Efficient and Robust Multicast Routing in Mobile Ad Hoc Networks

Ravindra Vaishampayan
Department of Computer Science
University of California Santa Cruz
ravindra@cse.ucsc.edu

J.J. Garcia-Luna-Aceves
Department of Computer Engineering
University of California Santa Cruz
jj@cse.ucsc.edu

***Abstract*— This paper argues that tree based protocols can have packet delivery ratios comparable to mesh based protocols if the tree construction algorithm can fix and detect broken links quickly, and at the same time have a much lower data packet overhead due to the absence of redundancy.**

We present such a protocol and call it *robust multicasting in ad hoc networks using trees (ROMANT)*. ROMANT does not require a unicast routing protocol or the preassignment of cores to groups. We compare ROMANT with ODMRP and MAODV which are the state of the art in mesh based and tree based protocols respectively. The results from a wide range of scenarios of varying mobility, group members, number of senders, traffic load and number of multicast groups show that ROMANT attains a comparable or better packet delivery ratio than ODMRP and MAODV, and a much lower control overhead which is almost constant for a fixed number of groups, and varies sublinearly with increasing groups.

***Keywords*— Ad hoc networks, routing, multicasting, multicast mesh, multicast tree.**

I. Introduction

Mobile ad hoc networks have applications in a wide range of areas including disaster relief and military. Most of these scenarios need one to many or many to many communication. In fact, some networks may need multicast routing only and not need unicast routing at all. This makes multicasting a very important feature in such networks. As a result, it is important to have a multicasting protocol that provides a high packet delivery ratio even in extreme conditions (e.g., high mobility and high traffic load). It is equally important for such protocols to have a low overhead, because bandwidth and battery power are extremely precious in these kinds of networks.

Over the past few years, several multicast routing protocols have been proposed for ad hoc networks [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. For the purposes of our discussion, the approaches taken to date can be classified into tree-based and mesh-based approaches.

A tree-based multicast routing protocol establishes and maintains either a shared multicast routing tree or multiple source-based multicast routing trees (one for each group source) to deliver data packets from sources to receivers of a multicast group. Recent examples of tree-based multicast routing approaches are the multicast ad hoc on-demand distance vector protocol (MAODV) [4], and the adaptive demand-driven multicast routing protocol (ADMR) [8]. In contrast, a mesh-based multicast routing protocol maintains a mesh consisting of a connected component of the network containing all the receivers of a group. Two well-known examples of mesh-based multicast routing protocols are the core assisted mesh protocol (CAMP) [1] and the on-demand multicast routing protocol (ODMRP) [2].

MAODV maintains a shared tree for each multicast group, consisting of only receivers and relays. Sources wishing to send to the group acquire routes to the group on demand in a way similar to the ad hoc on demand distance vector (AODV) [17] protocol. Each multicast tree has a group leader, which is the first node to join the group in the connected component. The group leader in each connected component periodically transmits a group hello packet to become aware of reconnections. Receivers join the shared tree with a special route request. The route replies coming from different multicast tree members specify the number of hops to the nearest tree member. The node wishing to join the tree joins through the node reporting the freshest route with the minimum hop count to the tree.

ADMR maintains source-based trees, i.e., a multicast tree for each source of a multicast group. A new receiver performs a network-wide flood of a multicast solicitation packet when it needs to join a multicast tree. Each group source replies to the solicitation, and the receiver sends a receiver join packet to each source answering its solicitation. An individual source-based tree is maintained by periodic keep-alive packets from the source, which allow routers to detect link breaks in the tree by the absence of data or keep-alive packets. A new source of a multicast group also sends a network-wide flood to allow existing group receivers to send receiver joins to the source. MZR [15] like ADMR, maintains source based trees. MZR performs zonal routing; hence, the flooding of control packets is less expensive. Compared to approaches based on shared trees, the use of source-based trees creates much more state at routers participating in many groups, each with multiple sources.

ODMRP requires control packets originating at each source of a multicast group to be flooded throughout the ad hoc network. The control packet floods help repair the link breaks that occur between floods. The limitations of ODMRP are the need for network-wide packet floods and requiring that the sources of multicast packets for a group be part of the group's multicast mesh, even if such sources are not interested in receiving multicast packets sent to the group. DCMP [14] is an extension to ODMRP in that it designates certain senders as cores and reduces the number of senders performing flooding. NSMP [16] is another extension to ODMRP aiming to restrict the flood of control packets to a subset of the entire network. However, both DCMP and NSMP do not entirely eliminate ODMRP's drawback of multiple control packet floods per group.

CAMP avoids the need for network-wide floods from each source to maintain multicast meshes by using one or more cores per multicast group. A receiver-initiated approach is used for receivers to join a multicast group by sending unicast join requests towards a core of the desired group. The drawbacks of CAMP is that it needs the pre-assignment of cores to groups and a unicast routing protocol to maintain routing information about the cores, and this may incur considerable overhead in a large ad hoc network.

ROMANT is based on a receiver-initiated group joining scheme that does not require an underlying unicast routing protocol to operate or the pre-assignment of cores to groups. The first receiver joining a group becomes the core of the group and starts transmitting core announcements periodically. An election takes place if more than one receivers join at the same time. Each such announcement specifies a sequence number, the address of the group, the address of the core, the sending node, and the distance to the core. Routers use the best core announcements they receive to send their own core announcements to their neighbors, and over time each router has one or multiple paths to the elected core of each known group in the ad hoc network. To join a multicast group, a receiver sends a join announcement to its next-hop towards the core of the group, which it learns from core announcements. Nodes receiving join announcements intended for them join the group and also send a join announcement periodically to their next-hops for the group core. In this way join announcements propagate from each receiver towards the core, establishing the various branches of the tree. Similarly, a multicast data packet for a group is forwarded from its source towards the core of the group, using next-hop information obtained in core announcements, and is flooded within the tree of the group as soon as it reaches the first tree member. As Section II-E shows, multicast data packets do not have to be encapsulated in unicast data packets to reach a mesh member from a given source outside the group.

Section II describes ROMANT in detail. Section III compares ROMANT to ODMRP [2] and MAODV [4]. Comparison with ODMRP serves to illustrate why the packet delivery ratio of ROMANT matches or exceeds that of ODMRP even though ROMANT has far less redundancy. As expected the data packet overhead of ROMANT is much lower than that of ODMRP as ROMANT is a tree based protocol. However even the control packet overhead for ROMANT is much lower. Comparison with MAODV explains why ROMANT is able to maintain a high packet delivery ratio in a wide range of simulation scenarios while MAODV is not, even though both are tree based protocols.

We did not compare ROMANT with CAMP, because our approach is intended to work without the need of any unicast routing protocol or predefined cores. We did not compare ROMANT with DCMP, MZR, ADMR and NSMP because the improvement of these approaches over ODMRP in terms of control packet overhead as described in [14], [15], [8] and [16] and is significantly lower than what we achieve in ROMANT. MZR also had a lower packet delivery ratio at high mobility. Our main objective of comparing ROMANT with MAODV was to illustrate some of the reasons as to why tree based protocols have not been able to match mesh based protocols in terms of packet delivery ratio, and how ROMANT corrects those problems.

Section IV presents our conclusions.

II. ROMANT Description

A. Overview

ROMANT supports the IP multicast service model of allowing any source to send multicast packets addressed to a given multicast group, without having to know the constituency of the group. Furthermore, sources need not join a multicast group in order to send data packets to the group.

Like CAMP and MAODV, ROMANT uses a receiver-initiated approach in which receivers join a multicast group using the address of a special node (core in CAMP or group leader in MAODV), without the need for network-wide flooding of control or data packets from all the sources of a group. Like MAODV, ROMANT eliminates the need for a unicast routing protocol and the pre-assignment of cores to multicast groups.

ROMANT implements a distributed algorithm to elect one of the receivers of a group as the core of the group, and to inform each router in the network of at least one next-hop to the elected core of each group. The election algorithm used in ROMANT is essentially the same as the spanning tree algorithm introduced by Perlman for internetworks of transparent bridges [18]. Within a finite time proportional to the time needed to reach the router farthest away from the eventual core of a group, each router has one or multiple paths to the elected core.

Every receiver connects to the elected core along *any one* shortest path between the receiver and the core. All nodes on such a shortest path between any receiver and the core collectively form the tree. A sender sends a data packet to the group also along *any one* shortest path between the sender and the core. When the data packet reaches a tree-member, it is flooded within the tree, and nodes maintain a packet ID cache to drop duplicate data packets.

ROMANT uses two kinds of control packets. The *core announcement* and the *join announcement*. Each core announcement specifies a sequence number, the address of the group (group ID), the address of the core (core ID), the distance to the core, and the sending router. Successive core announcements have a higher sequence number than previous core announcement announcements sent by the core. With the information contained in such announcements, nodes elect cores, and each node in the network learns of one or more routes to the core. A join announcement specifies the sender, the intended group (group ID), and the parent of the node sending the announcement, in the multicast tree. Join announcements help create and maintain the multicast tree.

B. Core Election

When a receiver joins the group, it checks to see if it has ever received a core announcement for that group. If so, then it does not participate in a core election and the current core of the group remains unchanged. On the other hand, if it has never received a core announcement for that particular group, then it considers itself the core of the group and starts transmitting *core announcement* packets periodically to its neighbors stating itself as the core of the group and a 0 distance to itself. Nodes propagate core announcements based on the best core announcements they receive from their neighbors. A core announcement with higher core ID is considered better than a core announcement with a lower core ID. Eventually, each connected component has only one core. If one receiver joins the group before other receivers, then it becomes the core of the group. If several receivers join the group concurrently, then the one with the highest ID becomes the core of the group.

A core election is also held if the network is partitioned. The election is held in the partition which does not have the old core. A node detects a partition if it does not receive a fresh core announcement for 3 x *core_announcement_interval*. Once a receiver detects a partition, it behaves in exactly the same way it would upon joining the group, and participates in the core election.

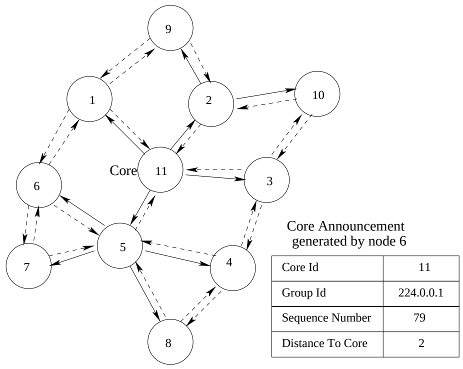
C. Connectivity Lists and Core Announcement Propagation

A node that believes itself to be the core of a group transmits core announcements periodically for that group. As the multicast announcement travels through the network, it establishes a *connectivity list* at every node in the network. Using connectivity lists, nodes are able to establish a mesh, and route data packets from senders to receivers.

A node stores all the core announcements it receives from its neighbors in the connectivity list. Fresher core announcements from a neighbor (i.e., one with a higher sequence number) overwrite entries with lower sequence numbers for the same group. Hence, for a given group, a node has only one entry in its connectivity list from a particular neighbor.

Each entry in the connectivity list, in addition to storing the core announcement, also stores the time when it was received, and the neighbor from which it was received. The node then generates its own core announcement based on the best entry in the connectivity list.

For the same core ID, core announcements with higher sequence number are considered better. For the same core ID and sequence number, core announcements with smaller distances to the core are considered better. When all those fields are the same, the multicast announcement that arrived earlier is considered better. After selecting the best core announcement, the node generates the fields of its own core announcement in the following way:



Core Announcement
generated by node 6

Core Id	11
Group Id	224.0.0.1
Sequence Number	79
Distance To Core	2

Connectivity List at node 6 : Core Id = 11 Group Id = 224.0.0.1

Neighbor	Core Announcement			Time (ms)
	Sequence Number	Distance To Core	Parent	
5	79	1	11	12152
1	79	1	11	12180
7	79	2	5	12260

Fig. 1. Dissemination of core announcements

- Core ID: The core ID in the best core announcement
- Group ID: The group ID in the best core announcement
- Sequence number: The sequence number in the best core announcement
- Distance to core: One plus the distance to core in the best core announcement

Connectivity lists store information about the one or more route that exist to the core. When a core change occurs for a particular group, then the node clears its old connectivity list and builds a new one, specific to the new core. Hence the group ID and core ID entries are same for all neighbors, and are not stored separately. Figure 1 illustrates the propagation of core announcements and the building of connectivity lists. The solid arrows indicate the neighbor from which a node receives its best core announcement. Node 6 has three entries in its connectivity list for neighbors 5, 1, and 7. However it chooses the entry it has received from 5 as the best entry, because it has the shortest distance to core and has been received earlier that the one from node 1. It uses this entry to generate its own core announcement. When a node node wants to send data packets to the group it forwards it to the node from which it received its best core announcement. If that link is broken then it tries its next best and so on. Hence each node in the network has one or more routes to the core. The core announcement sent by the core has distance to core set to zero.

Fresh core announcements are generated by the core every three seconds, after which they are disseminated throughout the network within a relatively short time. A node thus receives all core announcements with the latest sequence number within a short period of time from all its neighbors. After receiving a core announcement with a fresh sequence number, nodes wait for 100 ms to collect core announcements from multiple neighbors before generating their own core announcement. A node may send its core announcement before receiving the core announcements of some of its neighbors with the same or longer distance to core. e.g. in Figure 1 node 6 generates its core announcement at time = 12252 ms, which is 100 ms after it receives its first core announcement of sequence number 79. It receives a core announcement from node 7 later, at time = 12260 ms.

D. Tree Establishment and Maintenance

Initially only receivers consider themselves tree-members. In order to establish a connection to the core each receiver periodically transmits join announcements. i.e. once every three seconds. To generate a join announcement a node also accesses its connectivity list to obtain its best core announcement. A node generates the fields of the join announcement in the following way :

- Group ID: The ID of the group it wants to join
- Sender: Its own ID
- Parent : The node from which it received its best core announcement.

If a non-member receives a join announcement with its node ID in the parent field, it considers itself to be a tree member, and similarly generates periodic join announcements. Hence a join announcement sent by each receiver, triggers the generation of join announcements by all nodes lying on a shortest path between the receiver and the core. As all receivers establish a path to the core it follows that the resulting multicast tree connects all receivers together. This is illustrated in Figure 2. The tail of the arrow indicates the node sending the join announcement. The head of the arrow indicates the node in the parent field of the join announcement. If a node which is already a tree member receives a join announcement from a new node, it does not need to generate an additional join announcement as it already has established a path to the core. e.g. if node N3 decides to join the group it directs its join announcement towards node N4. This does not trigger the generation of join announcement in N4 as it is already a tree member. It only generates a join announcement once three seconds have elapsed since it generated its last one.

Every time a node receives a fresh batch of core announcements, the best entry in its connectivity list may change depending on the mobility of the network in the last three seconds. Thus, every time a node sends a join announcement, the parent field of the join announcement may vary depending on the latest “best entry” in the connectivity list. Thus every three seconds each branch of the tree is established afresh, and each receiver connects to the core along the “best path”. As a result, certain nodes which were at one time on the best path from a receiver to the core, may no longer be on it. Such nodes will no longer receive join announcements with their node ID in the parent field. If no such join announcement is received by the node for a period of 3 x join_announcement_interval, then the node no longer considers itself to be a tree member and stops generating join announcements.

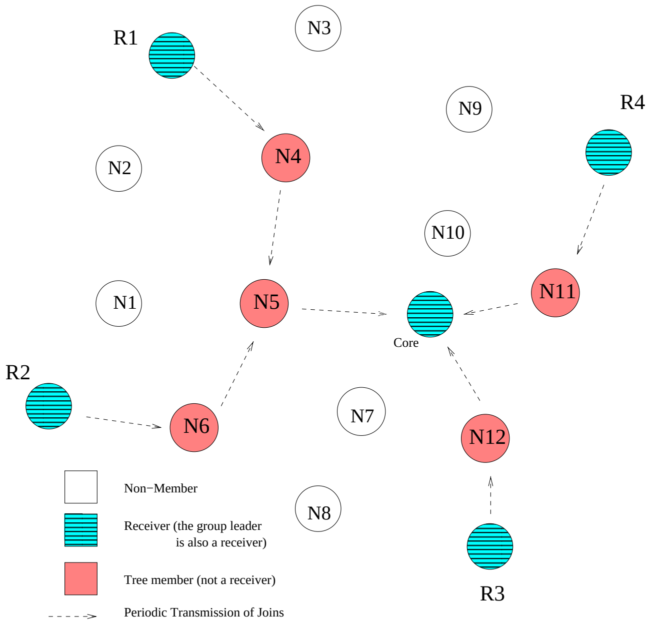

R1
N3
N9
R4
N2
N4
N10
N1
N5
N11
Core
R2
N7
N6
N12
Non−Member
Receiver (the group leader is also a receiver)
N8
Tree member (not a receiver)
R3
Periodic Transmission of Joins

Fig. 2. Tree formation in ROMANT

E. Data Packet Forwarding

Similar to how join announcements are send, a node sends a data packet to the node from which it received its best core announcement. A node learns the MAC address of its neighbors simply by examining the MAC headers of their core announcements.. When a node forwards a data packet, it sets the destination MAC address to the MAC address of the node from which it received its best core announcement. A non-member on receiving a data packet, drops the packet if the destination MAC address is not the same as its own MAC address. Otherwise it sets the MAC address to the address of the node from which it received its best core announcement. This process continues, until the data packet reaches a tree-member. From there, the packet is flooded throughout the tree, with a packet ID cache used to drop duplicate packets. Tree-members forward all data packets without looking at their MAC addresses. Unlike PIM sparse mode [19], where a multicast data packet is encapsulated inside a unicast packet till it reaches the rendezvous point, there is no need for encapsulation in ROMANT as the data packet moves from sender to the tree member.

F. Implicit Acknowledgments

The routing of data packets from senders to receivers is also used to detect broken links and update the connectivity list. Assume node X transmits a data packet to node Y, from whom it received its best core announcement.

Because all communication is broadcast, X also receives the data packet when it is forwarded by Y. This serves as an implicit acknowledgment of the packet transmission. If X does not receive an implicit acknowledgment within ACK_TIMEOUT, then it detects that the link X-Y is broken and removes Y from its connectivity list. The ability of nodes to detect broken links as an integral part of routing data packets, is another key contribution of ROMANT.

G. Multiple Groups

When multiple groups exist, nodes aggregate all the fresh core announcements they receive, and broadcast them every *core_announcement_interval*. However, core announcements representing groups being heard for the first time or resulting in a new core are forwarded immediately, without aggregation. This is to avoid delays in critical operations like core elections.

H. Recycling Sequence Numbers

Like other unicast or multicast routing protocols using sequence numbers, ROMANT needs to recycle sequence numbers and handle failures that cause a core to reset the sequence number assigned to a multicast group.

Because the sequence number of a core announcement is only increased by the core of the group, the same mechanisms used for the handling of sequence numbers in such link-state routing protocols as OSPF or in the spanning tree algorithm [18] suffice to ensure that nodes can trust the most recent core announcement. In particular, when a node recovers from a failure, it must apply a hold-down time long enough to ensure that no node in the MANET still considers the recovered node to be the core of any group.

III. Performance Comparison

We compared the performance of ROMANT against the performance of ODMRP [2] and MAODV [4] which are the state of the art mesh based and tree multicast routing protocols for ad hoc networks. ODMRP has been implemented in Qualnet as part of the Qualnet distribution. The MAODV code for Qualnet was obtained from a third party[1] who wrote the code independently of our effort following the MAODV IETF Specification [20]. We have used the same simulation parameters as [3], where the designers of ODMRP compare the performance of ODMRP with several other protocols. Figure 3 lists the details about the simulation environment.

We employed RTS/CTS when packets were directed to specific neighbors. All other transmissions used CSMA/CA. Each simulation was run for four different seed values. To have meaningful comparisons, all timer values (i.e., interval for sending JOIN requests and JOIN tables in ODMRP and the interval for sending core announcements and join announcements in ROMANT) were set to 3 seconds. We have also implemented and tested ROMANT in Linux 2.4.20-8, Red Hat Release 9, with the code having derived from our Qualnet implementation.

Simulator	Qualnet 3.5
Total Nodes	50
Simulation Time	900 seconds
Node Placement	Random
Pause Time	0
Mobility Model	Random Waypoint
Radio Range	250m
Channel Capacity	2 Mbps
MAC protocol	IEEE 802.11−1997
Data packet size	512 bytes

Fig. 3. Simulation Environment

Metric	Meaning
Packet Delivery Ratio	$\frac{\text{data packets delivered}}{\text{data packets expected}}$ *
Control Overhead	$\frac{\text{total control packets transmitted}}{\text{data packets delivered}}$
Total Overhead	$\frac{\text{total control packets transmitted + total data packets transmitted}}{\text{data packets delivered}}$

*data packets expected = data packets sent X number of receivers

Fig. 4. Metrics used for Performance Evaluation

The metrics used for our evaluation were **packet delivery ratio**, **control overhead** and **total overhead**, which are defined in Figure 4. **Total overhead** is a more important metric than **control overhead** because we are concerned about the number of packets transmitted to get a certain number of data packets to the receivers, regardless of whether those packets were data or control.

A. Simulation Scenarios

Several experiments were carried out to determine the effect of mobility, number of senders, number of members, traffic load and number of multicast groups on the performance metrics for each protocol. The details of each experiment performed are as follows:

- Experiment 1 : Mobility varied across {0, 5, 10, 15, 20} m/s. Senders = 5, Members = 20, Traffic Load = 10 pkts/sec, Multicast groups = 1.
- Experiment 2 : Senders varied across {1, 2, 5, 10, 20}. Mobility = 5 m/s, Members = 20, Traffic Load = 10 pkts/sec, Multicast groups = 1.
- Experiment 3 : Members varied across {5, 10, 20, 30, 40}. Mobility = 5 m/s, Senders = 5, Traffic Load = 10 pkts/sec, Multicast groups = 1.
- Experiment 4 : Traffic Load varied across {1, 2, 5, 10, 25, 50} pkts/sec. Mobility = 0, Senders = 5, Members = 20, Multicast groups = 1.
- Experiment 5 : Multicast Groups varied across {1, 2, 5, 10}. Mobility = 5 m/s, Senders = 5 per group, Members = 20 per group, Traffic Load = 20 pkts/sec.

[1] We thank Venkatesh Rajendran for providing the simulation code of MAODV.

Experiments 1, 2, 3, 4 and 5 determined the effect of mobility, number of senders, number of members, traffic load, and number of multicast groups respectively. For the traffic load test we set mobility to 0 because we wanted to focus of packet drops caused by congestion. Both the senders and members were chosen randomly from among the 50 nodes. Traffic load was equally distributed among all senders. For the multiple groups experiment, random allocation of nodes to groups could result in a single node being a member of multiple groups. Experiments 1, 2, 3 and 4 are the same that were carried out in [3] where the designers of ODMRP [2] compare its performance to CAMP [1], AMRIS [7] and AMROUTE [6]. Experiments 5 is an additional experiment that we have carried out which we believe is important in evaluating the effectiveness of a multicast protocol.

B. *Core Stability*

The stability of a core is important for the effective performance of ROMANT. Frequent core changes in addition to leading to control overhead, would also lead to a significant number of packet drops because the tree would always be in a state of reconstruction. This is avoided because ROMANT satisfies two properties: a) Core elections are not triggered if the partitions and reconnections are occurring rapidly b) Nodes do not detect a partition when one has not occurred.

The first condition is met because nodes detect a partition in ROMANT only if they fail to receive a core announcement from a core for three consecutive core_announcement_interval's i.e. 9 seconds. Hence if partitions and reconnections are frequently occurring then nodes will not detect a partition. Only when a node is partitioned from the core for a period of 9 seconds consecutively does it detect a partition, and participate in the election if it is a receiver.

Nodes may detect a partition when it has not occurred if they consistently don't receive core announcements from the core. Other multicasting protocols would face similar problems if important control information was consistently lost. Providing an analytical model to predict erroneous partition detection, based on the probability of successful packet delivery per link is beyond the scope of this paper. Please note that a node receives a core announcement along *all* paths connecting it to the core. It detects a false partition only when it is not able to receive even a single core announcement on any path, for three consecutive core_announcement_interval's. For each experiment we carried out we also measured the average number of core changes detected by each node during the course of the experiment. In addition to experiments described in Section III-A we carried out an experiment with the following parameter values: mobility = 0 m/s, senders = 5, members = 20, traffic load = 10 pkts/sec, and multicast groups = 1. We varied the terrain size from 800 m X 800 m to 12800 m X 12800 m. We wanted to detect the occurrence of false partitions in sparser networks where the average number of neighbors per node would be significantly lower. The idea was to distinguish the real core changes from the false core changes. The core changes detected when the nodes are mobile may be real or false. The core changes detected when nodes are not moving have to be false. Having zero core changes in situations of zero mobility is a strong indicator that false core changes do not occur. The results are shown in Table I. Table I indicates that the core changes are zero for all scenarios with no mobility. viz. mobility experiment for mobility = 0, traffic load experiment, and the terrain-size experiment with zero mobility. Even for scenarios with mobility core changes are relatively small for a 900 second simulation time.

mobility(m/s)	0	5	10	15	20
core changes	0	5.78	3.86	5.14	5.14

senders	1	2	5	10	20
core changes	4.5	4.5	5.79	4.5	4.5

members	5	10	20	30	40
core changes	0	1.29	5.79	5.79	10.93

terrain-size(m)	800 X 800	1600 X 1600	3200 X 3200	6400 X 6400	12800 X 12800
core changes	0	0	0	0	0

traffic-load (pkts/sec)	1	2	5	10	25	50
core changes	0	0	0	0	0	0

TABLE I

CORE CHANGES FOR DIFFERENT SCENARIOS

C. *ROMANT vs MAODV*

1) Broken routes: Data packets may be lost due to broken routes as a result of mobility, or due to collisions. Greater packet losses due to collisions would occur in a protocol with a higher overhead. Data packet loss due to broken routes is expected to be larger for a tree based protocol like ROMANT as it offers only a single route between senders and receivers as opposed to a mesh based protocol like ODMRP which offers multiple routes. In ROMANT, packets flow from senders to receivers in two steps as described in Section II-E. In the first step they are routed to the tree using connectivity list information and in the second step they are flooded within the tree. ROMANT is able to significantly restrict its losses due to broken routes, because in both steps it ensures that links do not remain broken for long.

In the first step, only one data packet is lost per broken link, as the link break is immediately detected if an implicit acknowledgment is not received as described in Section II-F. A new route to the tree does not have to be established, as subsequent data packets are simply directed towards the next best entry in the connectivity list. In contrast, in MAODV three hello packets sent once every three seconds have to be lost before a link break is detected. The broken link is then fixed by transmission of route requests and route replies. In this interval, a significant number of data packets could be lost depending on the rate of traffic generation.

In the second step, ROMANT does not detects link breaks in the multicast tree nor does it fixes them. It simply generates a brand new tree once every core_announcement_interval. i.e. three seconds. (Depending on the mobility of the nodes a large number of nodes than were present in the old tree may also be in the new tree). This global, proactive approach to tree maintenance as opposed to a local, reactive approach as in MAODV where individual link breaks are detected and fixed also results broken links existing for a shorter time. As a brand new tree is built every three seconds as described in Section II-D, a link may remain broken for a time between 0 and 3 seconds depending on when it is broken. In the MAODV approach on the other hand, with an allowed_hello_loss of two it means than only when three hello packets are lost is a link break detected. Hence it takes between 6 and 9 seconds for a broken link to be detected. Fixing a link break takes further time as it involves sending RREQ, RREP, and MACT packets. Probably the most important reason for the lower broken links in the case of proactive tree maintenance as in ROMANT is because the average link life for the chosen simulation parameters (radio range, mobility, and terrain size) is significantly more than 3 seconds. Thus the tree is rebuilt before its links have a chance to break.

2) Control Overhead: Although intuitively we feel that the proactive maintenance of the multicast tree in ROMANT would result in a higher overhead compared to an on-demand approach as in MAODV, that is not correct. The global, proactive approach to tree maintenance adopted by ROMANT is possible because of the periodic flood of core announcements by the core, which establishes connectivity lists at each node as described in Section II-D. Periodic flood of a control packet in core based protocols is needed anyway to detect partitions and reconnections. MAODV also has a similar packet which is flooded called the "group hello". MAODV *does not utilize* this flooding for tree maintenance. Hence the overhead incurred in fixing broken links i.e. RREQ, RREP and MACT packets is an additional overhead having no counterpart in ROMANT. In order to maintain tree connectivity only tree members in ROMANT transmit join announcements. However in MAODV *all* nodes transmit hello packets.

3) Analysis of Results: Based on simulation results shown in Figs. 5(a), 5(c), 5(d) and 8(b) we can see that the packet delivery ratio of MAODV is low in scenarios with high mobility, large numbers of members, high traffic loads or multiple multicast groups. We also note that the drop in the packet-delivery ratio is not gradual. When a certain threshold is crossed in terms of mobility, number of members, traffic load or multicast groups, we see from these figures that the packet-delivery ratio drops drastically. We call this threshold the "stress threshold". This is accompanied by a corresponding increase in packet overhead, as shown in Figs. 7(a), 7(c), 7(d) and 8(d)

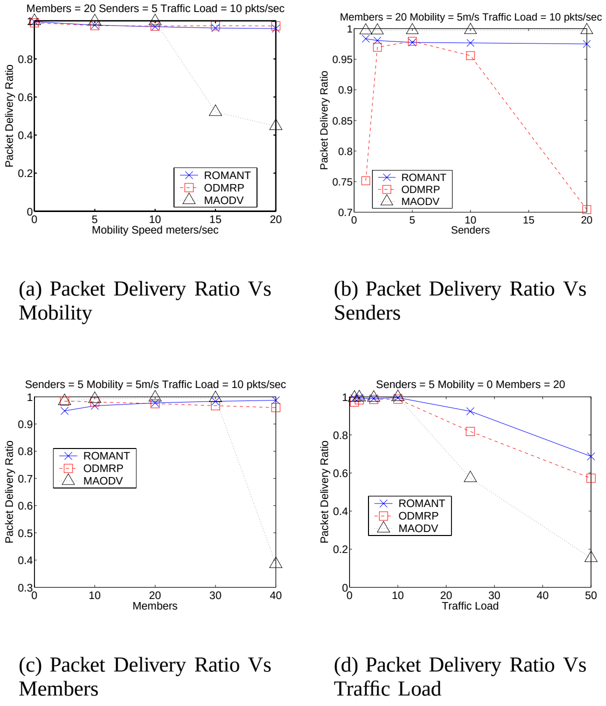
A careful analysis of the packets sent in all four scenarios show that a large number of RREQ (with Join flag set), RREP and MACT packets are sent. These are the packets associated with tree reconstruction. This indicates that the multicast tree is unstable and needs significant reconstruction activity. A multicast tree becomes unstable when the likelihood of links breaking increases. Links are assumed to break if neighbors do not hear each other's hello packets. The multicast tree can become unstable due to different reasons. In the case of high mobility, links actually break when nodes move in and out of each other's range. In the case of large numbers of members, the multicast tree is much larger. Assuming that a certain fraction of links are broken, a larger number of links means that a larger number of links are broken. In the case of higher traffic load, the links are not really broken; however, a larger number of packets are lost due to collisions. Hence, when hello packets are lost due to collisions, nodes infer erroneously that links have been broken. We call this phenomenon an "apparent link break". In case of multiple multicast groups multiple trees are maintained, one for each group. The tree maintenance packets of one tree interfere with another which also leads to apparent link breaks.

Our analysis leads us to believe that MAODV's response to fixing broken links is its greatest limitation. The fact that nodes believe that links are being broken indicates that the network is operating in stress mode, and MAODV responds by injecting three kinds of packets, i.e., RREQ, RREP and MACT packets. As a result, many RREQ packets may be flooded if a RREP packet is not received soon enough. The injection of these packets may in fact lead to more apparent link breaks due to the loss of more hello packets in collisions, which in turn leads to the injection of more RREQ, RREP and MACT packets, in an attempt to fix these new link breaks. As a result of this cyclic nature of congestion, there is sharp decrease in packet delivery ratio and a sharp increase in control overhead as the network crosses a certain "stress threshold." ROMANT on the other hand is less susceptible to link breakages because it proactively maintains the multicast tree as explained in section III-C.1. Even when a link breaks, a node does not need to inject control packets to rebuild it. It is able to lookup an alternate route using its connectivity list.

Another possible reason for the fast degradation of MAODV after a certain threshold value may be due to looping of multicast packets. Whether our previous conjectures or multicast looping are the reasons for MAODV's poor performance, it is clear that ROMANT offers a much better alternative.

D. ROMANT vs ODMRP

1) Comparison of the protocols: ROMANT is a tree based protocol whereas ODMRP is a mesh based protocol. There are however a number of additional aspects which merit closer inspection. An important difference between the two protocols is in the construction of the routing structure. i.e. the mesh in ODMRP and the tree



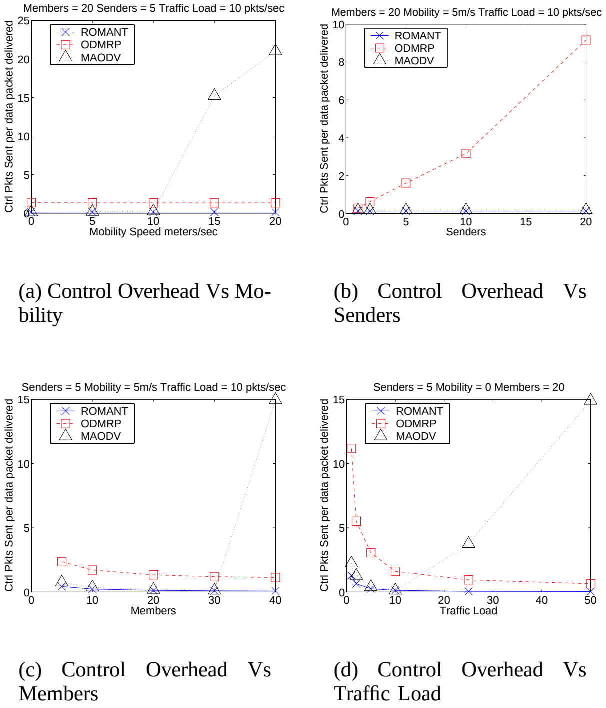
(a) Packet Delivery Ratio Vs Mobility

(b) Packet Delivery Ratio Vs Senders

(c) Packet Delivery Ratio Vs Members

(d) Packet Delivery Ratio Vs Traffic Load

Fig. 5. Protocol comparison results for Packet Delivery ratio



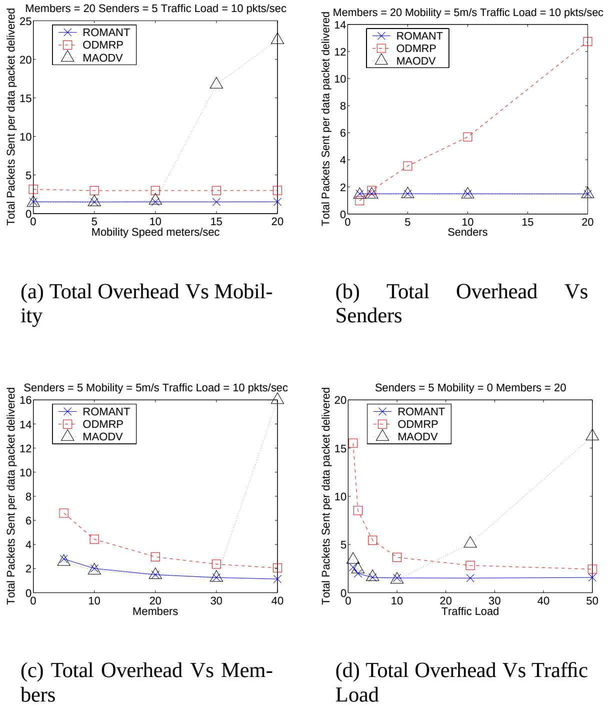
(a) Control Overhead Vs Mobility

(b) Control Overhead Vs Senders

(c) Control Overhead Vs Members

(d) Control Overhead Vs Traffic Load

Fig. 6. Protocol comparison results for Control Overhead



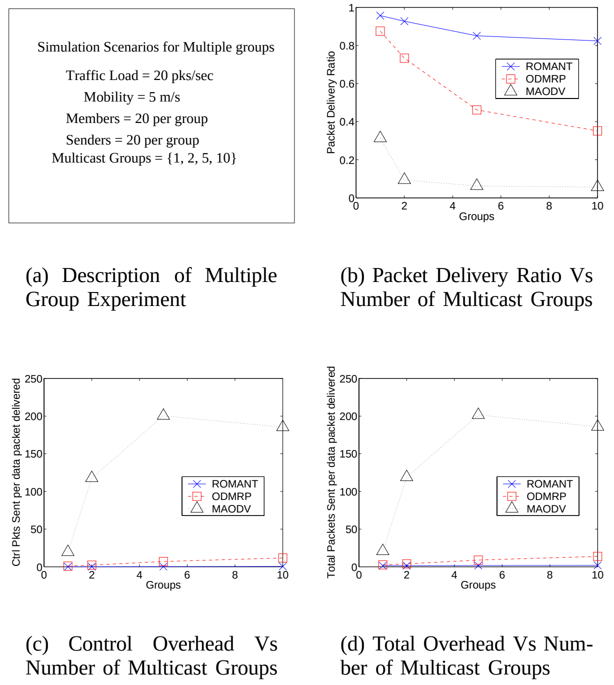
(a) Total Overhead Vs Mobility

(b) Total Overhead Vs Senders

(c) Total Overhead Vs Members

(d) Total Overhead Vs Traffic Load

Fig. 7. Protocol comparison results for Total Overhead



Simulation Scenarios for Multiple groups

Traffic Load = 20 pks/sec
Mobility = 5 m/s
Members = 20 per group
Senders = 20 per group
Multicast Groups = {1, 2, 5, 10}

(a) Description of Multiple Group Experiment

(b) Packet Delivery Ratio Vs Number of Multicast Groups

(c) Control Overhead Vs Number of Multicast Groups

(d) Total Overhead Vs Number of Multicast Groups

Fig. 8. Protocol comparison results for Number of Multicast Groups

in ROMANT. The establishment of the mesh in ODMRP is sender initiated and whereas the establishment of the tree in ROMANT is receiver initiated. Figures 9 and 10 illustrate the mesh established by ODMRP and ROMANT respectively, where nodes R1, R2 and R3 are receivers and nodes S1, S2 and S3 are senders. The forwarding group of ODMRP contains 16 nodes whereas the tree of ROMANT contains only 6 nodes. Hence, a data packet sent by node S3 is retransmitted by 16 nodes in ODMRP, whereas in ROMANT is is retransmitted only by 7 nodes (mesh members and node N15). The mesh in ODMRP (i.e., the forwarding group) is simply the union of the shortest paths connecting all senders to all receivers. This can lead to a significant and unnecessary data packet overhead if all senders are not also receivers. For example, as Fig. 9 shows, nodes N4, N8 and N12 retransmit packets from every sender, whereas they need to retransmit packets only from sender S1. Similarly, nodes N17, N18 and N19 need to retransmit packets only from node S3.

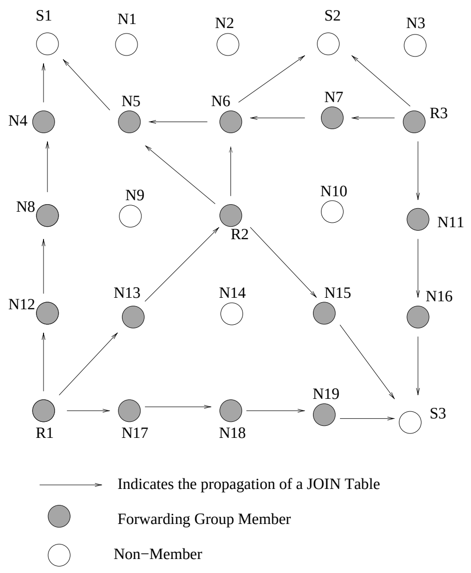


Fig. 9. Mesh establishment in ODMRP

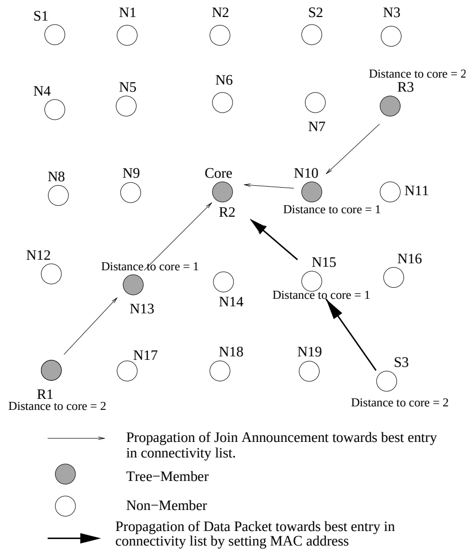


Fig. 10. Tree establishment in ROMANT

In addition to a higher data packet overhead, ODMRP also has a higher control packet overhead compared to ROMANT. Both ROMANT and ODMRP have two kinds of control packets. Core announcements and join announcements in the case of ROMANT and JOIN requests and JOIN[2] tables in the case of ODMRP. Both Core announcements and JOIN requests are flooded throughout the the network. Depending on the number of senders, the overhead of JOIN requests can be significantly more because JOIN requests are flooded by *every sender* whereas the core announcements are flooded by only the core. Every tree member in ROMANT transmits a join announcement and every mesh member transmits a JOIN Table. Because the number of tree members in ROMANT is significantly less than the number of mesh members in ODMRP, as we have shown in figures 10 and 9, the overhead of JOIN Tables is also more than that of join announcements.

Data packets may be lost due to broken routes as a result of mobility, or due to collisions. As ODMRP has a significantly higher overhead as described above, packet loss due to collisions can be expected to be higher in ODMRP. Though packet loss because of broken routes is expected to be lower in ODMRP as it offers multiple routes from sender to receiver, ROMANT does a good job in fixing broken links quickly as described in Section III-C.1.

2) Analysis of Results: As we can see from Figs. 5(a) and 5(c), the packet delivery ratio of ROMANT is comparable to that of ODMRP for varying mobility and number of multicast members. However, for increasing numbers of senders, increasing traffic load, and increasing number of multicast groups, the packet delivery ratio ROMANT is much better than that of ODMRP, as shown in Figs. 5(b), 5(d), and 8(b).

As described in Section III-D.1 and as can be seen from figures 6(a), 6(b), 6(c), 6(d), 8(c), 7(a), 7(b), 7(c), 7(d), 8(d) the control as well as total overhead of ODMRP is always higher than that of ROMANT. However for the scenarios involving varying mobility and number of multicast groups it is not enough to precipitate packet drops, as is shown in Figures 5(a) and 5(c). The main reason for ROMANT showing a high packet delivery ratio is that it is able to restrict packet losses due to broken routes as described in Section III-C.1.

However for the scenario involving multiple senders as the number of senders is increased beyond 10, the per-source flooding of ODMRP increases the overhead to a level which precipitates large scale collisions and packet drops, as shown is Figure 5(b). Similarly, when the number of multicast groups is increased, per source flooding per group has the same effect as shown in Figure 8(b). In both these scenarios packet delivery ratio of ROMANT is not

[2] In order to distinguish between the join's of ROMANT and ODMRP lowercase "join" refers to ROMANT and uppercase "JOIN"refers to ODMRP.

effected as a) Only the core is ROMANT performs flooding irrespective of the number of senders b) In case of multiple groups the core announcements of the different groups are aggregated as described in Section II-G. c) ROMANT restricts the number of packets lost due to broken routes. As the traffic load is increased both protocols suffer a drop in packet delivery ratio. However, ODMRP suffers a much larger drop because its higher control and data packet overhead results in network saturation much earlier. As a result, when the traffic load is increased beyond 10 packets/second, the packet delivery ratio of ROMANT is higher than that of ODMRP, as shown in Fig. 5(d).

E. *Control Overhead Bound*

As we have mentioned earlier, the control overhead of ROMANT does not vary much. As long as the core remains unchanged nodes generate a core announcement every time they receive a fresh announcement (one with higher sequence number). The core generates a core announcement every core_announcement_interval, which results in each node in the network generating a core announcement every core_announcement_interval i.e. every three seconds. Additional core announcements are generated every time a node detects a core change. This however does not result in a significant increase because the number of core changes that occur per node are very small as described in Section III-B. Another source of control overhead in ROMANT is the generation of join_announcements by each tree member once every join_announcement_interval i.e. every three seconds.

Table II shows the average control overhead, and its standard deviation for all the three protocols. Experiments 1, 2, 3, 4 and 5 refer to the same experiments described in Section III-A. The average number of control packets generated per node in more than 5000 for ODMRP and more than 6000 for MAODV. In ROMANT is in the 450 - 470 range for experiments 1 to 4. For experiment 5 it is higher because as the number of groups is increased although the core announcements are aggregated the join announcements are not. The average control overhead, as well as its standard deviation for experiment 5 is still much lower for ROMANT compared to the other two protocols. Low standard deviation indicates that the values do not vary much for different experiment scenarios, indicating that control overhead incurred by a node does not change much on changing mobility, number of senders, number of members or traffic load. ODMRP has the highest average and standard deviation for experiment 2. This indicates that the control overhead of ODMRP changes drastically on changing the number of senders. Other than experiment 2, MAODV has high values for all other experiments, both for the average control overhead as well as its standard deviation.

Exp No	1	2	3	4	5
ROMANT Avg	461.2	457.3	453.7	462.9	1033.5
ROMANT Std	5.0	7.3	12.5	6.2	645.9
ODMRP Avg	4690.1	8549.4	4757.1	5414.7	17590.8
ODMRP Std	83.0	9091.6	2315.8	1290.9	10827.6
MAODV Avg	12867.4	648.3	8807.3	10539.2	72338.8
MAODV Std	16830.7	22.9	18253.1	16717.5	19883.9

TABLE II
CTRL OVERHEAD AVERAGE AND STANDARD DEVIATION PER NODE

IV. CONCLUSIONS AND FUTURE WORK

The robust multicasting in ad hoc networks using trees (ROMANT) protocol is based on the simple idea of using core announcements to elect a core for the group and inform all routers of their distance and next-hops to the core. Each receiver connects to the core by periodically transmitting join announcements resulting in the formation of the multicast tree. In addition to providing the lowest control overhead compared to ODMRP and MAODV, ROMANT provides a very tight bound for the control overhead. In other words, the control overhead of ROMANT is almost constant when mobility, number of senders, number of members, or traffic load are changed. Even though it is a tree based protocol ROMANT provides comparable or better packet delivery ratio than ODMRP because ROMANT's mechanism for building the multicast tree and forwarding data packets from senders to receivers significantly restricts broken links. ROMANT does not depend on the existence of pre-designated cores or any unicast routing protocol.

MAODV's proved to be scalable with respect to the number of senders, but the link repair mechanism in MAODV was especially vulnerable in situations of real or perceived link breakages (e.g., high mobility, high traffic load, or a large multicast tree). ODMRP's main weaknesses were the the lack of scalability with respect to the number of senders, and large data-packet overhead due to path redundancy. ROMANT was also more scalable in terms of number of multicast groups compared to the other two protocols.

Our current research focuses on the integration of directional antennas, which could result in lower data-packet overhead because non-member nodes may not have to receive multicast packets for a group.

V. ACKNOWLEDGMENT

We would like to thank Venkatesh Rajendran for providing us with the code for the MAODV simulation in Qualnet 3.1 [21].

REFERENCES

[1] J. J. Garcia-Luna-Aceves and E.L. Madruga, "The core assisted mesh protocol," *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, vol. 17, no. 8, pp. 1380–1394, August 1999.

[2] S.J. Lee, M. Gerla, and Chian, "On-demand multicast routing protocol," in *Proceedings of WCNC*, September 1999.

[3] 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, Tel Aviv, Israel*, March 2000.

[4] E. Royer and C. Perkins, "Multicast operation of the ad hoc on-demand distance vector routing protocol," in *Proceedings of Mobicom*, August 1999.

[5] Park and Corson, "Highly adaptive distributed routing algorithm for mobile wireless network," in *Proceedings of IEEE INFOCOM*, March 1997.

[6] J. Xie snd R.R. Talpade, A. McAuley, and M. Liu, "Amroute: Ad hoc multicast routing protocol," *Mobile Networks and Applications (MONET)*, December 2002.

[7] C.W. Wu and Y.C. Tay, "Amris: A multicast protocol for ad hoc wireless networks," *Proceesings of IEEE MILCOM*, October 1999.

[8] J.G. Jetcheva and David B. Johnson, "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, October 2001.

[9] A. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (cbt)," in *Proceedings of ACM SIGCOMM*, September 1993.

[10] L. Ji and M. S. Corson, "A lightweight adaptive multicast algorithm," in *Proceedings of IEEE GLOBECOM 1998*, December 1998, pp. 1036–1042.

[11] L. Ji and M.S. Corson, "Differential destination multicast - a manet multicast routing protocol for small groups," in *Proceedings of IEEE INFOCOM*, April 2001.

[12] P. Sinha, R. Sivakumar, and V. Bharghavan, "Mcedar: Multicast core extraction distributed ad-hoc routing," in *Proceedings of the Wireless Communications and Networking Conference, WCNC*, September 1999, pp. 1313–1317.

[13] C.K. Toh, G. Guichala, and S. Bunchua, "Abam: On-demand associativity-based multicast routing for ad hoc mobile networks," in *Proceedings of IEEE Vehicular Technology Conference, VTC 2000*, September 2000, pp. 987–993.

[14] S.K. Das, B.S. Manoj, and C.S. Ram Murthy, "A dynamic core based multicast routing protocol for ad hoc wireless networks," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, June 2002.

[15] V. Devarapalli and D. Sidhu, "Mzr: A multicast protocol for mobile ad hoc networks," in *ICC 2001 Proceedings*, 2001.

[16] S. Lee and C. Kim, "Neighbor supporting ad hoc multicast routing protocol," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, August 2000.

[17] C. Perkins and E. Royer, "Ad hoc on demand distance vector (aodv) routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

[18] R. Perlman, "An algorithm for distributed computation of a spanning tree in an extended lan," in *ACM Special Interest Group on Data Communication(SIGCOMM)*, 1985, pp. 44–53.

[19] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "Protocol independent multicast - sparse mode (pim-sm)," *Internet-Draft, draft-ietf-pim-sm-v2-new-08.txt*.

[20] E.M. Royer and C.E. Perkins., "Multicast ad hoc on demand distance vector (maodv) routing," *Internet-Draft, draft-ietf-draft-maodv-00.txt*.

[21] Scalable Network Technologies, "Qualnet 3.5," http://www.scalable-networks.com/.