Mobile Networks and Applications 7, 429–439, 2002
© 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.

AMRoute: Ad Hoc Multicast Routing Protocol

JASON XIE
University of Wisconsin, USA

RAJESH R. TALPADE and ANTHONY MCAULEY
Telcordia Technologies, USA

MINGYAN LIU
University of Maryland, USA

Abstract. The Ad hoc Multicast Routing protocol (AMRoute) presents a novel approach for robust IP Multicast in mobile ad hoc networks by exploiting user-multicast trees and dynamic logical cores. It creates a bidirectional, shared tree for data distribution using only group senders and receivers as tree nodes. Unicast tunnels are used as tree links to connect neighbors on the *user-multicast tree*. Thus, AMRoute does not need to be supported by network nodes that are not interested/capable of multicast, and group state cost is incurred only by group senders and receivers. Also, the use of tunnels as tree links implies that tree structure does not need to change even in case of a dynamic network topology, which reduces the signaling traffic and packet loss. Thus AMRoute does not need to track network dynamics; the underlying unicast protocol is solely responsible for this function. AMRoute does not require a *specific* unicast routing protocol; therefore, it can operate seamlessly over separate domains with different unicast protocols. Certain tree nodes are designated by AMRoute as *logical cores*, and are responsible for initiating and managing the signaling component of AMRoute, such as detection of group members and tree setup. Logical cores differ significantly from those in CBT and PIM-SM, since they are not a central point for data distribution and can migrate dynamically among member nodes. Simulation results (using ns-2) demonstrate that AMRoute signaling traffic remains at relatively low level for typical group sizes. The results also indicate that group members receive a high proportion of data multicast by senders, even in the case of a highly dynamic network.

Keywords: IP Multicast, mobile ad hoc networks, network protocols, routing

1. Introduction

Mobile Ad hoc Networks (MANETs [8]) are envisioned to have dynamic, sometimes rapidly-changing, random, multi-hop topologies that are likely composed of relatively bandwidth-constrained wireless links. Individual nodes and even whole networks of nodes may continuously move, or disappear, leading to highly dynamic topologies. The TCP/IP protocol suite is expected to be used for robust communication over heterogeneous network technologies in MANETs. Efficient IP Multicast is needed for disseminating information to large numbers of nodes.

This document describes the Ad hoc Multicast Routing protocol (AMRoute), which enables the use of IP Multicast in MANETs. Existing multicast protocols [1,5,10] do not work well in MANETs as the frequent tree reorganization can cause excessive signaling overhead and frequent loss of datagrams. The tree reorganization in MANETs is more frequent as compared to conventional static networks, since the multicast protocols have to respond to network dynamics in addition to group dynamics. AMRoute solves this problem by tracking group dynamics only; the underlying unicast routing protocol is relied upon for tracking network dynamics, which it is required to do anyway. AMRoute emphasizes robustness even with rapidly changing membership or highly dynamic networks; it does not attempt to provide the absolute minimum bandwidth or latency guarantees in a given topology. The two key features of AMRoute that make it robust and efficient in MANETs are:

- user-multicast trees, where replication and forwarding is only performed by group members over unicast tunnels,
- dynamic migration of core node according to group membership and network connectivity.

The user-multicast tree includes only the group senders and receivers as its nodes. Each node is aware of its tree neighbors only, and forwards data on the tree links to its neighbors. Multicast state is maintained by the group nodes only, and is not required by other network nodes. In fact, AMRoute does not even require non-member nodes to support any IP multicast protocol. The elimination of state in other network nodes clearly saves node resources, especially when compared with broadcast-and-prune native multicast protocols that require per source and per group state at all network nodes. More importantly, especially in highly dynamic ad hoc networks, user-multicast trees also eliminate the need to change the tree as the network changes. Neighboring tree nodes are inter-connected by IP-in-IP tunnels, similar to the approach adopted for connecting multicast routers on the MBONE. Consequently, assuming unicast connectivity is maintained among member nodes, the AMRoute distribution tree will continue to function despite network changes.

Each group in the network has at least one logical core that is responsible for discovering new group members and creating/maintaining the multicast tree for data distribution. This logical core is significantly different from the core in CBT and the RP in PIM-SM as it is not the central point on the data path (only for signaling), it is not a preset node (chosen from among currently known members) and it can change dynamically. The absence of a single point of failure is an important requirement for MANETs.

AMRoute includes some of the best features of other multicast protocols. Like CBT [1] and PIM-SM [5] it uses shared trees, so only one tree is required per group, thus improving its scalability. Like DVMRP [10], CBT and PIM the protocol is independent from specific semantics of the underlying unicast routing protocol. Although some efficiency gains are possible by tying the protocol closely with a unicast protocol, we see the independence as more important. Its independence allows use of the optimal ad hoc unicast protocol for the network and can work transparently across domains supporting different unicast protocols. Like DVMRP and PIM-DM, it does not rely on core nodes in the data path. Like DVMRP and PIM-DM, it provides robustness by periodic flooding for tree construction. However, AMRoute periodically floods a small signaling message instead of data.

Simulation results (using *ns*-2) indicate that as long as the load on the network is not significantly high, AMRoute offers a very good data delivery ratio. This does not change even if the refresh rate is reduced to a minimal level. The broadcast and unicast signaling load is comparable favorably with that imposed by the ad hoc unicast routing protocols.

This paper is organized as follows. Section 2 explicitly states the assumptions made during AMRoute design. A conceptual explanation of the AMRoute design is presented in section 3, followed by a description of the operation in section 4. Experimental results using an *ns*-2-based simulation are presented in section 5. Other approaches for multicast routing in MANETs are discussed in section 6. A discussion and plans for further work are presented in section 7.

2. Assumptions (or lack thereof)

AMRoute assumes the existence of an underlying unicast routing protocol that can be utilized for unicast IP communication between neighboring tree nodes. However, no assumptions are made about the syntax or semantics of the unicast protocol, and the same unicast protocol is not required to be used in all domains. The actual paths followed by the two directions of a unicast tunnel connecting neighboring group members may be different. It is not required that all network nodes support AMRoute or any other IP Multicast protocol. Non-multicast routers only need to support unicast, and forward the control packets (without any AMRoute protocol processing) if the IP-level TTL is non-zero (and decrement the TTL field before forwarding).

Both multicast receivers and senders are required to explicitly join the multicast tree and perform data forwarding,

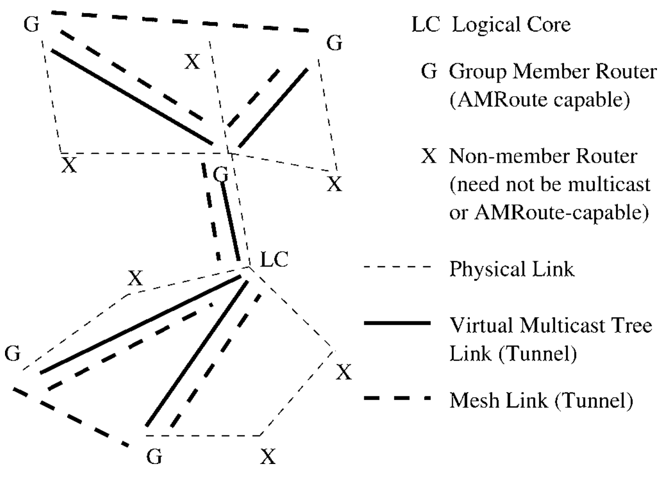
G
G
X
X
G
X
LC
X
G
X
G
X
LC Logical Core
G Group Member Router (AMRoute capable)
X Non-member Router (need not be multicast or AMRoute-capable)
Physical Link
Virtual Multicast Tree Link (Tunnel)
Mesh Link (Tunnel)

Figure 1. A virtual user-multicast tree.

which is a slight departure from the classical IP Multicast model. All group members[1] must be capable of processing IP-in-IP encapsulation. No group members need have more than one interface or act as unicast routers (we can build a tree entirely from host computers). However, at least one member (ideally all) must be capable of being AMRouter: forwarding and replicating datagrams to other members. AMRoute routers must be able to set the TTL field in the IP headers (decrementing the inner IP TTL field before a datagram is repackaged and forwarded).

3. Concepts

AMRoute creates a per group multicast distribution tree using unicast tunnels connecting group members. The protocol has two main components: mesh creation and tree creation. Only the logical core node initiates mesh and tree creation; however, the core can migrate dynamically according to group membership and network connectivity. Bidirectional tunnels are created between pairs of group members that are *close* together, thus forming a mesh. Using a subset of the available mesh links, the protocol periodically creates a multicast distribution tree.

3.1. User-multicast distribution trees

Figure 1 shows a user-multicast tree connecting six members. The group members forward and replicate multicast traffic along the branches of the virtual tree. The datagram that is sent between logically neighboring nodes is physically sent on a unicast tunnel through potentially many intermediate routers. The path taken by the unicast tunnel can change without affecting the user-multicast tree.

There are two key advantages to implementing our multicast protocol using only members:

- Provided there remains a path among all nodes connected by mesh branches, the user-multicast tree need not change

[1] Group members include receivers and senders.

because of network changes (e.g., because of node movement). This independence improves robustness and reduces signaling overhead.

- Non-members are not required to perform packet replication or even support any IP multicast protocol. This places the processing and storage overhead needed to support the multicast tree only on members.

The penalty paid for using a user-multicast tree is reduced efficiency. Clearly, bandwidth efficiency is reduced, as non-member routers are not allowed to perform packet replication. Furthermore, the delay is often increased. However, the difference between an optimal tree using all network nodes and an optimal tree using only member nodes is theoretically bounded.

In a stable network, the worst case bandwidth overhead for forwarding on a user multicast tree is less than twice as that for an optimal tree using all network nodes. This worst case overhead occurs in case of a simple star network with a non-member central node directly connecting n member nodes: a tree using the central node will take n hops, while a user multicast tree will take $2(n-1)$ hops. The worst case delay overhead depends on the time it takes to do packet replication. Assuming all nodes can only send one packet at a time, then the worst case delay overhead is again twice as much as the other case. (However, if a router can send on multiple interfaces simultaneously, then the worst case delay overhead will be higher.)

In a dynamic network, the overhead becomes more complex to calculate. The bandwidth overhead for using an optimal tree must include the signaling overhead of maintaining the tree. On the other hand, a user multicast tree has the overhead of the tree becoming less optimal as nodes move. The balance between these overheads depends on the mobility pattern.

Just as there are many ways to create native multicast trees, there are many ways to create user-multicast trees. We now describe a way that is well suited to ad hoc networks.

3.2. Logical core and non-core members

In the AMRoute protocol each group has at least one logical core[2] that is responsible for initiating signaling actions, specifically: (a) mesh joins (discovering new group members and disjoint mesh segments and (b) multicast tree creation. A non-core node cannot initiate these two operations, and can act only as a passive responding agent. Limiting the number of nodes that perform these two functions (ultimately to a single logical core) ensures that AMRoute can scale, without causing excessive signaling or processing overhead. Non-core nodes may need to respond to signaling messages initiated by the core node, thus making signaling traffic proportional to group size. But this is much better than permitting non-core nodes to initiate signaling messages, which would result in significantly larger signaling traffic proportional to square of the group size (each group member sends and receives signaling messages from every other member).

[2] The term *logical core* and *core* are used interchangeably for AMRoute, and imply the same entity.

The AMRoute logical core node is different from a CBT core and a PIM-SM rendezvous point in several fundamental aspects. In particular, it avoids robustness problems while permitting scalability since it:

- is not a central point for all data. Forwarding can continue on working branches of the tree regardless of the status of the logical core and links to the logical core;
- is not a preset node. Each multicast tree segment designates one of its nodes to be the core based on the *core resolution algorithm*;
- changes dynamically. The core node migrates according to group membership and network connectivity.

An AMRoute segment can temporarily have more than one core node for a group after new nodes join or disjoint segments merge together. A network node designates itself as a core when first joining a group. As a logical core a node can quickly discover new group members and join the mesh and tree with its closest neighbors, not just to the existing core. When multiple core nodes exist in a segment, they will advertise their presence by sending out tree creation messages. Core nodes use the reception of tree creation messages from other cores to decide whether to remain as a core. The core resolution algorithm is run in a distributed fashion by all nodes. The goal of the algorithm is to ensure that any group segment has exactly one core node and that the core node migrates to maintain a robust and efficient multicast tree.

An AMRoute segment can also have no core nodes because the core node disappears (e.g., leaves the group) or an existing segment is split into multiple disjoint segments (e.g., because of link or node failure). If a segment does not have a core node, one of the nodes will designate itself as the core node at some random time, on not receiving any tree creation messages.

A key issue with any algorithm that assigns a single core node is that it can centralize the multicast tree and indeed the mesh links on itself. AMRoute prevents centralization in the following ways:

- A non-core node is not allowed to graft to its own logical core. Without this limitation, all group members would ultimately be connected to the core.
- All nodes, including the core, are only allowed to have a limited number of tree links. If the limit is reached the node must drop the link furthest (at highest cost) from its current location.
- A logical core will take responsibility as core for a limited time or until some event makes changing the core desirable. A new logical core can be picked; for example when the core's mesh connectivity limit is reached.

Clearly the core resolution and change algorithms are key to the robustness and performance of the AMRoute protocol. However, it is also desirable to contain the complexity of the algorithms. Simulations are hence being undertaken to determine the tradeoffs between simplicity, robustness and efficiency.

4. Operation

We now discuss the operation of AMRoute, with an emphasis on explaining the design choices.

4.1. AMRoute messages

AMRoute uses five control messages for signaling purposes and one data message format. These are sent directly over IP. JOIN_REQ is the only message broadcast using an expanding ring search, while the other control messages are unicast between group members. The purpose of the control messages can be summarized as follows:

- JOIN_REQ. This is broadcast periodically by a logical core for detecting other group segments.
- JOIN_ACK. This is generated by a tree node in response to receiving a JOIN_REQ from a different logical core.
- JOIN_NAK. This is generated by a tree node if its application leaves the group.
- TREE_CREATE. This is generated periodically by a logical core to refresh the group spanning tree.
- TREE_CREATE_NAK. This is generated by a tree node to convert a tree link into a mesh link.
- DATA_MESSAGE. This is generated by a sender on receiving data from a multicast application.

A group member uses a monotonically increasing sequence number for messages generated for each group. These are used by its tree neighbors for ordering of received messages, so that older messages are not processed after newer ones have been received, and for dropping duplicate messages. The sequence number wrap-around problem is alleviated by using a large field (32 bits), which translates to about 2^{31} messages before wrap-around occurs. A worst-case signaling message rate of 0.4 messages per second per group member (from section 5) would result in a significantly large period of $2^{31}/0.4$ s before the sequence number wrapped-around. Considering state-of-the-art wireless link bandwidth of 10 Mbps, which translates to 2441 packets per second (512 byte packet size), wrap-around for data messages will not occur for 879 755 s (244 hours). A policy decision can be made restricting an ad hoc network multicast group validity to a time period less than 244 hours.

4.2. Mesh creation

An AMRoute mesh is a graph where each node is a member (sender or receiver) of the group and every link is a bidirectional unicast tunnel (figure 1). While the mesh establishes connectivity across all the members of a group, a tree is needed for forwarding the data. We use a two step process of creating a mesh before the tree because the mesh is simpler and more robust.

It is much simpler to maintain a mesh (that could potentially have loops) than a tree (with no loops) at every stage of member mutual discovery phase. For example, a very naive merging algorithm could result in a loop when three disjoint trees discover each other. In addition, the redundant mesh links contribute towards increased robustness in the case of node/link failures. (Note that the use of unicast tunnels between neighboring nodes of the mesh itself contributes towards robustness in the face of intermediate node/link failures along routes between them as the unicast protocol is expected to establish a separate route around the failed network node/link.)

4.3. Joining and leaving a group

To create a mesh encompassing all the members (senders and receivers) of a specific group, mechanisms are needed to allow members to discover each other. The expanding ring search mechanism based on TTL-limited broadcasts is adopted. All core nodes periodically broadcast JOIN_REQ messages.

Each member begins by identifying itself as the core of a 1-node mesh consisting of only itself. The core node sends JOIN_REQ packets with increasing TTL to discover other members of the group. When a member (core or non-core) node receives a JOIN_REQ from a core for a different mesh for the same group, the node responds back with a JOIN_ACK. A new bidirectional tunnel is established between the core and the responding node of the other mesh. Due to mesh mergers, a mesh will have multiple cores. One of the cores will emerge as the "winning" core of the unified mesh due to the core resolution algorithm.

If a node leaves a group, it sends out a single JOIN_NAK message to its neighboring nodes. If it subsequently receives any data or signaling message for that group, it can send out further JOIN_NAK messages.

4.4. Becoming a passive non-core node

Only logical core nodes initiate the discovery of other disjoint meshes, while both core and non-core nodes respond to the discovery messages. It is simpler and more scalable (in bandwidth usage) to have only the core node initiate discovery as compared to every node of the mesh. However, to avoid the situation where every merger adds a link to a core (which might result in too many links from the core), non-core nodes can participate in the mergers by responding to discovery messages received from other cores.

4.5. Tree creation

This section discusses the creation of a tree for data forwarding purposes once a mesh has been established. The core is responsible for initiating the tree creation process. From the point of view of individual nodes of the mesh, this phase involves identifying the subset of links incident on it that belong to the tree.

The core sends out periodic TREE_CREATE messages along all the links incident on it in the mesh. (Note that TREE_CREATE messages are sent along the unicast tunnels in the mesh and are processed by group members only, while JOIN_REQ messages are broadcast messages that are processed by all network nodes.) The periodicity of the TREE_CREATE messages depends on the size of the mesh and on the mobility of the nodes of the mesh. As the mesh nodes are mobile, the number of hops between neighbors keeps changing dynamically. So newer and more optimal trees might be created when TREE_CREATE messages are sent out. Group members receiving non-duplicate TREE_CREATEs forward it on all mesh links except the incoming, and mark the incoming and outgoing links as tree links. If a node has a collection of neighbors all 1-hop away on the same broadcast capable interface, then a possible optimization would be for the node to send a single broadcast message to all 1-hop neighbors simultaneously.

The simplistic approach of forwarding incoming TREE_CREATE on all outgoing mesh links can result in significant signaling overhead due to TREE_CREATEs and consequent TREE_CREATE_NAKs. One solution would be to restrict the number of tree neighbors that a node could have. However, simply picking the first *n* mesh links is not sufficient. This is because a new member will not be able to attach itself to an existing group member if the existing member had already reached its limit on the number of tree neighbors. The new member would need to continue broadcasting the JOIN_REQ until it attached to an existing member that did not have sufficient neighbors. It is thus necessary to permit a member to chose *deserving* members as tree neighbors over others. We use the number of TREE_CREATE_NAKs received from group members as the criteria for picking the *n* tree neighbors for a node. Members are considered more deserving if they have sent fewer TREE_CREATE_NAKs. Thus a new member would be most deserving compared to existing neighbors, and would get the TREE_CREATE from the group member. This approach also reduces the possibility of loops by restricting the tree formation.

4.6. Purging of tree links

If a link is not going to be used as part of the tree, the TREE_CREATE message is discarded and a TREE_CREATE_NAK is sent back along the incoming links. On receiving a TREE_CREATE_NAK, a group member marks the incoming link as a mesh link and not a tree link. Thus, each non-core node considers the link along which a non-duplicate TREE_CREATE message was received and every other link along which no TREE_CREATE_NAK message was received to be part of the tree for a specific group. (Core considers every link incident on it to be part of the tree.) Note that all these tree links are bidirectional tunnels.

The choice of using ACK or NAK in response to the TREE_CREATE messages is dictated by whether robustness or saving bandwidth is more important. If an ACK-based (positive acknowledgment) scheme is used, then data may not be delivered along links where ACKs were lost. This results in loss of data, but no waste of bandwidth. However, if a NAK (negative acknowledgment) based scheme is used, loss of NAKs can only result in same data being forwarded more than once (which is discarded by the downstream node on reception).

When data arrives at a group member along one of the tree links, it is forwarded along all other tree links. However, if it arrives along a non-tree link, a TREE_CREATE_NAK message is sent back along that link and the data is discarded.

4.7. Transient loops

The tree created by the *n*th TREE_CREATE message might not be the same as the one created by $(n-1)$th message. A situation may exist where some nodes are forwarding data according to the older tree and some according to the newer tree, which may result in loops or data loss. Such a transient phase is to be expected due to the dynamic nature of ad hoc networks.

We reduce the effect of loops by using sequence numbers in the data packets. These sequence numbers are on a per multicast group and per sender basis, which implies that a group member can determine whether a data packet is a duplicate and hence to be dropped. We reduce the processing and state required for implementing such a solution at the group members by requiring them to keep track of only the last data sequence number (per sender per group). This tradeoff is necessary to reduce the effect of loops at the cost of losing some out-of-sequence data packets. As shall be seen in the simulation results, this mechanism does not cause significant data packet loss. In addition, group members do not forward packets that have a TTL value of one.

4.8. Effect of node failures and group leaves

Nodes leaving a group or node failures are only partially handled by the redundant links in the mesh. In some situation, node failures might result in splitting the mesh into multiple disjoint meshes, where only one of these meshes has the core. Each node in the mesh expects to periodically receive TREE_CREATE messages. In case this message is not received within a specific period, the node designates itself to be the core after a random time. The node whose timer expires the earliest succeeds in becoming the core and initiates the processes of discovering other disjoint meshes as well as tree creation. Multiple cores that may arise in this case are resolved by the core resolution procedure.

4.9. Core resolution

In case of mesh mergers, there may be multiple active cores in the new mesh. Nodes in the mesh become aware of this situation when they receive TREE_CREATE messages from multiple cores. The nodes execute a deterministic core resolution algorithm to decide on a unique core for the mesh. They forward TREE_CREATE messages arriving from the unique

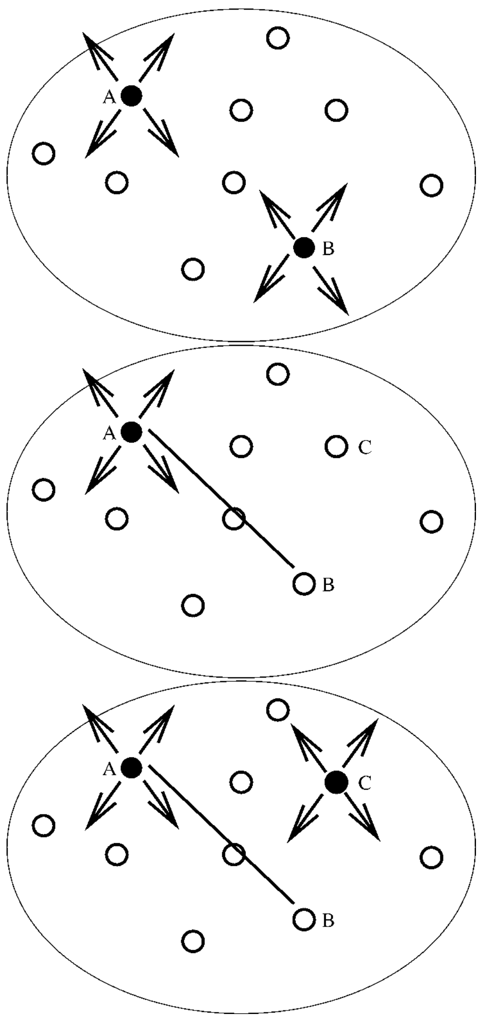

Figure 2. Formation of mesh and tree (1).

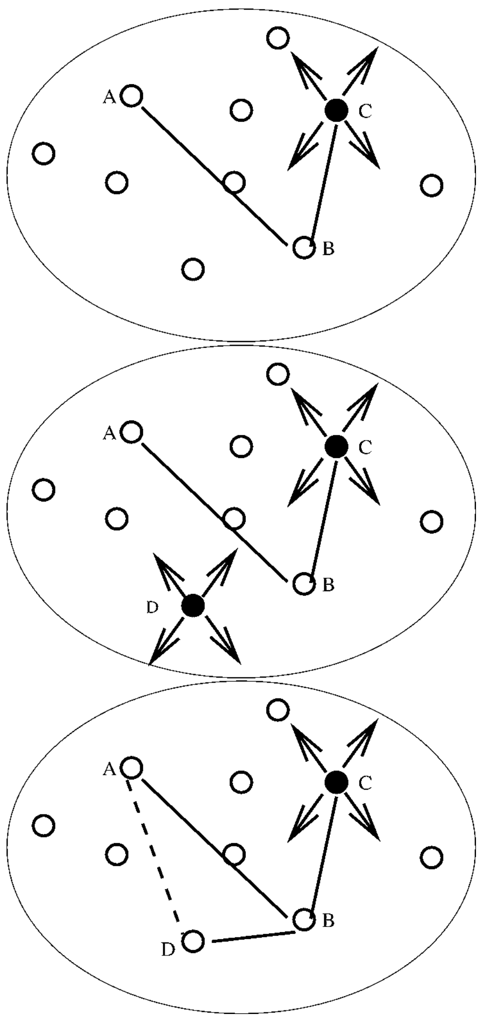

Figure 3. Formation of mesh and tree (2).

core and discard TREE_CREATE messages from other cores. As the multiple cores in the mesh will also become aware of the existence of other cores, they will also execute the same core resolution algorithm. All the cores except the "winning" core will demote themselves to non-core state. The simple core resolution algorithm that we used picked the winning core to be the one with the highest IP address.

4.10. Picking which branch to use for the tree

We adopt the simplest approach for picking a mesh-branch as a tree-link by simply accepting the first TREE_CREATE message that is received, and discarding any duplicate TREE_ CREATE messages using the sequence number included in each TREE_CREATE message. This results is a reasonable tree, but it is not necessarily the most bandwidth efficient (e.g., using minimum number of total hops) or lowest latency.

4.11. Pictorial representation

Figures 2 and 3 demonstrate the formation of an AMRoute tree in an ad hoc network. Nodes A and B simultaneously join a group, elect themselves as logical cores and start transmitting JOIN_REQ with expanding TTL. When either of them receives the other's JOIN_REQ, node B loses the core resolution procedure and is relegated to being a non-core node. There now exists a tree link (tunnel) connecting nodes A and B. Node C now joins the group, elects itself as a logical core, and starts transmitting JOIN_REQ with increasing TTL. Node B is closer to node C, and so will receive the JOIN_REQ from C before it reaches node A. A mesh link will be formed between B and C. The core resolution mechanism at B will determine that C is the winner. B will forward TREE_CREATEs from C to A. A will also determine that C wins, and relegate itself to non-core node. There now exists tree links from C to B and from B to A. Eventually JOIN_REQ from C will reach A, but since A is on the same mesh as B, it ignores this JOIN_REQ. This step is a tradeoff between reducing dynamic tree changes that can result in packet loss, and optimizing the tree structure. A new group member D can now join this mesh by transmitting JOIN_REQ, which are received at B. The core resolution at B results in C remaining the core, and D is grafted onto the tree at B. The JOIN_REQ from D may also have been received

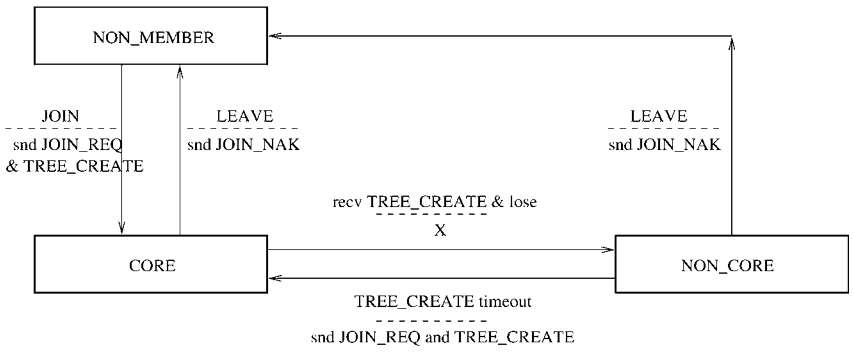

Figure 4. AMRoute state diagram.

by A, but D may receive the TREE_CREATE from B before getting it from A. So the mesh link between A and D does not get converted to a tree link.

4.12. State diagram

AMRoute's simplicity is illustrated by the state diagram in figure 4, which shows the three main AMRoute states and state transitions (with causing events and resulting actions). The states can be interpreted as follows:

- NON-MEMBER – a node does not belong to the multicast group.
- CORE – a node currently recognizes itself to be a logical core.
- NON-CORE – a node is currently a non-core member in the multicast group.

A node transitions from the NON_MEMBER state when an application on the node joins a group and transitions to it from all other states when the application leaves the group. A node transitions to the CORE state when an application joins a group, and by default sets itself to be a logical core. A logical core sends out periodic JOIN_REQ messages and TREE_CREATE messages. A logical core becomes a non-core node if it loses in the core resolution procedure that ensues when it receives a TREE_CREATE message from another core belonging to the same multicast group, which means the other core becomes the new core. A non-core member expects periodic TREE_CREATE messages from a logical core. If it does not receive one within the specified period, the associated timer expires and the node resets itself to be a core.

4.13. Timers

A logical core keeps two timers, namely the JOIN_REQ _SEND timer and the TREE_CREATE_SEND timer. The expiry of JOIN_REQ_SEND causes the node to compute the new TTL value to use for the expanding ring search, broadcast a new JOIN_REQ with this TTL value and restart the timer. To facilitate fast discovery of the group when a group member is not connected to any other group member, the JOIN_REQ_SEND timer has a significantly smaller value as compared to that when the core is connected to other group members. The TREE_CREATE_SEND timer is used to send out periodic TREE_CREATE messages.

A non-core member uses a TREE_CREATE_RECV timer. When it expires, the node waits for a random amount of time before it sets itself to be a core, and starts sending out JOIN_REQs and TREE_CREATEs. This period can be set to be random to reduce the possibility of multiple non-core nodes becoming cores simultaneously.

5. Simulation results

AMRoute was simulated using Network Simulator 2 (*ns*-2 [7]). The Dynamic Source Routing (DSR [2]) protocol was used as the underlying unicast layer. The simulations were carried out on a Pentium II 366 MHz machine running Redhat Linux 5.2. A 50 node ad hoc network was simulated, with one multicast group comprising of up to 20 members and up to 5 senders. Node mobility and wireless functionality (IEEE 802.11) were provided by CMU [3].

Node mobility in the CMU model is simulated using the "random waypoint" model. Each node begins the simulation by being stationary for *pause* seconds. It can then move towards a random destination in a 1500 m $\times$ 300 m grid at a speed of up to 20 m/s. On reaching the destination, it pauses for the same *pause* seconds before repeating the behavior. For each simulation run, the *pause* time for all nodes could be either of 0, 30, 60, 120, 300, 600 or 900 s, with each value denoting a different level of node mobility, 1 indicating least mobility (*pause* time of 900 s), through 7 indicating most mobility (*pause* time of 0 s). Values of other control variables for the simulation runs, unless otherwise noted, were: TREE_CRT_FLOODING_THRESHOLD = 3 (upper bound on number of tree neighbors that a group member can have), RING_EXPANSION_TTL = 3 (initial value of TTL for JOIN_REQ from core), INIT_JOIN_REQ_SEND = 3 s (initial value of JOIN_REQ timer at core), CORE_JOIN_REQ_ SEND = 10 s (value of timer at core after at least one other member has been discovered), TREE_CREATE_SEND = 10 s, TREE_CREATE_RECV = 20 s, size of data packets = 512 bytes.

Each simulation data point was obtained by using 10 different network scenarios, so as to reduce the effect of specific network configuration and node mobility patterns on the results. Thus the total number of simulation runs used for obtaining the results below were 840 (10 network scenarios $\times$ 7 *pause* times$\times$4 group sizes $\times$ 3 experiment sets). Each simulation run lasted for 900 simulation seconds. The parameters of interest were signaling overhead and data delivery ratio. The signaling overhead imposed by AMRoute in terms of total broadcast and unicast packets generated during the simulation run was measured. The broadcast signaling load includes the total number of JOIN_REQ packets, while the unicast signaling load includes the total number of TREE_CREATE, JOIN_ACK, JOIN_NAK and TREE_CREATE_NAK packets generated by all the nodes in the simulation. Since AMRoute is not dependent on a *specific* unicast protocol, the signaling generated by DSR is not considered in the measurements. The data delivery ratio is the ratio of total number of unique data packets received by all group members to the total number of unique data packets transmitted by the senders. Since we wanted to study the effect of node mobility and group size, and not of group dynamics, all group members had joined the group before data transmission was initiated by any sender. No group member left the group during the simulation.

5.1. *Effect of group size*

The first simulation was aimed at understanding the effect of group size on the results. The group size was varied from 5 through 20, in increments of 5. The number of senders (they were also group members) were 5, with each transmitting 1 data packet every 5 s. Figure 5 indicates that the data delivery ratio is very good in case of low to medium mobility, and is reasonable (above 91%) even in the case of continuous mobility. The effect on the data delivery ratio of dropping duplicate data packets using a simplistic implementation (section 4.7) thus seems to be minimal. The data delivery ratio also seems to be independent of group size. This is because a larger group size results in an improvement in tree connectivity, which leads to an improvement in data packets delivered to group members and negates any adverse effect of increased signaling traffic.

The unicast signaling load varies with the size of the group. This load is comparable to that imposed by the unicast ad hoc routing protocols in [3], and so can be considered reasonable. The slight decrease in unicast signaling with increase in mobility (for group size $\geqslant$10) can be attributed to weaker connectivity for the mesh, which translates to group members having fewer neighbors and hence exchanging fewer signaling messages. This effect is not evident for group size 5 since connectivity is sparse even in the absence of mobility. The broadcast signaling load is also dependent on the size of the group, and is within reasonable limits. However it increases slightly with mobility, especially for group size $\geqslant$10. This is because group members have to use more JOIN_REQs to discover other group members due to the increase in mobility.

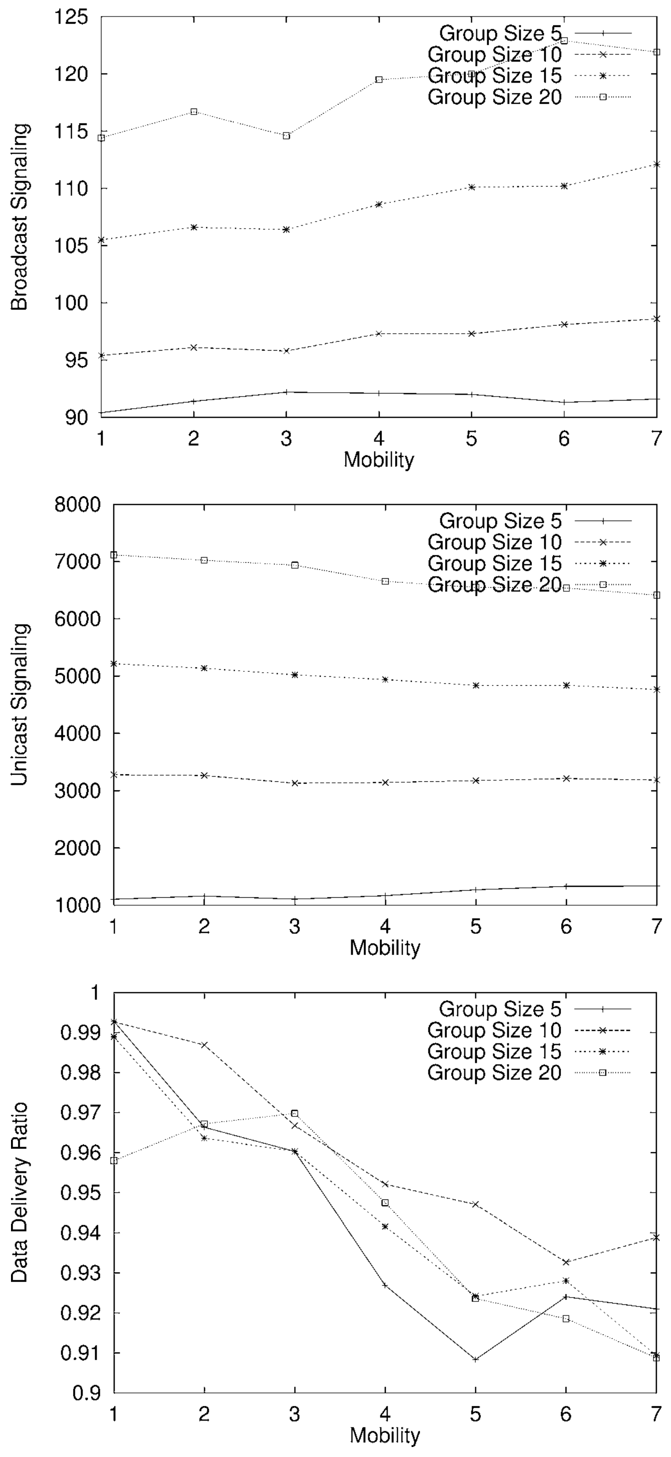

Figure 5. Effect of group size on signaling traffic and data delivery ratio.

5.2. *Effect of increased data rate*

The data transmission rate of each sender was increased to 1 packet per second. As is seen in figure 6, the data delivery ratio suffered in case of higher mobility for group sizes $>$10. This could be because the network could not support the

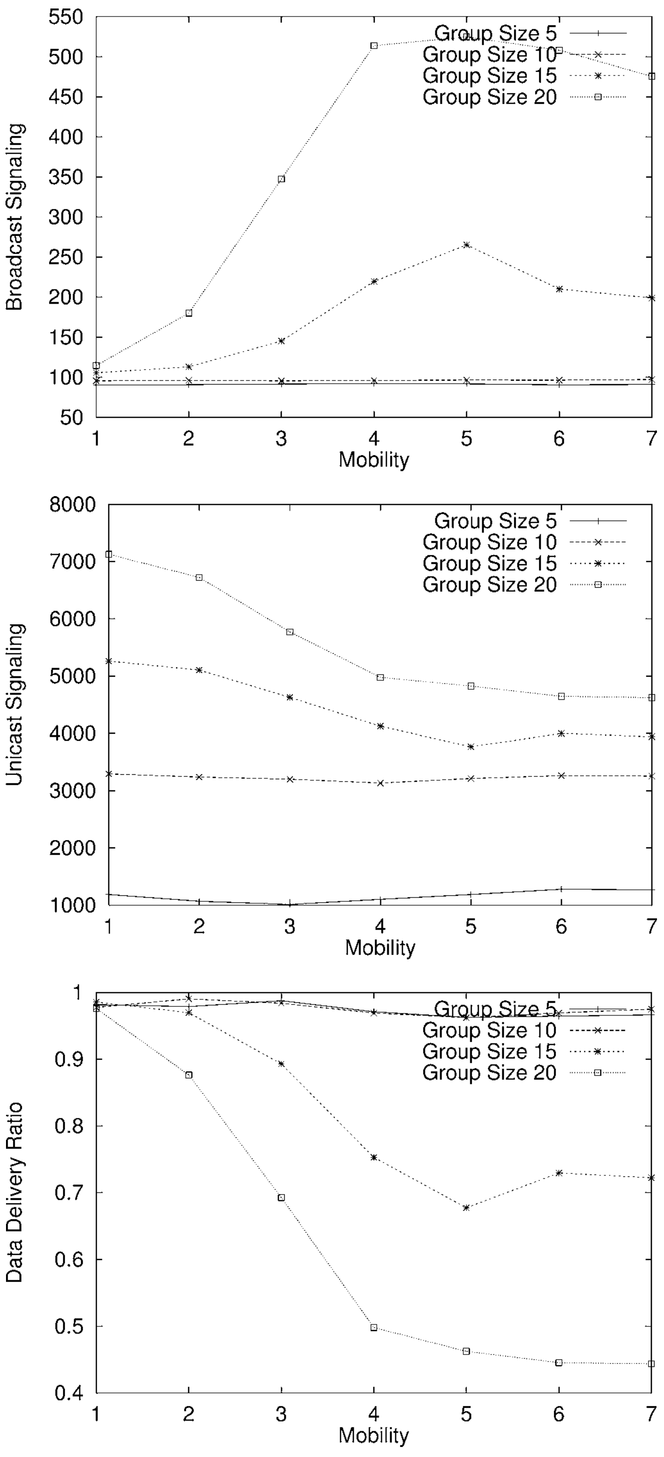

Figure 6. Effect of increased data rate on signaling traffic and data delivery ratio.

higher traffic load that is imposed with a larger group (more multicast data packets are replicated and circulated in the network).

The high packet drop rate is also reflected in the broadcast signaling load, which is higher for high mobility and group sizes >10. The signaling load is higher since JOIN_REQs have a higher loss probability, which implies group members require additional transmissions to get connected to the multicast mesh. The unicast signaling load is similar to the previous case in general, since it is timer-driven and, hence, not affected by network drop rate. However there is a noticeable drop in unicast signaling for high mobility and group sizes >10. This is due to the sparser mesh connectivity since more JOIN_REQs are dropped by the network and more time is required to connect all the nodes to the mesh.

5.3. Reducing the unicast signaling load

The unicast signaling load was further reduced by reducing the rate at which TREE_CREATEs are generated by the core (TREE_CREATE_SEND = 15 s, TREE_CREATE_RECV = 30 s; data rate was kept at 1 packet per 5 s as in section 5.1). Interestingly, this did not significantly affect the data delivery ratio (figure 7). This would lead one to believe that the virtual tree structure is usable even at low refresh and high mobility rates.

6. Related work

Since the introduction of AMRoute in August 1998, three other approaches have been proposed for multicast routing in MANETs.

The Ad hoc Multicast Routing protocol utilizing Increasing ID-numbers (AMRIS [11]) assigns an identifier to each node in a multicast session. A per-multicast session delivery tree rooted at a special node (by necessity a sender) in the session joins all the group members. The tree structure is maintained by assigning identifiers in increasing order from the tree root outward to the other group members. Election algorithms (not yet specified) may be required to choose one special node if multiple senders are present in a multicast session. All nodes in the network are required to support AMRIS and any node can be on a tree. All nodes are required to process the tree setup and maintenance messages that are transmitted by the root periodically. No global state is required to be maintained by nodes on the tree, and repairs to damaged links are performed locally. AMRIS currently assumes the existence of bidirectional links between network nodes. It also assumes that applications (multicast sessions) are long-lived, and hence, sacrifices route discovery latency to route recovery latency.

The On-Demand Multicast Routing Protocol [6] uses a mesh-based approach for data delivery, rather than the traditional tree-based approaches. It requires sources rather than receivers to initiate the mesh building by periodic flooding of control packets. Receivers are required to periodically exchange information with their neighbors locally. All nodes in the network are required to be involved in the protocol, with per group and per source state being maintained at intermediate nodes and group members. Additionally, nodes on the mesh seem to require maintenance of a cache to detect duplicate data and control messages that may arise due to the use of a mesh. The use of soft state implies that receivers/sources

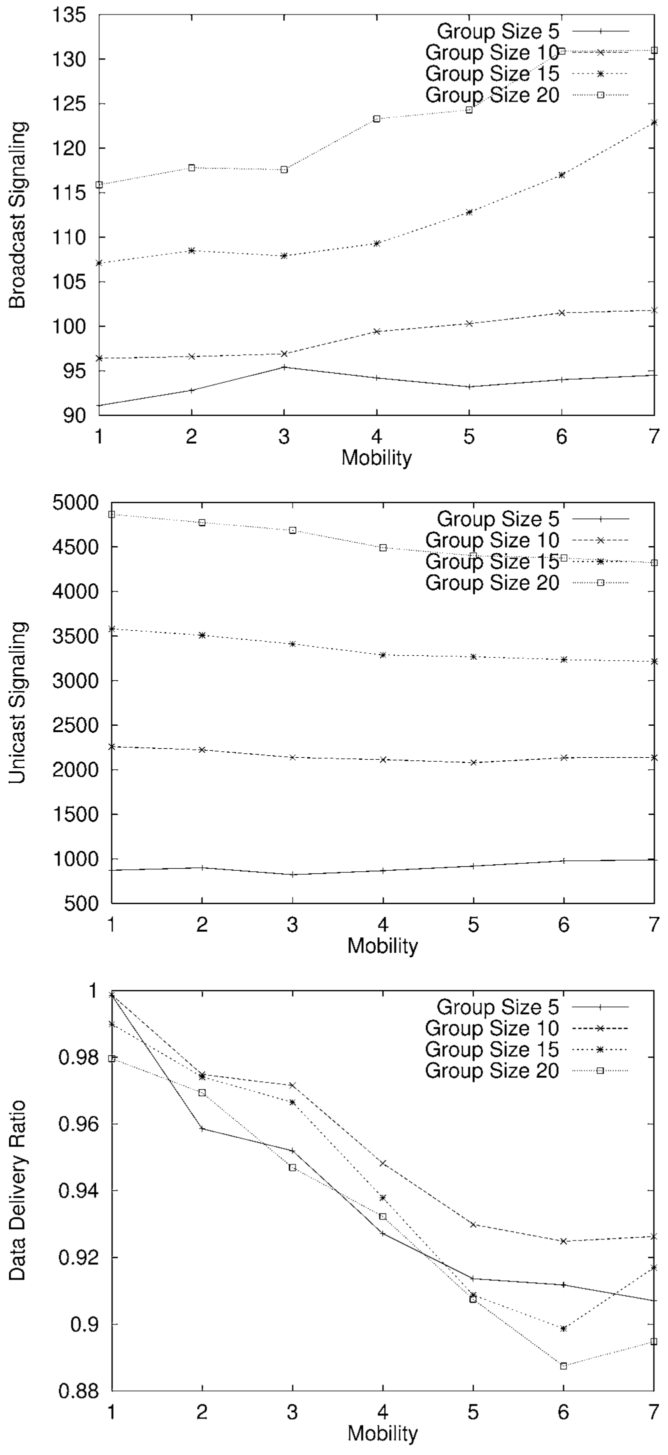

Figure 7. Reducing the unicast signaling load.

do not need to generate an explicit control message to leave the mesh.

The Ad hoc On Demand Distance Vector (AODV [9]) protocol was primarily designed for unicast, but has since been extended to support multicast. It uses the concept of a group leader, which is similar to a logical core, for managing the per-group, shared, bidirectional tree. Non-member nodes and non-senders can also be part of the multicast tree, and hence need to support AODV.

The Dynamic Source Routing (DSR [2]) protocol emulates multicast by controlled flooding of data packets, using the TTL field to limit the scope of the flooding. Individual nodes within the scope of the TTL value used are required to filter based on the multicast address specified in the destination field of the data packet. While this approach ensures that the multicast data reaches all nodes within the specified scope, both node and network resources are inefficiently utilized.

7. Discussion

The Ad hoc Multicast Routing protocol (AMRoute) presents a novel approach for robust IP Multicast in mobile ad hoc networks by exploiting user-multicast trees and dynamic logical cores. It creates a bidirectional, shared tree for data distribution using only the group senders and receivers as tree nodes. Unicast tunnels are used as tree links to connect neighbors on the *user-multicast tree*. Thus, AMRoute does not need to be supported by network nodes that are not interested/capable of multicast, and group state cost is incurred only by group senders and receivers. Also, the use of tunnels as tree links implies that tree structure does not need to change even in case of a dynamic network topology, which reduces the signaling traffic and packet loss. Thus, AMRoute does not need to track network dynamics; the underlying unicast protocol is solely responsible for this function. AMRoute does not require a *specific* unicast routing protocol; therefore, it can operate seamlessly over separate domains with different unicast protocols. Certain tree nodes are designated by AMRoute as *logical cores*, and are responsible for initiating and managing the signaling component of AMRoute, such as detection of group members and tree setup. Logical cores differ significantly from those in CBT and PIM-SM, since they are not a central point for data distribution and can migrate dynamically among member nodes.

Simulation results indicate that as long as the load on the network is not significantly high, AMRoute offers a very good data delivery ratio. This does not change even if the refresh rate is reduced to a minimal level. The broadcast and unicast signaling load is comparable favorably with that imposed by the ad hoc unicast routing protocols.

AMRoute performance is influenced by the characteristics of unicast routing protocol being used, which was DSR in this case. Further simulations are needed to explore any specific effects of using other unicast protocols (DSR, AODV). A performance comparison of AMRoute with PIM, DVMRP and the other ad hoc multicast routing approaches under the same simulation environment, mobility patterns and traffic load would be useful.

Acknowledgement

This article was prepared through collaborative participation in the Advanced Telecommunications & Information Distribution Research Program (ATIRP) Consortium sponsored by the US Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or US Government.

References

[1] T. Ballardie, Core based Trees (CBT) multicast routing architecture, RFC 2201 (September 1997).

[2] J. Broch, D. Johnson and D. Maltz, Dynamic source routing protocol for mobile ad hoc networks, Internet Draft, `draft-ietf-manet-dsr-02` (June 1999).

[3] J. Broch, D. Maltz, D. Johnson, Y. Hu and J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, In *Proceedings of ACM MobiCom* (October 1998).

[4] S. Corson and J. Macker, Mobile ad hoc networking: Routing protocol performance issues and evaluation considerations, RFC 2501 (January 1999).

[5] D. Estrin et al., PIM-SM: Protocol specification, RFC 2362 (June 1998).

[6] S.-J. Lee, W. Su and M. Gerla, On-Demand Multicast Routing Protocol for ad-hoc networks, Internet Draft, `draft-ietf-manet-odmrp-01` (June 1999).

[7] Network Simulator (Version 2), `http://www-mash.cs.berkeley.edu/ns/`

[8] C. Perkins, Mobile ad hoc networking terminology, Internet Draft, `draft-ietf-manet-term-00` (October 1997).

[9] C. Perkins, E. Royer and S. Das, Ad hoc On Demand Distance Vector (AODV) routing, Internet Draft, `draft-ietf-manet-aodv-03` (June 1999).

[10] T. Pusateri, Distance Vector Multicast Routing Protocol, Internet Draft, `draft-ietf-idmr-dvmrp-v3-09` (September 1999).

[11] C. Wu, Y. Tay and C.-K. Toh, Ad hoc Multicast Routing protocol utilizing Increasing ID-numbers (AMRIS) functional specification, Internet Draft, `draft-ietf-manet-amris-spec-00` (November 1998).

Jason Xie is a software engineer in Asera Inc. (a Silicon-Valley start-up). He participated in the AM-Route research at Telcordia when he was a graduate student in the Department of Computer Sciences at the University of Wisconsin-Madison. He completed his undergraduate degree from the Computer Science Department at the University of North Carolina at Chapel Hill, and subsequently worked for Bell Northern Research.
E-mail: jasonxie@cs.wisc.edu

Rajesh R. Talpade is a Senior Research Scientist at Telcordia Technologies. He is currently the Technical Program Manager for a CECOM-funded project on providing scalable QoS for IP networks. He is also involved with other research projects including mobile IP multicast (ARL-funded) and IP network provisioning software. Dr. Talpade joined Telcordia after completing a PhD in computer science from the Georgia Institute of Technology, Atlanta. His thesis was in the area of Internet Multicast, with focus on IP Multicast over ATM networks and Reliable Multicast. Rajesh is the primary author of an RFC on Multicast Servers Architectures (RFC 2149) within the Internetworking over NBMA working group of the IETF. He has published several papers based on his PhD and research at Telcordia Technologies.
E-mail: rrt@research.telcordia.com

Anthony McAuley received his PhD from Hull University, England, in 1985. He worked as a research fellow in Caltech in 1985–1987. Since 1987 he has been at Telcordia and is currently a Director in the Wireless IP Networking Research group. His current research projects include protocols for complete network autoconfiguration, self-managed virtual networks and architectures and protocols for future IPv4 and IPv6 wireless and home networking systems. He has built several mobile internetworking systems on Linux that included novel software he wrote for autoconfiguration, protocol boosters, multicast proxies, and a Transport Protocol. In the past he has also worked on IP multicasting in ad hoc and large-scale networks (including the Comprehensive Test Ban Treaty network), efficient error correction and detection codes, and design of VLSI chips (for everything from microprocessors to asynchronous packet switches).
E-mail: mcauley@research.telcordia.com

Mingyan Liu received the B.S. degree in electrical engineering from the Nanjing University of Aero. and Astro., Nanjing, China, in 1995, and the M.S. degree in systems engineering and Ph.D. degree in electrical engineering from the University of Maryland, College Park, in 1997 and 2000, respectively. She joined the University of Michigan, Ann Arbor, in September 2000, and is currently an Assistant Professor with the Department of Electrical Engineering and Computer Science. Her research interests are in the areas of performance modeling and analysis of hybrid communication networks, network dimensioning, wireless and mobile ad hoc networks, and communication protocol design and analysis.
E-mail: daphnel@isr.umd.edu