than the mean range (i.e., < 125m) and half have a range larger than the mean range, so on average half of the links in the network should be unidirectional as a node is equally likely to have a link with a unidirectional neighbor as it is to have a link with a bidirectional neighbor.

Two-Power Model

Unlike the random-power model in which one parameter (the minimum range) changes between scenarios, in the two-power model two parameters and their interaction determine the routing characteristics of the network; these are the number of low power nodes, or N_{low}, and the wireless range of the low power nodes, R_{low} (Section 4.3.2). Similarly to the random-power model, the total number of neighbors in the two-power model increases monotonically with the increase in average range, i.e., with the decrease in N_{low} and the increase in R_{low}, and also with the decrease in the differences between the high and low power ranges, R_{high} and R_{low} (Figure 4.4). The increase in the total number of neighbors is larger for higher average ranges because the number of links grows faster when the connectivity in the network is higher and the average range is higher.

The total number of neighbors increases faster with the decrease in N_{low} than with the increase in R_{low}, indicating that the effect of the number of low power nodes is stronger than the effect of the magnitude of their range, even though the average range in the network is equally affected by both parameters. The reason for this is that N_{low} has a stronger effect than R_{low} on the in-neighbors component of the total number of neighbors (Figure 4.5): unlike the number of in-neighbors in the random-power model scenarios which depends on the average range in the network, in the two-power model the number of in-neighbors is more dependent on the number of nodes with different ranges (i.e., the pairs of nodes which can form unidirectional links) and to a lesser extent on the magnitude of the differences between these ranges. As a result, the number of in-neighbors increases with an increase in N_{low} up to $N_{low} = N_{high}$, which is the point at which the number of pairs of nodes with different ranges is highest, and when N_{low} becomes larger than N_{high}, the number of pairs of nodes with different ranges begins to decrease, which leads to a decrease in the number of in-neighbors as well.

The values for maximum number of total, in- and bi-neighbors are similar to the corresponding maximum values in the random-power model, while the minimum values in the two-power model reach values that are about four times smaller; the two-power model produces scenarios with a wider range of these particular routing characteristics than the random-power model.

Unlike the random-power model, the number of in-neighbors and the fraction of in-neighbors metrics in the two-power model follow different trends: the number of in-neighbors metric achieves its maximum at $N_{low} = 50$ and $R_{low} = 0.1$ of R_{high}, while the in-neighbors fraction metric achieves its maximum at $N_{low} = 90$ and $R_{low} = 0.1$ of R_{high} (Figure 4.6). The in-neighbors fraction increases monotonically as R_{low} decreases, and for values of $R_{low} = 0.1$ and 0.2, it increases monotonically with an increase in N_{low} as well. For values of R_{low} higher than 0.2, however, the behavior becomes more complex as the increase in the in-neighbors fraction stops and turns into a decrease as N_{low} increases. The point at which this switch occurs is different for different R_{low} values and as R_{low} increases, the switch happens for increasingly lower values of N_{low}. This is due to the fact that for lower values of R_{low}, the effect of the magnitude of the differences in range overwhelms the effect of the number of nodes with different ranges, while for higher values of R_{low} the magnitude of the differences in range decreases and thus the dominant effect is that of the number of low power nodes, which results in the maximum in-neighbors fraction for each value of R_{low} occurring at an increasing smaller value of N_{low}.

Three-Power Model

Similarly to the random- and two-power models, network connectivity in the three-power model, as expressed by the total number of neighbors, increases with the increase in average range, i.e., with the decrease in the number of low and medium power nodes, N_{low} and N_{medium} respectively

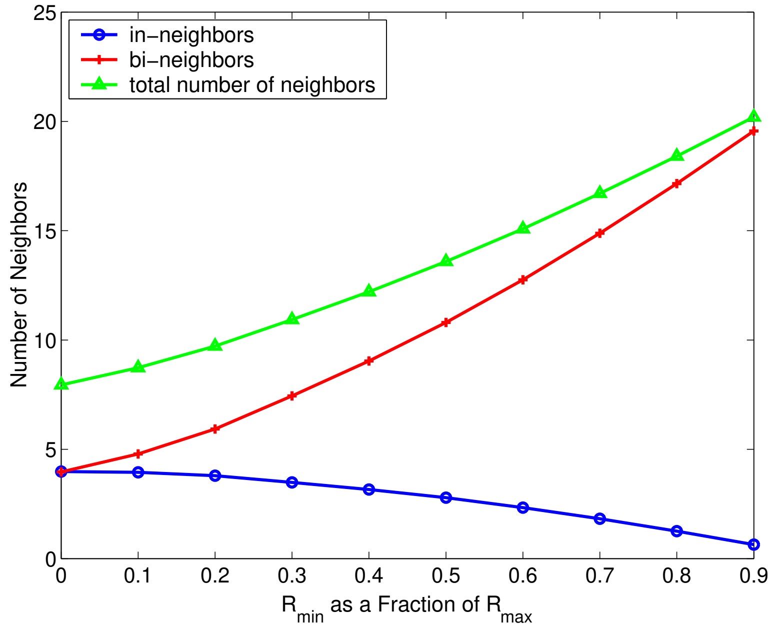

Figure 4.2: Random-Power Model: Number of Neighbors (1 m/s)

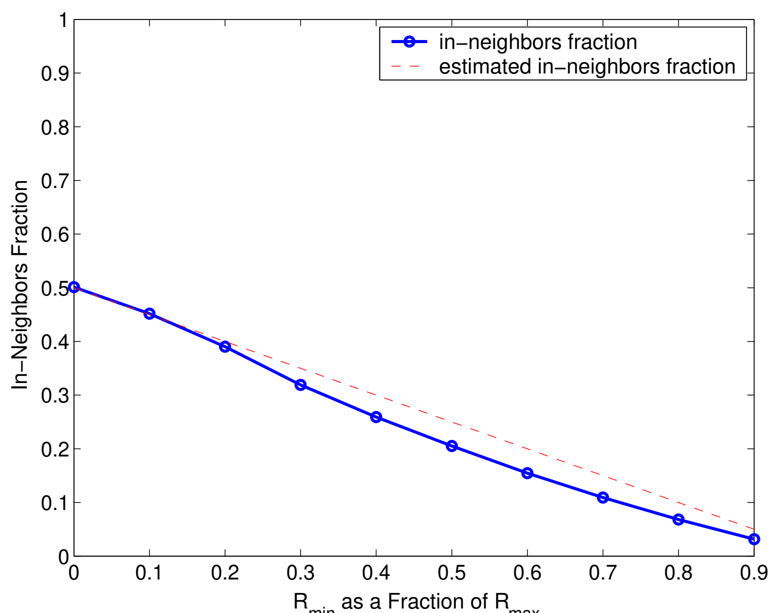

Figure 4.3: Random-Power Model: In-Neighbors as a Fraction of the Total Number of Neighbors (1 m/s)

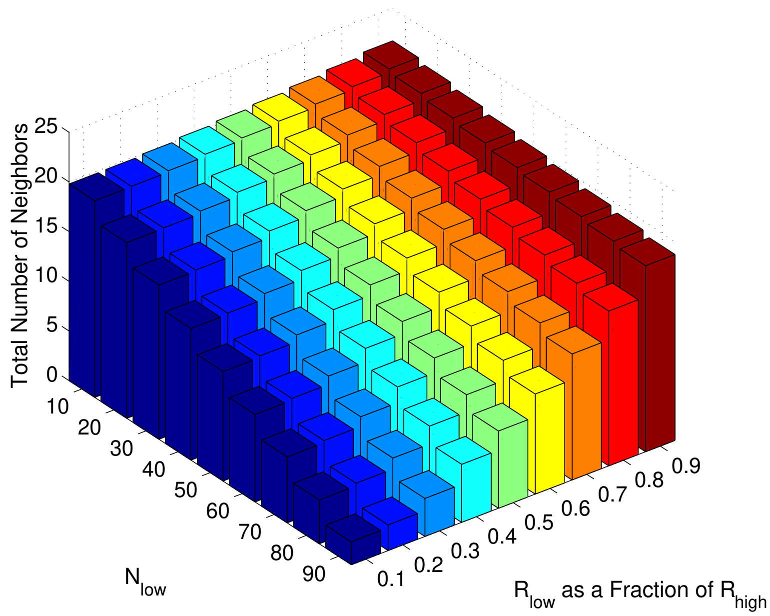

Figure 4.4: Two-Power Model: Total Number of Neighbors (1 m/s)

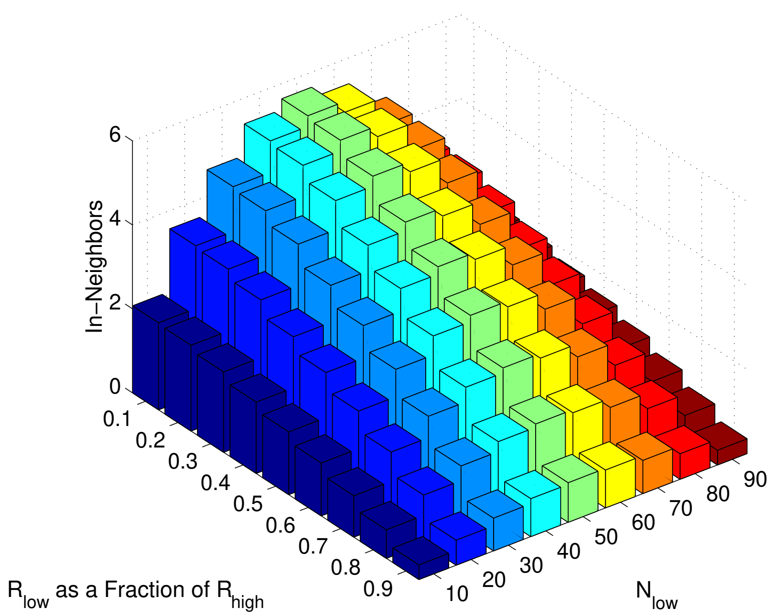

Figure 4.5: Two-Power Model: Number of In-Neighbors (1 m/s)

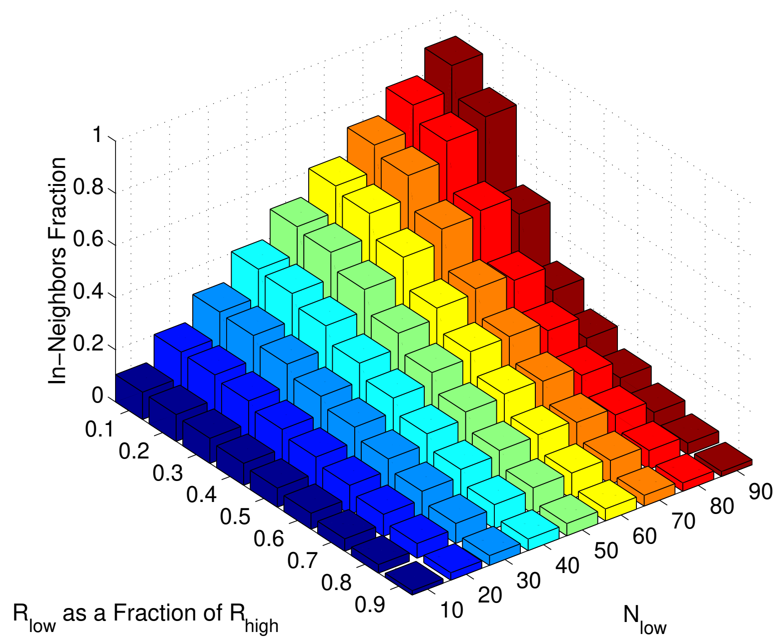

Figure 4.6: Two-Power Model: In-Neighbors as a Fraction of the Total Number of Neighbors (1 m/s)

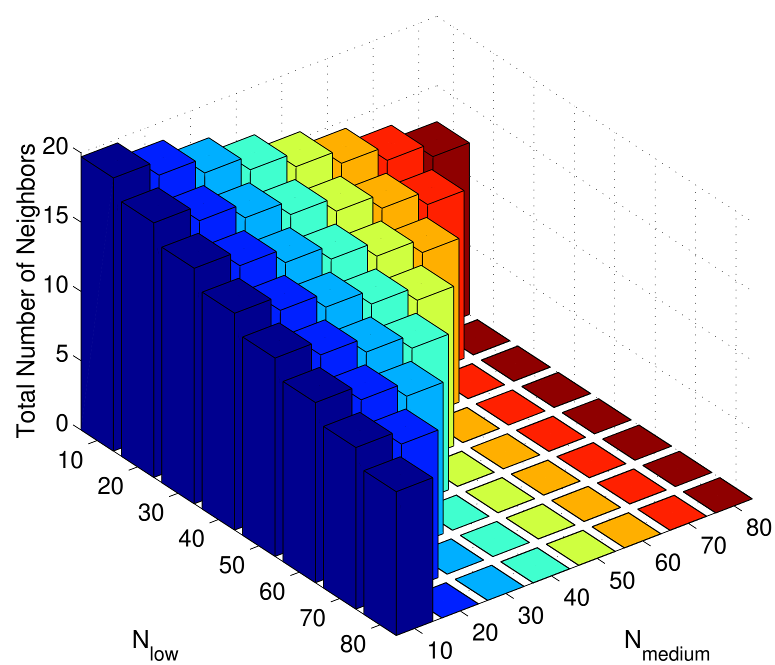

Figure 4.7: Three-Power Model (Scenario 1): Total Number of Neighbors (1 m/s)

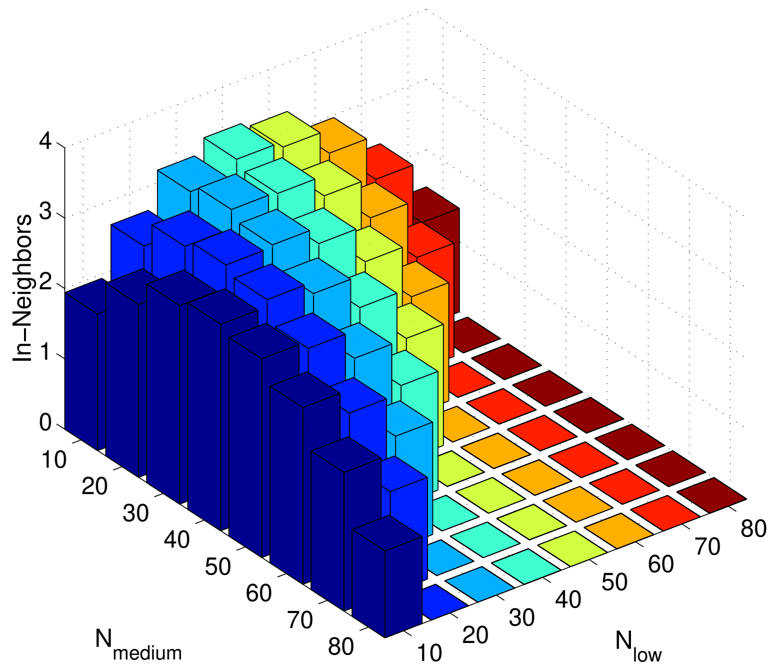

Figure 4.8: Three-Power Model (Scenario 1): Number of In-Neighbors (1 m/s)

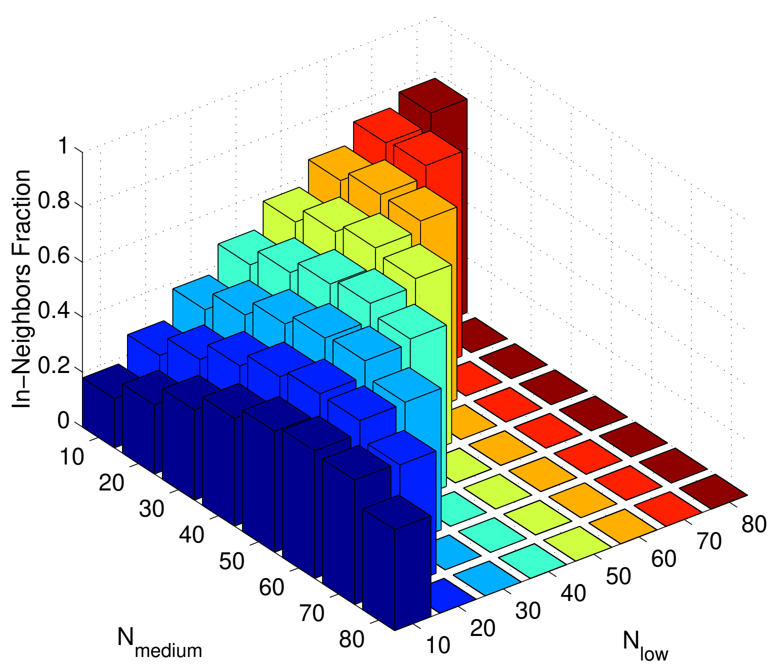

Figure 4.9: Three-Power Model (Scenario 2): In-Neighbors as a Fraction of the Total Number of Neighbors (1 m/s)

(Figure 4.7). The total number of neighbors is generally higher for Scenario 1 because the average range in this scenario is higher and therefore reachability is higher.

The number of in-neighbors in both Scenarios 1 and 2 increases with the increase in N_{low} and N_{medium} (Figure 4.8, Scenario 2 graph has similar shape) up to a point and then decreases. The number of in-neighbors metric is more dependent on the value of N_{low} than on the value of N_{medium} indicated by a higher slope of the graph as N_{low} increases than when N_{medium} increases. This is due to the fact that the low power nodes have larger differences in range with the high power nodes on average and are thus more likely than the medium power nodes to be involved in unidirectional links with the high power nodes. The values for the number of in-neighbors metric in Scenario 2 are higher by 1.55 on average, while the values for the number of bi-neighbors metric are lower by 6.86 on average, as the network is much less connected than in Scenario 2 and the average range is lower.

The maximum value for the number of in-neighbors in both Scenarios 1 and 2 occurs at $(N_{low},N_{medium},N_{high}) = (40,10,50)$. For each successive value of N_{low}, the highest value of the in-neighbors metric occurs at a smaller value of N_{medium}. In addition, up to the maximum value of the number of in-neighbors for each value of N_{low}, N_{high} is 50. This indicates that the effect of the magnitude of the differences between the ranges of different nodes has a higher impact on the number of unidirectional links in these parameterizations of the three-power model than does the number of such differences (in contrast to the two-power model). Since the highest magnitude of the differences in range occurs when a low power node forms a link with a high power node, the highest value for the number of in-neighbors is achieved when the number of low power nodes and high power nodes are as similar as possible, which happens at point (40,10,50). The reason why N_{high} has a higher value than N_{low} at the maximum point is that this is the case in which the highest average magnitude of differences in ranges is created as the difference between R_{high} and R_{medium} is larger than the difference between R_{medium} and R_{low} (Section 4.3.2).

The values of the in-neighbors fraction metric in Scenario 1 follow the trend of the values of the in-neighbors metric except that the in-neighbors fraction metric reaches its maximum (0.23) at $(N_{low},N_{medium},N_{high}) = (50,10,40)$ rather than at (40,10,50), the point at which the number of in-neighbors is highest. The maximum is not at $N_{high} = 50$, since at this value, the number of bidirectional links is high as well (these are bidirectional links between the high power nodes) and compensates for the higher number of unidirectional links. As in the two-power model, a higher number of unidirectional links does not necessarily result in a higher fraction of unidirectional links.

Unlike Scenario 1 but similarly to the two-power model, the in-neighbors fraction metric in Scenario 2 exhibits a different behavior from the in-neighbors metric (Figure 4.9). It achieves values of up to 0.8, which is about 3.4 times higher than Scenario 1, and increases monotonically with successively higher values of N_{low}, reaching its maximum value at (80,10,10). Since the ranges in Scenario 2 are lower and the differences between them higher than in Scenario 1 (Section 4.3.2), the level of unreachability is high (Figure 4.15) and as a result the in-neighbors fraction is also high: when the number of low power nodes is high in Scenario 2, pairs of nodes that are reachable usually include a low power node and a medium or high power node (pairs of low power nodes are likely to be unreachable) and such pairs are likely to be connected via a unidirectional link due to the differences in range.

Summary

The two- and three-power models can produce unidirectional network scenarios with a larger range of values for the neighbor-related metrics than the random-power model. In general, models that

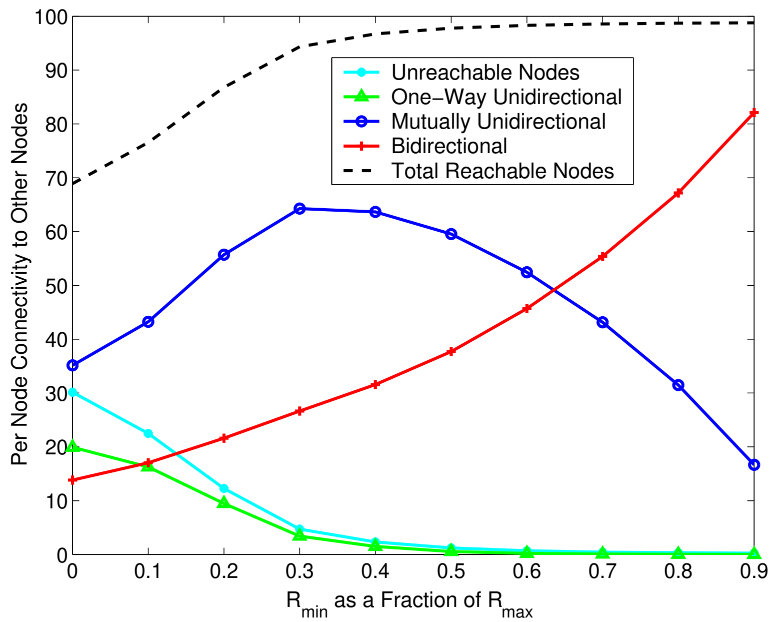

Figure 4.10: Random-Power Model: Per Node Connectivity to Other Nodes (1 m/s)

have a larger number of configurable parameters can achieve a larger set of routing characteristics, but are also more complex to analyze. For example, in the random-power model, the number of in-neighbors and the fraction of in-neighbors follow the same trend, as average range increases across the set of unidirectional scenarios (Section 4.5.1), while in the two-and three-power models, the number of in-neighbors and the in-neighbors fraction may exhibit different behaviors from each other (Section 4.5.1). In addition, some metrics do not exhibit a uniform behavior in scenarios produced by successive parameter values of a given model, e.g., the in-neighbors fraction in the two-power model exhibits a non-uniform behavior for different R_{low} values, achieving its maximum values for different values of N_{low}. These findings indicate that choosing a range of values for parameterizing a model for generating unidirectional links and expecting that the likelihood that a routing protocol would encounter unidirectional links changes monotonically with each successive parameterization is not necessarily a good strategy; the routing characteristics of the scenarios produced by each parameterization need to be analyzed in detail.

4.5.2. Node Reachability Metrics

The node reachability metrics (Section 4.4.2) reflect the likelihood that a node can reach an arbitrary node in the network (e.g., the likelihood of partitions), and the type of reachability it would have with that node, e.g., via a bidirectional, mutually unidirectional, or one-way unidirectional path. The higher the likelihood of encountering a unidirectional path, the higher the overhead the routing protocol would have to expend due to having to route over unidirectional links, or due to trying to find alternate bidirectional paths. Only shortest paths are discussed in this section as these are the paths the routing protocol is most likely to use.

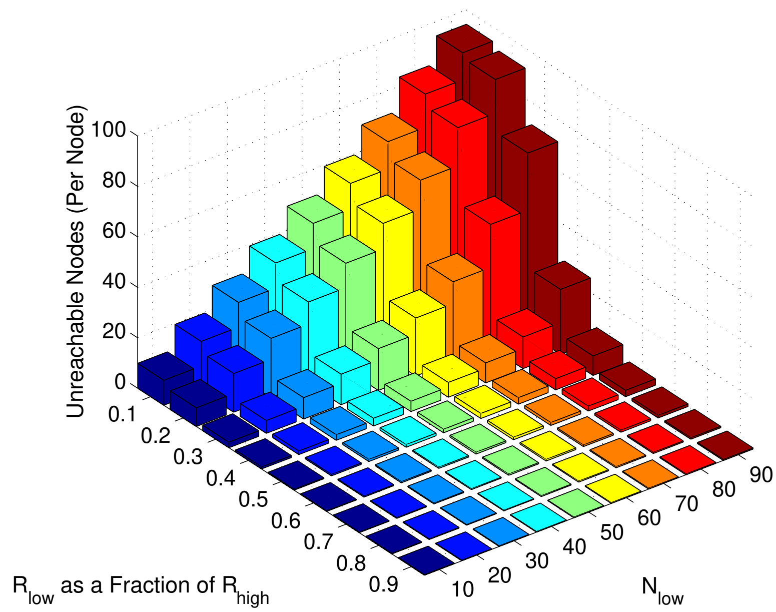

Figure 4.11: Two-Power Model: Number of Nodes Unreachable from a Node (1 m/s)

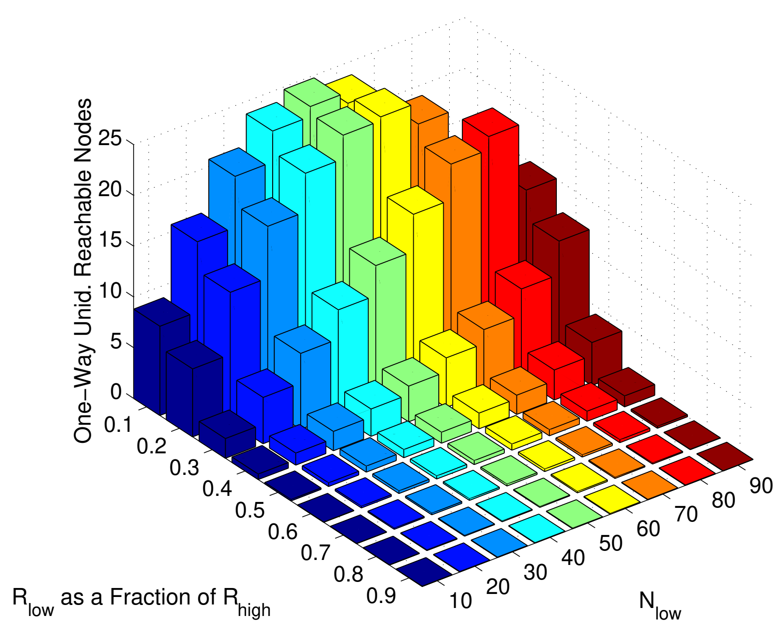

Figure 4.12: Two-Power Model: Number of One-Way Unidirectionally Reachable Nodes Per Node (1 m/s)

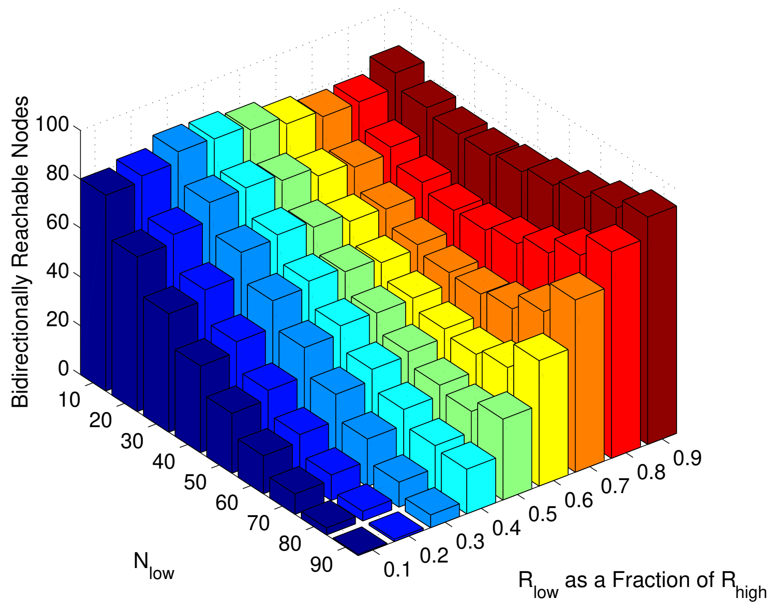

Figure 4.13: Two-Power Model: Number of Bidirectionally Reachable Nodes Per Node (1 m/s)

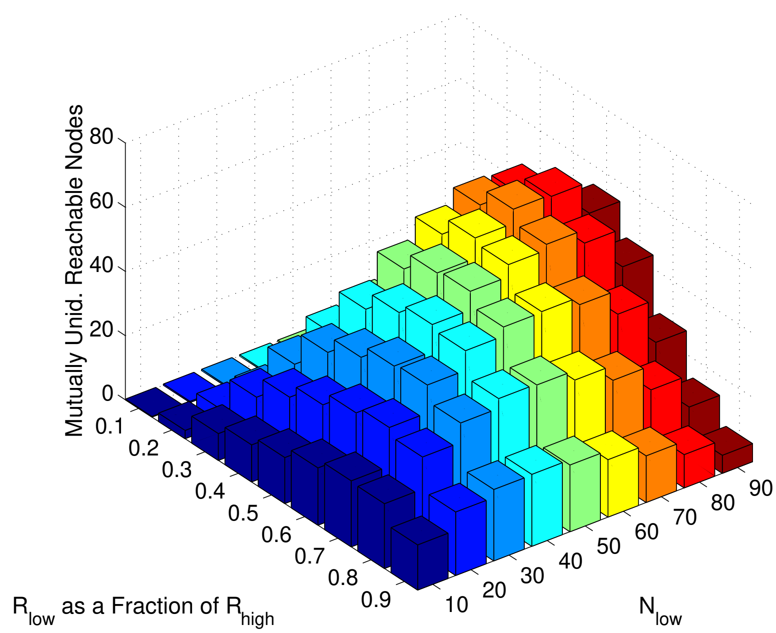

Figure 4.14: Two-Power Model: Number of Mutually Unidirectionally Reachable Nodes Per Node (1 m/s)

Random-Power Model

The number of unreachable nodes per node starts out high (30% of all nodes) at $R_{min} = 0$m (Figure 4.10). As the average range in the network increases, previously unreachable nodes become unidirectional or bidirectional neighbors, and as a result, the number of unreachable nodes drops sharply and becomes negligible at $R_{min} = 0.3$-0.4. A significant number of nodes are reachable only one-way at $R_{min} = 0$m as well. The similarity between the curves of the unreachable nodes and one-way unidirectionally reachable nodes metrics is due to the fact that poorly connected nodes (e.g., ones close to the edges of the site), flap between being barely connected and being partitioned as they move about. In addition, the smaller the average range, the more likely it is for a node to be partitioned (rather than unidirectionally connected to the network). As a result the unreachable nodes graph has higher values than the one-way unidirectionally reachable nodes graph, and the differences are more pronounced for lower values of R_{min}.

The number of bidirectionally reachable nodes per node rises monotonically as the average range in the network increases (similarly to the number of bi-neighbors metric, Figure 4.2). The number of unidirectionally reachable nodes per node, on the other hand, does not follow the monotonic decrease in the number of in-neighbors but instead rises up to $R_{min} = 0.3$ and only then begins to decrease. This initial increase is due to the initial increase in connectivity and average wireless range which conceptually draws nodes closer together as a result of which unidirectional links become bidirectional and unreachable links become unidirectional. The number of unidirectionally reachable nodes per node starts to fall when R_{min} becomes larger than 0.3 of R_{max}, since node unreachability becomes negligible and the dominant effect of the increase in average range is that of unidirectional links becoming bidirectional.

The number of unidirectionally reachable nodes is higher than what the average in-neighbors fraction would suggest (Section 4.5.1) because the addition of unidirectional links has a stronger impact on the unidirectional paths metric than on the in-neighbors fraction metric; the addition of one unidirectional link may make multiple paths unidirectional and vice versa, the removal of one unidirectional link may make multiple paths bidirectional.

Two-Power Model

Unreachability increases with a decrease in average range, and at $N_{low} = 90$ and $R_{low} = 0.1$ of R_{high}, it reaches a value of 98% (Figure 4.11), which is over three times the maximum number of unreachable nodes per node in the random-power model. Unlike the random-power model, the one-way unidirectionally reachable nodes metric (Figure 4.12) does not follow the same trend as the unreachable nodes metric because in addition to average range, it is also influenced by the number of pairs of nodes with differing ranges; in the random-power model both of these parameters change together as only one parameter (the minimum range) is varied. As a result, the maximum value for the one-way unidirectionally reachable nodes metric occurs at the same values of R_{low} and N_{low} as the highest value of the in-neighbors metric ($R_{low} = 0.1$ and $N_{low} = 50$).

The number of bidirectionally reachable nodes per node increases monotonically with N_{low}, for values of $R_{low} = 0.1$ and 0.2 (Figure 4.13). For larger values of R_{low} it decreases with an increase in N_{low} up to a point, and then starts to increase since N_{low} becomes higher than N_{high} after which point the majority of bidirectional paths are between low power nodes (since R_{low} is high) and these paths dominate the overall shortest-path reachability. As R_{low} increases, the number of bidirectional paths increases as well, except that for values of N_{low} smaller than 40 there is a temporary decrease in the number of bidirectional paths around $R_{low} = 0.6$ and 0.7. This decrease is due to the fact that at some point nodes that were previously unreachable are now reachable due to the higher average range, however, they are reachable only unidirectionally as the higher range is not high enough to make them bidirectional neighbors.

The number of mutually unidirectionally reachable nodes metric behaves differently from the number of in-neighbors metric. It increases for higher values of N_{low} and R_{low} (for low N_{low} and R_{low} values), and then begins to decrease (Figure 4.14). The initial increase is due to increased connectivity in a poorly connected network in the case of N_{low}, and an increased number of pairs of nodes with differing ranges in the case of R_{low}. The subsequent decrease is due to a decrease in the number of unidirectional links.

Three-Power Model

There is very little unreachability in Scenario 1 (up to 2.7 unreachable nodes per node, figure not shown). The number of unreachable nodes per node rises with a decrease in average range, i.e., for higher N_{low} and N_{medium}. Similarly to the random-power model, the number of one-way unidirectionally reachable nodes graph tracks the unreachable nodes graph very closely and has slightly lower values.

In Scenario 2 unreachability also increases with both an increase in N_{low} and N_{medium}, except due to the smaller values of R_{low} and R_{medium}, the unreachability is much higher than in Scenario 1, reaching up to 90 unreachable nodes per node (i.e., 91% of the nodes) (Figure 4.15). Unlike Scenario 1, in Scenario 2 the number of one-way unidirectionally reachable nodes initially increases with the increase in average range, and then starts to decrease along with the number of unreachable nodes due to the increasing connectivity in the network which leads to the formation of a higher number of mutually unidirectionally reachable and bidirectionally reachable nodes (Figures 4.16 and 4.15).

The mutually unidirectionally reachable nodes and the bidirectionally reachable nodes' graphs in Scenario 1 follow complementary trends (Figures 4.17 and 4.18) as virtually all paths are either bidirectional or mutually unidirectional due to the high level of connectivity. In Scenario 2, the bidirectionally reachable nodes graph follows a complementary trend to the unreachable nodes graph (Figure 4.15) as nodes are more likely to be unreachable or bidirectionally connected to other nodes than to be unidirectionally connected, due to the large differences between the ranges of different nodes. The number of mutually unidirectionally reachable nodes in Scenario 2, increases with an increase in N_{medium} as the medium power nodes have a stronger impact on the growth of the number of unidirectional paths in the network than do the low power nodes, since the medium power nodes are more likely to participate in unidirectional links than the low power nodes which are more likely to be unreachable (due to their small ranges). Unlike in the other power models and Scenario 1, the connectivity of the network remains low even in the most connected configuration of Scenario 2 and as a result, the number of mutually unidirectionally reachable paths does not reach a peak value and then top off but only increases in this set of experiments. The in-neighbors fraction does top off (Figure 4.9) but the number of unidirectional links in the network has a weaker effect on the composition of paths than on the fraction of unidirectional links, as one unidirectional link may cause multiple paths to be unidirectional.

Summary

The analysis of the node reachability metrics shows that attempting to predict the routing characteristics of the network based on knowledge of only the number or fraction of unidirectional links in the network can be misleading as the values of these metrics are not indicative of the types of

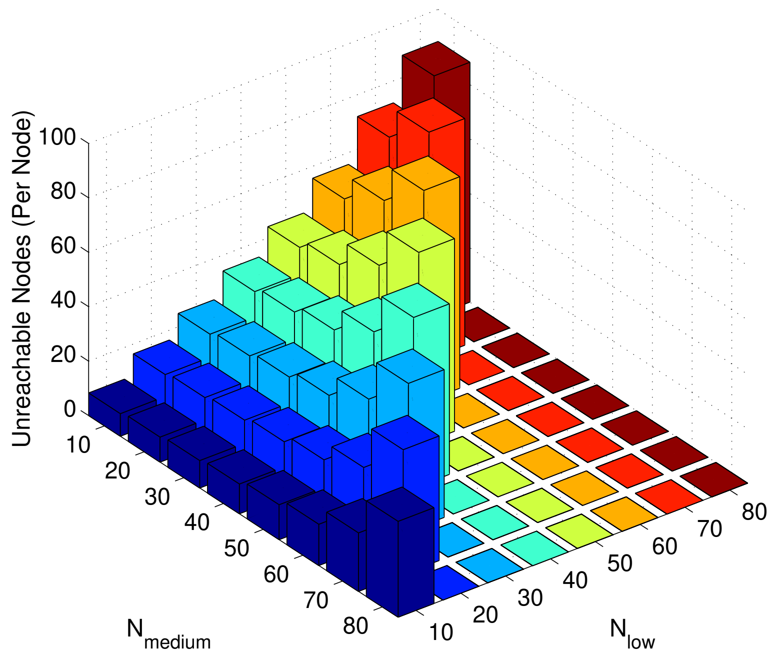

Figure 4.15: Three-Power Model (Scenario 2): Number of Nodes Unreachable from a Node (1 m/s)

paths that the routing protocol is likely to encounter. In addition, picking seemingly similar parameterizations can lead to network scenarios with very different characteristics, and vice versa, picking seemingly different parameterizations can lead to network scenarios with similar routing characteristics. Understanding the reachability characteristics of the network would enable protocol designers to better analyze routing protocol behavior, as different routing mechanisms are sensitive to the presence of different kinds of paths. Similarly to the neighbor-related metrics (Section 4.5.1), the two- and three-power models provide a wider range of values for the reachability metrics but are more complex to analyse than the random-power model.

4.5.3. Path Characteristics Metrics

The path characteristics metrics (Section 4.4.3) characterize a scenario in terms of the average and maximum shortest-path length in the network, the path length benefit of using a shorter unidirectional path instead of a longer bidirectional path, and the level of unidirectionality of a unidirectional path, i.e., the fraction of links on each unidirectional path that are unidirectional. Only shortest paths are discussed in this section as these are the paths the routing protocol is most likely to use.

Random-Power Model

As the average range in the network increases, so does the number of bidirectional paths. As a result, the likelihood of encountering a bidirectional path that is as short as the corresponding unidirectional path between a pair of nodes increases as well. This trend is reflected in the path length benefit of using unidirectional links which decreases from 1.4 to 1 and remains equal to 1 for values of R_{min} greater than 0.6, which is the point at which the dominant reachability between nodes in the network starts to be via bidirectional paths (Figure 4.10). The path length benefit in the random-power model is relatively small (Figure 4.19).

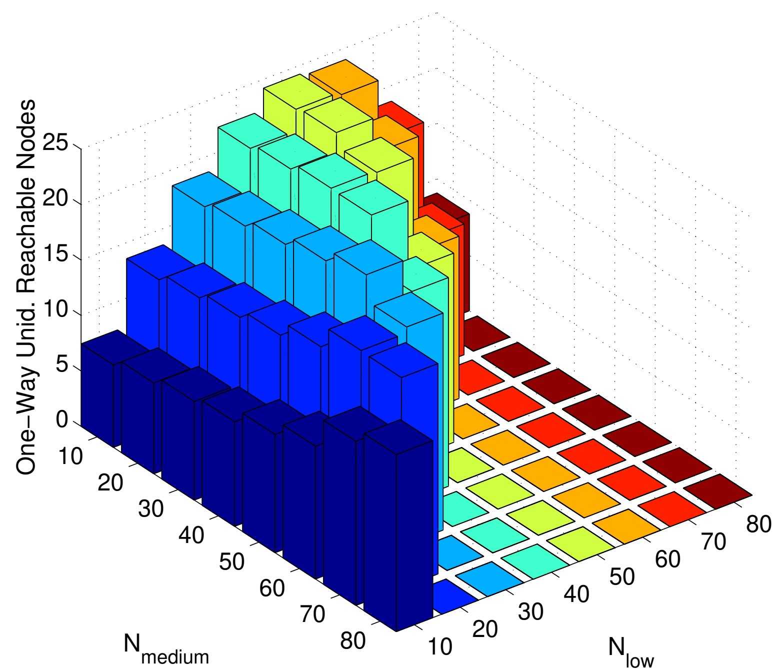

Figure 4.16: Three-Power Model (Scenario 2): One-Way Unidirectionally Reachable Nodes Per Node (1 m/s)

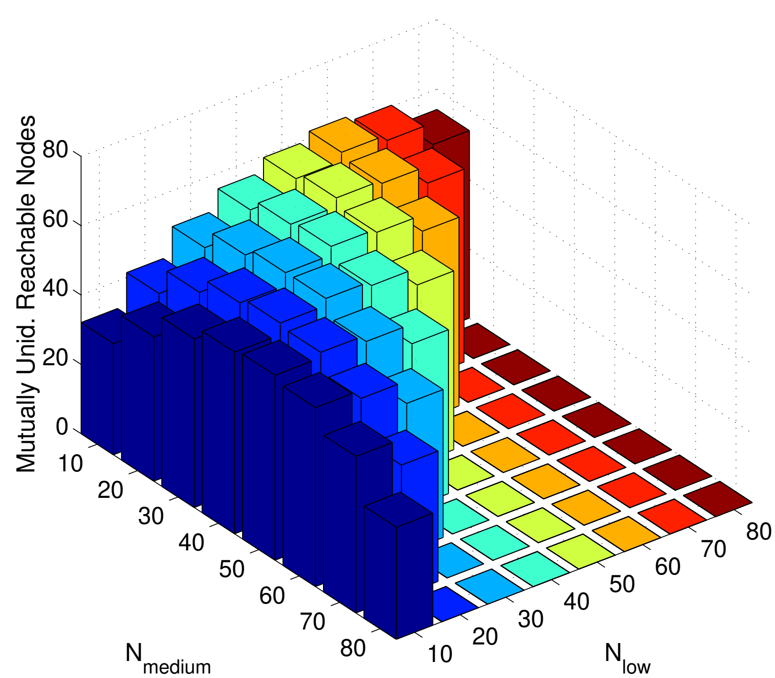

Figure 4.17: Three-Power Model (Scenario 1): Mutually Unidirectionally Reachable Nodes Per Node (1 m/s)

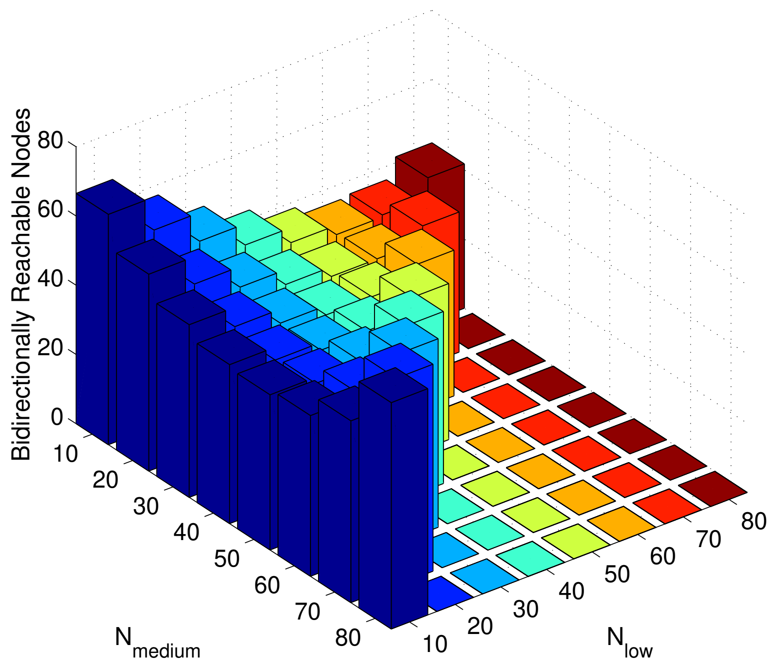

Figure 4.18: Three-Power Model (Scenario 1): Bidirectionally Reachable Nodes Per Node (1 m/s)

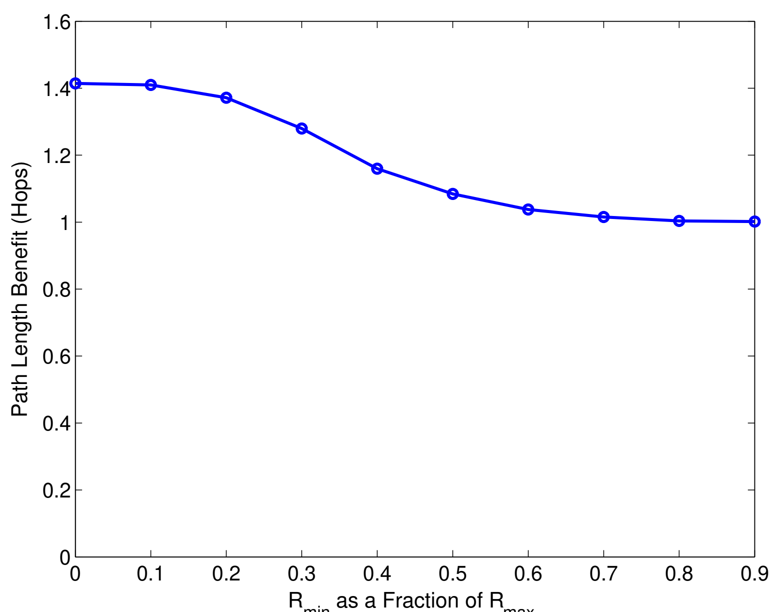

Figure 4.19: Random-Power Model: Path Length Benefit of Using a Unidirectional Path over a Bidirectional Path (1 m/s)

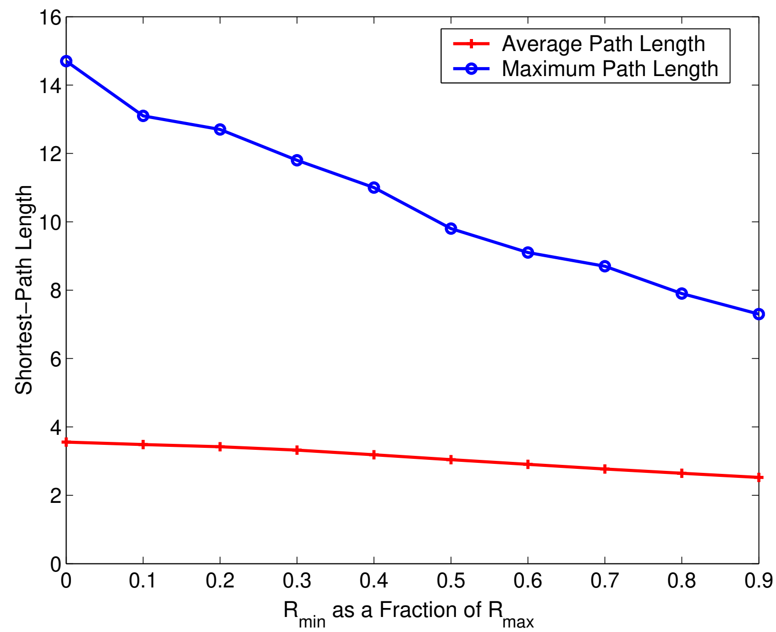

Figure 4.20: Random-Power Model: Path Length (1 m/s)

The average and maximum shortest-path lengths in the network decrease as connectivity increases and more paths become available. The maximum shortest-path length varies between 14.8 and 7.3 for $R_{min} = 0$ and $R_{min} = 0.9$ of R_{max}, respectively, while the average shortest-path length varies between 3.6 and 2.3 for the same R_{min} values (Figure 4.20). The decrease in average shortest-path length is small due to the dominant contribution of 1- and 2-hop paths, whose number increases the fastest with increased connectivity.

The unidirectional links per path fraction is about 45% at $R_{min} = 0$m and decreases monotonically to about 38%. There are several competing factors that influence this metric: 1) As R_{min} increases, there are less unidirectional links and since links are shared between paths, the conversion of a unidirectional link to a bidirectional link (as a result of the higher average range) may affect more than one path (leading to a lower unidirectional links per path fraction). 2) Path length decreases with an increase in R_{min} and shorter paths do not share as many unidirectional links because there are less nodes they can have in common (leading to a lower unidirectional links per path fraction). 3) Shorter paths can have a higher fraction of undirectional links with a smaller number of unidirectional links on them (which leads to a higher unidirectional links per path fraction). The combination of these factors leads to a slow decrease in the unidirectional links per path fraction.

Two-Power Model

The average path length benefit of using unidirectional paths reaches a maximum of 2.16 hops, which is higher than in the random-power model. This is due to the wider range of values for the number of unreachable and unidirectionally reachable nodes in the two-power scenarios (Section 4.5.2). As a result, at low values of N_{low} and R_{low}, where the number of unidirectional links is high, longer bidirectional paths are less likely to exist than shorter ones; a single unidirectional link can make the path unidirectional, and the more links in a path, the more likely it is that one of them may be unidirectional. As in the random-power model, the path-length benefit declines with the increase in the number of bidirectional paths.

The wider range of values for the unreachable and unidirectionally reachable nodes (relative to the random-power model), leads to a wider range for the shortest and maximum shortest-path metrics as well, which vary between 1.87 and 4.57, and 7 and 21.3, respectively. The range of values for the unidirectional links per path metric is also wider in the two-power model (0.12 to 0.59). Similarly to the random-power model, this metric is highest for scenarios with the highest in-neighbors fraction and lowest for scenarios with the lowest in-neighbors fraction, (N_{low},R_{low}) = (90,0.1) and (10,0.9) respectively.

Three-Power Model

The path length benefit in both Scenarios 1 and 2 decreases with the increase in the number of bidirectional paths, just as in the random- and two-power models. In Scenario 1, the maximum path length benefit is only 1.22, because the number of unreachable nodes is very low. The fraction of nodes that a node cannot reach on average reaches 91% in Scenario 2 leading to a higher path length benefit of using unidirectional links (up to 2.9 hops).

Similarly to the random- and two-power models, path length increases with an increase in unreachability, for both Scenarios 1 and 2. The average shortest-path length and maximum shortest-path length reach values of 3.57 and 12.7 in Scenario 1, and 4.45 and 20.8 in Scenario 2 respectively.

The unidirectional links per path fraction metric achieves ranges of 0.34 to 0.42 for Scenario 1 and 0.33 to 0.57 for Scenario 2, which are wider than the ones in the random-power model but narrower than the ones in the two-power model.

Summary

The analysis of the path characteristics metrics reveals that unidirectional links can have a significant impact on the length of network routes, and that unidirectional paths in the three power models generally contain a significant fraction of unidirectional links. The path length benefit of using unidirectional paths is fairly small though of course, in some cases bidirectional paths are actually not available, so protocols that attempt to find a bidirectional path and are not able to route over unidirectional paths, are not going to be able to deliver their data and may incur unnecessary overhead.

As in the case of the neighbor-related and reachability metrics, the two- and three-power models provide a larger set of values for each metric than the random-power model and thus provide protocol designers with more choices for experimenting with routing protocols in unidirectional networks with different characteristics.

4.5.4. Link and Path Changes Metrics

The link and shortest-path changes metrics reflect the level of mobility in the network and the challenge to the routing protocol in terms of distinguishing between changes in the directionality of a link versus a link breaking, which affects protocol efficiency (Section 4.4).

Random-Power Model

The number of links in the network increases with an increase in average range and as a result, the number of link changes increases as well (Figure 4.21); node movement in the presence of a higher number of links causes a higher number of link changes. However, the fraction of all links that experience a change actually decreases from 0.009 at $R_{min} = 0$ to 0.0065 at $R_{min} = 0.9$ of R_{max}.

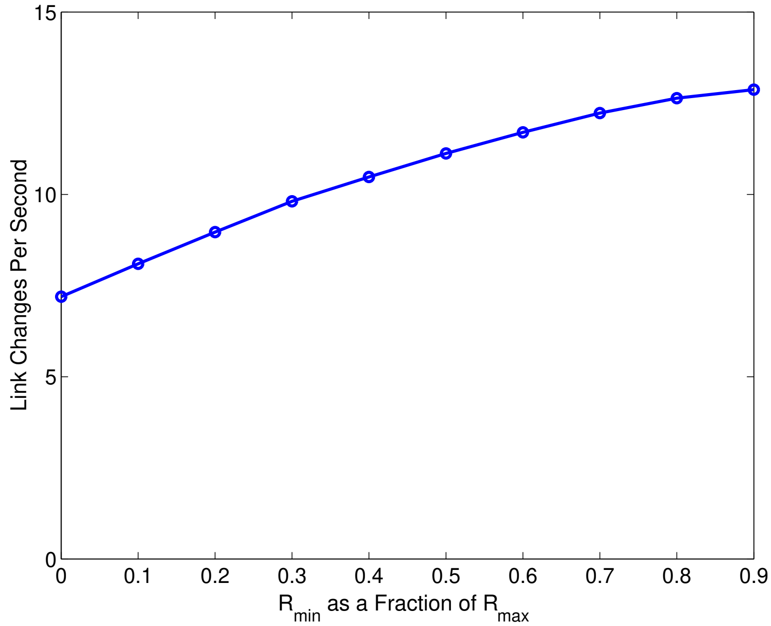

Figure 4.21: Random-Power Model: Number of Link Changes Per Second (1 m/s)

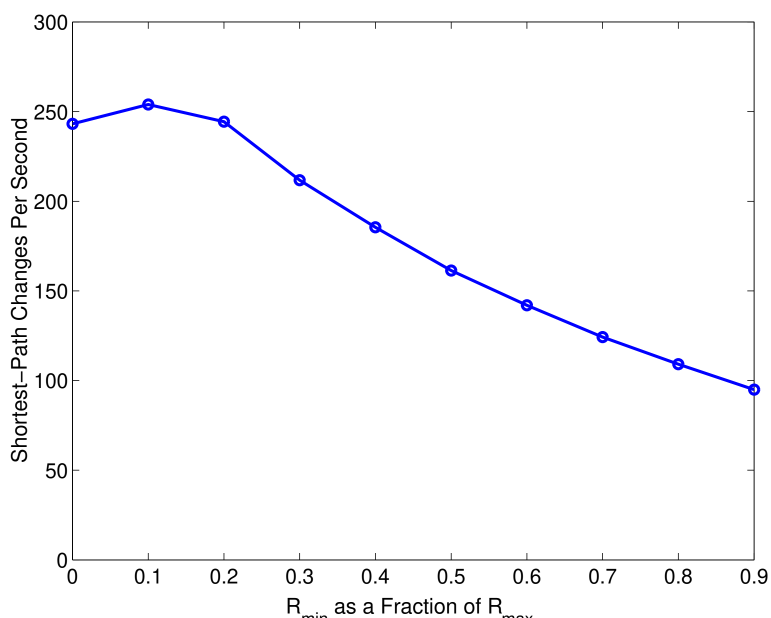

Figure 4.22: Random-Power Model: Number of Shortest-Path Changes Per Second (1 m/s)

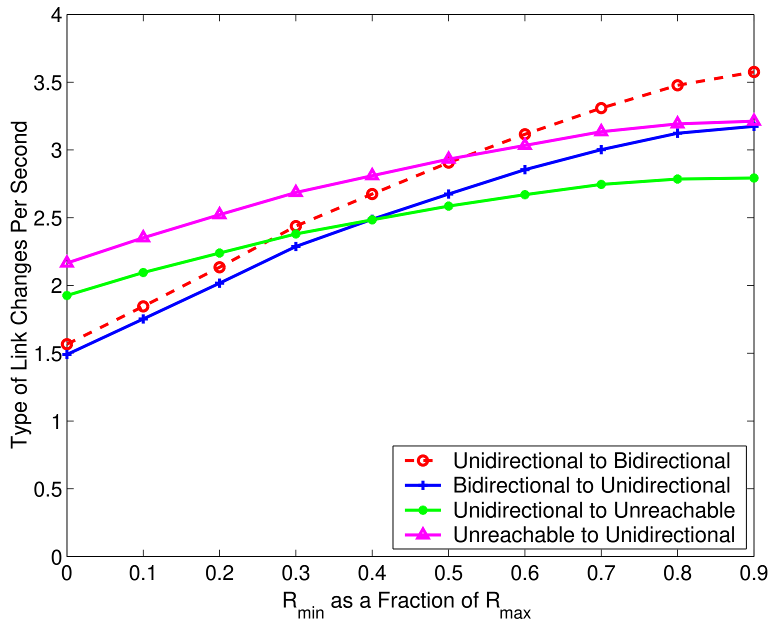

Figure 4.23: Random-Power Model: Link Changes Per Second (1 m/s)

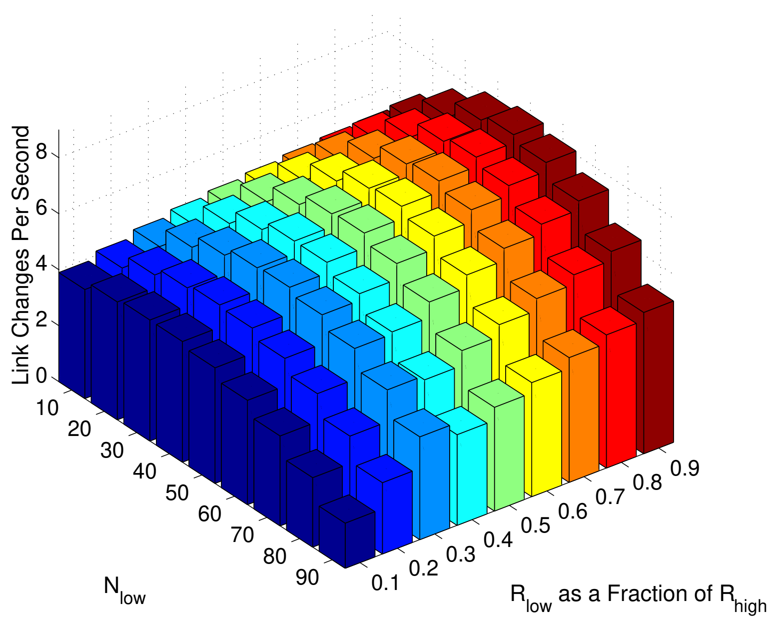

Figure 4.24: Two-Power Model: Number of Link Changes Per Second (1 m/s)

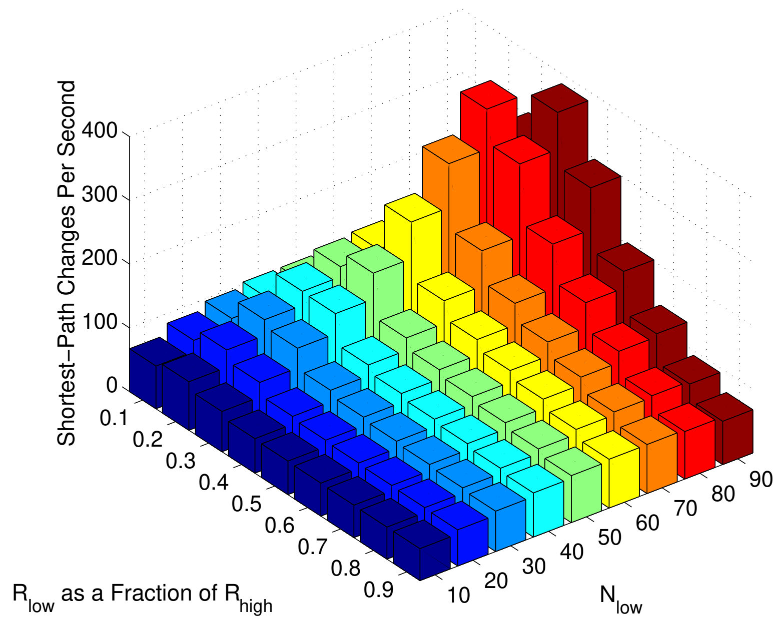

Figure 4.25: Two-Power Model: Number of Shortest-Path Changes Per Second (1 m/s)

This decrease is due to the fact that the rise in the number of links outpaces the increase in the number of link changes since at a higher average range, the distance to be traversed and thus the time required to cause a link change is longer.

The number of shortest-path changes and the fraction of shortest paths that change decrease with an increase in average range (Figure 4.22), with the fraction of shortest-path changes decreasing from 0.035 to 0.01 at $R_{min} = 0$ and 0.9 of R_{max} respectively. This is due to the fact that as average range increases, connectivity increases, leading to the presence of redundant links (i.e., links that do not improve reachability or path length) and when a node encounters such a link or when such a link breaks, the shortest paths between it and other nodes are unaffected. This effect is reflected in the ratio of the fraction of shortest-path changes to the fraction of link changes which starts out at 3.9 and decreases monotonically, reaching a value of 1.51 at $R_{min} = 0.9$ of R_{max}.

I have divided link changes into several groups to highlight several different types of link-related events that may affect the routing protocol (Section 4.4). The only statistically significant differences between the four types of link changes are for the smallest and largest values of R_{min} (Figure 4.23). For small values of R_{min}, the likelihood of unidirectional links becoming unreachable and unreachable links becoming unidirectional is higher than the likelihood of unidirectional links becoming bidirectional and bidirectional links becoming unidirectional. For the highest values of R_{min}, this trend is reversed. These effects are due to the fact that at low values of average range, movement is more likely to cause two nodes to go from being unidirectional neighbors to being disconnected and vice versa, while for high values of the average range in the network, movement is more likely to cause them to become bidirectional neighbors.

Two-Power Model

The number of link changes per second is influenced by the number of links in the network and also by the number of unidirectional links in the network (Figure 4.24). The highest number of link changes occurs at the point of highest value for the number of in-neighbors ($N_{low} = 50$) and highest number of links ($R_{low} = 0.9$). Even though the maximum number of link changes per second in the two-power model is 8.16, while the maximum number of link changes per second in the random-power model is 14.17, the maximum fraction of link changes approaches 0.01 in both models.

The fraction of shortest-path changes, shown in Figure 4.25, reaches a maximum of 0.12 which is 3.43 times higher than the maximum value of the fraction of shortest-path changes in the random-power model. The fact that a similar fraction of broken links in the two models causes a different fraction of broken routes is due to the differences in path length between the models (Sections 4.5.3 and 4.5.3); longer paths are more affected by link changes than shorter ones.

Similarly to the random-power model, the number of changes between unreachable and unidirectional links is higher than the number of changes between unidirectional and bidirectional links when average range is low and the reverse is true for high average range.

Three-Power Model

The number of link changes in Scenario 1 increases with average range, e.g., as both N_{low} and N_{medium} increase (Figure 4.26), and also with the number of unidirectional links in the network, which is why the link changes curve follows the same trend exhibited by the number of in-neighbors curve. When the number of unidirectional links is highest, the number of link changes is also highest as unidirectional links participate in the most types of link changes – unidirectional to bidirectional or unidirectional to unreachable and vice versa. Scenario 2 follows the same trend as well, except there are less link changes (about 2 at maximum) since the connectivity is lower as there are less links overall.

Similarly to the random- and two-power models, the number of shortest-path changes is influenced by the lengths of the paths in the network. The shortest-path changes in Scenario 1 increase monotonically with both N_{low} and N_{medium} following the average shortest-path length trend. The number of shortest-path changes in Scenario 2 (Figure 4.27) follows the same pattern except that it has higher values than Scenario 1 up to $N_{low} = 50$ (due to the much lower connectivity) and after that starts to decline (rather than increase monotonically as in Scenario 1) due to the higher level of unreachability (Figure 4.15). The fraction of link changes reaches the same maximum value of 0.01 in both Scenarios as in the random- and two-power models, whereas the shortest-path changes fraction in Scenario 2 reaches a maximum of 0.08 (which is smaller than in the two-power model and larger than in the random-power model) and is about 4 times higher than the maximum value in Scenario 1 (which itself is higher than the maximum value of the metric in the random-power model). As mentioned in Section 4.5.4, a similar fraction of broken links in all of the models causes a different fraction of shortest-path changes in each model due to the differences in path length between scenarios generated by each model. (Section 4.5.3); longer paths are more affected by link changes than shorter ones.

The different types of link changes in Scenario 1 have similar values to each other and follow the same trend as the link changes metric.

In Scenario 2, due to the higher level of unreachability, the unidirectional to unreachable link changes have higher values than the unidirectional to bidirectional link changes and vice versa.

Summary

The analysis of the link and path change metrics shows that knowing the number or fraction of link changes, does not always help in predicting the number and fraction of shortest-path changes, as these are also dependent on path length and the types of paths that exist in the network. My findings

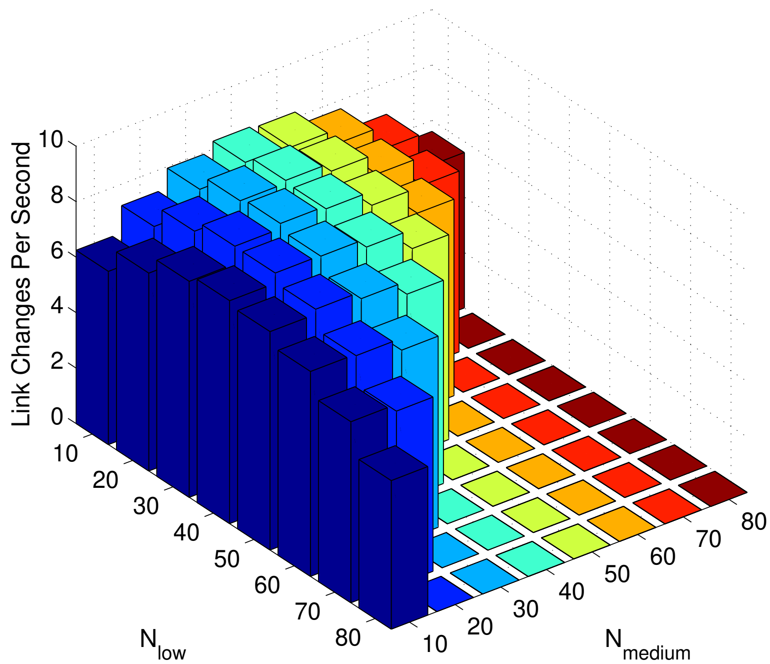

Figure 4.26: Three-Power Model (Scenario 1): Number of Link Changes Per Second (1 m/s)

reinforce once again, that it is important to analyse the routing characteristics of the network using a rich set of metrics, as different metrics expose different aspects of the routing environment that often cannot be predicted based on the values of other metrics.

4.5.5. Effects of Mobility and Speed of Movement

To explore the effects of mobility, I performed the same set of experiments described in Section 4.5 but with a maximum speed of movement of 20 m/s, instead of 1 m/s. In addition, I repeated the experiments for a static network.

High Mobility

The values of most of the metrics for the random-power model at 20 m/s are nearly identical to the values of the metrics in the 1 m/s scenarios because they are influenced by node density rather than mobility, and the random waypoint model maintains a similar density between scenarios with different maximum speeds at a pause time of 0 as discussed in Section 3.4.2 of Chapter 3. The metrics that are influenced by the level of mobility in the network are the link and shortest-path changes metrics for which the shapes of the graphs are the same but the absolute values are higher at 20 m/s due to the higher speed of movement. The number of link changes is about 12.7 times higher, and the number of shortest-path changes is about 11.5 times higher at 20 m/s, which match the differences in the ratios of link and shortest-path change metrics between 1 and 20m/s in the scenarios without power variations (Table 4.1).

The metrics for the two- and three-power models at 20 m/s exhibit the same relative behavior as in the random-power model.

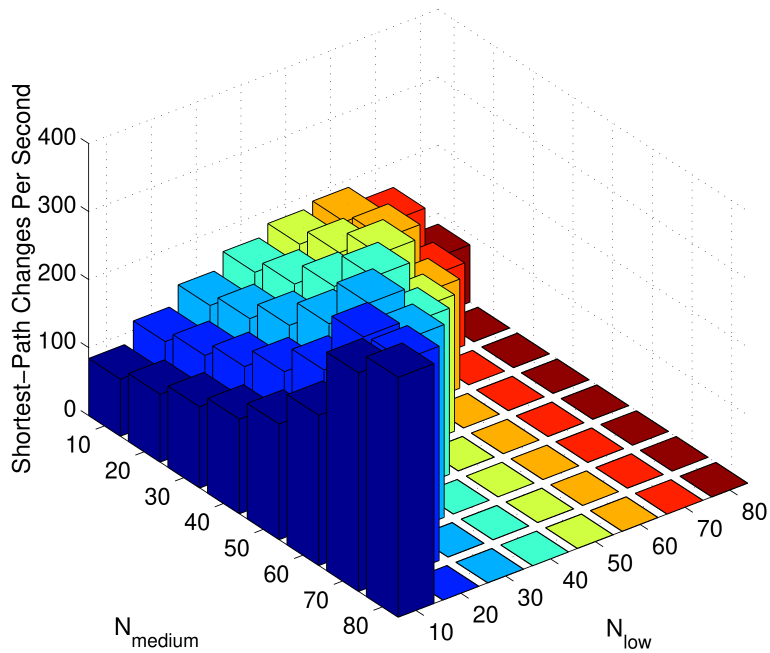

Figure 4.27: Three-Power Model (Scenario 2): Number of Shortest-Path Changes Per Second (1 m/s)

Static Networks

In a static network, the total number of neighbors is 74% of the total number of neighbors in a mobile scenario, and the total number of paths is 95% of that of the mobile scenarios. This lower connectivity is present in scenarios without power variation as well (Table 4.1) and is due to the difference in the distribution of nodes on the site between a static network and a network in which the nodes move according to a random waypoint model (Section 4.5). The difference in the distribution of nodes on the site, and the consequent difference in connectivity between the static and mobile scenarios is the cause of differences in their routing characteristics. The general trend in the values of the metrics in the static network is the same as in the mobile ones, except that the number of neighbors is lower, the number of unreachable nodes is higher, and the path lengths are higher. These differences are generally in the range of 5% to 15%.

4.5.6. Routing Example Analysis

In this section, I briefly revisit the AODV example introduced in Section 4.1, which can now be analyzed using the metrics studied in the preceding sections.

As mentioned in Section 4.1, the number and fraction of in-neighbors decrease monotonically in the parameterizations of the two-power model, moving from left to right in Figure 4.1 as shown in Figures 4.5 and 4.6. These simple metrics, however, do not explain the non-monotonic behavior of the routing protocol, which can now be interpreted with the help of the metrics I developed. In particular, Figure 4.28 shows the reachability metrics for the unidirectional scenarios used in simulating AODV.

AODV generates ROUTE REQUESTs when it encounters a unidirectional or broken link, and when a destination node is unreachable. The peak of the ROUTE REQUEST graph is at R_{low} of 0.3; this is the scenario with the highest number of unreachable and one-way unidirectionally reachable

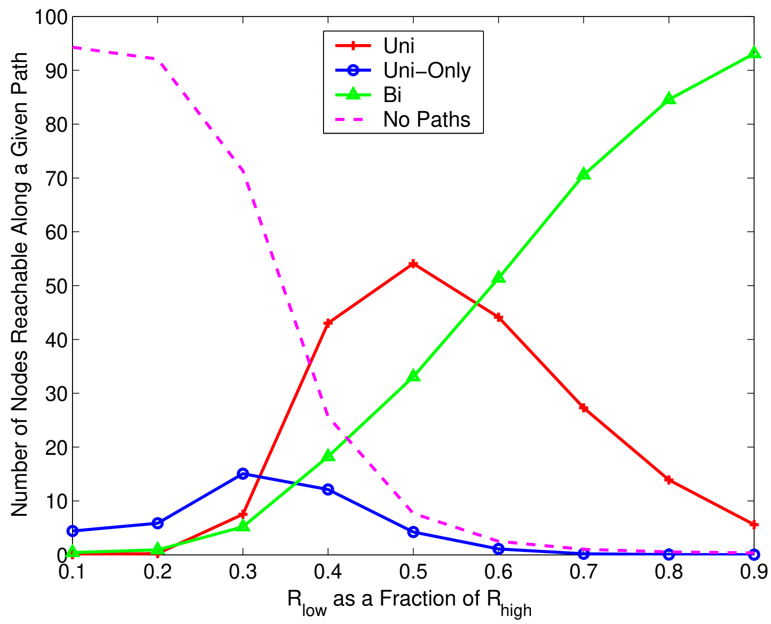

Figure 4.28: Two-Power Model: Node Reachability Metrics for $N_{low} = 90$ (1 m/s)

paths, and these cause the largest number of route discoveries. For values of $R_{low} = 0.1$ and 0.2, connectivity is lower than at 0.3, but it is so low that ROUTE REQUEST packets are not able to propagate to many nodes. At $R_{low} = 0.4$, the number of unreachable nodes is much lower than at 0.3, and even though the number of mutually unidirectional paths is higher (triggering protocol reaction), the number of bidirectional paths is also higher, which is why the ROUTE REQUEST curve begins to decline. This decline turns into a slow increase at $R_{low} = 0.6$, which is the point at which the number of bidirectional paths in the network begins to exceed the number of unidirectional paths. At first glance, it seems that the number of ROUTE REQUESTs should begin to decline as the protocol is more likely to find bidirectional paths. However, as the number of bidirectional paths grows, so does the number of broken links as well as the number of bidirectional to unidirectional link changes (Figure 4.24). As a result, the protocol is forced to perform more local repairs, both in response to broken links and also in response to links changing from bidirectional to unidirectional, which are perceived by the protocol as broken links.

4.6. Summary

In this chapter, I have studied the impact of unidirectional links on the routing characteristics and routing difficulty of multi-hop wireless ad hoc networks. My analysis focused on *mobile* networks composed of heterogeneous devices with different transmission capabilities. To generate such networks, I used the three most commonly used power variation models for simulations of ad hoc networks with unidirectional links: the random-power model, the two-power model, and the three-power model, each parameterized with a wide range of values. I introduced a set of metrics to help in exposing the routing characteristics of the resultant network scenarios, as well as the routing difficulty that each poses to the routing protocol. My analysis shows that it is important to examine the behavior of the network from multiple viewpoints, as the difficulty and effects of a unidirectional scenario on routing protocol performance can be interpreted only with knowledge of the routing characteristics of the network and insight into their interactions. My findings enable protocol designers to better choose a set of network scenarios that truly explore a wide range of a routing protocol's behaviors in the presence of unidirectional links, and to better understand the subtleties of the interplay between routing mechanisms and network states.

Chapter 5

Multicast Routing in Ad Hoc Networks with Unidirectional Links

In this chapter, I study the effect of unidirectional links on multicast protocol performance, and propose extensions for multicast routing in networks with unidirectional links, which I evaluate in the context of ADMR.

5.1. Motivation and Contributions

Most proposed unicast routing protocols and all multicast protocols for ad hoc networks assume that all links in the network are *bidirectional*, such that over any link, each of the two endpoint nodes is able to receive packets sent by the other. However, there are many real-world scenarios in which links in an ad hoc network may work in only one direction. Such *unidirectional* links may occur, for example, due to the use of heterogeneous wireless devices or devices that are configured differently, whereby each device may be transmitting at a different power level. Device characteristics and configuration may vary because of differences in sophistication of the device, or differences in energy availability. Unidirectional links may also occur due to signal interference in the vicinity of a node, caused by transmissions by other nodes, or by other devices, including jammers, which emit signal at a frequency overlapping the one used by the ad hoc network: when there is interference around node **A**, node **A** may be unable to receive packets from node **B** even though **B** may have no trouble receiving packets from **A**.

Routing protocols usually need to use both directions of a link, for example, for per-hop acknowledgments of data packets, neighbor detection, routing table updates, or route discoveries. During a route discovery for a destination, a source node floods a control packet to which the destination responds by sending a reply along the path back towards the source, first traversed by the source's flood.

A protocol that does not even consider the presence of unidirectional links will generally treat the lack of connectivity in one direction of a link as packet loss and perform unnecessary retransmissions, increasing overhead, energy consumption, and network congestion. Some protocols detect the presence of unidirectional links only during route setup [30] and then attempt to avoid them entirely in data packet forwarding. However, if there are no routes between some nodes that need to communicate that consist of only bidirectional links, such protocols cannot establish connectivity between them. In addition, protocols that do not support unidirectional links may be forced to route along longer paths, thus increasing packet delay and the probability of packet losses and broken routes. Such protocols are also forced to concentrate all traffic on routes in the network that are composed of only bidirectional links, which may cause link overload, congestion, and battery exhaustion.

There has been a growing amount of research on routing over unidirectional links in wireless ad hoc networks in the context of *unicast* routing protocols, but no previous studies have been performed on the effect of unidirectional links on *multicast* routing performance, and no specific mechanisms have been proposed to enable multicast routing in the presence of unidirectional links. In addition, with the exception of the work of Kim et al [22], who propose a scheme that requires power control and GPS information, and the partial solution by Pearlman et al [28], all previous proposed protocol extensions for routing over unidirectional links employ periodic link directionality information exchange by all nodes in the network [1, 27, 31–34, 40, 43]. The size of each packet in this exchange varies among the proposed solutions, and each packet is transmitted either as a 1-hop broadcast (i.e., broadcast only to a node's neighbors) or is allowed to travel several hops away in order to update a locality or cluster defined by the protocol. Solutions based on periodic control packet transmissions increase network load and energy consumption, even when no unidirectional links are present in the network. The on-demand approach to routing should avoid all such periodic packets.

In this chapter, I present the first study of the effects of unidirectional links on multicast routing performance. In particular, I focus on on-demand multicast routing through detailed simulations of the Adaptive Demand-Driven Multicast Routing protocol (ADMR) [15], and the On-Demand Multicast Routing Protocol (ODMRP) [38] in networks with unidirectional links. In addition, I propose a set of on-demand mechanisms that enable efficient multicast routing over unidirectional links. I use these mechanisms to extend ADMR, and evaluate the performance of the extended protocol, ADMR-U. This is the first attempt to specifically extend a multicast protocol to route over unidirectional links, and also the first proposal for unidirectional link extensions that are activated only when the protocol encounters a unidirectional link, without also requiring the use of GPS (position) information. ADMR employs a diverse set of on-demand mechanisms and thus makes it possible to test almost all of the newly proposed mechanisms in one protocol.

In networks with unidirectional links, ADMR-U matches or outperforms ADMR in terms of packet delivery ratio by up to 45%, and also lowers ADMR's packet overhead by up to 68%. In addition, the unidirectional extensions do not degrade ADMR's performance in networks with only bidirectional links and do not increase its control packet overhead.

5.2. Related Work

Previous work on routing in ad hoc networks in the presence of unidirectional links has focused on unicast.

Marina and Das [26] present several approaches to dealing with unidirectional links in the context of the Ad Hoc On-Demand Distance Vector protocol (AODV) [30]. Two of the approaches, "blacklisting" and "hello," work by eliminating the unidirectional links from route computation. The third approach, which the authors call *Reverse Path Search*, works by traversing multiple (in some instances all) paths from the destination to the source in order to find at least one bidirectional path. These approaches differ from my work in several ways. First, none of these approaches enables the routing protocol to use unidirectional links. Second, the Reverse Path Search technique is similar to the limited exploration for bidirectional links proposed here, except that the scope of the Reverse Path Search is not limited (except by the number of paths back to the source), it does not establish connectivity if no bidirectional path to the source exists, and it incurs overhead even if there are no unidirectional links on the path to the source.

Pomalaza-Raez [31] presents extensions that enable distance-vector protocols to route over unidirectional links. The idea is to modify the format of the routing tables periodically exchanged by the nodes in the network to include information on inbound links as seen by the node transmitting the table. This approach causes increased control overhead even in the absence of unidirectional links and is inefficient, as shown by Prakash [32], who shows an analysis of the memory and transmission requirements of distance-vector-based protocols that account for unidirectional links; that work shows that such protocols require O(n^2) storage space at each node, and nodes have to exchange O(n^2) information with each other at every iteration (i.e., every periodic interval), where n is the number of nodes in the network. The same work also presents a distance-vector protocol based on destination sequence numbers (introduced by DSDV [29]), which operates over unidirectional links. This protocol utilizes periodic beacons sent by each node in the network. Each of these beacons includes a list of nodes from which the sending nodes overhears beacons. In addition, periodic routing information messages need to be exchanged to distribute reachability information (including information on link directionality) to each destination in the network.

Bao and Garcia-Luna-Aceves [1] propose a link-state protocol able to operate over unidirectional links, called the Unidirectional Link-State Protocol (ULP). ULP is based on the Link Vector Algorithm (LVA) [8]. In LVA, each node maintains its own routing tree, and sends periodic link state updates to its neighbors for links that it uses to reach various destinations, as well as for links it is no longer using. In addition to the link state updates, ULP utilizes periodic neighbor Hello packets sent by each node in the network to gather link directionality information. The collected link information is stored as a directed graph, which ULP uses to compute inclusive cycles for each link, and thus is able to discover paths in both directions between pairs of nodes. This solution incurs overhead even when unidirectional links are not present in the network and requires periodic packet transmissions.

The Dynamic Source Routing protocol (DSR) [18, 19] uses separate paths for bidirectional traffic (the reverse direction can be used for end-to-end acknowledgments) between a source and a destination; the protocol discovers a reverse path without knowing in advance whether unidirectional links are present. Recent versions of the protocol also suggest "blacklisting" as a way to deal with unidirectional links, i.e., detection and elimination of unidirectional links from the route computation, a mechanism similar to that used by AODV [30]. However, in this mechanism, unidirectional links are only detected but cannot be used for routing, which limits the connectivity of the network.

Kim et al [22] propose an end-to-end acknowledgment scheme similar to DSR in which there are two paths, a forward (data) path and a back path from the receiver to the source, for end-to-end acknowledgments. In addition, they propose hop-by-hop acknowledgment techniques in which each node relies on GPS to determine how to modify its transmission power to reach a particular neighbor in order to be able to acknowledge a packet it received (these acknowledgments may be passive or active). This scheme is only applicable to environments in which GPS is available and packet-level power-control is desirable; in addition, it does not attempt to find a potentially cheaper bidirectional path if possible but instead uses the unidirectional links that it finds during its search for the destination.

A more general approach to supporting unidirectional links is presented by Nesargi and Prakash [27]. In their work, unidirectional link support is not part of the routing protocol but is a sublayer below it. Link directionality information is detected through the use of periodic neighbor Hello packets. The sublayer tunnels routing packets and acknowledgment packets to the head of the unidirectional link by encapsulating them in a network-layer header. The sublayer and the routing protocol need to be able to cooperate, as the sublayer occasionally needs routing information from the routing protocol. This scheme works for proactive protocols but would need to be extended to work for on-demand protocols: the proposed mechanisms require periodic control packet exchanges and generate control traffic regardless of whether unidirectional links are present in the network.

Another approach based on the idea of a sublayer is described by Ramasubramanian et al [34]. The protocol is called the Sub Routing Layer (SRL) protocol and relies on periodic reverse path updates transmitted by each node in the network. Each packet contains routing information for nodes that are within NEIGHBORHOOD_SIZE hops away and thus the packets here are smaller than in distance-vector-based unidirectional extensions. Also only some of the messages contain full updates, including information on all reverse routes; others contain only information on routes whose length has changed. This scheme generates additional overhead regardless of whether there are any unidirectional links in the network and is based on periodic control mechanisms.

Unidirectional extensions in the context of a hierarchical routing algorithm based on dominating sets are presented by Wu and Li [43]. Similar to the work of Ramasubramanian et al [34], each node periodically sends beacons with neighbor information, except that each beacon is transmitted several hops away to cover the possible length of the reverse route.

Sinha et al [40] suggest protocol-specific unidirectional extensions for the Zone Routing Protocol (ZRP) [11]. Link directionality information is collected through periodic unit transmissions within each ZRP zone; each unit contains information on the inbound and outbound links of a node, as well as, inbound and outbound trees.

A partial scheme for handling unidirectional links, that does not involve periodic control packet exchange, is described by Pearlman et al [28]. The authors explore the use of multi-hop acknowledgments on the reverse direction of a unidirectional link. These multi-hop acknowledgments are sent via unicast routes if such routes are available at the network layer, or are flooded with a limited TTL, or network-wide. This approach is on-demand in nature, but it is not obvious how much overhead is expended to discover unicast paths in the first case, and the latter case generates significant overhead, as pointed out by the authors.

5.3. Effect of Unidirectional Links on Multicast Routing Performance

In this section, I examine the effects of unidirectional links on the performance of on-demand multicast routing protocols for ad hoc networks. These protocols do not currently include mechanisms to detect or utilize unidirectional links.

I base my study on ADMR and the On-Demand Multicast Routing Protocol (ODMRP) [38], as they represent two different points in the multicast design space. ADMR attempts to operate efficiently by discovering and maintaining multicast state on-demand and by pruning forwarding state that is no longer needed. ODMRP relies on periodic floods to create and refresh multicast state and uses redundant data forwarding of packets to increase the likelihood of successful packet delivery. ADMR is described in detail in Chapter 2 and ODMRP is described in Section 3.3.2 in Chapter 3.

5.3.1. Evaluation Methodology

Simulation Setup

I simulated the performance of ADMR and ODMRP using the simulation setup described in Section 3.4.2 of Chapter 3. I modified the IEEE 802.11 [13] link layer to treat all packet transmissions as broadcasts, since standard 802.11 does not allow unicast communication over unidirectional links (in either direction of the link). The power levels of the nodes in the network are assigned at the beginning of each simulation according to a *two-power model*, in which each node has one of two power levels and is classified as a low- or high-power node, representing a scenario where some nodes are carried by pedestrians while others are carried in vehicles. I use 10 high-power nodes, with a range of $R_{high} = 250$m, and 90 low-power nodes, with a range of R_{low}, which is varied between 25 and 225m at 25m increments. I chose to use these scenarios because they represent a realistic heterogeneous ad hoc network, and because, after studying a variety of parameterizations of several power assignment models (Chapter 4), I determined that they would allow me to test the routing protocols in networks with a wide range of routing characteristics and unidirectional difficulty.

The nodes are distributed uniformly randomly over a 1200m $\times$ 800m area and move according to the random waypoint model. I use two maximum speeds of movement, 1 m/s and 20 m/s, and two pause times, 0s (continuous motion) and 900s (which was also the duration of the simulation runs and thus represents a static network).

I experimented with two multicast scenarios, which I refer to as the *light scenario* and the *heavy scenario*. The light scenario consists of 1 multicast source and 10 receivers, while the heavy scenario includes 3 multicast groups, 3 sources and 20 multicast receivers per group, for a total of 9 sources and 60 receivers. The multicast sources begin sending data and the multicast receivers join a multicast group at uniformly randomly chosen times between 0 and 180 seconds from the beginning of the simulation. Multicast data is sent at a rate of four 64-byte packets per second. This packet rate was chosen to continuously probe the routing ability of the protocols rather than to represent any particular application.

For each parameter combination, I ran 50 randomly generated scenarios, and each point in my graphs is the average of the results of these 50 scenarios.

Unidirectional Scenario Characteristics

Figure 5.1 shows the reachability metrics introduced in Chapter 4, for the scenarios with maximum node movement speed of 20 m/s and 0 s pause time. The reachability results are similar for maximum speed of 1 m/s and pause time of 0s; in static scenarios (900s pause time), the intersection of the bidirectional and unidirectional paths curves is between R_{low} of 150 and 175m, rather than between 125 and 150m (in mobile scenarios). At $R_{low} = 250$m, $R_{low} = R_{high}$ and all links in the the network are bidirectional.

Protocol Performance Metrics

I use two metrics to evaluate protocol performance, which were first defined in Chapter 3: *packet delivery ratio* is the total fraction of packets sent by the multicast application that are received by the multicast receivers, and *normalized packet overhead* is the number of control and data transmissions used by the protocol per successfully delivered data packet (e.g., normalized packet overhead of 5 means that the protocol makes 5 packet transmissions on average for each data packet that is delivered to a multicast receiver).

5.3.2. Simulation Results

When the majority of paths in the network are unidirectional, both ADMR and ODMRP deliver less than 80% of their data. In the light and lower mobility scenarios (Figure 5.2), ADMR performs better than ODMRP since it reacts to failures to establish multicast state by performing route discoveries. Some of these discoveries result in alternate paths that the protocol is able to use to deliver

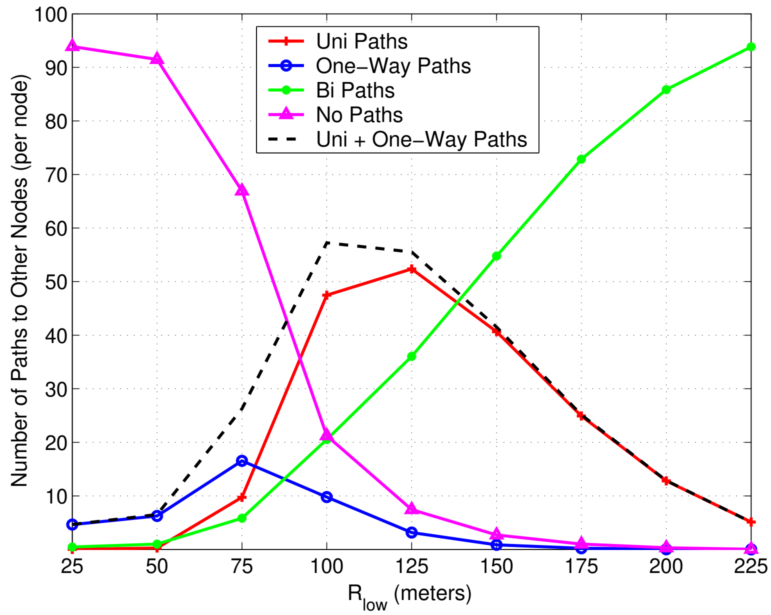

Figure 5.1: Node Reachability (0s pause time, 20 m/s)

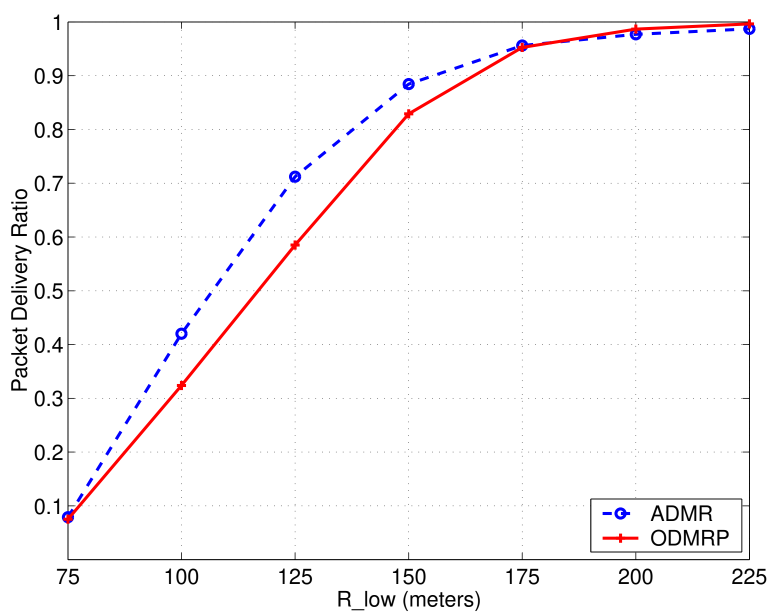

Figure 5.2: Packet Delivery Ratio: Light Scenario (0s pause time, 1 m/s)

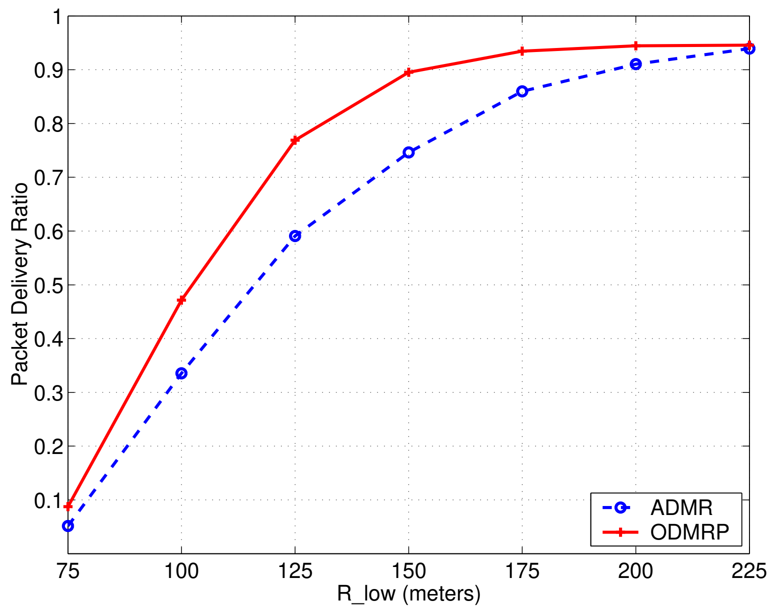

Figure 5.3: Packet Delivery Ratio: Heavy Scenario (0s pause time, 20 m/s)

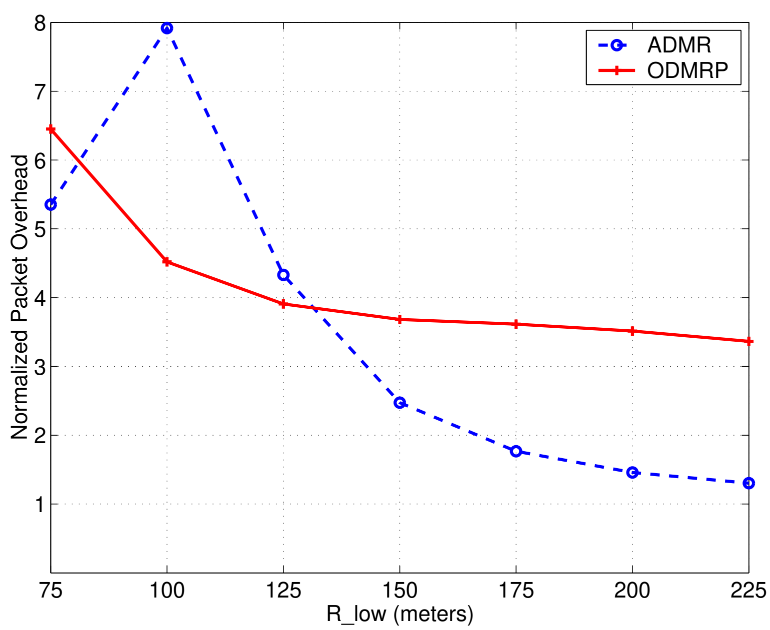

Figure 5.4: Normalized Packet Overhead: Light Scenario (0s pause time, 1 m/s)

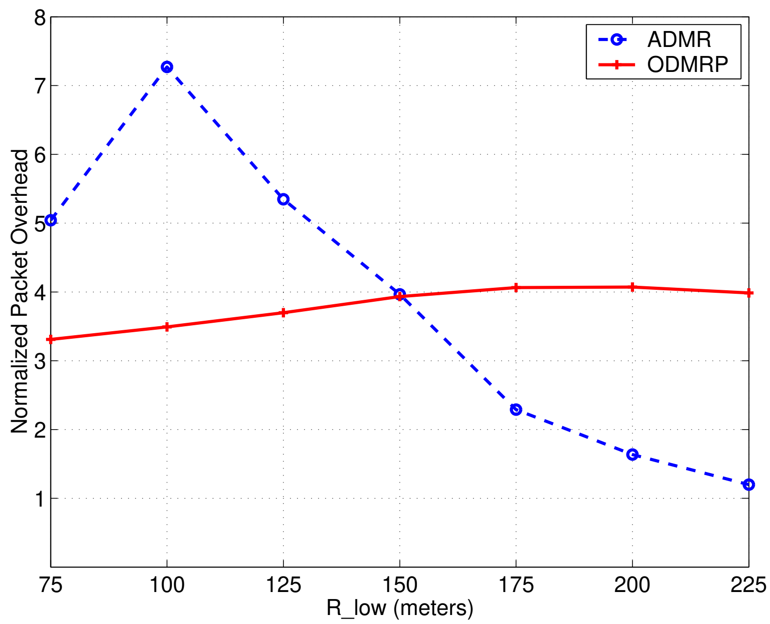

Figure 5.5: Normalized Packet Overhead: Heavy Scenario (0s pause time, 20 m/s)

its data. ODMRP performs better than ADMR in the heavy, higher mobility scenarios due to the redundant forwarding of data packets that it performs, which is higher at higher levels of mobility and for larger groups (Figure 5.3). I do not show the results for R_{low} values of 25 and 50m since network connectivity is very low in these scenarios and virtually no packets are delivered.

In the presence of unidirectional links, both reactive routing mechanisms and redundant data forwarding could be useful in improving protocol performance. However, both need to adapt to current network conditions to avoid generating high overhead. For example, ADMR's high normalized packet overhead and non-monotonic behavior in the presence of unidirectional links is due to the protocol's reactive nature (Figures 5.4 and 5.5). The peak of ADMR's overhead occurs at moderate reachability levels and when the number of one-way and unidirectional paths is highest (dotted curve in Figure 5.1). In this case, ADMR's data floods are able to reach many of the multicast receivers and its MULTICAST SOLICITATION floods are likewise able to reach many of the multicast sources, but the unicast packets sent in response to these floods (UNICAST KEEPALIVES and RECEIVER JOINS) are unable to set up multicast state as they attempt to traverse unidirectional links along the reverse path to the originator of each flood. Because ADMR attempts to establish multicast state among all sources and receivers that are active in the network but are currently not part of the multicast session, a data flood elicits a MULTICAST SOLICITATION flood and vice versa until multicast state is set up along paths between them.

ODMRP's normalized overhead is especially high at higher levels of mobility and for larger multicast groups (due to its highly redundant forwarding), and while its packet delivery ratio is high as a result, it can be an unnecessary and serious drain on network resources.

5.4. Mechanisms for Multicast Routing over Unidirectional Links

In this section, I present unidirectional mechanisms that enable multicast protocols to route over unidirectional links. On-demand multicast routing protocols in general contain one or more of the following mechanisms that must be extended to work in the presence of unidirectional links:

- *Control Packet Unicast*: Multicast state discovery and repair in on-demand multicast routing protocols frequently involve the use of unicast control packets that need to traverse the reverse of paths discovered by a flooded state discovery packet. After one or more attempts to forward such a packet along a link fails, the link is considered broken, even though it may simply be unidirectional. As a result, the routing protocol may attempt to discover alternate routes, thus generating more overhead; or, in some protocols, multicast forwarding state will not be set up until the next scheduled route discovery, thus disrupting the flow of multicast data towards the multicast receivers.

- *Per-Hop Pruning*: Each multicast forwarding node may require acknowledgments for multicast data packets that it forwards, in order to make decisions about pruning itself from the multicast forwarding mesh (or tree), e.g., lack of acknowledgments may indicate that the forwarding state is no longer needed and can be pruned. Usually in this mechanism, a single (passive or explicit) acknowledgment from any node, received over some period of time is sufficient to postpone pruning. When all nodes in the multicast forwarding mesh that receive packets from a forwarding node **A** do so over a unidirectional link from **A**, node **A** will not receive any of their acknowledgments and will prune its forwarding state, disconnecting the mesh. When a node overhears a packet it forwarded being forwarded by another node, it considers it a passive acknowledgment that its next hop node has received the packet. To enable this mechanisms, nodes include in each packet's header, the address of the node from which they received the packet, or their own address if they originated the packet.

- *Per-Hop Disconnection Detection*: Some protocols require acknowledgments of data packets in order to detect broken links. For example, a forwarding node **A** expects an acknowledgment from each multicast node that received a packet forwarded by **A**, and when such an acknowledgment is not received, the node initiates multicast repair to reconnect the multicast forwarding mesh (or tree). A unidirectional link between **A** and a downstream node **B** would prevent **A** from receiving **B**'s acknowledgments and would cause **A** to perform multicast repair, and in some cases would disrupt the flow of data towards the multicast receivers because the forwarding state may be marked as invalid upon detection of the disconnection.

In the rest of this section, I describe how to extend the above mechanisms to enable efficient multicast routing over unidirectional links. While the proposed unidirectional mechanisms below are particularly appropriate for on-demand multicast protocols, some can also be used to extend proactive multicast protocols, and also unicast protocols.

In a network with only bidirectional links, the proposed mechanisms below incur only negligible byte overhead over the original routing protocol mechanisms, since several additional fields are added to some of the protocol's control packets. *No additional control packets are sent unless the protocol encounters a unidirectional link.*

5.4.1. Link Directionality Detection

In prior work, link directionality detection has typically employed periodic 1- or 2-hop beacons broadcast by each node in the network. Such techniques may not seriously affect the performance of protocols that already use beaconing for other purposes, as the two types of beacons can often be combined. However, on-demand routing protocols usually do not employ such periodic beaconing and so adding it to the protocol would incur a significant amount of extra overhead. Instead of collecting link directionality information proactively, my approach relies on passive monitoring of received packets at a node; no control packets are sent specifically to discover the directionality of a link.

I call the data structure in which each node records information about the directionality of adjacent links, a *Neighbor Table*. Node **A** records in its Neighbor Table the existence of a link *from* node **B** when it receives a packet from **B**. A node **A** can determine that it is able to deliver packets to **B**, if it overhears a transmission of a packet by **B** which **B** received from **A** (passive acknowledgment). Alternatively, node **A** determines that it cannot reach node **B** when it cannot deliver a packet *to* **B** as indicated by a lack of acknowledgment (passive or explicit) from **B**. Node **A** considers the link it shares with node **B** unidirectional if it is able to receive packets from **B** but is unable to deliver packets to **B**. Neighbor Table entries are expired after some period of time after last updated to ensure freshness of the link directionality information.

The neighbor information collected by each node can be used by the routing protocol to avoid choosing unidirectional links for routing, to find alternate bidirectional links if possible, or to employ routing mechanisms for traversing unidirectional links. Neighbor link directionality information is refreshed during request and reply cycle network-wide or localized floods used by the routing protocol. These floods immediately precede multicast state setup in on-demand routing protocols, which is exactly when link directionality information is needed.

5.4.2. Localized Bidirectional Path Search

Localized Bidirectional Path Search is an extension to enable Control Packet Unicast over unidirectional links. Its goal is to forward unicast control packets over alternate bidirectional links whenever they encounter a unidirectional link.

Localized Bidirectional Path Search requires that the originator of the unicast control packet include in it, the unique sequence number included in the flooded packet that triggered it, as well as the maximum allowable path length to its destination. The sequence number is used for duplicate detection, while the maximum path length is used to limit the amount of time and effort the protocol would spend looking for a bidirectional path.

A node **A** assumes that it has encountered a unidirectional link when it does not receive a (passive or explicit) acknowledgment after a limited number of retransmissions of a unicast control packet to a neighbor node **B** (Figure 5.6). In this case, node **A** will broadcast a NEIGHBOR QUERY packet to check if a neighboring node has a route to the control packet's destination, e.g., node **S** (Figure 5.7). The unique sequence number and maximum path length from the unicast control packet are copied into the NEIGHBOR QUERY packet; this packet also includes a list of neighbors allowed to process it. The list of neighbors contains the addresses of nodes with which **A** has bidirectional links according to its Neighbor Table, and is used to ensure that only nodes that share bidirectional links with **A** respond to the query (otherwise **A** will not receive their responses), and also to limit the number of neighbors that would respond to the NEIGHBOR QUERY. Only a neighbor listed in the packet can respond to it, and only if 1) the neighbor has not previously forwarded the unicast control packet for which local exploration is being performed (the unique sequence number allows for detection of duplicates) and 2) the neighbor has a path to **S** that is shorter than the maximum allowable path length specified in the NEIGHBOR QUERY. Nodes that satisfy the above conditions respond with a NEIGHBOR REPLY packet (Figure 5.8). Transmission of this packet is scheduled after a random delay proportional to the length of the current node's path to **S**. This delay is intended to allow **A** to receive the response with the shortest path to **S** first and also to reduce the probability of collisions due to multiple nodes sending NEIGHBOR REPLY packets simultaneously.

When node **A** receives a NEIGHBOR REPLY from node **C**, it forwards the unicast control packet to **C** which is now responsible for forwarding it onto **S**, using the same steps as **A** (Figure 5.9). On the other hand, if **A** does not receive a response to its NEIGHBOR QUERY within some period of time, it can process the unicast control packet as described in Section 5.4.3.

Unlike previously proposed approaches, when using Localized Bidirectional Path Search, the protocol explores only a small area around the encountered unidirectional link rather than exploring multiple paths simultaneously and potentially forwarding the packet along all possible paths to its destination. In addition, the Localized Bidirectional Path Search will only choose a bidirectional path over a unidirectional path if the bidirectional path is sufficiently shorter than the unidirectional path. The mechanism can be configured based on the relative cost of routing along a unidirectional or a bidirectional link in the context of a given routing protocol.

5.4.3. Limited Multi-Hop Flooding

Limited Multi-Hop Flooding can be used for Control Packet Unicast, e.g., when the Localized Bidirectional Path Search fails (Section 5.4.2), or for unicast control packets that precede state setup packets for which a node does not receive a passive acknowledgment after some number of retransmissions. Such packets traverse the reverse path of the one that will be subsequently traversed by a state setup packet. The Localized Bidirectional Path Search is not needed when forwarding such packets since they are not setting up forwarding state but only need to be delivered to the destination.

To deliver a control packet across a unidirectional link as may be required for control packet unicast, a node **A** that cannot reach **B** directly and does not have a multi-hop route to **B**, can send the packet as a limited multi-hop flood. This mechanism has been used in various forms in the context of unicast routing protocols. I propose to modify it as follows: first node **A** sends the packet as a 2-hop flood and if no passive or explicit acknowledgment is received from **B**, **A** sends the packet as a 4-hop flood. If this flood also fails, rather than sending more floods, it would be more efficient for a new potentially shorter path between the source and destination of the unicast control packet to be explored. Such a path would be discovered by end-to-end mechanisms within the protocol which would detect the failure of the packet to reach its destination. In addition, the link between **A** and **B** may no longer exist as **B** may have moved away. As a result, repeated and more aggressive flooding attempts to reach **B** will generate unnecessary overhead and will result in the packet traversing a path between nodes that are no longer adjacent in either direction of the link.

The reception of a multi-hop flooded packet at node **B** when **B** is not the final destination of the packet can be confirmed through passive acknowledgment mechanisms when it gets forwarded by **B** onto its next hop towards its destination. When **B** is the packet's final destination, it can explicitly confirm the reception of the packet by reforwarding it, or better yet, other packets that are already part of the protocol's operation can confirm the reception of the control packet. For example, once multicast state along a path is set up, node **B** will start forwarding data packets onto **A** and node **A** can consider the first such data packet as a confirmation that the unicast state setup packet has reached its destination.

In addition, Limited Multi-Hop Flooding can serve to disseminate information on link directionality in areas of the network where the protocol encounters unidirectional links. To do that a special header can be attached to the flooded packet and be updated by each node, which contains a list of neighbors from which the transmitting node has recently received packets. All of a nodes' neighbors that are on this list and receive the flooded packet directly from it will be able to conclude that they share a bidirectional link with that node. Subsequent state setup packets will be able to use this information when performing Localized Bidirectional Path Search for example.

5.4.4. Selective Source-Routed Multicast Acknowledgments

When the links between a node **A** and all its downstream neighbors in the multicast mesh (or tree) are unidirectional, node **A** will not be able to receive acknowledgments from these neighbors and will prune itself from the multicast mesh (Per-Hop Pruning). In order for **A** to continue forwarding packets, a downstream node would have to start sending acknowledgments to **A** over a multi-hop path. I call the acknowledgment mechanism introduced here, *Selective Source-Routed Multicast Acknowledgments*.

I add an additional flag to the multicast header of each data packet, called the *No Ack* flag. This flag is set by each node **A** forwarding a multicast packet, when **A** has not received any acknowledgments (passive or explicit) for some number of consecutive packets that it forwarded. A set flag indicates to the nodes that receive their data through **A** that **A** will soon prune itself from the forwarding mesh unless it receives an acknowledgment for the data that it forwards. This mechanism allows nodes to discover that a link to their parent node in the mesh is unidirectional both when the mesh is being set up and also in cases when a link that was previously bidirectional has become unidirectional (e.g., due to nodes with different ranges moving farther apart). In contrast to previously proposed mechanisms that require periodic 1-hop beacons to be sent by each node in the network, the mechanism proposed here does not send any control packets to detect that a link is or has become unidirectional.

A node **B** which receives some number of consecutive multicast packets with a set *No Ack* flag from node **A**, and which does not have acknowledgment state for **S** and **G**, resets its counter of received packets with a set *No Ack* flag and floods an ACKNOWLEDGMENT PATH DISCOVERY packet with a small hop limit (e.g., 2) and destination **A** (Figure 5.10). This packet includes the sequence number from the last data packet for **S** and **G** that **B** received from **A**. This sequence number is used to consolidate floods sent by multiple child/descendent nodes for the same parent node **A** by treating them as duplicates of each other; only one such flood needs to reach **A**.

Each node that forwards an ACKNOWLEDGMENT PATH DISCOVERY packet adds its address in a field in the packet's header. When node **A** receives this packet, it records the route collected in its header along with the address of the source of the packet, node **B** in this case, provided that the sequence number in the packet header is more recent than the one **A** has already recorded when it received a previous ACKNOWLEDGMENT PATH DISCOVERY, or is the same as the recorded one but the received packet contains a shorter route.

Node **A** attaches the shortest most recent route received in an ACKNOWLEDGMENT PATH DISCOVERY to the next few data packets (e.g., 2 or 3) that it forwards for **S** in order to "announce" to its descendents in the multicast mesh which one of them it has chosen to send it active acknowledgments. This arbitration improves efficiency by ensuring that only one node at a time will be sending active acknowledgments to a given upstream node and by choosing the one with the shortest acknowledgement path.

When node **B** receives a multicast packet with an attached acknowledgment route in which it is listed as a first hop, node **B** records this route and uses it to source route an ACTIVE ACKNOWLEDGMENT to **A** every time it receives a multicast packet from **A** with a set *No Ack* flag (Figure 5.11). Since this flag is only set every few packets, ACTIVE ACKNOWLEDGMENTs are sent once in a while rather than in response to each data packet. In addition, if a link between the parent node and one of its children becomes bidirectional, passive or 1-hop explicit acknowledgments will

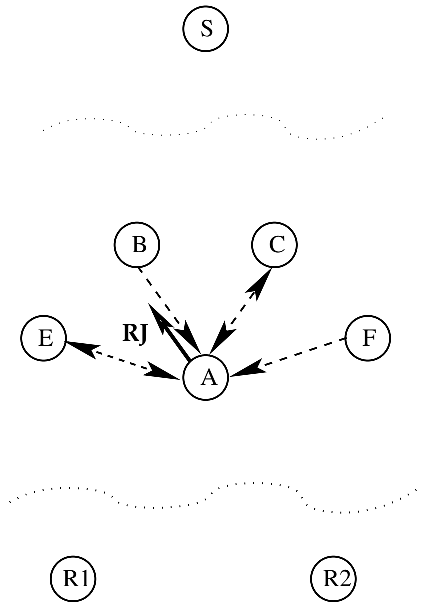

Figure 5.6: Node **A** attempts to send an ADMR RECEIVER JOIN to node **B**. The packet transmission is indicated by a solid arrow; the dashed arrows indicate link directionality.

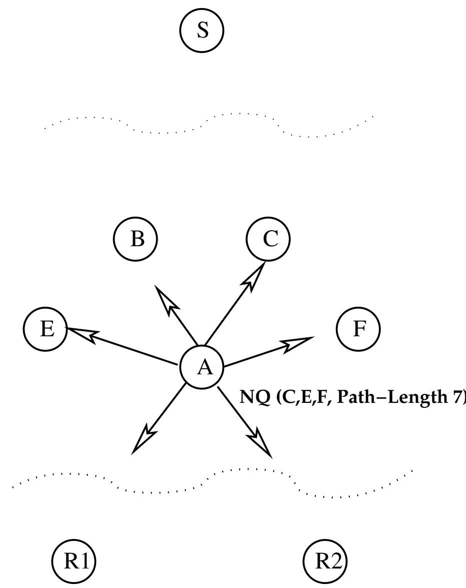

Figure 5.7: Node **A** broadcasts a NEIGHBOR QUERY to be processed by **C**, **E** and **F**; the maximum allowable path length to the source is 7 hops.

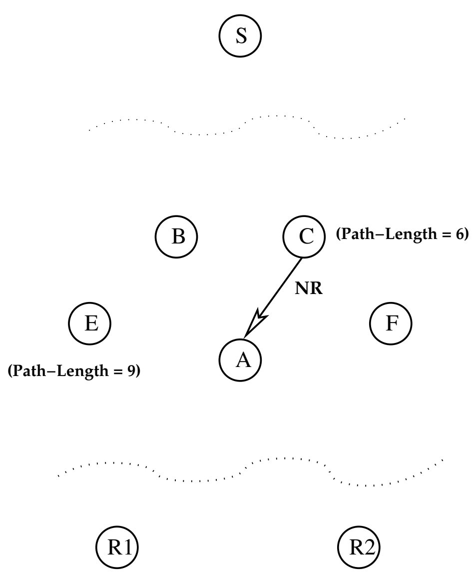

Figure 5.8: Node **C** sends a NEIGHBOR REPLY to node **A**.

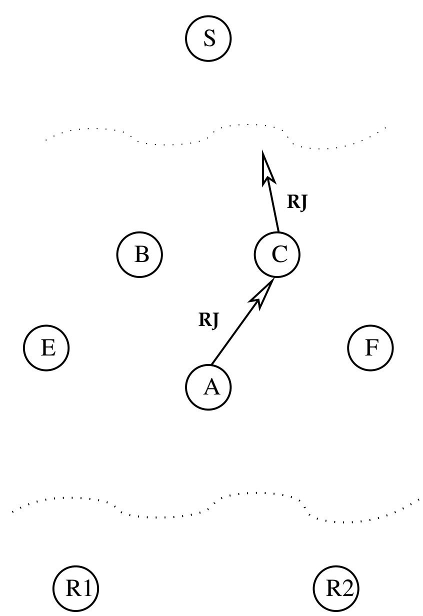

Figure 5.9: Node **A** forwards the RECEIVER JOIN towards the source over the newly discovered bidirectional link through node **C**.

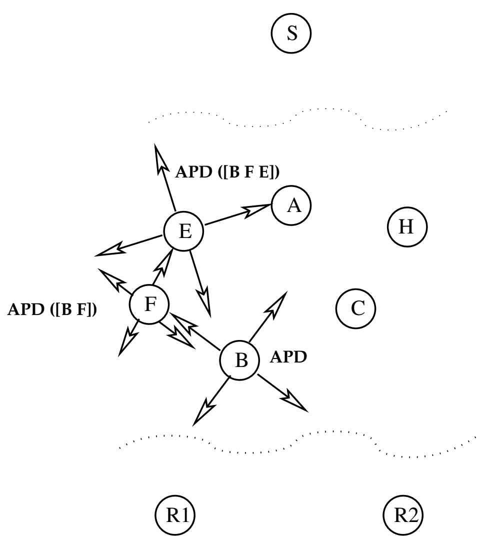

Figure 5.10: ACKNOWLEDGMENT PATH DISCOVERY flood towards node **A**.

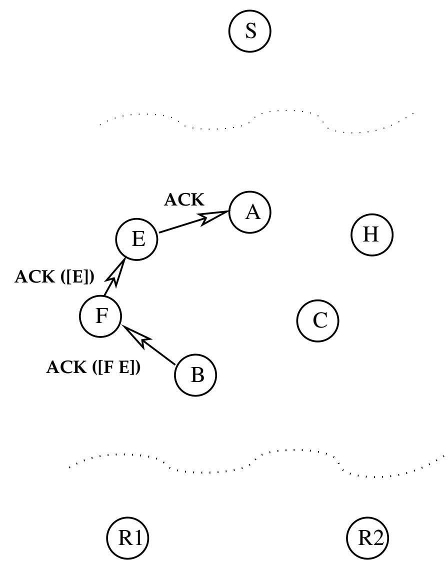

Figure 5.11: Source routing of ACTIVE ACKNOWLEDGMENT to node **A**.

begin to arrive at the parent, the *No Ack* flag in data packets will no longer be set, and the active acknowledgments will stop.

When a node **C** who has a recorded source route for active acknowledgments, receives a multicast packet with an attached acknowledgment route in which it is not listed as a first hop, node **C** deletes its acknowledgment state as another node has been chosen to send active acknowledgments. When an active acknowledgment route breaks, the parent node will stop receiving acknowledgments, and will set its *No Ack* flag, which will eventually trigger the children to discover a new acknowledgment path.

Using source routing for active acknowledgments removes the need for per-hop maintenance of the acknowledgment route which can be expensive since some links along it may also be unidirectional.

Selective Source-Routed Multicast Acknowledgments can also be used to extend Per-Hop Disconnection Detection, which requires explicit acknowledgments by all nodes that receive a data packet. In this case, aggregation of ACKNOWLEDGMENT PATH DISCOVERY packets should be disabled, and the *No Ack* flag can be associated with the address of the specific node that needs to send an acknowledgment.

5.4.5. Mesh/Tree Reconfiguration

Mesh (or tree) Reconfiguration is an optimization that applies to multicast protocols that utilize Per-Hop Pruning. It can be used by protocols that use both multicast trees and multicast meshes though it is more effective in the presence of redundant forwarding state which is typical of multicast meshes. The idea is to explore forwarding state redundancy in order to reconfigure the forwarding mesh automatically so as to decrease reliance on unidirectional links for data forwarding within the mesh. The proposed mechanisms detect changes in link directionality of links in the multicast mesh, and “encourage” forwarding nodes to choose parent nodes who do not require active acknowledgments and if possible prune parents that do. This optimization does not add *any* overhead to the protocol.

A node **A** in the multicast mesh for a group **G** and source **S** may receive multicast packets from several upstream nodes in the mesh, say **B**, **C** and **D**. Such a node typically records the address of the node from which it last received a non-duplicate multicast packet as its parent in the mesh, and sends a (passive or active) acknowledgment to that parent, say node **B**. However, even though **B** forwards the multicast packets on2to **A** first, node **B** may have a unidirectional link to **A** and thus require ACTIVE ACKNOWLEDGMENTs in order to continue forwarding packets (Section 5.5.4). On the other hand, node **C** may have a bidirectional link to **A** and would not require active acknowledgments. Another node **D** may not have a bidirectional link to **A** but may already be receiving passive acknowledgments from another of its descendent nodes.

Under the Mesh Reconfiguration mechanism, node **A** chooses the most desirable parent (e.g., the one which requires the least overhead to keep active), say node **C**, and sends passive acknowledgments to that parent rather than the node from which it received the data packet first. This causes the undesirable parents to eventually expire. Of course, if node **A** stops receiving packets from the most desirable parent, it switches to the next most desirable parent, etc. Parent desirability depends on the number of descendant (child) nodes of the parent node and on whether the parent node has at least one bidirectional child from which it can receive passive acknowledgments. The number of children is important since at most one ACTIVE ACKNOWLEDGMENT is sent per parent node, and it would be more efficient to concentrate children around a smaller number of parents. It is more likely that a child would not need to start sending active acknowledgments to prevent such a parent from expiring its multicast state, even if some of its child nodes leave the mesh.

Forwarding nodes can be classified into four levels in descending order of desirability: Level 1 are parent nodes who have more than 1 child node and are receiving passive acknowledgments. Level 2 are parent nodes that have 1 child and are receiving passive acknowledgments. Level 3 are parent nodes that have more than 1 child and are not receiving passive acknowledgments. Level 4 are parent nodes that have 1 child and are not receiving passive acknowledgments. Nodes know if a parent receives passive acknowledgments and whether the parent has one or more children based on 3 bits in the header of each multicast packet. These bits are updated by the parent node upon forwarding each packet based on collected information on the number of recently received ACTIVE ACKNOWLEDGMENTs and the number of recently overheard passive acknowledgments.

5.5. ADMR-U

In this section, I describe how I extend each of ADMR's mechanisms to enable it to operate in the presence of unidirectional links. The extended protocol, ADMR-U, uses unidirectional extensions based on the mechanisms introduced in Section 5.4. Because these extensions are only activated when ADMR-U encounters a unidirectional link: *ADMR-U does not incur any additional control packet overhead or performance penalties compared to ADMR when no unidirectional links are present in the network.*

Each node in ADMR-U maintains a Neighbor Table and updates this table with link directionality information as described in Section 5.4.1.

5.5.1. New Multicast Source

In this section, I describe the operation of ADMR when a multicast source first starts sending data.

Operation Over Bidirectional Links. When an application on node **S** originates a multicast packet for some group **G** for which **S** is not currently a multicast source, the routing layer attaches an ADMR header to the packet and sends the packet as a network-wide flood.

Multicast receivers for group **G** and source **S** reply to the data flood by sending a RECEIVER JOIN packet back towards the source along the shortest-delay path followed by the flooded data packet. Each node forwarding the RECEIVER JOIN sets up forwarding state for the source and group listed in the packet, and stores its distance from the source (recorded in the hopcount field of the RECEIVER JOIN).

Unidirectional Extensions. In order to enable the RECEIVER JOIN packet to set up forwarding state along a path to the source in the presence of unidirectional links, I have extended the protocol with Localized Bidirectional Path Search and Limited Multi-Hop Flooding, introduced in Sections 5.4.2 and 5.4.3.

When a node using ADMR-U encounters a unidirectional link, it performs Localized Bidirectional Path Search around the nodes forwarding the RECEIVER JOIN to attempt to find bidirectional links on alternate (potentially bidirectional) paths towards the source. The last node that forwards the RECEIVER JOIN packet considers the first received data packet from the source to be an acknowledgment that the JOIN has reached the source, so no explicit acknowledgment is necessary.

In addition, in order to increase the probability that the RECEIVER JOIN packet will not encounter unidirectional links, a node which receives the source's data flood from a node with which it has a unidirectional link (according to its Neighbor Table), delays forwarding the packet for a short amount of time, e.g., several milliseconds. This mechanism is designed to allow for copies of the flooded packet to traverse bidirectional links before traversing unidirectional links, which in turn would cause the RECEIVER JOIN to be sent along paths with less (or no) unidirectional links.

5.5.2. Receiver Application Join

In this section, I describe the mechanisms that ADMR employs to enable a multicast receiver to join a multicast group when an application on the receiver issues a multicast join request.

Operation Over Bidirectional Links. Each node who is a forwarder or receiver for **G** and **S** and receives a source-specific MULTICAST SOLICITATION, instead of rebroadcasting it, it unicasts it along the reverse links in the multicast mesh towards **S**. Each node in the mesh remembers the node from which it received its last data packet as the next hop node along a reverse path back to the source.

When a source for group **G** receives a MULTICAST SOLICITATION, it unicasts a UNICAST KEEPALIVE packet back towards the receiver. Upon receiving the UNICAST KEEPALIVE, the multicast receiver sends a RECEIVER JOIN towards the source, as described in Section 5.5.1.

Unidirectional Extensions. To enable UNICAST KEEPALIVE packets to be forwarded to the receiver over unidirectional links, I use Limited Multi-Hop Flooding (Section 5.4.3). I do not use Localized Bidirectional Path Search in order to speed up the multicast state setup, and also to avoid duplication of effort; the RECEIVER JOIN will be performing Localized Bidirectional Path Search (Section 5.4.2). The RECEIVER JOIN serves as an acknowledgment that the UNICAST KEEPALIVE was successfully delivered to its destination, so no explicit acknowledgment needs to be sent by the destination.

The mechanism for forwarding source-specific MULTICAST SOLICITATION packets towards the source by nodes on the mesh for the source and group listed in the packet uses Limited Multi-Hop Flooding except that if a node forwarding the MULTICAST SOLICITATION packet upstream towards the source has a multi-hop path to its parent in the mesh (used for acknowledgment purposes, Section 5.5.4), the MULTICAST SOLICITATION can be unicast towards the upstream node along this multi-hop path.

5.5.3. Broken Link Detection and Repair

In this section I describe the mechanisms employed by ADMR to detect and repair broken links.

Operation over Bidirectional Links. When a node **A** detects a broken link, as a result of missing data packets or keep-alives, it initiates local repair but only after deferring for a period proportional to its hop count from the multicast source (copied from the hopcount field of the last forwarded data or keep-alive packet) to ensure that the broken link is adjacent to it rather than further upstream. Node **A** begins the local repair by multicasting a RECONNECT NOTIFICATION packet down the multicast mesh to notify the forwarding nodes below it that it is performing local repair and they should cancel their local repair timers. When a receiver node receives the REPAIR NOTIFICATION packet, it postpones its repair timer instead of canceling it. If this timer expires before the receiver has started receiving data again, the local repair must have failed and the receiver performs global repair by (re)joining the group as described in Section 5.5.2.

In networks with bidirectional links, the REPAIR NOTIFICATION packet is also overheard by the parent node of **A**, say node **B**, if the parent is within range of **A**. Upon overhearing the packet, **B** will send a 1-hop REPAIR NOTIFICATION to **A** to notify it that the link between them is actually not broken. This mechanism increases the likelihood that only nodes adjacent to a broken link will perform a local repair and thus eliminate futile local repair efforts.

After node **A** has sent a REPAIR NOTIFICATION and has not received a REPAIR NOTIFICATION from its parent node, it sends a 3-hop flooded RECONNECT packet which includes the hop count of the last received data packet from the source. When this packet reaches a forwarding mesh node with a smaller hop count to the source than the one recorded in the RECONNECT, this node unicasts the packet to its parent in the multicast mesh, which unicasts it to its parent, etc., until the packet reaches the source. The source then unicasts a RECONNECT REPLY packet back towards the receiver, which sets up forwarding state for **S** and **G** along the path to the originator of the local repair.

Unidirectional Extensions. The REPAIR NOTIFICATION packet sent when a broken link is detected, will not reach the transmitting node's parent when the link between the parent and the child is unidirectional. This may lead to a node performing a repair when the link to its parent is not actually broken but can be used to deliver data. Since repairs are scheduled at a time proportional to the distance of the node from the source, it should be rare that a child node would initiate repair before its parent and since even if it did, the protocol would still operate correctly, I decided not to add any new mechanisms here.

The mechanism for forwarding the RECONNECT packet as a unicast towards the multicast source by mesh nodes is the same as the one used to forward source-specific MULTICAST SOLICITATIONS (Section 5.5.2).

The RECONNECT REPLY packet is forwarded using Limited Multi-Hop Flooding. In addition, once it is received by the node across the unidirectional link, node **B**, **B** needs to send a RECEIVER

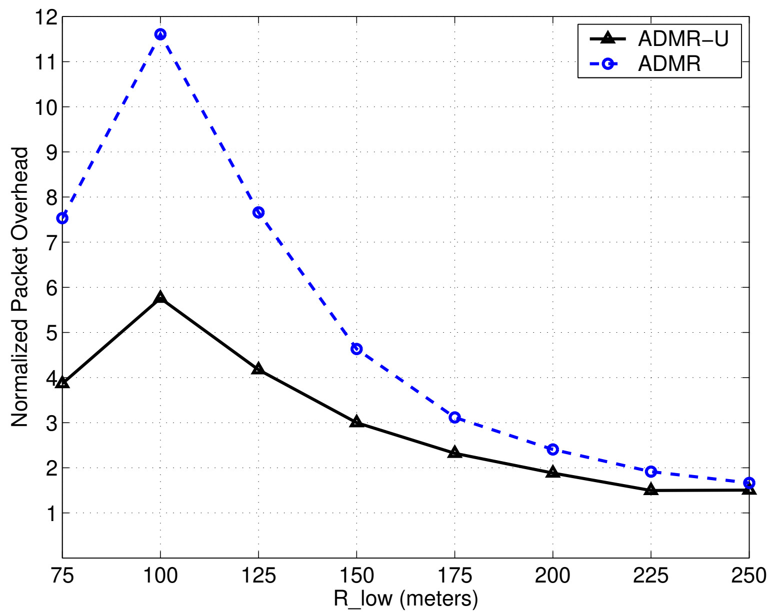

Figure 5.12: Normalized Packet Overhead: Light Scenario (0s pause time, 20 m/s)

JOIN packet towards the originator of the RECONNECT REPLY flood, node **C**, in order to attempt to discover a bidirectional path to it.

5.5.4. Pruning of Multicast State

In this section, I describe the mechanisms used by ADMR to automatically expire unneeded forwarding state.

Operation Over Bidirectional Links. Each node which has forwarding state for a group **G** and source **S** automatically expires its forwarding state when it determines that it is no longer needed for multicast forwarding, because for example, a downstream receiver has left the multicast group, or has crashed, or because, as a result of a disconnection and an ensuing repair, some forwarding state may no longer be needed.

The decision to expire forwarding state at a node is based on whether the multicast packets (data or keep-alives) that this node transmits are subsequently transmitted by other nodes. For each multicast packet it transmits, a node **A** expects to overhear at least one subsequent transmission.

Multicast receivers that are not also forwarders for the multicast group, send explicit acknowledgments (possibly the last received data packet) to the node from which they received their last data packet, unless they overhear another (passive or explicit) acknowledgment directed at the same node.

Unidirectional Extensions. In order to enable the mesh pruning mechanism to operate in the presence of unidirectional links, I extend ADMR with the Mesh Reconfiguration mechanism (Section 5.4.5) and the Selective Source-Routed Multicast Acknowledgments (Section 5.4.4), which are used for both data packets and keep-alives.

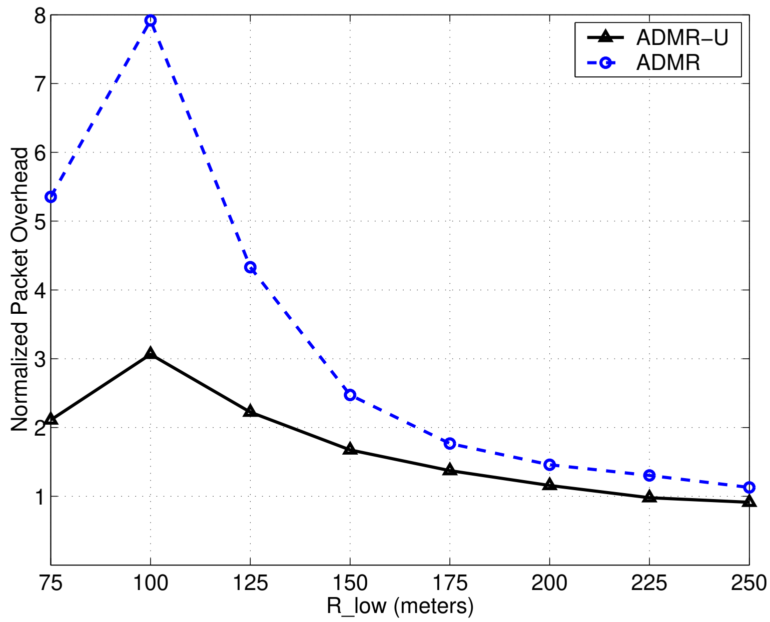

Figure 5.13: Normalized Packet Overhead: Light Scenario (0s pause time, 1 m/s)

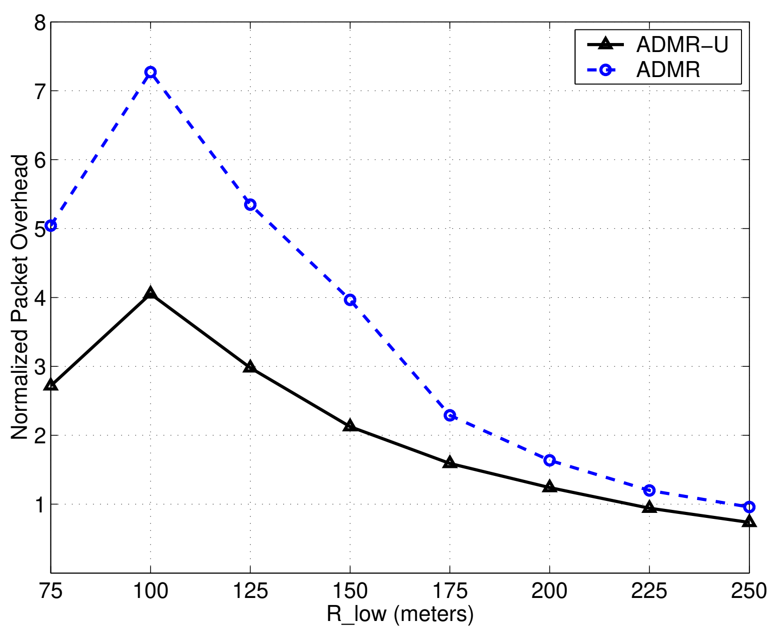

Figure 5.14: Normalized Packet Overhead: Heavy Scenario (0s pause time, 20 m/s)

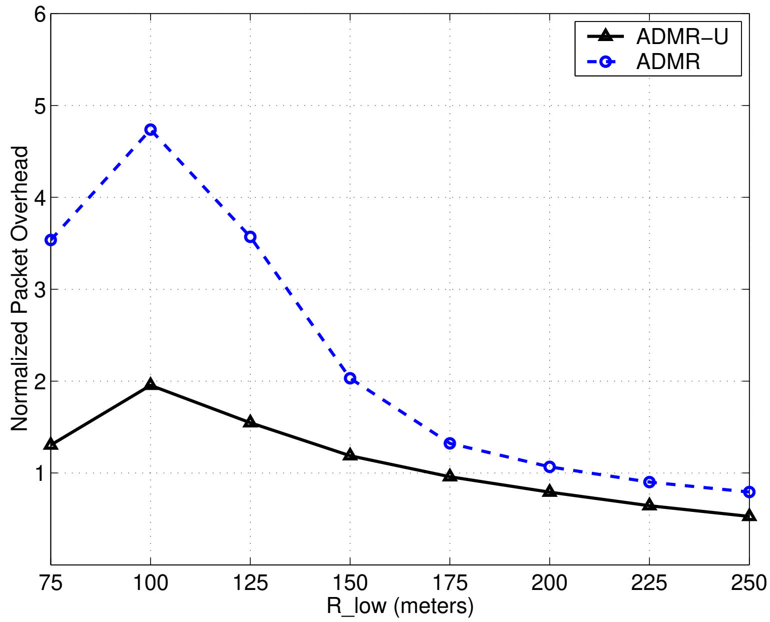

Figure 5.15: Normalized Packet Overhead: Heavy Scenario (0s pause time, 1 m/s)

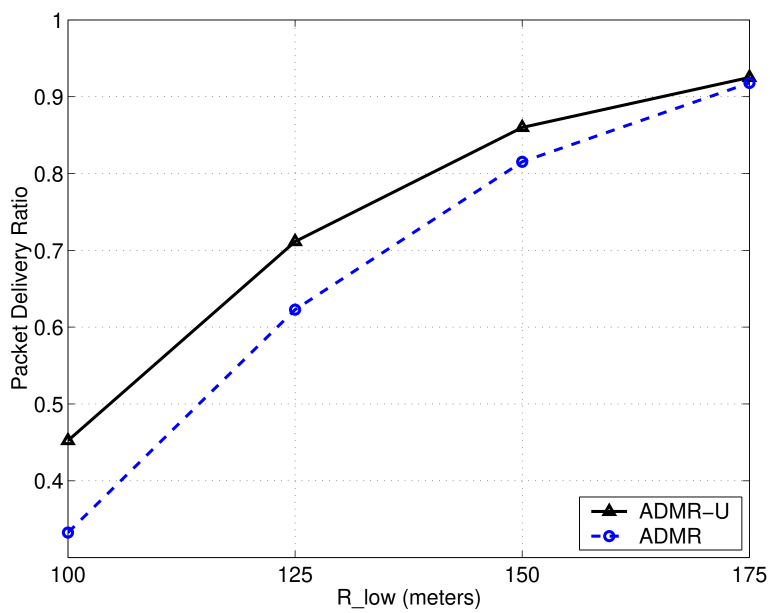

Figure 5.16: Packet Delivery Ratio: Light Scenario (0s pause time, 20 m/s)

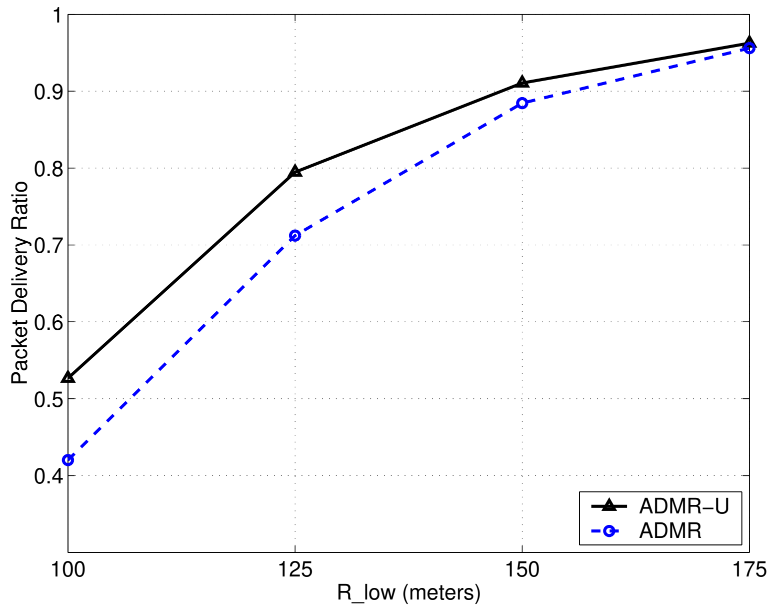

Figure 5.17: Packet Delivery Ratio: Light Scenario (0s pause time, 1 m/s)

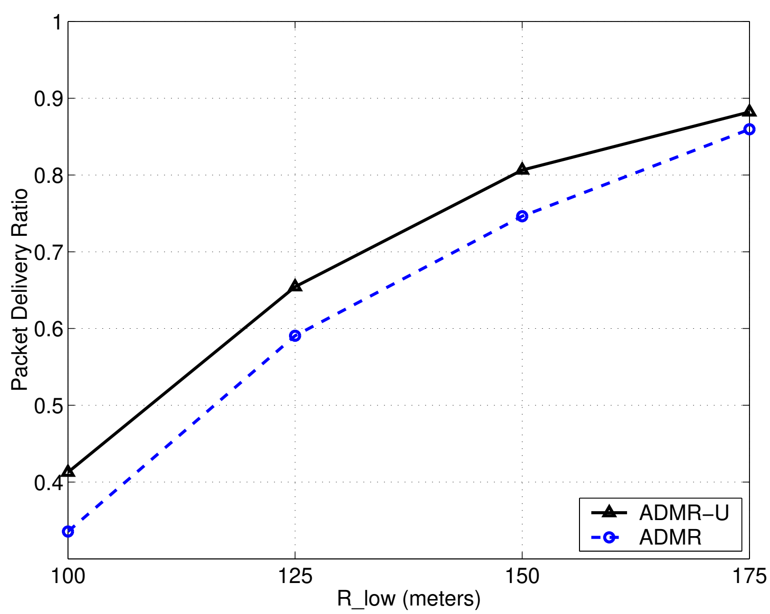

Figure 5.18: Packet Delivery Ratio: Heavy Scenario (0s pause time, 20 m/s)

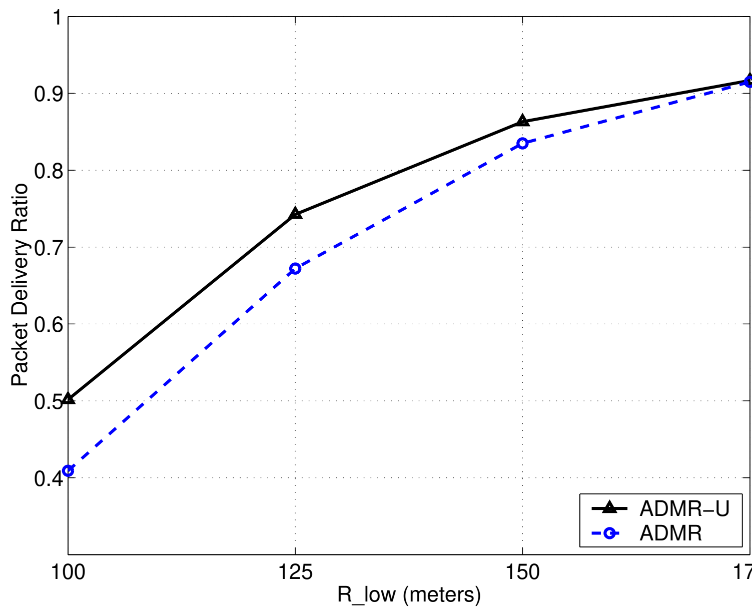

Figure 5.19: Packet Delivery Ratio: Heavy Scenario (0s pause time, 1 m/s)

5.5.5. ADMR-U Simulation Results

For my evaluation of ADMR-U, I use the simulation methodology presented in Section 5.3.1.

In networks with only bidirectional links, ADMR-U matches ADMR's packet delivery ratio, while matching or reducing ADMR's normalized packet overhead (Figures 5.12– 5.15, R_{low}= 250). This reduction is greatest in the static scenarios (37%) and is due to the local retransmissions of unicast control packets that ADMR-U performs while trying to verify the unidirectionality of a link; this leads to more reliable multicast state setup, as unicast control packets lost due to collisions are retransmitted, and as a result join and repair attempts do not need to be repeated and/or flooded.

When $R_{low} > 175$, reachability is high and network paths are predominantly bidirectional (Figure 5.1); in this case, both protocols deliver between 90 and 99% of their packets and incur only slightly more overhead than in networks with only bidirectional links. When $R_{low} < 100$, reachability is low and network paths are predominantly unidirectional (Figure 5.1); in this case, both protocols deliver very few of their packets (< 15%). ADMR-U incurs less overhead than ADMR in these scenarios, since it is better able to establish working paths between sources and receivers when such paths exist; it is also better able to distinguish between broken links and unidirectional links and performs fewer route repairs. In the rest of this section, I discuss only scenarios with R_{low} values between 100 and 175.

As discussed in Section 5.3.2, ADMR's reactive behavior causes it to generate a high level of overhead at moderate reachability levels and when the number of one-way and unidirectional paths is high (e.g., R_{low}= 100m). ADMR-U is able to reduce this overhead of ADMR by up to 50% in the 20 m/s scenarios (Figures 5.12 and 5.14). The results for the 1 m/s and static scenarios are similar except that the reduction in overhead is even more dramatic (up to 68%) (Figures 5.13 and 5.15). At lower levels of mobility, ADMR-U is able to set up multicast state with even less control overhead, while delivering even more data.

In addition to reducing ADMR's overhead, ADMR-U equals or exceeds ADMR's packet delivery ratio in *all* cases. ADMR-U delivers up to 45% more packets than ADMR in the mobile scenarios (Figures 5.17- 5.16 and 5.19-5.18) and up to 55% more packets in the static scenarios.

As discussed in Section 5.3.2, there are two ways to improve a protocol's ability to deliver data efficiently in challenging network environments such as the ones characterized by a high level of unidirectionality: ability to adapt to network conditions and multicast packet forwarding redundancy. The unidirectional extensions enable ADMR to adapt to the presence of unidirectional links in the network. I believe that adding buffering to the protocol as a form of packet forwarding redundancy will enable it to perform even better, as it will reduce the number of packets lost due to broken links and repair attempts. Some protocols, e.g., ODMRP, would not benefit from such buffering as they do not detect broken links or perform route repair; in fact ODMRP's redundant data forwarding is already a form of buffering, as multiple redundant copies of each data packet are stored and forwarded by many nodes the network.

5.6. Summary

Routing in wireless ad hoc networks is significantly more difficult when unidirectional links are present in the network, but there are many real-world scenarios in which unidirectional links may exist. The effect of unidirectional links on routing protocol performance has only been explored in the context of unicast, and while some approaches for routing over unidirectional links have been proposed, none of the proposed unidirectional extensions have been designed specifically for multicast. In addition, existing protocol extensions typically employ mechanisms for link directionality detection and dissemination that increase routing overhead even if unidirectional links are not present in the network, and in most cases utilize inefficient mechanisms such as periodic control packet exchanges by all nodes in the network.

In this chapter, I explored the effect of unidirectional links on the performance of the Adaptive Demand-Driven Multicast Routing protocol (ADMR) and the On-Demand Multicast Routing Protocol (ODMRP), and presented the design and evaluation of unidirectional extensions for on-demand multicast routing protocols. I used these extensions to enable ADMR to route over unidirectional links. The extended protocol, ADMR-U, has a number of desirable properties: the unidirectional extensions are only activated when ADMR encounters a unidirectional link and do not affect the operation of the protocol when the network consists of only bidirectional links; ADMR-U routes over unidirectional links without utilizing any proactive or periodic mechanisms; and it detects when a bidirectional link has become unidirectional and vice versa, adjusting its operation accordingly, without the use of any control packets. In networks with unidirectional links, ADMR-U matches or outperforms ADMR in terms of packet delivery ratio, by up to 45%, and also lowe rs its packet overhead by up to 68%.

Chapter 6

Conclusions and Future Work

Previous efforts to design general-purpose on-demand multicast routing protocols for ad hoc networks have utilized periodic (non-on-demand) mechanisms within some portions of the protocol. The overall on-demand nature of such protocols derives from the fact that significant portions of their operation are active only for active multicast groups. However, the periodic mechanisms within the protocol are responsible for core routing functionality and significantly affect overall protocol performance.

- *My thesis in this dissertation is that on-demand multicast that does not rely on periodic techniques is more efficient and performs better than multicast that utilizes such techniques.*

To support my thesis statement, I designed a new multicast protocol, the Adaptive Demand-Driven Multicast Routing protocol (ADMR) for multi-hop wireless ad hoc networks. ADMR uses no periodic control packet network-wide floods, periodic neighbor sensing, or periodic routing table exchanges, and adapts its behavior based on network conditions and application sending pattern, allowing efficient detection of broken links and expiration of routing state that is no longer needed. I conducted an extensive performance evaluation of ADMR in a wide range of simulation scenarios and showed that it works well and compares well against protocols that utilize proactive mechanisms, typically generating three to five times less packet overhead.

In addition, in this dissertation, I studied the impact of unidirectional links on the routing characteristics of ad hoc networks, and used this study to explore the effect of unidirectional links on multicast routing performance. Using the lessons learned from this work, I extended ADMR with mechanisms that enable it to route over unidirectional links and showed that the unidirectional extensions improve the performance of the protocol by increasing packet delivery ratio by up to 45% and decreasing overhead by up to 68%.

The main research contributions of this dissertation are as follows:

- The design of ADMR, the first general purpose-multicast routing protocol for ad hoc networks that does not utilize any periodic control packet transmissions.
- A detailed comparative performance evaluation of ADMR, ODMRP and MAODV which explores the effectiveness of different multicast mechanisms in a wide range of ad hoc network simulation scenarios.
- A detailed study of the impact of unidirectional links on the routing characteristics of ad hoc networks.
- The first study of the impact of unidirectional links on multicast protocol performance in ad hoc networks.

- The design and evaluation of the first unidirectional routing extensions specifically for multicast, which are also the first such extensions that operate entirely on-demand without at the same time requiring GPS information.

In future research, I plan to study ADMR's performance in a real implementation testbed. In addition, I would like to further explore the area of *routing protocol self-tuning*. ADMR adapts its timeout values based on the sending pattern of application programs. Further avenues for protocol adaptation include tuning of timeout values based on current network conditions. To operate well in the dynamic ad hoc networking environment, the routing protocol should be able to detect the state of the network and adjust its operation, by self-tuning its parameters. For example, if mobility in part of the network is high, some protocol timeout values may need to be reduced for faster broken link detection in that part of the network, and inversely, if mobility declines, the protocol timeout values there may be increased, to reduce network control traffic. This kind of self-tuning is a challenging problem. For example, detecting current network conditions is difficult because different conditions sometimes display similar symptoms; a perceived link break may be due to node movement or due to interference because of congestion. While in the case of increased mobility, reducing timeout values would be the appropriate response, in the case of congestion, reducing network control traffic would be the better option. I plan on investigating these issues through both simulation and implementation experimentation.

Bibliography

[1] Lichun Bao and J.J. Garcia-Luna-Aceves. Link-State Routing in Networks with Unidirectional Links. In *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)*, pages 358–363, October 1999.

[2] Bommaiah, McAuley, and Talpade. AMRoute: Adhoc Multicast Routing Protocol. Internet-Draft, draft-talpade-manet-amroute-00.txt, February 1999. Work in progress.

[3] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, October 1998.

[4] S. Bunchua and C.-K. Toh. Performance Evaluation of Flooding-Based and Associativity-Based Ad Hoc Mobile Multicast Routing Protocols. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, September 2000.

[5] C.-C. Chiang, Mario Gerla, and Lixia Zhang. Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks. *ACM Baltzer Journal of Cluster Computing: Special Issue on Mobile Computing*, 1(2):187–196, 1998.

[6] Steve Deering. Host Extensions for IP Multicasting. RFC 1112, August 1989.

[7] Kevin Fall and Kannan Varadhan, editors. *ns* Notes and Documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997. Available from http://www-mash.cs.berkeley.edu/ns/.

[8] J.J. Garcia-Luna-Aceves. Distributed, Scalable Routing Based on Vectors of Link States. *IEEE Journal on Selected Areas in Communications*, 13(8):1383–1395, October 1995.

[9] 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 99*, pages 784–792, March 1999.

[10] Zygmunt J. Haas. A Routing Protocol for the Reconfigurable Wireless Network. In *1997 IEEE 6th International Conference on Universal Personal Communications. Bridging the Way to the 21st Century, ICUPC '97*, volume 2, pages 562–566, October 1997.

[11] Zygmunt J. Haas and Marc R. Pearlman. The performance of query control schemes for the zone routing protocol. In *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 167–177, August 1998.

[12] Hugh Holbrook and Brad Cain. Source-Specific Multicast for IP. Internet-Draft, draft-holbrook-ssm-arch-01.txt, November 2000. Work in progress.

[13] IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York, New York, 1997.

[14] Jorjeta G. Jetcheva, Yih-Chun Hu, David Maltz, and David B. Johnson. A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-simple-mbcast-00.txt, November 2000. Work in progress.

[15] Jorjeta G. Jetcheva and David B. Johnson. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks. In *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 33–44, October 2001.

[16] L. Ji and M. S. Corson. A Lightweight Adaptive Multicast Algorithm. In *Proceedings of IEEE GLOBECOM '98*, pages 1036–1042, December 1998.

[17] L. Ji and M. S. Corson. Differential Destination Multicast (DDM) Specification. Internet-Draft, draft-ietf-manet-ddm-00.txt, July 2000. Work in progress.

[18] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[19] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-05.txt, March 2001. Work in progress.

[20] H-J. Kang and M-J. Lee. A Multi-source Multicast Routing Protocol for Ad-Hoc Networks. In *Lecture Notes in Computer Science Journal (LNCS)*, pages 309–320, January 2002.

[21] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, November 1998.

[22] Dongkyun Kim, C.-K. Toh, and Yanghee Choi. On Supporting Link Asymmetry in Mobile Ad Hoc Networks. In *IEEE Symposium on Ad Hoc Mobile Wireless Networks in conjunction with IEEE GLOBECOM 2001*, pages 2798–2803, November 2001.

[23] T. Kunz and E. Cheng. Multicasting in ad-hoc networks: Comparing MAODV and ODMRP. In *Proceedings of the Workshop on Ad Hoc Communications*, September 2001.

[24] S. Lee and C. Kim. Neighbor Supporting Ad Hoc Multicast Routing Protocol. In *Proceedings of the First Annual Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc 2000)*, August 2000.

[25] 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 IEEE INFOCOM 2000*, pages 565–574, March 2000.

[26] Mahesh K. Marina and Samir R. Das. Routing Performance in the Presence of Unidirectional Links in Multihop Wireless Networks. In *Proceedings of the 2002 ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002)*, pages 12–23, June 2002.

[27] Sanket Nesargi and Ravi Prakash. A Tunneling Approach to Routing with Unidirectional Links in Mobile Ad-Hoc Networks. In *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)*, pages 16–18, October 2000.

[28] M.R. Pearlman, Z.J.Haas, and B.P. Manvell. Using Multi-Hop Acknowledgements to Discover and Reliably Communicate over Unidirectional Links in Ad Hoc Networks. In *Wireless Communications and Networking Conference (WCNC)*, pages 532–537, September 2000.

[29] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994.

[30] Charles E. Perkins, Elizabeth M. Royer, and Samir R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. Internet-Draft, draft-ietf-manet-aodv-08.txt, March 2001. Work in progress.

[31] Carlos A. Pomalaza-Raez. A Distributed Routing Algorithm for Multihop Packet Radio Networks with Uni- and Bi-Directional Links. In *IEEE Transactions on Vehicular Technology*, pages 579–585, August 1995.

[32] Ravi Prakash. Unidirectional Links Prove Costly in Wireless Ad Hoc Networks. In *Proceedings of the ACM DIAL M Workshop*, pages 15–22, August 1999.

[33] Ravi Prakash. A Routing Algorithm for Wireless Ad Hoc Networks with Unidirectional Links. *ACM/Baltzer Wireless Networks Journal*, 7(6):617–626, November 2001.

[34] Venugopalan Ramasubramanian, Ranveer Chandra, and Daniel Mosse. Providing a Bidirectional Abstraction for Unidirectional Ad Hoc Networks. In *Proceedings of IEEE Infocom*, June 2002.

[35] Venugopalan Ramasubramanian and Daniel Mosse. Statistical Analysis of Connectivity in Unidirectional Ad Hoc Networks. In *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW)*, pages 109–116, August 2002.

[36] Giovanni Resta and Paolo Santi. An Analysis of the Node Spatial Distribution of the Random Waypoint Mobility Model for Ad Hoc Networks. In *Principles of Mobile Computing (POMC) Workshop*, October 2002.

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

[38] S.-J.Lee, Mario Gerla, and C.-C. Chiang. On-Demand Multicast Routing Protocol. In *Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC '99*, pages 1298–1304, September 1999.

[39] P. Sinha, R. Sivakumar, and V. Bharghavan. MCEDAR: Multicast core extraction distributed ad-hoc routing. In *Proceedings of the Wireless Communications and Networking Conference, WCNC '99*, pages 1313–1317, September 1999.

[40] Prasun Sinha, Srikanth Krishnamurthy, and Son Dao. Scalable Unidirectional Routing with Zone Routing Protocol (ZRP) extensions for Mobile Ad-Hoc Networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1329–1339, September 2000.

[41] C.-K. Toh, Guillermo Guichala, and Santithorn Bunchua. ABAM: On-Demand Associativity-Based Multicast Routing for Ad Hoc Mobile Networks. In *Proceedings of IEEE Vehicular Technology Conference, VTC 2000*, pages 987–993, September 2000.

[42] C.W. Wu, Y.C. Tay, and C-K. Toh. Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS). Internet-Draft, draft-ietf-manet-amris-spec-00.txt, November 1998. Work in progress.

[43] Jie Wu and Hailan Li. Domination and Its Applications in Ad Hoc Wireless Networks with Unidirectional Links. In *Proceedings of the International Conference on Parallel Processing (ICPP)*, pages 189–200, August 2000.