On-Demand Multicast Routing in Ad Hoc Networks with Unidirectional Links

Jorjeta G. Jetcheva and David B. Johnson

December 15, 2004
CMU-CS-04-175

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

This work was supported in part by NASA under grant NAG3-2534 at Rice University; by NSF under grants CNS-0209204, CNS-0325971, CNS-0338856, and CNS-0435425 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 at Carnegie Mellon University. The views and conclusions contained here are those of the authors 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, unidirectional links, routing, multicast, mesh networks.

Abstract

Many techniques used by routing protocols for ad hoc networks assume that links in the network can be used equally well in both directions between the two endpoint nodes of the link. However, there are many real-world situations in which wireless links may physically work in only one direction, resulting in degradation of routing performance in the network. In this paper, we present the first study of the effect of unidirectional links on the performance of *on-demand multicast* routing protocols for ad hoc networks and present mechanisms that enable such protocols to route efficiently over unidirectional links. We evaluate these mechanisms in the context of the Adaptive Demand-Driven Multicast Routing protocol (ADMR), and simulate the extended protocol, ADMR-U, in a wide range of mobile and static network scenarios with unidirectional links. In networks with only bidirectional links, the unidirectional extensions do not get activated and cause no overhead; in networks with unidirectional links, ADMR-U matches or outperforms ADMR in terms of packet delivery ratio, and lowers ADMR's packet overhead.

1. Introduction

A group of mobile wireless nodes that wish to communicate may self-organize into an *ad hoc network*. There are two main approaches in the design of routing protocols for ad hoc networks: the *proactive* approach and the *on-demand* (or reactive) approach. Proactive protocols discover network topology information through periodic control packets and operate in a manner independent of current communication needs and network conditions. On-demand protocols discover routing information only when it is needed for current communication, and are generally more efficient, as they scale their overhead according to communication demands in the network [3].

Most proposed *unicast* routing protocols and all *multicast* routing 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 situations 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 node 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**.

Most routing protocols generally 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, where a source node floods a control packet to which the destination of the discovery responds by unicasting a reply back towards the source.

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 [15] 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 route breaks. Such protocols are also forced to concentrate all traffic on routes in the network that are composed of only bidirectional links, which may cause excessive link overload, congestion, and battery exhaustion.

There has been a growing amount of research on routing over unidirectional links in wireless ad hoc networks for *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. Multicast routing is more difficult than unicast routing, for example because nodes may join or leave a multicast group at any time, making multicast addresses and group membership difficult to track, and because nodes along a route must generally forward each data packet to a number of next-hop nodes rather than a single next hop, making route maintenance solutions more complex. In addition, with the exception of the work of Kim et al [11], which proposes a scheme that requires power control and GPS location information, and the partial solution by Pearlman et al [14], all previous proposed protocol extensions for routing over unidirectional links employ periodic link directionality information exchange by all nodes in the network [2, 13, 16–21]. Such proactive solutions increase network load and energy consumption, even when no unidirectional links are present in the network, and are particularly inappropriate for on-demand protocols, which should avoid such periodic packets.

In this paper, we present the first study of the effects of unidirectional links on multicast routing performance, and we propose a set of on-demand mechanisms that enable efficient multicast routing over unidirectional links. We perform our study in the context of the Adaptive Demand-Driven Multicast Routing protocol (ADMR) [8], and we extend ADMR using our new mechanisms to create a new version of the protocol, ADMR-U, that efficiently supports unidirectional links; we present the design of ADMR-U and evaluate its performance. We chose ADMR since it is a general-purpose on-demand multicast protocol for ad hoc networks that has been shown to perform well [8] and because it employs a diverse set of on-demand mechanisms, thus allowing us to test almost all of our proposed unidirectional link mechanisms in one protocol.

This work is the first attempt to specifically extend a multicast protocol to route over unidirectional links, and unlike previous proposals for unicast routing over unidirectional links, the unidirectional link extensions in ADMR-U are automatically activated *only* when the protocol encounters a unidirectional link, without also requiring the use of GPS (location) information. In networks with only bidirectional links, the unidirectional extensions do not get activated and thus do not degrade ADMR's performance. 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%.

Section 2 of this paper describes our proposed mechanisms for enabling on-demand multicast routing protocols to route over unidirectional links. We present an overview of ADMR in Section 3 and the design of ADMR-U in Section 4. Sections 5 and 6 present our methodology and evaluation results, respectively. We discuss related work in Section 7 and conclude in Section 8.

2. Mechanisms for Multicast Routing 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.

- *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, we 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.*

2.1. 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, our approach relies on passive monitoring of received packets at a node; no control packets are sent specifically to discover the directionality of a link.

We 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 acknowldgement (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.

2.2. 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 link.

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 1). 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 2). 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 3). 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 4). 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 2.3.

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.

2.3. 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 2.2) 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 a packet since it is not setting up forwarding state but only needs to be delivered to its destination which would then send a state setup packet, which would perform Localized Bidirectional Path Search to make sure less unidirectional links are included in the forwarding mesh/tree.

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. We propose to configure 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.

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** on to 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 be used 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 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.

2.4. 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. We call the acknowledgment mechanism introduced here, Selective Source-Routed Multicast Acknowledgments.

We 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 node **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

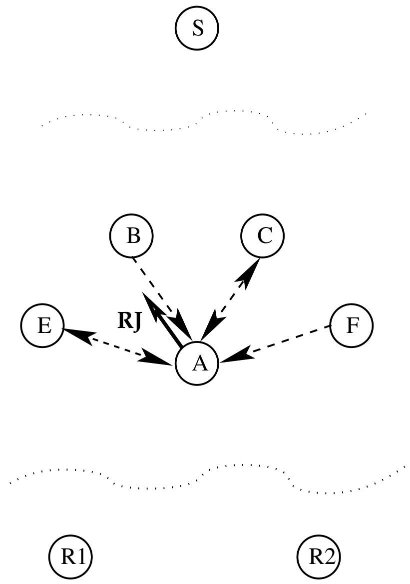

Figure 1: Node **A** attempts to send an ADMR RECEIVER JOIN to node **B**. The packet transmission is indicated by a solid arrow; the dashed arrows indicate link directionality.

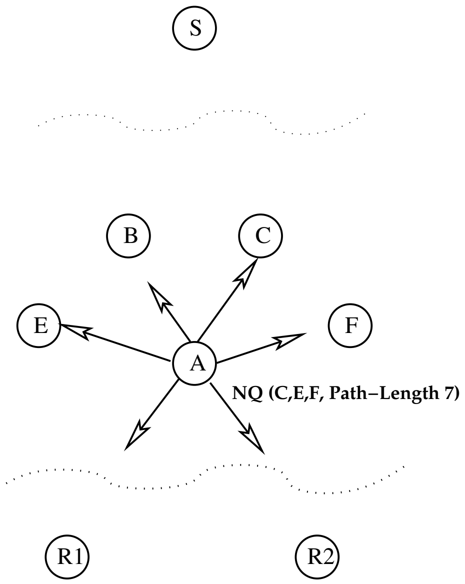

Figure 2: Node **A** broadcasts a NEIGHBOR QUERY to be processed by **C**, **E** and **F**; the maximum allowable path length to the source is 7 hops.

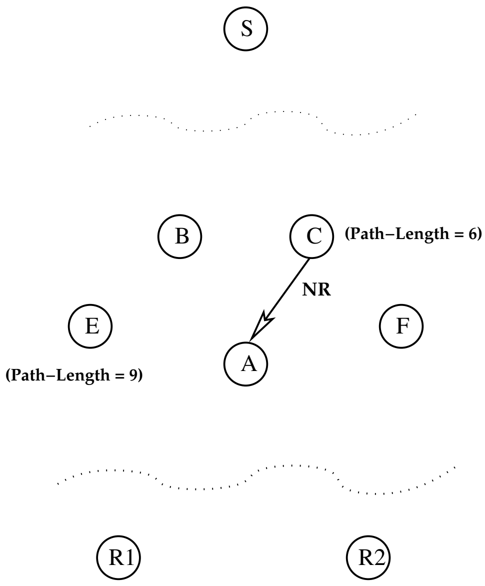

Figure 3: Node **C** sends a NEIGHBOR REPLY to node **A**.

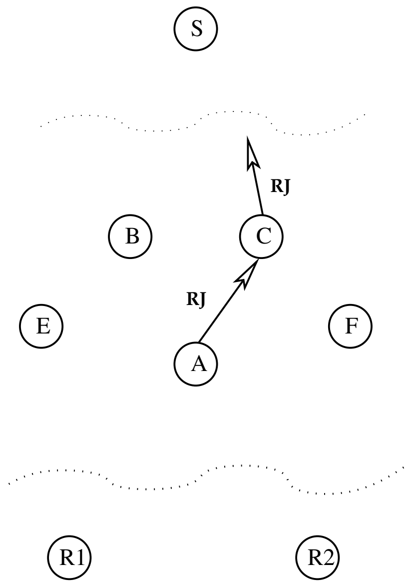

Figure 4: Node **A** forwards the RECEIVER JOIN towards the source over the newly discovered bidirectional link through node **C**.

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 from another node **A** with thet *No Ack* flag set, 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). This ACKNOWLEDGMENT PATH DISCOVERY 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 acknolwedgement 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 6).

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 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.

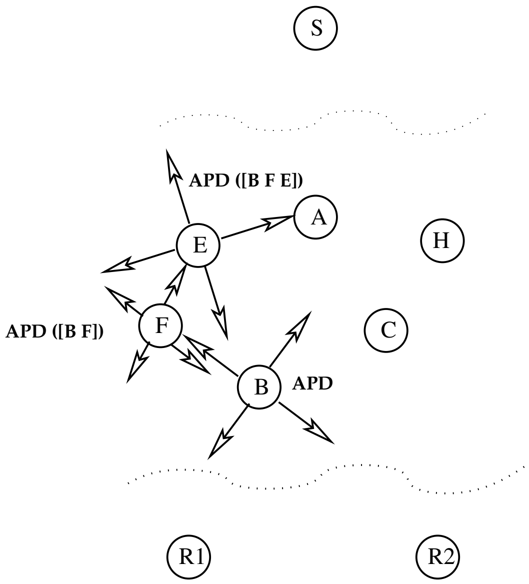

Figure 5: ACKNOWLEDGMENT PATH DISCOVERY flood towards node **A**.

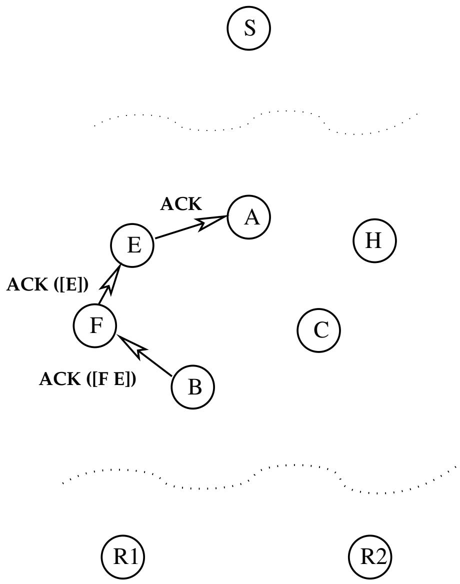

Figure 6: Source routing of ACTIVE ACKNOWLEDGMENT to node **A**.

2.5. 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 multicast mesh, say **B**, **C** and **D**. A node **A** 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 on to **A** first, node **B** may have a unidirectional link to **A** and thus require ACTIVE ACKNOWLEDGMENTs in order to continue forwarding packets (Section 4.4). 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 the packet based on collected information on the number of recently received ACTIVE ACKNOWLEDGMENTs and the number of recently overheard passive acknowledgments.

3. ADMR Overview

ADMR does not use periodic control packet transmissions of any kind. It operates efficiently by adapting to application behavior and network conditions and by performing routing control functions only when necessary for current communication.

Multicast sources and receivers in ADMR cooperate to create a source-based forwarding mesh. Data packets 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** along the shortest-delay path through the mesh to the receiver members of the multicast group. The mesh flooding mechanism increases robustness to link failure as packets may follow any currently available mesh links toward the receivers. Duplicate detection based on a unique sequence number carried in the header of each multicast packet ensures that each packet is forwarded at most once by each mesh node.

ADMR starts to perform its multicast mesh maintenance functions as soon as multicast state is created in the network. These include broken link detection and repair, and expiration of multicast state that is no longer needed for forwarding. ADMR performs its mesh maintenance functions efficiently by adapting to the behavior of each multicast application. A multicast source using ADMR monitors the traffic pattern of its multicast source application. This traffic pattern, characterized by the inter-arrival time of the packets sent by the application, is distributed in the header of each multicast data packet. In the event that a node in the forwarding mesh does not receive a packet within several packet inter-arrival times, it can detect that its link to the mesh is broken and initiate local repair (a limited-hop flood by the node which detected the broken link). ADMR uses its observations of an application's sending pattern to detect sources that have become inactive and will not be sending any more data. Once a source becomes inactive, multicast forwarding state is silently expired without the need to send an explicit shutdown message. To monitor for broken links in the multicast forwarding mesh when the source is temporarily not sending data and also to detect when a source is no longer sending data, ADMR sends a limited number of keep-alives (which could also be retransmitted data packets) at increasing inter-packet times. When the source application has not sent any data for a period of time that constitutes a significant deviation from its sending pattern, the keep-alives stop and the entire mesh silently expires. A significant deviation from a source's sending pattern is an indication that the source is likely to be inactive for a while, in which case it would be wasteful to maintain routing state in the network.

ADMR also prunes individual branches of the mesh automatically when they are no longer needed for forwarding.

4. ADMR-U

In this section, we describe how we 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 2. 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 2.1.

4.1. New Multicast Source

In this section, we 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. The ADMR header contains a number of fields including a sequence number generated by the source of the packet and used for duplicate detection, and an expected inter-arrival time at which the application is expected to send packets (initially set to a default value or supplied by the application).

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). The set of nodes with forwarding state for a group **G** and source **S** constitutes the forwarding mesh for **S** and **G**.

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, we have extended the protocol with Localized Bidirectional Path Search and Limited Multi-Hop Flooding, introduced in Sections 2.2 and 2.3.

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.

4.2. Receiver Application Join

In this section, we 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. When an application on a node joins a multicast group **G**, if the node is not already a receiver or a forwarder for the group, it floods a MULTICAST SOLICITATION packet.

Each MULTICAST SOLICITATION packet contains a unique sequence number for duplicate detection purposes, as well as the address of the multicast group and source (in the case of a source-specific join). Each node in the network forwards a non-duplicate copy of the flooded packet, and records the sequence number from the packet. When a node who is a forwarder or receiver for **G** and **S** 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. This packet can be either a data packet or an empty packet with an ADMR header in it. Upon receiving the UNICAST KEEPALIVE, the multicast receiver sends a RECEIVER JOIN towards the source, as described in Section 4.1.

Unidirectional Extensions. To enable UNICAST KEEPALIVE packets to be forwarded to the receiver over unidirectional links, we use Limited Multi-Hop Flooding (Section 2.3). We 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 2.3). 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 4.4), the MULTICAST SOLICITATION can be unicast towards the upstream node along this multi-hop path.

4.3. Broken Link Detection and Repair

In this section we 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 (Section 3), 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 4.2.

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. 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, we 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 4.2).

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** sends a RECEIVER JOIN packet towards the originator of the RECONNECT REPLY flood, node **C**.

4.4. Pruning of Multicast State

In this section, we 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, we extend ADMR with the Mesh Reconfiguration mechanism (Section 2.5) and the Selective Source-Routed Multicast Acknowledgments (Section 2.4), which we use for both data packets and keep-alives.

5. Methodology

5.1. Simulation Setup

We simulated the performance of ADMR using the ns-2 simulator [4] with Monarch wireless multicast extensions [1]. We modified the IEEE 802.11 [7] 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. We use 100 nodes, 10 of which are high-power nodes, with a range of R_{high} = 250m, and 90 of which are low-power nodes, with a range of R_{low}, which is varied between 25 and 225m at 25m increments. We chose to use these scenarios, since they represent a realistic heterogeneous ad hoc network, and since, after studying a variety of parameterizations of several power assignment models, we determined that they would allow us 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 [3]. Each node independently picks a random destination and speed from an interval $(0, \text{max\_speed})$ 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. We 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 our simulation runs and thus represents a static network).

We experimented with two multicast scenarios, which we 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, we ran 50 randomly generated scenarios, and each point in our graphs is the average of the results of these 50 scenarios.

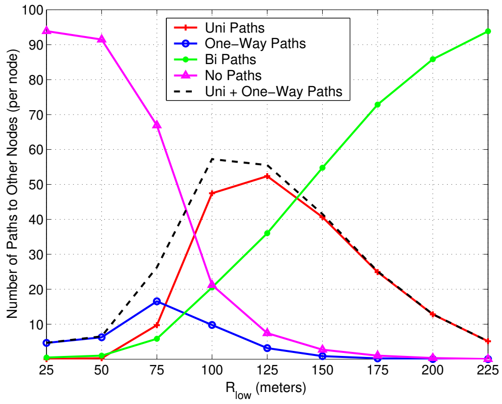

Figure 7: Node Reachability (0s pause time, 20 m/s maximum speed)

5.2. Unidirectional Scenario Characteristics

We define four *reachability metrics*, depending on the type of directionality of the shortest paths between nodes in the network: the average number of nodes to which a node has no path (*No Paths*), the average number of nodes to which a node has at least one bidirectional path (*Bi Paths*), the average number of nodes to which a node has a path but which do not have a path back to it (*One-Way Paths*), and the average number of nodes to which a node's shortest path is unidirectional and who's shortest path to this node is also unidirectional (*Uni Paths*). Figure 7 shows these metrics 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. At R_{low} of 250m, $R_{low} = R_{high}$ and all links in the the network are bidirectional.

Average shortest-path length in these scenarios varies between 3 and 4 hops, while the maximum shortest-path length varies between 7 and 25 hops, with path length decreasing with an increase in R_{low}.

5.3. Protocol Performance Metrics

We use two metrics to evaluate protocol performance: *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).

6. Performance Evaluation

6.1. Effect of Unidirectional Links on Multicast Routing Performance

Even though only 20% of the paths in the network scenario with R_{low}= 100m are bidirectional (Figure 7), ADMR is able to deliver between 33% and 42% of the packets (Figures 8, 9, 11 and 12). Overall, ADMR delivers up to 70% of the data packets in scenarios where the majority of the paths

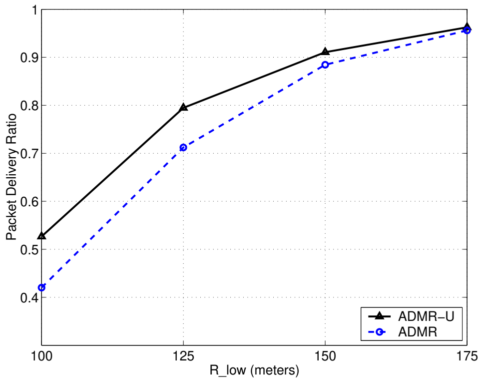

Figure 8: Packet Delivery Ratio: Light Scenario (0s pause time, 1 m/s)

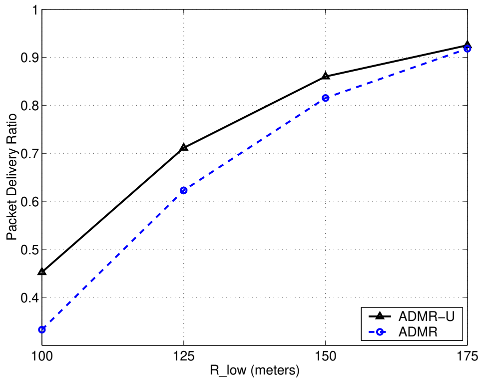

Figure 9: Packet Delivery Ratio: Light Scenario (0s pause time, 20 m/s)

in the network are unidirectional, i.e., for $R_{low} < 150$m. We do not show our results for R_{low} values lower than 100m since network connectivity is very poor in these scenarios and very few packets are delivered (Figure 7, No Paths curve).

The reason for the higher fraction of data delivered than might be expected from the connectivity metrics is that the protocol is very aggressive about multicast state setup and maintenance as long as there is at least one active source and one joined receiver for a given group. ADMR's reactive behavior is useful in improving its packet delivery ability; however, because of its unawareness of unidirectional links, it generates high normalized packet overhead in these cases (Figures 10 and 13). The non-monotonic behavior in the presence of unidirectional links is due to the protocol's reactive nature; the peak in the curve occurs at moderate reachability levels and when the number of one-way and unidirectional paths is highest (dotted curve in Figure 7). 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 paths 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, or when multicast state cannot be set up, as long as the floods are able to reach their destinations.

Adding unidirectional extensions to ADMR tempers its aggressiveness and reduces its overhead, by allowing it to distinguish between broken and unidirectional links and to then react appropriately. The extensions also enable the protocol to deliver more packets because it can then use more network paths and links.

6.2. Comparison between ADMR-U and ADMR

In networks with only bidirectional links, ADMR-U matches ADMR's packet delivery ratio, while matching or reducing ADMR's normalized packet overhead (Figures 10 and 13, $R_{low} = 250$). This reduction is greatest in scenarios with lower mobility and is due to the local retransmissions of unicast control packets that ADMR-U performs while trying to verify the unidirectionality of a link (Sections 2.2 and 2.3); these retransmissions lead to more reliable multicast state setup, as

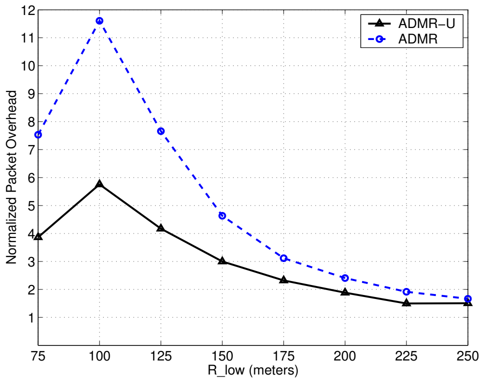

Figure 10: Normalized Packet Overhead: Light Scenario (0s pause time, 20 m/s)

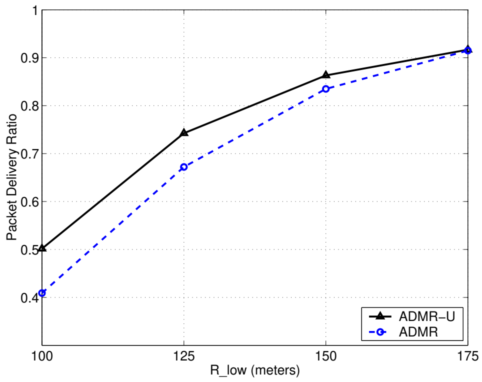

Figure 11: Packet Delivery Ratio: Heavy Scenario (0s pause time, 1 m/s)

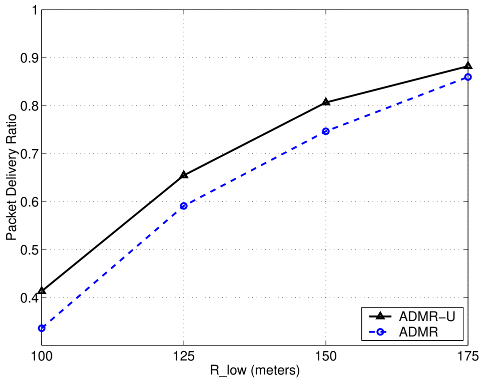

Figure 12: Packet Delivery Ratio: Heavy Scenario (0s pause time, 20 m/s)

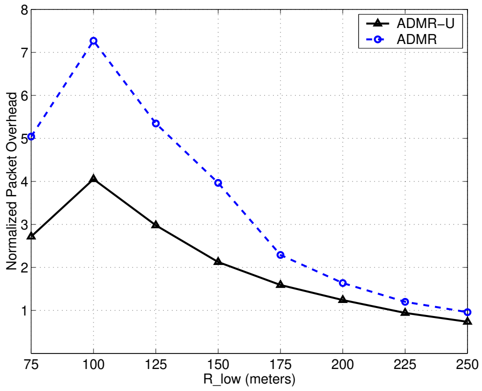

Figure 13: Normalized Packet Overhead: Heavy Scenario (0s pause time, 20 m/s)

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 7); in this case, both protocols deliver between 90 and 99% of data 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 7); in this case, both protocols deliver very few of the data 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, we discuss only scenarios with R_{low} values between 100 and 175.

As discussed in Section 6.1, 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 by up to 50% in the 20 m/s scenarios (Figures 10 and 13). 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%). At lower levels of mobility, ADMR-U is able to set up multicast state with even less control overhead, while delivering even more of the data packets.

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 8 and 9, and Figures 11 and 12) and up to 55% more packets in the static scenarios.

7. Related Work

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

Marina and Das [12] present several approaches to dealing with unidirectional links in the context of the Ad Hoc On-Demand Distance Vector protocol (AODV) [15]. 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 our work in several ways. First, none of their approaches enables the routing protocol to use unidirectional links. Second, the Reverse Path Search technique is similar to the Localized Bidirectional Path Search proposed here (Section 2) but unlike the limited bidirectional path search, its scope 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 [16] 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 [17].

Bao and Garcia-Luna-Aceves [2] 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) [5], in which 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. 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) [9, 10] 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 [15]. However, unidirectional links are only detected but cannot be used for routing, which limits the connectivity of the network.

Kim et al [11] 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 [13]. 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. This scheme works for proactive protocols but would need to be extended to work for on-demand protocols.

Another approach based on the idea of a sublayer is described by Ramasubramanian et al [19]. The protocol is called the Sub Routing Layer (SRL) protocol and relies on periodic reverse path updates transmitted by each node in the network.

Unidirectional extensions in the context of a hierarchical routing algorithm based on dominating sets are presented by Wu and Li [21]. Similar to the work of Ramasubramanian et al [19], 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 [20] suggest protocol-specific unidirectional extensions for the Zone Routing Protocol (ZRP) [6]. 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 [14]. 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.

8. Conclusion

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 paper, we explored the effect of unidirectional links on multicast routing performance and presented the design and evaluation of unidirectional extensions for on-demand multicast routing protocols, which we evaluated in the context of ADMR. The extended protocol, which we call 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 and is able to adjust 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 and also lowers its packet overhead.

References

[1] Monarch Project Wireless Multicast Extensions. Rice University, Department of Computer Science, Houston, Texas. http://www.monarch.cs.rice.edu/multicast_extensions.html.

[2] 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.

[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] 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/.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-dsr-09.txt, April 2003. Work in progress.

[11] 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.

[12] 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.

[13] 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.

[14] 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 *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 532–537, September 2000.

[15] Charles E. Perkins, Elizabeth M. Royer, and Samir R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. Internet-Draft, draft-ietf-manet-aodv-13.txt, February 2003. Work in progress.

[16] 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.

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

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

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

[20] 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)*, September 2000.

[21] 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.