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

Jorjeta G. Jetcheva
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891 USA
jorjeta@cs.cmu.edu

David B. Johnson
Department of Computer Science
Rice University
Houston, TX 77005-1892 USA
dbj@cs.rice.edu

http://monarch.cs.rice.edu/

Abstract

The use of on-demand techniques in routing protocols for multi-hop wireless ad hoc networks has been shown to have significant advantages in terms of reducing the routing protocol's overhead and improving its ability to react quickly to topology changes in the network. A number of on-demand *multicast* routing protocols have been proposed, but each also relies on significant periodic (non-on-demand) behavior within portions of the protocol. This paper presents the design and initial evaluation of the Adaptive Demand-Driven Multicast Routing protocol (ADMR), a new on-demand ad hoc network multicast routing protocol that attempts to reduce as much as possible any non-on-demand components within the protocol. Multicast routing state is dynamically established and maintained only for active groups and only in nodes located between multicast senders and receivers. Each multicast data packet is forwarded along the shortest-delay path with multicast forwarding state, from the sender to the receivers, and receivers dynamically adapt to the sending pattern of senders in order to efficiently balance overhead and maintenance of the multicast routing state as nodes in the network move or as wireless transmission conditions in the network change. We describe the operation of the ADMR protocol and present an initial evaluation of its performance based on detailed simulation in ad hoc networks of 50 mobile nodes. We show that ADMR achieves packet delivery ratios within 1% of a flooding-based protocol, while incurring half to a quarter of the overhead.

1. Introduction

Multicast routing is becoming an important networking service in the Internet for supporting applications such as remote conferencing, resource discovery, content distribution, and distributed games. In wireless *ad hoc networks*, these and other uses are expected to also be important. An ad hoc network is a group of wireless mobile nodes which self-organize into a network in order to communicate. Such networks can operate without the need for existing infrastructure or configuration. Each mobile node in the network acts as a router and forwards packets on behalf of other nodes. This "multi-hop" forwarding allows nodes beyond direct wireless communication range of each other to communicate. Routing protocols for ad hoc networks must discover such paths and maintain connectivity when links in these paths break due to effects such as node motion, radio propagation, or wireless interference.

Most prior work in ad hoc network routing has focused on routing of *unicast* packets, but a number of ad hoc network *multicast* routing protocols have been proposed over the past few years as well [19, 25, 8, 14, 13, 24, 23, 2, 4, 22], using a variety of basic routing algorithms and techniques. Of these multicast routing protocols, a few attempt to operate in an *on-demand* fashion [19, 22, 14, 13, 24, 12], in which the operation of the protocol is driven by the presence of data packets being sent rather than by continuous or periodic background activity of the protocol. For routing of *unicast* packets in ad hoc networks, such on-demand operation has been shown to have substantial advantages in terms of the routing protocol's overhead and ability to react quickly to routing changes in the network. We believe that similar benefits also hold for multicast routing in ad hoc networks.

However, designing an ad hoc network multicast routing protocol that operates *entirely* on-demand is difficult, although several such multicast protocols have been proposed [14, 12]. These protocols perform well in scenarios with small groups [14] or in networks in which mobility is very high and flooding is the only way to deliver packets successfully [12]. However, these protocols do not scale well and are not efficient to use as general-purpose multicast protocols.

Previous efforts to design general-purpose multicast protocols for ad hoc networks have utilized various 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 may substantially limit the benefits of the protocol's otherwise on-demand operation. For example, the On-Demand Multicast Routing Protocol (ODMRP) [19] builds multicast meshes through periodic network-wide control packet floods. The protocol relies on these floods to repair link breaks in the mesh that occur between the floods. The Multicast Ad Hoc On-Demand Distance Vector protocol

This work was supported in part by the NASA Cross Enterprise Technology Development Program under Grant Number NAG3-2534. 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, Rice University, Carnegie Mellon University, or the U.S. Government.

(MAODV) [22] requires continuous periodic neighbor sensing for link break detection, and periodic "group hello messages" for multicast forwarding state creation. The 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 (ABAM) protocol [24] requires continuous periodic neighbor sensing for link break detection and distribution of link characteristics. In addition, these protocols rely on explicit "prune" messages for deletion of forwarding state that is no longer needed. Loss of an explicit prune message because of wireless interference or because the sender of the prune message has moved out of range of the intended recipient of the prune, leads to significant unnecessary overhead as nodes continue forwarding packets even though there are no receivers for the group that are interested in receiving them downstream.

This paper presents the design and initial evaluation of the Adaptive Demand-Driven Multicast Routing protocol (ADMR), a new on-demand multicast routing protocol for wireless ad hoc networks that attempts to reduce as much as possible any non-on-demand components within the protocol.

In ADMR, source-based forwarding trees are created whenever there is at least one source and one receiver in the network. ADMR monitors the traffic pattern of the multicast source application, and based on that can detect link breaks in the tree, as well as sources that have become inactive and will not be sending any more data. In the former case, the protocol initiates local repair procedures, and then global repair if the local repair fails. In the latter case, multicast forwarding state is silently expired without the need to send an explicit shutdown message. To enable monitoring for link breaks in the multicast forwarding tree when the source is not sending data temporarily, ADMR sends a limited number of keep-alives at increasing inter-packet times. When the source 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 tree 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 tree automatically, when they are not necessary for forwarding. These pruning decisions are based on lack of passive acknowledgements from downstream, instead of relying on the receipt of an explicit prune message.

Each multicast data packet is forwarded from the sender to the multicast receivers using MAC-layer multicast transmissions along the shortest-delay path between nodes with forwarding state for the group.

To deal with partitions, ADMR occasionally sends an existing multicast data packet instead as a network flood, taking the place of the multicast distribution of this existing packet. This data packet flood is used only at infrequent intervals (e.g., once per several tens of seconds) and only when new data is being sent to the given multicast group; in addition, this flood is not a required protocol mechanism and does not represent core functionality.

ADMR also detects when mobility in the network is too high to allow timely multicast state setup and maintenance, without requiring GPS or other positioning information or additional control traffic. When such high mobility is detected, ADMR temporarily switches to flooding of each data packet, and after a short time, the protocol again attempts to operate efficiently with multicast routing, as the mobility in the network may have decreased.

To summarize, the novel features of ADMR include:

- ADMR uses no periodic network-wide floods of control packets, periodic neighbor sensing, or periodic routing table exchanges; and requires no core (no other protocol has all these properties in one protocol).
- ADMR adapts its behavior based on application sending pattern, allowing efficient detection of link breaks and expiration of routing state that is no longer needed.
- Bursty sources are handled by sending a limited number of keep-alives along the multicast tree, in order to distinguish lack of data from disconnection.
- ADMR uses passive acknowledgements for efficient automatic tree pruning.
- If there are no receivers, sources only flood infrequent existing data packets (to heal partitions) and do not transmit other data or control packets.
- ADMR can detect high mobility without the use of GPS or other positioning information or additional control traffic, and can switch to flooding for some period of time before reverting back to normal multicast operation.

Section 2 of this paper describes the design of the ADMR protocol, including the data structures and packet types used and the operation of the different aspects of the protocol. Section 3 describes our simulation methodology in evaluating ADMR, and Section 4 presents an evaluation of ADMR based on detailed simulations in ad hoc networks of 50 mobile nodes moving at average speeds of 1 m/s and at 20 m/s. We also compare this performance to that of ODMRP [19] running on the same simulation scenarios; we compare ADMR against ODMRP since it is the best-studied on-demand multicast protocol for ad hoc networks. In Section 5, we discuss related work, and finally, Section 6 summarizes and presents conclusions.

2. ADMR Protocol Description

2.1. Overview

Multicast senders and receivers using ADMR cooperate to establish and maintain forwarding state in the network to allow multicast communication. We assume 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 distance. 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 the sender.

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 for only specific senders [10]. 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.

The multicast forwarding state for a given multicast group **G** and sender **S** in ADMR is conceptually represented as a loosely-structured multicast forwarding tree rooted at **S**. Each multicast packet is dynamically forwarded from **S** along the shortest-delay path through the tree to the receiver members of the multicast group.

Only members of the multicast forwarding tree forward multicast packets, and each node forwards each packet at most once. In addition, packets are not constrained to follow any particular branches or parent/child links while being forwarded. For example, in Figure 1, receiver **R1** receives packet **X** through node **B** but receives packet **Y** through node **D**. This can happen when node **D** acquires the media before **B** and forwards packet MHΦΦ**Y** first, or when **B** does not receive the packet correctly due to wireless interference and is therefore unable to forward it. (All figures pertaining to the protocol description depict transmission of packets as arrows. For clarity, reception of a packet is generally only shown for nodes that will forward the packet further, i.e., we do not depict each transmission as a broadcast received by all nodes within range, even though the wireless medium we are considering is a broadcast medium.)

We refer to the flood of a packet constrained to the nodes in the multicast forwarding tree as a *tree flood*, and to the more general type of flood of a packet through all nodes as a *network flood* (Figure 2). This use of flooding within the multicast forwarding tree is similar to the "forwarding group" concept introduced in the FGMP protocol [4] and used also in ODMRP [19], except that our forwarding state is specific to each sender rather than being shared for the entire group. When a sender using ADMR sends a multicast packet, it floods within the multicast distribution tree *only* towards the group's receivers, whereas with FGMP or ODMRP, the packet also floods back towards any other senders that are not receivers. Although this difference requires us to maintain source-specific state in forwarding nodes, such state is required anyway in order to support the source-specific multicast service model [10]. 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.

If the MAC layer in use in the network supports multicast addressing and packet transmission, ADMR takes advantage of it by causing receivers and nodes in the multicast forwarding tree to join the MAC-layer multicast group corresponding to the network-layer multicast group address. By utilizing MAC-layer multicast when available, ADMR limits the overhead on other nodes in the network due to multicast packet transmission.

Each multicast packet originated by some node **S** for multicast group **G** contains a small ADMR header, including a number of fields used by the protocol in forwarding the packet and in maintaining the multicast distribution tree for **S** and **G**. The *sequence 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 **S**. The *hop count* is initialized by **S** to 0 and is incremented by each node forwarding the packet. For a packet being forwarded, the *previous hop address* in the ADMR header is the MAC-layer transmitting source address from which this packet was received, copied from the MAC-layer header of the packet before forwarding it; when a packet is originated, this field is initialized to 0.

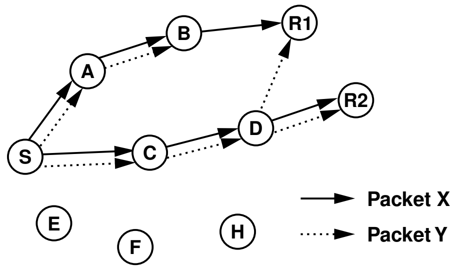

Figure 1 Multicast Data Packet Forwarding

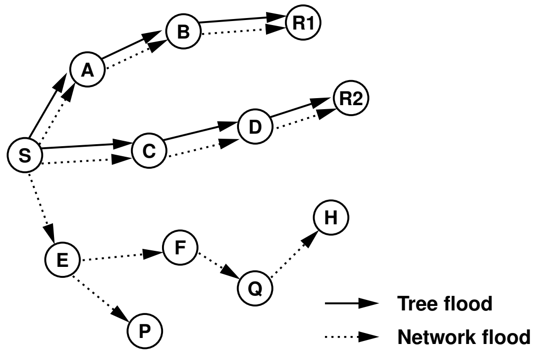

Figure 2 Tree Flood vs. Network Flood

The ADMR header also includes the *inter-packet time* (interval) at which new packets should be expected from this sender **S** for this group **G**. This field in the ADMR header is initialized by **S** based on dynamically tracking the average interval at which it originates multicast packets for group **G**. If the application layer at node **S** originates no new multicast packets for **G** within some multiple (e.g., 1.5) of this current inter-packet time, the routing layer at **S** begins originating "keep-alive" packets for **G**; the 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 distribution tree for **S** and **G**. The inter-packet time between keep-alives 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 this sender and group 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 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 tree is scheduled to expire, if the sender application does not begin to send new multicast data packets before that time.

Absence of data packets and keep-alives within a multiple of the inter-packet time is an indication of forwarding tree disconnection. 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 inter-packet time, it performs a local repair procedure to reconnect to the tree. If the local repair procedure fails, receivers who got their previous packet through **F** and are now disconnected from the source, perform a global reconnect procedure by sending a network flood.

ADMR performs automatic pruning of branches of the multicast tree that are no longer needed for forwarding. Pruning decisions are based on lack of passive acknowledgements from downstream, instead of relying on the receipt of an explicit prune message.

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. Although it may be useful to share information between the unicast and multicast protocol, not doing so improves modularity and portability. We are also currently studying the trade-offs of various levels of unicast-multicast cooperation within ADMR.

ADMR currently operates only over bidirectional links. We are working on extending the specification to handle unidirectional links as well.

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 address for which this node is an active sender. Each entry in the Sender Table includes the current inter-packet time for this node sending to the group, and a count of consecutive keep-alive packets sent to the group since the last data packet sent to the group by this node.
- *Membership Table*: Logically contains one entry for each combination of multicast group address and sender address for which this node is either a receiver member or a forwarder. Each entry in the Membership Table includes a flag to indicate if this node is a receiver, a flag to indicate if this node is a forwarder, the current inter-packet time for the sender sending to this group, and the current value of the keep-alive count from packets received for the group.
- *Node Table*: Logically contains one entry for each other node in the network from which this node has received a tree flooded or network flooded ADMR packet. Each entry in the Node Table includes the sequence number from the ADMR header of the most recent such packet, plus a bitmap representing a number of previous sequence numbers of packets from this sender, used to detect and discard duplicate packets during a flood: if the bit corresponding to some sequence number in this bitmap is set, the packet is assumed to be a duplicate; all sequence numbers prior to that 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 [18]. Each entry in the Node Table also includes the previous hop address, taken from the MAC-layer transmitting source address of the packet received from this sender with this sequence number that contained the minimum hop count 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.

2.3. Multicast Packet Forwarding

Any packet with a multicast or broadcast destination address containing an ADMR header will be flooded. The type of flooding is indicated by the *flood type* flag in the packet's ADMR header. For most packets, the flood type flag is set to cause a *tree flood* of the packet, such that the packet will be forwarded only among those nodes belonging to the multicast forwarding tree indicated by the source address (the original sender) and destination address (the multicast group address) in the packet (Figure 2). When a node receives such a packet, it checks its Membership Table entry for this group and source to determine if it should forward the packet; the packet thus flows along the tree from the sender to the group receivers but is not constrained to follow specific branches in the tree and is thus able to automatically be forwarded around temporarily broken links or failed forwarding nodes in the tree (Figure 1). If, instead, the flood type flag in the ADMR header indicates a *network flood* for the packet, the packet will be flooded among all nodes.For either type of flood, each node's Node Table and the sequence number in a packet's ADMR header reliably limit any node that should forward the packet to do so at most once.

When a node receives such a 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 the source of the packet. If the new hop count is less than that already recorded in the Node Table entry, this node updates the entry with the new hop count and sets the previous hop address in the entry to the MAC-layer source address from which it received the packet. 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 this group and source. If so, it passes the packet up within the protocol stack to allow the packet to be processed as a received multicast packet.

2.4. 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 inter-packet time in this entry may be set to a default value, may be assumed based on the IP 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 and flags it to send the packet as a network flood, as described in Section 2.3.

After sending this packet, node **S** buffers for a short time subsequent multicast packets that it might originate to group **G**, rather than sending them immediately as they are generated, in order to allow the routing state in the network to be formed for receivers interested in this group and sender. Once **S** receives at least one RECEIVER JOIN packet, **S** then begins sending any buffered packets to the group as normal multicast packets. The packet exchange which takes place when a new source becomes active is depicted in Figure 3. Most subsequent multicast packets for group **G** from node **S** will be flooded only within the members of the multicast forwarding tree established for this group and sender (a tree flood). However, it is possible that some interested receivers did not receive this initial packet from **S**. To allow for such occurrences, node **S** uses a network flood rather than a tree flood for certain of its subsequent existing multicast data 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 our simulations, the first data packet after 5 seconds since the initial network flood data packet, is sent as a network flood; the first data packet after 10 additional seconds is also sent as a network flood, as is one data packet after each subsequent 30 seconds. These network floods are sent only when there is data to be sent by **S** to **G**.

2.5. Receiver 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, with the group address **G** as the destination address of the packet. If the group is a source-specific multicast group, the specific sender address **S** requested by the application is included after the ADMR header in the packet.

The forwarding of the packet through the network follows the procedure described in Section 2.3. However, in the case of source-specific multicast, the specified source does not forward the MULTICAST SOLICITATION packet. Also in this case, if a node receiving the MULTICAST SOLICITATION has a Node Table entry for

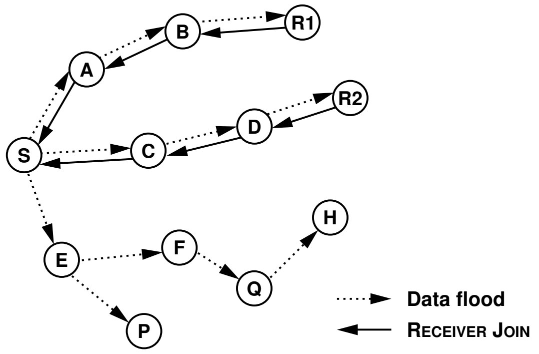

Figure 3 New Source

this source, and has a Membership Table entry for this group and source indicating that it is a forwarder, this node will instead *unicast* the Multicast Solicitation only to the previous hop address indicated in that Node Table entry; the packet thus follows the tree towards the source, decreasing the overhead and speeding up the receiver join.

When any source **S** for multicast group **G** receives the Multicast Solicitation packet (or the single source, in the case of a source-specific multicast group join), the source replies to the Multicast Solicitation to advertise to **R** its existence as a sender for the group. This reply may take one of two forms. If the next scheduled network flood of an existing multicast data packet (Section 2.4) is to occur soon, **S** may choose to advance the time for this network flood and use it as the reply for the Multicast Solicitation from **R**. 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**, following the path taken by **R**'s Multicast Solicitation packet; each node forwarding this unicast keep-alive packet unicasts it to the address recorded in the previous hop address field of that node's Node Table entry for **R**, created when it forwarded **R**'s Multicast Solicitation as it traveled toward **S** (Figure 4). When forwarding this 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 path back to **S** in each entry's previous hop address field.

When node **R** receives the reply from its Multicast Solicitation, it will process it as described in Section 2.6, since **R** will view this reply as a packet from a multicast source that it is not yet connected to. **R** also sends a Receiver Join packet back to **S**, creating the forwarding state to connect it to the multicast forwarding tree for this group and source (Figure 5).

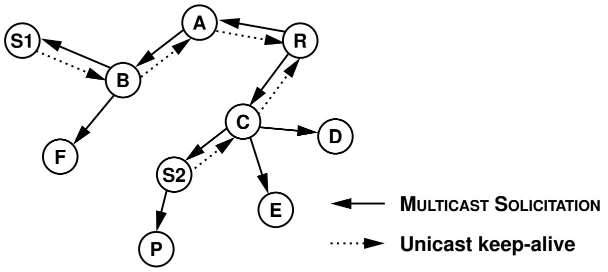

Figure 4 Sources respond to receiver's Multicast Solicitation

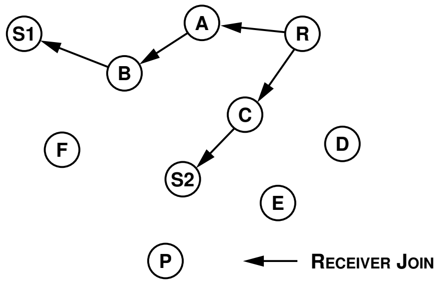

Figure 5 Receiver sends Receiver Join

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 the Receiver Join from **R** within a short time. If **S** does not receive the Receiver Join, it will retransmit its reply to **R**'s Multicast Solicitation (which again, may be in the form of **S**'s next network flood of an existing multicast data packet or may use a unicast ADMR keep-alive packet). If the timer expires a second time and **S** has not received a Receiver Join from **R**, then **S** assumes that the path that the unicast keep-alive is trying to traverse, created by the forwarding of **R**'s Multicast Solicitation to **S**, is broken, and **S** advances its next scheduled network flood of a multicast data packet to reply to **R**. No further specific retransmissions of the reply are attempted, although the normal occasional network flood mechanism of selected existing multicast data packets from **S** to the group will eventually reach **R**.

2.6. Receiving from a New Multicast Source

When a node **R** receives any multicast packet, in addition to forwarding the packet if required by the forwarding procedure described in Section 2.3, node **R** also checks the entry for this sender and group in its Membership Table to determine if it is a receiver member. If so, then **R** processes it as a multicast packet that it is intended to receive, passing the packet up to the next layer within its receiving protocol stack.

In addition, if the packet was sent as a tree flood (rather than as a network flood), this indicates that the receiver node **R** is currently connected to the multicast forwarding tree for this sender and group. The node considers itself to remain connected until detecting that it has become disconnected, as described in Section 2.8.

If, instead, this received packet was sent as a network flood or the packet is a unicast keep-alive (Section 2.5), and if the receiver **R** is not currently connected to the multicast forwarding tree for this sender and group, then **R** replies with a Receiver Join packet, to cause the necessary nodes along the path back to the sender **S** to become forwarders. Node **R** initializes the inter-packet time field in its Receiver Join packet to the inter-packet time from the ADMR header of the received packet. The Receiver Join packet then follows the path established by the forwarding of the received multicast data packet or keep-alive packet, as recorded in the previous hop address field in each node's Node Table entry for this sender **S**. Each node that forwards the Receiver Join, if it does not already have a Membership Table entry for this group and source, creates a new entry, with the inter-packet time initialized from the Receiver Join packet's header; each node that forwards the Receiver Join also sets the forwarder flag in its Membership Table entry.

If there are multiple new receivers for a given multicast group **G** near each other in the network, many Receiver Join packets will traverse the same paths or subpaths on their way to the source **S**. However, in order to make each node along these paths a forwarder for **G** and **S**, as necessary, it is enough for one RECEIVER JOIN packet to be received and 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, but doing so would leave the connection of these new receivers susceptible to the loss of the single RECEIVER JOIN packet forwarded. To reduce overhead and yet provide resilience to such packet loss, each node will forward at most 3 RECEIVER JOIN packets for the last sequence number it has recorded in its Node Table entry for **G** and **S**.

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 inter-packet time contained in the ADMR header of the received packet that triggered its RECEIVER JOIN; in our simulations, we use a timer value of 3 times the inter-packet time. If this timer expires before any new multicast packets have been received from the source, the receiver 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.

2.7. Local Subtree Repair

Forwarders or receiver members of some multicast group may become disconnected from the multicast forwarding tree for the group, as nodes in the network move or as wireless transmission conditions change. Each forwarder or receiver for some multicast group **G** and source **S** detects that it has become disconnected from the multicast forwarding tree 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 **G** and sender **S** for which it is either a forwarder or a receiver member, and resets this timer each time it receives a packet for the group. The timer value is based on the inter-packet time value in the ADMR header of the last received packet, plus a time proportional to the node's hop count from the source **S**, as determined by the forwarding of the last packet from **S** that updated the node's Node Table entry for **S**. This small increase in disconnection timeout value as a function of hop count is intended to generally allow nodes closer to **S** (i.e., closer to a broken link on the path from **S**) to detect the disconnection before nodes further from **S**. This property is not required for correct operation of the protocol, but it improves the efficiency of the repair process.

When some node **C** detects disconnection, it initiates a *local repair* of the multicast forwarding tree, as illustrated in Figure 6. Node **C** first sends a REPAIR NOTIFICATION packet to the other nodes in the subtree "below" node **C** in the multicast distribution tree for group **G** and sender **S**. Here, the subtree "below" is defined by the previous hop address recorded in each node's Node Table for sender **S**, such that any node whose previous hop for **S** is node **C** or is some other node below **C** is defined to be below **C** in the tree. Although as described in Section 2.3, each multicast packet is forwarded through the tree without regard to such relationships, this relationship represents the set of nodes that received the previous multicast packet through **C** and who will thus possibly detect the disconnection themselves later, due to the increase in disconnection timer values with hop count from **S**.

To forward the REPAIR NOTIFICATION packet to the nodes in the subtree below **C**, each node accepts and forwards 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**. In addition, the sequence number and bitmap

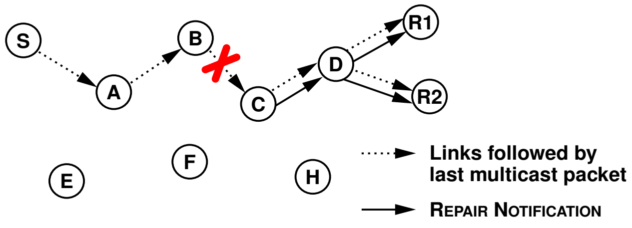

Figure 6 Node downstream of break initiates local repair

in each node's Node Table entry (Section 2.2) are used to avoid duplicates in the forwarding of the REPAIR NOTIFICATION packet.

After sending the REPAIR NOTIFICATION packet, node **C** waits for a *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 this other node will perform the repair.

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 is indeed the one that is broken. The REPAIR NOTIFICATION will be received by nodes directly below **C** in the forwarding tree, and if the link from **C** to its parent **B** in the tree (Figure 6) 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 the multicast source **S**. If the REPAIR NOTIFICATION is received by this parent node, it recognizes that one of the nodes directly below it in the tree (node **C**) is performing a local repair. The parent then sends a one-hop REPAIR NOTIFICATION to **C**, causing it to cancel its local repair as described above.

When a receiver member of the group receives a REPAIR NOTIFICATION, it postpones its disconnection timer for an interval of time determined by an estimate of the amount of time the local repair is expected to take.

After the short delay described above, if node **C** has not received a REPAIR NOTIFICATION initiated by an upstream node for this group and source, node **C** sends a hop-limited RECONNECT packet as a form of network flood (Figure 7). The RECONNECT packet identifies the group and source for which the local repair is being performed. The hop count limit (TTL) for the RECONNECT packet (e.g., 3) limits this flood to only reaching nodes near **C**.

In addition to the normal handling of a network flood in deciding whether or not to forward the RECONNECT packet, nodes that are forwarders for the group **G** and source **S** being repaired treat the packet specially. Such a node (e.g., node **A** in Figure 7), if it has not received a REPAIR NOTIFICATION for this repair, assumes that it is upstream of the repair node **C** and that it is therefore still connected

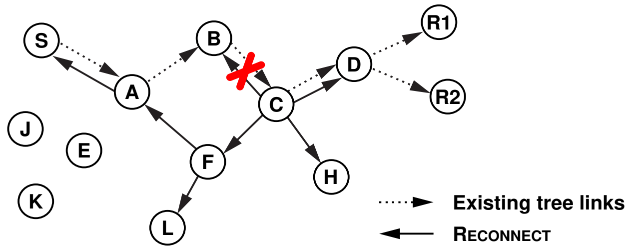

Figure 7 Disconnected node sends RECONNECT packet

to the source **S** in the tree. Rather than forwarding the RECONNECT packet as part of the hop-limited network flood, the node instead reinitializes the packet's hop count limit (TTL) to the default value and unicasts the packet to the node listed as its parent in the previous hop address field in its Node Table entry for **S**. This packet is no longer treated as a network flood packet, and is instead forwarded by each node in turn to its parent in the same way, until reaching **S**. If the node is in fact not upstream from the repair node **C** and its unicast RECONNECT reaches **C**, node **C** will discard the packet.

If the RECONNECT reaches **S**, node **S** responds with a RECONNECT REPLY packet, as illustrated in Figure 8. This RECONNECT REPLY packet is unicast back to the repair node **C** along the path the RECONNECT took to reach **S**, as recorded in the Node Table entry at each node for **C**. Each node through which the RECONNECT REPLY packet is forwarded on the path to **C** becomes a forwarder for the multicast group **G** and source **S**, and creates an entry in its Membership Table to record this if it is not already a forwarder for it.

2.8. Receiver-Initiated Repair

If the local repair procedure as described in Section 2.7 succeeds, the multicast forwarding tree will be reconnected and the receiver members will continue to receive data as expected. If the disconnection timer expires at some receiver member **R** for a group **G** and source **S**, this is an indication that the local repair has probably failed, perhaps because the amount of mobility in the network has been too great to allow the type of hop-limited repair attempted. In this case, node **R** performs its own individual repair by rejoining the group and source in the same way as when it originally joined, as described in Section 2.5.

Each receiver keeps track of how many times it had to perform a re-join to a group because of disconnection. When this number reaches a threshold, the receiver sets the "high mobility" flag in the ADMR header of the RECEIVER JOIN packet. When the source receives some number of RECEIVER JOINs with this flag set, it switches to flooding mode in which each subsequent multicast packet is set as a network flood. The high number of re-joins indicate that the multicast state setup cannot keep up with the high mobility in the network and only flooding can deliver the data successfully. 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. Tree Pruning

Each forwarder node in the multicast forwarding tree for some group **G** and source **S** automatically expires its own state and leaves the tree when it determines that it is no longer necessary for multicast forwarding. Similarly, the multicast source **S** automatically expires its state and stops transmitting multicast data packets when it determines that there are no downstream receiver members of the group for this source; the sender continues to send certain of its subsequent multicast packets as infrequent background network flood packets, but otherwise defers sending other multicasts for this group until receiving at least one new RECEIVER JOIN packet, as described in Section 2.4. This mechanism helps to prune nodes from the forwarding tree 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.

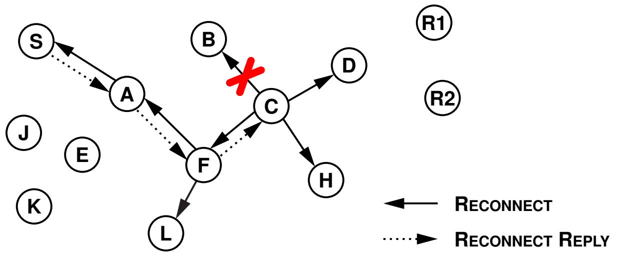

Figure 8 Source sends RECONNECT REPLY

The decision to expire this state is based at each such node on whether the multicast packets that it originates (at **S**) or forwards (at forwarder nodes) are subsequently forwarded by other nodes that received them from this node. In order to determine this for each multicast packet, a node **B** expects to hear at least one other node **C** that received the packet from **B** forward it. As described in Section 2.1, when node **C** receives and forwards a packet, **C** copies the MAC-layer source address of the received packet (i.e., node **B**'s address) into the previous hop address field in the packet's ADMR header, before forwarding the packet. If node **B** overhears **C** transmit this packet, **B** considers this as confirmation that it should continue forwarding subsequent multicast packets, so that nodes such as **C** can continue to receive them. On the other hand, if **B** fails to receive such confirmation for a number of consecutive multicast packets that it sends, then **B** decides that it is no longer necessary in the multicast forwarding tree for this group and source (or in the case of the source **S** itself, that no receiver members or forwarders remain). This technique is similar to the use of passive acknowledgements.

3. Evaluation Methodology

We evaluated the performance of ADMR through detailed packet-level simulation in a variety of mobility and communication scenarios. In addition, we have simulated the On-Demand Multicast Routing Protocol (ODMRP) [19], which has been shown to perform well in previous studies, and we compared its performance to that of ADMR.

3.1. Simulation Environment

We conducted our simulations using the *ns-2* network simulator [7] with our Monarch Project wireless and mobile *ns-2* extensions [3, 21]. The *ns-2* simulator is a publicly available discrete-event simulator, widely used in networking research. The Monarch Project wireless and mobile extensions incorporate models of signal strength, radio propagation, propagation delay, wireless medium contention, capture effect, interference, and arbitrary continuous node mobility. The radio model is based on the Lucent/Agere WaveLAN/OriNOCO IEEE 802.11 product, which provides a 2 Mbps transmission rate and a nominal transmission range of 250m, depending on capture effect and colliding packets. The link layer model is the Distributed Coordination Function (DCF) of the IEEE 802.11 wireless LAN standard [11]. We have extended the existing simulation models to enable multicast simulations with ADMR and ODMRP in the simulator.

3.2. Simulation Scenarios

In each simulation run, we simulate the behavior of 50 nodes forming a mobile ad hoc network in a 1500 m × 300 m area, operating over 900 seconds of simulated time. Each run of the simulator executes a scenario containing all movement behavior of the ad hoc network nodes and all application-layer communication originated by the nodes, generated in advance so that it can be replayed identically for the different routing protocols and variants studied. Each routing protocol was thus challenged by an identical workload.

The nodes in our simulations move according to the Random Waypoint model [3, 16]. Each node independently starts at a random location in the simulation area and remains stationary for a period of time called the *pause time*. The node then chooses a random new location to move to and speed to move at, both uniformly randomly generated, and once it reaches that new location, again remains stationary for the pause time. Each node independently repeats this movement pattern over the duration of each simulation run.

For our simulation experiments, we study runs with a maximum node movement speed of 20 m/s and others with a maximum node movement speed of 1 m/s. For each maximum node movement speed, we studied 7 different pause times: 0, 30, 60, 120, 300, 600, and 900 seconds; a pause time of 0 represents a network in which all nodes move continuously, whereas a pause time of 900 represents a stationary network. For each of these pause times and maximum node movement speeds, we randomly generated 10 different scenarios, and we present here the average over those 10 scenarios.

The multicast application-layer sources in our simulations generate constant bit rate (CBR) traffic, with each source originating 4 64-byte packets per second. This sending rate was chosen to significantly challenge the routing protocols' abilities to successfully deliver data packets in a mobile ad hoc network. It was not chosen to represent any particular application or class of applications, although it could be considered to abstractly model a very simple broadcast audio distribution application.

We used 4 different combinations of number of multicast groups, sources, and receivers. The first 3 cases consist of 1 multicast source and 5, 15, and 30 multicast receivers, respectively. These cases were designed to allow us to observe the behavior of the routing protocol in an environment that is understandable and possible to analyze. The fourth case consists of 3 groups with 3 sources and 10 receivers each. This case is designed to explore the behavior of the routing protocol in the presence of multiple, larger groups. The multicast sources start sending data and the multicast receivers join a group at uniformly randomly generated times between 0 and 180 seconds.

3.3. Performance Metrics

We evaluated the performance of ADMR and compared it to that of ODMRP using the following metrics:

- *Packet delivery ratio*: The fraction of multicast data packets originated by the application layer on a multicast source that are received by the application layer of the multicast receivers. For example, in a multicast group with 1 sender and 2 receivers, each multicast data packet originated should be received a total of 2 times across the 2 receivers; if 100 data packets were originated, 150 total received packets represents a packet delivery ratio of 0.75.
- *Normalized packet overhead*: The total number of all data and control packets transmitted by any node in the network (either originated or forwarded), divided by the total number of all data packets received across all multicast receivers. This metric represents the total packet overhead normalized by the successful results obtained in terms of data packets delivered.
- *Forwarding efficiency*: The average number of times each originated multicast data packet was transmitted by the routing protocol. This metric represents the efficiency of the basic multicast forwarding within the routing protocol.
- *Delivery latency*: The average time from when a multicast data packet is originated by a source until it is successfully received by a multicast receiver, counting each receiver individually.

3.4. Summary of the ODMRP Protocol

We use the On-Demand Multicast Routing Protocol (ODMRP) as a point of comparison for ADMR because it has been demonstrated to perform well and is well documented by its designers [19, 20]. In addition, our multicast forwarding tree flooding operates similarly to the forwarding group flooding in ODMRP, allowing us to better compare the other aspects and overall behavior of the two protocols.

ODMRP operates by periodically flooding the network with a control packet to re-create the multicast forwarding state. This mechanism serves two goals: to set up multicast forwarding state when a source first starts sending multicast data, and to recover from partitions and breaks in the forwarding tree as a result of node movement or a change in propagation conditions. The main trade-off in the protocol design is between overhead and the latency until tree breakage is repaired. Frequent network floods reduce the latency to tree breakage discovery and increase the packet delivery ratio but create a significant amount of packet overhead. Reducing the frequency of the floods reduces the overhead but increases the latency to tree breakage discovery and decreases the packet delivery ratio.

In particular, while a multicast source using ODMRP is active, the source periodically floods the network with a JOIN QUERY control packet. This packet is forwarded by every node in the network. In addition, nodes remember the address of the previous hop node from which they received the JOIN QUERY, so that when receivers for the group respond by sending a JOIN REPLY packet, this packet can be forwarded along the path that the JOIN QUERY took to reach the receiver. Each node that forwards the JOIN REPLY sets a *group forwarding flag* for the group indicated in the header of the packet. Each JOIN QUERY flood entirely re-creates the forwarding state for the group (reinitializes which nodes will be forwarders for the group), but this forwarding state expires after a multiple of the interval between successive JOIN QUERY floods.

When a node receives a packet for a multicast group for which it has a set forwarding flag, the node forwards the packet if it is not a duplicate. The individual unique identifiers of recently forwarded multicast packets are kept by each node for duplicate detection in the flooding process.

These procedures allow for redundant forwarding to each receiver, increasing the packet delivery ratio of the protocol: if a packet is dropped on one path as a result of collision or a link break, the receiver can receive it along another path. The benefits of this redundancy come at the cost of additional overhead and additional load on the network. A packet sent by one source is forwarded by all nodes that have their forwarding flags for the group set, allowing packets to traverse paths that may even lead "away" from the receivers and only towards the other sources. The more sources there are per group, the more redundancy and overhead is generated by each packet sent by any of the multicast sources.

Redundancy in ODMRP forwarding is also created by the fact that the lifetime of each forwarding flag setting is equal to a multiple of the periodic JOIN QUERY flooding interval. For example, in the published simulation results for ODMRP, the forwarding state lifetime is 3 times the JOIN QUERY flood interval, allowing some of the JOIN QUERY packets to be lost without losing forwarding state. However, in the worst case, there may be three sets of forwarding nodes for the traffic of each multicast source at any one time, if the network is highly mobile or the JOIN QUERY floods take different paths through the network. This redundancy again increases the packet delivery ratio of the protocol but also increases the overhead for each data packet and the overall load on the network.

3.5. ADMR and ODMRP Simulation Parameters

In our simulations of ADMR, we used 30 seconds for the periodic data flood interval, 1.2 for the multiplicative factor for the average inter-packet time in the absence of data, and 2 missing packets to trigger disconnection detection.

To compare different aspects of ADMR's performance to that of ODMRP, we evaluated three different variations on the ODMRP parameters. The "ODMRP-baseline" variation represents ODMRP using the parameter values chosen by ODMRP's designers in their published simulations of the protocol: 3 seconds for the JOIN QUERY flooding interval, and a forwarding state lifetime of 3 times this interval (a total of 9 seconds). The "ODMRP-3-1.1" and "ODMRP-4-1.1" variations both reduce the forwarding state lifetime to 1.1 times the JOIN QUERY flooding interval (rather than 3 times this interval); this change attempts to evaluate the effect of the forwarding redundancy present in the way that ODMRP rebuilds the forwarding state before the lifetime of the existing state has expired. The "ODMRP-4-1.1" variation also lengthen this JOIN QUERY flooding interval from 3 seconds up to 4 seconds, allowing us to evaluate the effect of the frequent complete rebuilding of the forwarding state in ODMRP.

4. Simulation Results

For each set of simulations of ADMR and the three variants of ODMRP to which we compared it, we studied 7 different pause times, 2 different maximum node movement speeds, and 4 different combinations of number of multicast groups, sources, and receivers. Each point in our performance graphs represents the average of 10 simulation runs for that combination of parameters and scenarios.

Figure 9 shows the packet delivery ratio of ADMR and ODMRP-baseline as a function of pause time in the 1-source, 15-receivers scenarios. The y-axis scale in this graph and other graphs in this paper for packet delivery ratio ranges from 0.8 to 1.0 to better show the detail in the performance curves plotted. Both ADMR and ODMRP deliver over 98% of the originated multicast data packets, even in highly mobile networks (small pause times). ODMRP-baseline delivers about 1% more packets on average than ADMR. However, to achieve this packet delivery ratio, ODMRP expends close to twice as much overhead as ADMR (Figure 10). In addition to frequently flooding the network and rebuilding its forwarding tree, ODMRP owes a large fraction of its overhead to redundant data packet forwarding (Figure 11); whereas ADMR forwards each data packet roughly 10 times on average in these scenarios, ODMRP forwards each packet roughly 17 times.

As described in Section 3.4, ODMRP creates forwarding state within nodes in the network, that is not expired when it is no longer needed but instead expires after a fixed timeout. This timeout is set to a multiple of the periodic JOIN QUERY flood interval in order to ensure that loss of the flood packets will not cause disruption in the delivery of multicast data. However, this mechanism leads to the creation of redundant state in the network, since new nodes may become forwarders for a group, while forwarders created during a previous periodic flood still have a set forwarding flag and may overhear packets for that group. While the redundancy that ODMRP creates increases its resilience to losses, it significantly increases the load on the network and the battery consumption of the nodes in the network. As a result of the high load, overall network performance degrades, and packet latency goes up slightly (Figure 12). ADMR also creates redundant state in the network when, as a result of tree breakage and repair, the forwarding tree no longer includes certain nodes that were part of the tree before the tree breakage. However, nodes that forward for a source in ADMR expire their forwarding state adaptively when there are no receivers downstream that are interested in receiving the multicast packets through them, as described in Section 2.9.

ODMRP-3-1.1 and ODMRP-4-1.1 incur a much lower overhead than ODMRP-baseline (Figures 10 and 11). However the packet delivery ratio of the protocol suffers dramatically under these parameterizations, especially in highly mobile networks at lower pause times (Figure 9). Since ODMRP does not explicitly react to link breaks but instead relies on redundant state and fixed periodic control floods to maintain multicast connectivity, it is not able to keep up with the link breaks introduced at high levels of mobility. In addition, ODMRP-3-1.1 and ODMRP-4-1.1 produce a higher delivery latency than ODMRP-baseline. This increase is due to a corresponding increase in the average path length (number of hops) used to deliver multicast data packets in the ODMRP-3-1.1 and ODMRP-4-1.1 cases (data not presented here to conserve space). Since ODMRP-3-1.1 and ODMRP-4-1.1 create less redundant forwarding nodes, packets are more constrained to follow a limited set of paths, whereas in ODMRP-baseline, the greater redundancy allows the data packet flooding among the forwarding nodes to automatically produce more direct paths by finding the shortest path through the nodes involved in the flood.

The performance results are similar in the other two 1-source scenarios (5 and 30 receivers). The difference in the 30-receiver from the 15-receiver results is that when a larger fraction of the nodes are receivers, a larger fraction of the nodes have forwarding state, and the density of nodes with forwarding state is higher. This creates a natural redundancy which both protocols exploit through the flood forwarding of the multicast data packets within the forwarding nodes. Also, since a larger number of forwarding nodes are required to connect the larger number of receivers, the number of other nodes that can redundantly become forwarding nodes in ODMRP is reduced. Furthermore, since the packet overhead is presented normalized to the number of receivers (the number of total received packets), these cases with more receivers generally show lower overhead. The opposite holds in the 5-receiver case: ODMRP incurs four times the overhead of ADMR, and delivers a similar fraction of the data packets, except in the ODMRP-3-1.1 and ODMRP-4-1.1 parameterizations, whose packet delivery ratio drops even more at high mobility rates than in the 15-receiver scenarios.

In terms of packet delivery ratio, at a maximum node movement speed of 1 m/s rather than 20 m/s, ADMR and all three variants of ODMRP deliver almost all of the originated multicast data packets. ODMRP-3-1.1 and ODMRP-4-1.1 perform better than they did at 20 m/s because the mobility in these scenarios is much lower and the need for redundant forwarding nodes is less. Overall overhead in these lower mobility scenarios also is less than in the cases with speeds of 20 m/s. In ADMR's case, this decrease is due to the fact that there are fewer tree breakages. In ODMRP's case, this is due to the creation of less redundant state, since the multicast forwarding tree is mostly the same between periodic rebuilds of the tree.

Figures 13 through 16 show the results for the higher load case of 3 groups, with 3 sources and 10 receivers each. Since in this scenario, there is more than 1 source per group, forwarding nodes in ODMRP for one source also forward packets on behalf of all other sources, creating additional redundancy and additional overhead. In

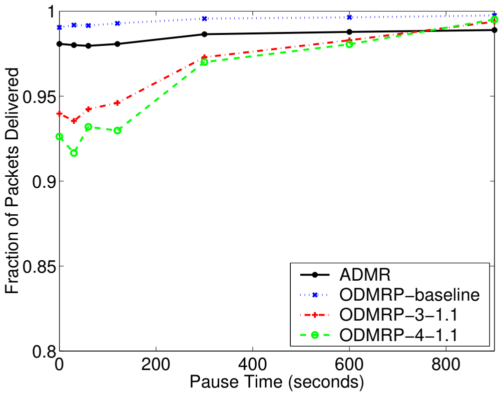

Figure 9 Packet Delivery Ratio: 1 source, 15 receivers, 20 m/s

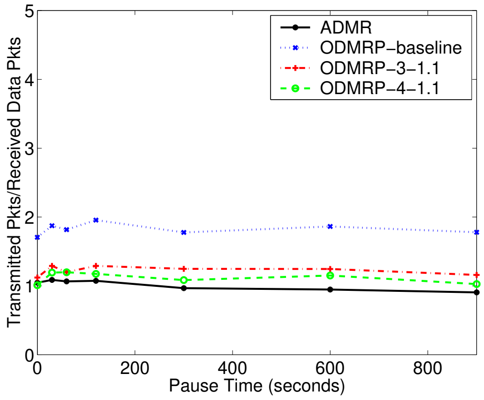

Figure 10 Normalized Packet Overhead: 1 source, 15 receivers, 20 m/s

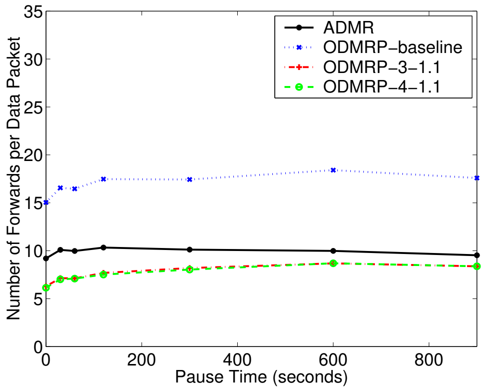

Figure 11 Forwarding Efficiency: 1 source, 15 receivers, 20 m/s

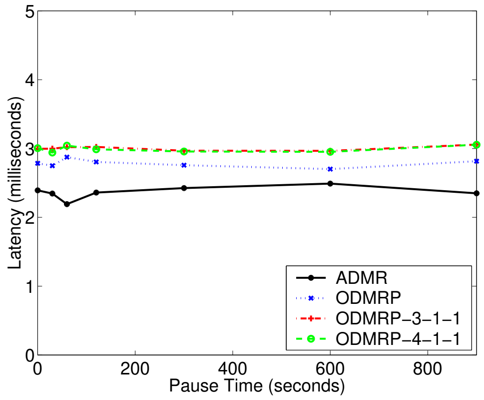

Figure 12 Delivery Latency: 1 source, 15 receivers, 20 m/s

particular, packets may be forwarded not only towards the receivers but also towards other sources in the group.

In these scenarios, the load on the network is significantly higher than in the previous cases discussed. As a result, the packet delivery ratio for all protocols decreases to an average of 96%. ODMRP performs slightly better than ADMR in these scenarios due to the extra redundancy it creates in the network (Figures 14 and 15). However, ODMRP causes 4 times the number of transmissions per packet than ADMR. In addition to maintaining its low overhead, ADMR maintains a similar level of latency as it did in the 1-source scenarios (Figure 16 and 12).

ODMRP-3-1.1 and ODMRP-4-1.1 incur a much lower overhead, as expected. Their throughput does not drop as significantly as it did in the 1-source scenarios at high levels of mobility, however, as there is still a lot of redundant forwarding that compensates for losses (Figure 15).

ODMRP performs well when a large percentage of the nodes in the network are receivers. ADMR also performs well in this case, though it achieves a more marked performance improvement when the receiver membership in the network is sparse. It is hard to find a parameterization of ODMRP that is able to deliver a large fraction of the data packets, yet has as small an overhead as ADMR. ODMRP is also very sensitive to mobility, especially if only a small fraction of the nodes in the network participate in forwarding its packets. ADMR's adaptive behavior allows it to scale its overhead as needed and to deliver its data efficiently and at a low latency.

5. Related Work

A number of protocols have been proposed for multicast routing in wireless ad hoc networks. Some are based on on-demand mechanisms [19, 22, 14, 24, 13, 12, 9], driven by data that needs to be delivered, whereas others rely on proactive (periodic) mechanisms to operate [4, 25, 2, 8, 23]. Our protocol falls in the former category and we therefore limit our discussion in this section to on-demand multicast protocols. We omit here discussion of ODMRP here as it was discussed in detail in Sections 3.4 and 4.

In the MAODV protocol [22], a "group leader" for each multicast group periodically floods a Group Hello control message throughout the ad hoc network. Multicast sources and receivers reply to this message, and these replies enable group forwarding state on the paths to the source. The resulting tree is rooted at the group leader. This tree is similar to the tree built by ODMRP in that packets sent by a source travel not only toward receivers for the group but also toward sources for the group. Unlike ODMRP, where only an active source floods control packets to rebuild the tree, in MAODV both multicast receivers and sources can become group leaders, and thus a multicast receiver may periodically flood the network even though there are no senders for the group. In addition, MAODV requires periodic neighbor sensing for link breakage detection. This neighbor sensing uses Hello messages which are periodically broadcast locally by each node in the network, if the node has not sent another broadcast packet within the periodic interval.

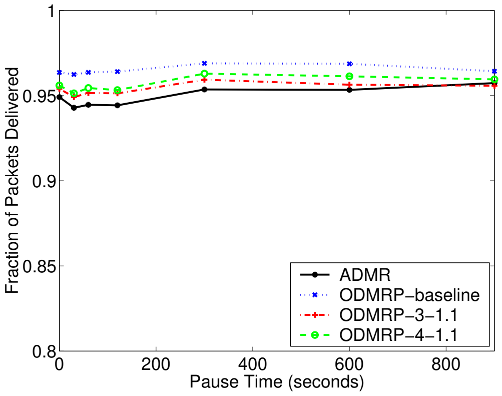

Figure 13 Packet Delivery Ratio: 3 groups, 3 sources, 10 receivers per group, 20 m/s

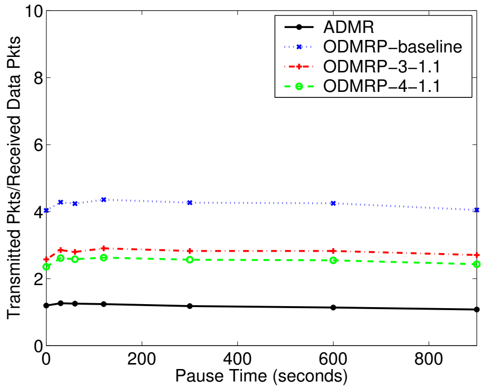

Figure 14 Normalized Packet Overhead: 3 groups, 3 sources, 10 receivers per group, 20 m/s

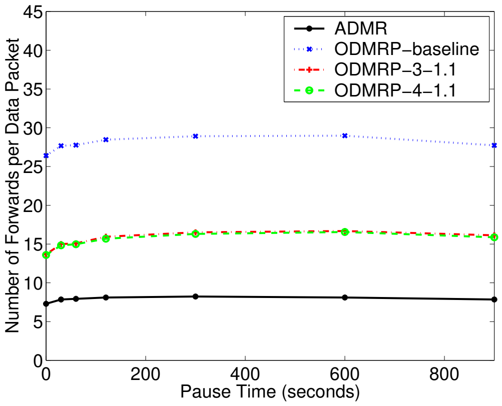

Figure 15 Forwarding Efficiency: 3 groups, 3 sources, 10 receivers per group, 20 m/s

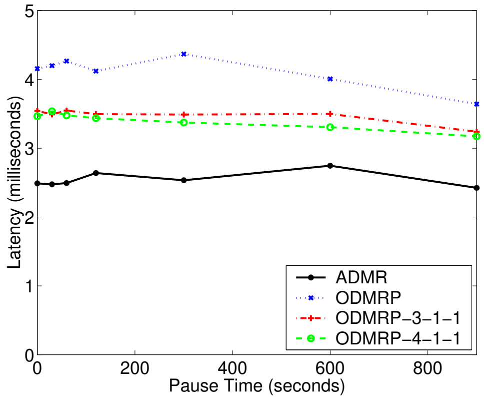

Figure 16 Delivery Latency: 3 groups, 3 sources, 10 receivers per group, 20 m/s

The LAM [13] protocol operates only in conjunction with TORA [5], an on-demand protocol for unicast routing in ad hoc networks. LAM's design is inspired by the Core Based Trees protocol [1]. Each group in LAM has a *core* associated with it. Receivers join a multicast group by establishing a route to the core node for the group they wish to join. Each source also establishes a route to the core and unicasts all of its packets to the core, which then forwards them to the multicast receivers. Core election algorithms are not discussed in LAM's specification. Detection and repair of link breaks is performed by TORA as part of its normal unicast operation. TORA, and as a result LAM, relies on the following assumptions for correct operation: that a lower-layer protocol ensures that each node has complete neighbor information; that all transmitted packets are received correctly and in order of transmission; and that each node is able to "broadcast" to all of its neighbors.

ABAM [24] is an on-demand protocol based on associativity. Associativity refers both to the stability of a link between two node and to the stability of a path which consists of multiple links. Stability information is acquired by each node through beacons, and can incorporate factors like signal strength and battery life. When a multicast source first becomes active, it floods a control packet in the network. This packet collects information on the "temporal, connection, and power stability" of each node that forwards it. The receivers wait until they receive some predefined number of copies of the flooded control packet and send a response back towards the source along the path that has the "best" associativity. ABAM reacts to tree breakage by first attempting a local repair, and if that fails, a global repair initiated by the receivers. Tree breakage is detected through periodic neighbor sensing similar to MAODV.

The DDM [14] multicast routing protocol logically encodes all multicast receivers in each data packet. It relies on the unicast forwarding information at each node along the path to each receiver to know the correct next hop towards that receiver. DDM is not a general-purpose multicast routing protocol because its header-encoded destination mechanism can only handle groups of limited size, and because it relies on the unicast routing protocol for all of its routing information.

The Simple Broadcast and Multicast Protocol [12] is based on the DSR [15, 16, 17] protocol. It does not incur any overhead when there is no data to be delivered to the multicast group. When data needs to be delivered, each data packet is flooded. The protocol is suitable for small networks or for networks characterized by a high level of mobility that makes the creation and maintenance of forwarding state hard or impossible to do.

ZRP [9] is a unicast routing protocol that can be used also to support multicast. However, few details of its multicast operation have been described.

6. Conclusion

In this paper, we have introduced the Adaptive Demand-Driven Multicast Routing protocol (ADMR) for multicast routing in wireless ad hoc networks. Previous efforts to design a general-purpose on-demand multicast routing protocol for ad hoc networks have utilized periodic (non-on-demand) mechanisms to enable some core routing functionality. ADMR uses no periodic network-wide floods of control packets, periodic neighbor sensing, or periodic routing table exchanges, and requires no core. The protocol adapts its behavior based on application sending pattern, allowing efficient detection of link breaks and expiration of routing state that is no longer needed. ADMR handles bursty sources by sending limited keep-alives for a period, to distinguish disconnections from lack of data. If there are no receivers in the network, sources only flood existing data at infrequent intervals (to heal partitions) and do not transmit other data or control packets. Furthermore, this is optional functionality and does not affect the main routing mechanisms used by the protocol.

We have presented an initial performance evaluation of ADMR and compared it to ODMRP, which has been shown to perform well and is perhaps the previously best-studied on-demand multicast protocol for ad hoc networks. ADMR delivers within 1% of the multicast data packets at approximately half to a quarter of the overhead generated by ODMRP. As a result of the lower load that ADMR imposes on the network, packet latency is up to 1.6 times lower than with ODMRP. ADMR's overhead scales gracefully with group size and with increased mobility. In addition, ADMR can detect when mobility in the network is too high to allow timely multicast state setup, without requiring GPS or other positioning information or additional control traffic; when such high mobility is detected, an ADMR source can switch to flooding for some period of time, after which it may attempt to operate efficiently with multicast again in case the mobility in the network has decreased.

Acknowledgements

We would like to thank the other members of the Monarch Project for their valuable feedback on the design and presentation in this paper. We also thank the anonymous reviewers for their helpful suggestions on the paper's presentation.

References

[1] A. Ballardie. Core Based Trees (CBT) Multicast Routing Architecture. RFC 2201, September 1997.

[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] 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.

[5] M. Scott Corson and Anthony Ephremides. A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, 1(1):61–81, feb 1995.

[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 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.

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

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

[11] 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.

[12] Jorjeta G. Jetcheva, Yih-Chun Hu, David A. Maltz, and David B. Johnson. A Simple Protocol for Multicast and Broadcast in Mobile Ad Hoc Networks. Internet-Draft, draft-ietf-manet-simple-mbcast-01.txt, July 2001. Work in progress.

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

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

[15] David B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] The Monarch Project. Monarch Project: Mobile Networking Architectures. Available at http://monarch.cs.rice.edu/.

[22] 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.

[23] P. Sinha, R. Sivakumar, and V. Bharghavan. MCEDAR: Multicast Core Extraction Distributed Ad-Hoc Routing. In *In Proceedings of the Wireless Communications and Networking Conference, WCNC '99.*, pages 1313–1317, September 1999.

[24] 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.

[25] 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.