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*— We present the protocol for unified multicasting through announcements (PUMA) in ad-hoc networks, which establishes and maintains a shared mesh for each multicast group, without requiring a unicast routing protocol or the preassignment of cores to groups. PUMA achieves a high data delivery ratio with very limited control overhead, which is almost constant for a wide range of network conditions. Using simulations in Qualnet 3.5, we compare PUMA with ODMRP and MAODV, which are representatives of mesh-based and tree-based multicast routing in ad hoc networks. 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 PUMA attains higher packet delivery ratios than ODMRP and MAODV, while incurring far less control overhead.**

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

I. Introduction

[1] The objective of a multicast routing protocol for mobile ad hoc networks (MANET) is to support the dissemination of information from a sender to all the receivers of a multicast group while trying to use the available bandwidth efficiently in the presence of frequent topology changes. Several multicast routing protocols have been proposed for MANETs [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. 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) [7]. 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) [15] 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. As the simulation results presented in Section III show, although the performance of MAODV is very good for small groups, low mobility, and light traffic loads, its performance degrades sharply once a given value of group size, mobility, or traffic load is reached, which is due to a sharp increase in the MAODV control packets transmitted to maintain the multicast tree of a group.

ADMR [7] 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 [13] 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 the sender initiated construction of the mesh. This method of mesh construction results in a much larger mesh as well as numerous unnecessary transmission of data packets

[1]This work was funded in part by the Baskin Chair of Computer Engineering at UCSC.

compared to a receiver initiated approach, as explained in Section III. DCMP [12] is an extension to ODMRP that designates certain senders as cores and reduces the number of senders performing flooding. NSMP [14] is another extension to ODMRP aiming to restrict the flood of control packets to a subset of the entire network. However, DCMP and NSMP fail to eliminate entirely 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.

This paper introduces and evaluates the *Protocol for Unified Multicasting through Announcements* (PUMA) for ad hoc networks. PUMA does not require any unicast routing protocol to operate, or the pre-assignment of cores to groups. Section II describes PUMA in detail. The novelty in PUMA derives from its use of very simple signaling (multicast announcements) to accomplish all the functions needed in the creation and maintenance of a multicast routing structure in a MANET. Multicast announcements are used to: elect cores dynamically, determine the routes for sources outside a multicast group to unicast multicast data packets towards the group, join and leave the mesh of a group, and maintain the mesh of the group. We note that, in a dynamic network, either sources must flood packets to reach the receivers of a multicast group (and also the rest of the nodes in the network), or the receivers elect an intermediary (which we call core) to serve as the point of contact between the group and non-members, and intermediaries must flood the news about their existence to the rest of the nodes. In PUMA, we chose the latter approach, and our analysis shows that the control overhead of PUMA is fairly independent from such factors as mobility, number of senders, number of receivers, traffic load, and number of multicast groups.

Section III uses discrete event simulations in Qualnet 3.5 [16] to compare PUMA with ODMRP and MAODV, which are representatives of the state of the art in mesh-based and tree-based multicasting in ad hoc networks. The simulation experiments address the impact of mobility, number of senders, number of members, traffic load, and number of multicast groups on the data packet delivery ratio and the packet overhead incurred by each protocol. The results of these experiments show that, for those conditions in which ODMRP or MAODV perform at their best, PUMA attains the same or better packet delivery ratios as ODMRP and MAODV, while incurring the same or far less overhead per packet delivered. Furthermore, the results also show that the savings in control overhead in PUMA can be *orders of magnitude* compared to the overhead of MAODV and ODMRP, depending on the mobility of nodes, traffic load, group size, and number of sources per group.

We did not compare PUMA with CAMP, because our approach is intended to work without the need of any unicast routing protocol or predefined cores. We did not compare PUMA with DCMP, MZR, NSMP, and ADMR because the improvement of these approaches over ODMRP in terms of control packet overhead [12], [13], [14], [7] is significantly lower than what we achieve in PUMA. MZR also had a lower packet delivery ratio at high mobility. Section IV presents our conclusions.

II. PUMA Description

A. Overview

PUMA 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, PUMA 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, PUMA eliminates the need for a unicast routing protocol and the pre-assignment of cores to multicast groups.

PUMA 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 PUMA is essentially the same as the spanning tree algorithm introduced by Perlman for internetworks of transparent bridges [17]. 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 all shortest paths between the receiver and the core. All nodes on shortest paths between any receiver and the core collectively form the mesh. A sender sends a data packet to the group along any of the shortest paths between the sender and the core. When the data packet reaches a mesh member, it is flooded within the mesh, and nodes maintain a packet ID cache to drop duplicate data packets.

PUMA uses a single control message for all its functions, the *multicast announcement*. Each multicast announcement specifies a sequence number, the address of the group (group ID), the address of the core (core ID), the distance to the core, a mesh member flag that is set when the sending node belongs to the mesh, and a parent that states the preferred neighbor to reach the core. Successive multicast announcements have a higher sequence number than previous multicast announcements sent by the same core. With the information contained in such announcements, nodes elect cores, determine the routes for sources outside a multicast group to unicast multicast data packets towards the group, notify others about joining or leaving the mesh of a group, and maintain the mesh of the group.

B. Connectivity Lists and propagation of Multicast Announcements

A node that believes itself to be the core of a group transmits multicast 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.

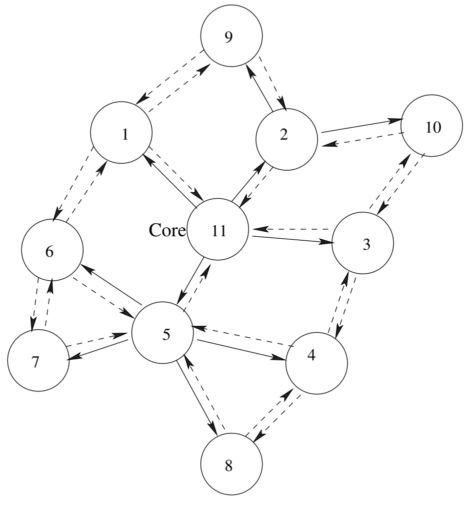
A node stores the data from all the multicast announcements it receives from its neighbors in the connectivity list. Fresher multicast announcements from a neighbor (i.e., one with a higher sequence number) overwrite entries with lower sequence numbers for the same group. Hence, for a given group, a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

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

For the same core ID, only multicast announcements with the highest sequence number are considered valid. For the same core ID and sequence number, multicast 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 multicast announcement, the node generates the fields of its own multicast announcement in the following way:

- Core ID: The core ID in the best multicast announcement
- Group ID: The group ID in the best multicast announcement
- Sequence number: The sequence number in the best multicast announcement
- Distance to core: One plus the distance to core in the best multicast announcement
- Parent: The neighbor from which it received the best multicast announcement
- *Mesh member: The manner in which this field is set is described in Section II-C*

The connectivity list stores information about one or more routes that exist to the core. When a core change occurs for a particular group, then the node clears the entries of its old connectivity list and builds a new one, specific to the new core. Figure 1 illustrates the propagation of multicast announcements and the building of connectivity lists. The solid arrows indicate the neighbor from which a node receives its best multicast announcement. Node 6 has three entries in its connectivity list for neighbors 5, 1, and 7. However it chooses the entry it receives from 5



Connectivity List at node 6
Core Id = 11 Group Id = 224.0.0.1, Seq No 79

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

Fig. 1. Dissemination of multicast announcements

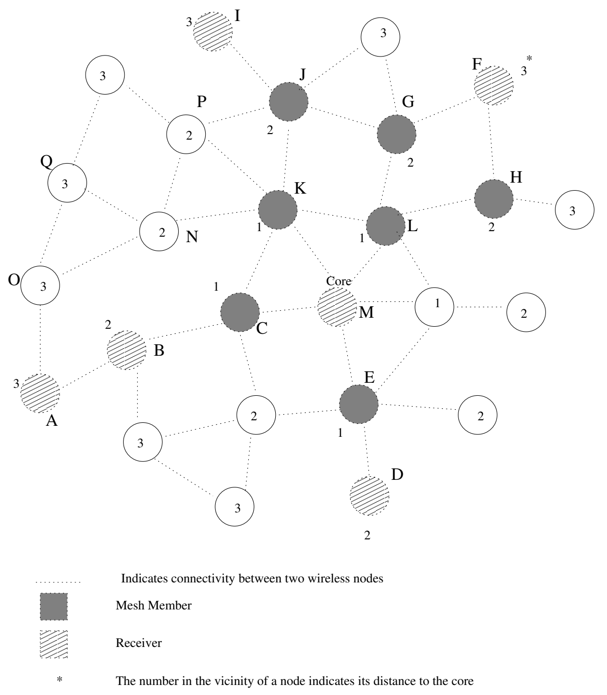
as the best entry, because it has the shortest distance to core and has been received earlier that the one from node 1. Node 6 uses this entry to generate its own multicast announcement, which specifies Core ID = 11, Group ID = 224.0.0.1, Sequence Number = 79, Distance to Core = 2 and Parent = 5. When a node wants to send data packets to the group it forwards it to the node from which it received its best multicast 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 multicast announcement sent by the core has distance to core set to zero and parent field set to *invalid_address*.

Multicast announcements are generated by the core every three seconds. After receiving a multicast announcement with a fresh sequence number, nodes wait for a short period (e.g. 100 ms) to collect multicast announcements from multiple neighbors before generating their own multicast announcement.

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

C. Mesh Establishment and Maintenance

Initially only receivers consider themselves mesh-members and set the mesh member flag to TRUE in the



Indicates connectivity between two wireless nodes
Mesh Member
Receiver
The number in the vicinity of a node indicates its distance to the core
Core

Fig. 2. Mesh Creation in PUMA

multicast announcements they send. Non-receivers consider themselves mesh-members if they have at least one *mesh child* in their connectivity list. A neighbor in the connectivity list is a mesh child if : (a) its mesh member flag is set; (b) the distance to core of the neighbor is larger than the nodes own distance to core; and (c) the multicast announcement corresponding to this entry was received in within a time period equal to two multicast announcement intervals. Condition (c) is used to ensure that a neighbor is still in the neighborhood. If a node has a mesh child and is hence a mesh member, then it means that it lies on a shortest path from a receiver to the core. As illustrated in Figure 2, Node M is elected as the core and all nodes in the network know their distance to the core by adding one to the best entry in their connectivity list as described in Section II-B. The receivers (nodes I, F, A, B, D and M) set the mesh member flag to 1 in their multicast announcements. Upon receiving the multicast announcement from F, nodes G and H consider themselves mesh-members. Node F qualifies as a mesh child for both of them, because its distance to the core (3) is larger than their own (2). Similarly, nodes J, K, L, C and E also consider themselves mesh members. Because a mesh-member serves as a mesh child of all nodes that have a distance to the core smaller than its own, it results in all of them becoming mesh members. The above scheme results in the inclusion of *all* shortest paths from the receiver to the core in the mesh. In our example, two paths of distance 3 from receiver F to the core M exist viz. F-G-L-M and F-H-L-M and both paths are part of the mesh.

Whenever a node generates a multicast announcement, it sets the mesh member flag depending on whether or not it is a mesh member at that point of time. In addition to generating a multicast announcement when it detects a core change, or when it receives a fresh multicast announcement, a node also generates a multicast announcement when it detects a change in its mesh member status. This could occur when a node detects a mesh child for the first time, or when a node that previously had a mesh child detects that it has no mesh children. The scheme would work even if a node did not generate a multicast announcement immediately after detecting a change in its mesh member status, and waited for the next batch of fresh multicast announcements to report its new mesh member status. This however, could lead to a delay in establishing the correct mesh, and could lead to packet drops as well as unnecessary transmissions of data packets. Also, as we discuss in Section III-E, this approach does not increase control packet overhead significantly.

D. Core Election

When a receiver needs to join a multicast group, it first determines whether it has received a multicast announcement for that group. If the node has, it adopts the core specified in the announcement it has received, and it starts transmitting multicast announcements that specify the same core for the group. Otherwise it considers itself the core of the group and starts transmitting *multicast announcements* periodically to its neighbors stating itself as the core of the group and a 0 distance to itself. Nodes propagate multicast announcements based on the best multicast announcements they receive from their neighbors. A multicast announcement with higher core ID is considered better than a multicast 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 *multicast 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.

E. Forwarding Multicast Data Packets

The parent field of the connectivity list entry for a particular neighbor corresponds to the node from which the neighbor received its best multicast announcement. This field allows nodes that are non-members to forward multicast packets towards the mesh of a group. A node forwards a multicast data packet it receives from its neighbor if the parent for the neighbor is the node itself. Hence, multicast data packets move hop by hop, until they reach mesh members. The packets are then flooded within the mesh, and group members use a packet ID cache to detect and discard packet duplicates. This is illustrated in Figure 2.

Nodes O and Q indicate in the multicast announcements that their parent is node N. Similarly, node P indicates in its multicast announcement that its parent is K. Assume that nodes O and P are senders. Node N forwards a data packet from O, but not from P, because only O has informed N that it considers N as its parent. Although node J is not the parent of P, it forwards the packet when it receives it from P, because mesh members do not consult their connectivity list before forwarding a packet. As a result, receiver I gets the packet sooner. Node J does not rebroadcast the packet when it receives it for the second time from K, because the ID of the packet is stored in its packet ID cache.

The routing of data packets from senders to receivers is also used to update the connectivity list. When a non-member transmits a packet, it expects its parent to forward the packet. Because all communication is broadcast, the node also receives the data packet when it is forwarded by its parent. This serves as an implicit acknowledgment of the packet transmission. If the node does not receive an implicit acknowledgment within ACK_TIMEOUT, then it removes the parent from its connectivity list.

F. Recycling Sequence Numbers

Like other unicast or multicast routing protocols using sequence numbers, PUMA 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 multicast announcement is only increased by the core of the group, and because the core floods its announcements periodically the same mechanisms used for the handling of sequence numbers in such link-state routing protocols as OSPF or in the spanning tree algorithm [17] suffice to ensure that nodes can trust the most recent multicast 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 PUMA against the performance of ODMRP [2] and MAODV [4] which are representatives of the state of the art multicast routing protocols for ad hoc networks. PUMA and MAODV are both receiver-oriented protocols. However, PUMA is a mesh-based protocol and provides multiple routes from senders to receivers. MAODV, on the other hand, is a tree-based protocol and provides only a single route between senders and receivers.

PUMA and ODMRP are both mesh-based protocols. However, every sender performs control packet flooding in ODMRP. Hence, depending on the number of senders there may be multiple nodes flooding the network periodically. In PUMA on the other hand, only one node, i.e., the core floods the network. The mechanisms for establishing a mesh are also different in PUMA and ODMRP. The mesh constructed in ODMRP is sender-initiated whereas

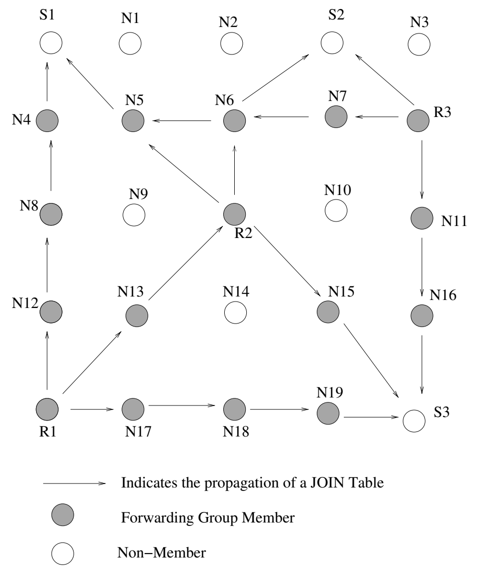


Fig. 3. Mesh structure of ODMRP

in PUMA it is receiver-initiated. Figures 3 and 4 illustrate the mesh established by ODMRP and PUMA 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 mesh of PUMA contains only 6 nodes. Hence, a data packet sent by node S3 is retransmitted by 16 nodes in ODMRP, whereas in PUMA is is retransmitted only by 7 nodes (mesh members and node N15). PUMA tends to concentrate mesh redundancy in the region where receivers exist by including all shortest paths from each receiver to the core, which is also a receiver. On the other hand, 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. 3 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.

We compared PUMA, ODMRP and MAODV using Qualnet [16]. Figure 5 lists the details about the simulation environment.

The distribution of Qualnet itself had the ODMRP code, which was used for ODMRP simulations. The MAODV code for Qualnet was obtained from a third party[2] who wrote the code independently of our effort following the MAODV IETF Specification [18]. 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 multicast announcements in PUMA) were set

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

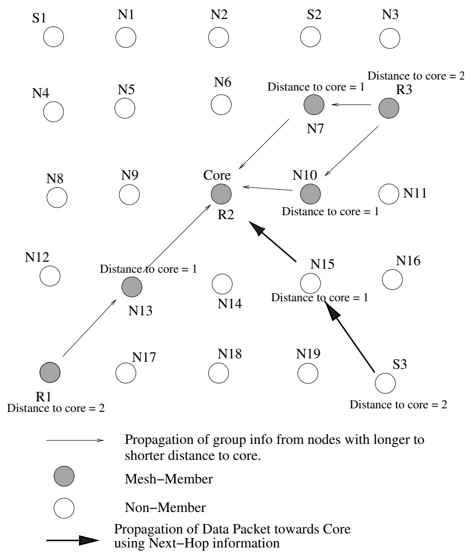


Fig. 4. Mesh structure of PUMA

Simulator	Qualnet 3.5
Total Nodes	50
Simulation Time	700 seconds
Simulation Area	1000m X 1000m
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. 5. Simulation Environment

to 3 seconds. The parameters for MAODV code are given in Figure 6. We have also implemented and tested PUMA in Linux 2.4.20-8, Red Hat Release 9, with the code having derived from our Qualnet implementation.

The metrics used for our evaluation were **packet delivery ratio**, **control overhead** and **total overhead**. Packet delivery ratio is defined as the the data packets delivered divided by the the data packets expected to be delivered. The data packets expected to be delivered is nothing but data packets sent times number of receivers. Control overhead is defined as control packets transmitted divided by the data packets delivered. Total overhead is defined as total packets transmitted (control packets + data packets) divided by data packets delivered. **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.

allowed hello loss	2
group hello interval	5 sec
hello interval	1 sec
hello life	3 sec
pkt id save	3 sec
prune timeout	750 ms
rev route life	3 sec
rreq retries	2
route discovery timeout	1 sec
retransmit timer	750 ms

Fig. 6. Maodv Parameters

A. Simulation Scenarios

Several experiments were carried out to determine the effect of mobility, number of senders, number of members, traffic load, number of multicast groups and terrain-size 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.
- Experiment 6 : Terrain size varied from 800 m X 800 m to 1600 m X 1600 m. Mobility = 0 m/s, Senders = 5, Members = 20, Traffic load = 10 pkts/sec, Multicast Groups = 1.

For the traffic load test (Experiment 4) 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-group test (Experiment 5), random allocation of nodes to groups could result in a single node being a member of multiple groups. The simulation environment and simulation scenarios are basically the same as those used by the developers of ODMRP [2] in [3] to compare their protocol against CAMP [1], AMRIS [6] and AMROUTE [5]. We have added experiments 5 and 6 because we believe that they are important experiments to evaluate the effectiveness of multicast protocols.

B. PUMA vs ODMRP

As we can see from Figs. 7(a) and 7(c), the packet delivery ratio of PUMA 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 PUMA is much better than that of ODMRP, as shown in Figs. 7(b), 7(d), and 10(b). The packet delivery ratio of PUMA is significantly higher than ODMRP for more than 10 senders, as shown in Fig. 7(b). This is because the per-source flooding of ODMRP leads to significant number of packet drops due to congestion as the number of senders is increased beyond 10. On the other hand, the only node that floods the network in PUMA is the core. Similarly, when the number of multicast groups is increased, per source flooding per group leads to congestion and packet drops. Increasing the number of multicast groups does not have a significant effect in PUMA, because the multicast announcements for the multiple groups are aggregated by every node, as shown in Figure 10(b). Similarly, the higher signaling overhead of ODMRP due to per-source flooding and higher data-packet overhead due to its mesh structure results in network saturation much earlier. As a result, when the traffic load is increased beyond 10 packets/second, the packet delivery ratio of PUMA is higher than that of ODMRP, as shown in Fig. 7(d).

The overhead of ODMRP is twice or more than that of PUMA for all simulation scenarios. This is expected, because: (a) the mesh structure of ODMRP results in numerous unnecessary transmissions, and (b) every sender in ODMRP performs control flooding, while only the core performs flooding in PUMA.

C. PUMA vs MAODV

Based on simulation results shown in Figs. 7(a), 7(c), 7(d) and 10(b) we can see that the packet delivery ratio of MAODV is very 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. 9(a), 9(c), 9(d) and 10(d)

A careful analysis of the packets sent in all four scenarios show that a large number of RREQ (with the join flag set), RREP and MACT packets are sent. These are the packets associated with tree reconstruction, and in MAODV indicates that the multicast tree is unstable and leads to significant signaling. 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 in MAODV. 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. 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.

MAODV's response to fixing links that appear to be broken 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 positive feedback 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." PUMA on the other hand is less susceptible to link breakages because it is a mesh based protocol. 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.

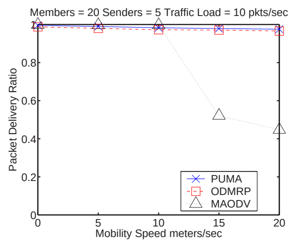
Another possible reason for the fast degradation of MAODV after a certain threshold value may be due to looping of multicast packets. Whether our previous conjectures or multicast looping are the reasons for MAODV's poor performance, it is clear that PUMA offers a far better alternative.

D. Core Stability

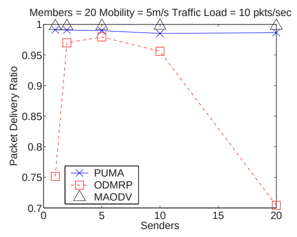
The stability of a core is important for the effective performance of PUMA. Frequent core changes, in addition to leading to control overhead, would also lead to a significant number of packet drops because the mesh would always be in a state of reconstruction. This is avoided because PUMA satisfies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

The first condition is met because nodes detect a partition in PUMA only if they fail to receive a multicast announcement from a core for three consecutive multicast announcement interval's (i.e., 9 seconds) from any neighbor. Hence, if partitions and reconnections are frequently occurring then nodes will not trigger a core election. Only when a node is partitioned from the core for a period of 9 seconds does a receiver node engage in the core election process.

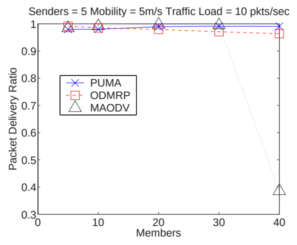
Nodes may detect a partition when it has not occurred if



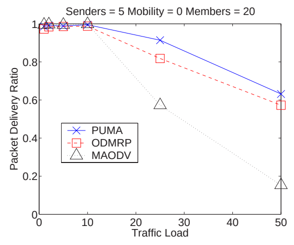
(a) Packet Delivery Ratio Vs Mobility



(b) Packet Delivery Ratio Vs Senders

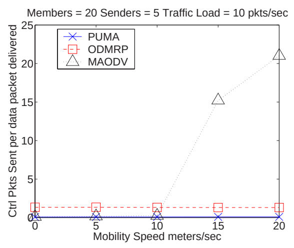


(c) Packet Delivery Ratio Vs Members

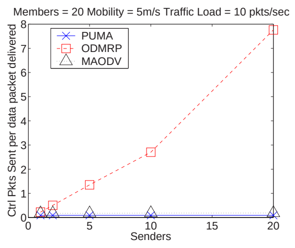


(d) Packet Delivery Ratio Vs Traffic Load

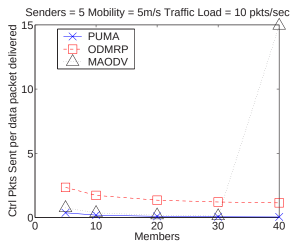
Fig. 7. Protocol comparison results for Packet Delivery ratio



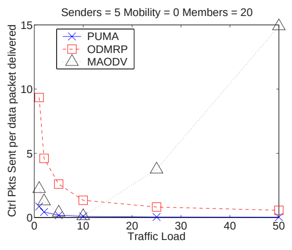
(a) Control Overhead Vs Mobility



(b) Control Overhead Vs Senders

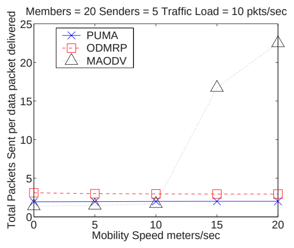


(c) Control Overhead Vs Members

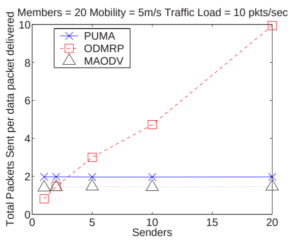


(d) Control Overhead Vs Traffic Load

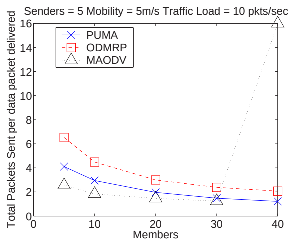
Fig. 8. Protocol comparison results for Control Overhead



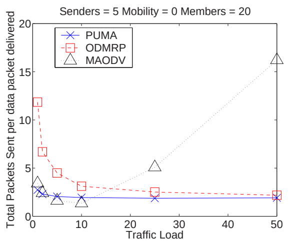
(a) Total Overhead Vs Mobility



(b) Total Overhead Vs Senders



(c) Total Overhead Vs Members



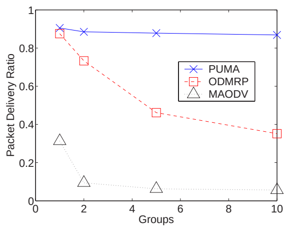
(d) Total Overhead Vs Traffic Load

Fig. 9. Protocol comparison results for Total Overhead

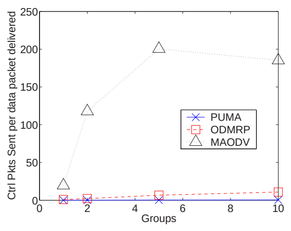
Simulation Scenarios for Multiple groups

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

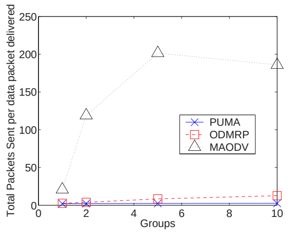
(a) Description of Multiple Group Experiment



(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. 10. Protocol comparison results for Number of Multicast Groups

mobility(m/s)	0	5	10	15	20
core changes	0	2.25	1.5	2.0	2.0

senders	1	2	5	10	20
core changes	1.75	1.75	2.25	1.75	1.75

members	5	10	20	30	40
core changes	0	0.5	2.25	2.25	4.25

terrain-size(m)	800 X 800	1058 X 1058	1265 X 1265	1442 X 1442	1600 X 1600
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

they consistently fail to receive multicast 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 multicast announcement on any path, for three consecutive multicast announcement intervals. 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. The idea of the terrain-size test (Experiment 6) was to detect the occurrence of false partitions in sparser networks where the average number of neighbors per node would be significantly lower. 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 terrain-size experiment. Real core changes can occur only if nodes are mobile. False core changes can occur irrespective of mobility. Having zero core changes in situations of zero mobility is a strong indicator that false core changes do not occur. Even for scenarios with mobility core changes are relatively small for a 700 second simulation time.

E. Control Overhead Bound

As we have mentioned earlier, the control overhead of PUMA does not vary much. As long as the core remains unchanged nodes generate a multicast announcement every time they receive a fresh announcement (one with higher sequence number). The core generates a multicast announcement every multicast announcement interval, which results in each node in the network generating a multicast announcement every multicast announcement interval i.e. every three seconds. This is the main source of control overhead at every node. Additional multicast 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-D.

Nodes also generate multicast announcements when they detect a change in their mesh-member status as described in Section II-C. Most of these kinds of multicast announcements are generated immediately after a core election when the mesh is being established. As the number of core elections is small, it follows that the generation of these kinds of packets is limited. In the steady state the network can be divided into three kinds of nodes a) Those that are at the heart of the mesh : These nodes do not generate these kinds of packets because they generally have multiple mesh children. Having one more or one less mesh child does not trigger them to generate these kinds of packets. b) Those that are far away from the mesh : these kinds of nodes never have mesh children and hence do not generate these kinds of multicast announcements. c) Those nodes that are at the periphery of the mesh : These nodes are most likely to move in and out of the mesh due to mobility, and hence generate these kinds of multicast announcements. However, nodes detect a link break only if they don't receive multicast announcements for 3 x multicast announcement interval i.e. 9 seconds, if such nodes move in and out of each others range rapidly, they do not generate multiple multicast announcements.

Table II shows the average control overhead, and its standard deviation for all the three protocols. Experiments 1, 2, 3, and 4 refer to the same experiments described in Section III-A. The average number of control packets generated per node in more than 2000 for ODMRP and more than 3000 for MAODV. In PUMA is in the 240 - 260 range. In the absence of multicast announcements triggered by a change in mesh member status, nodes would generate 700/3 i.e. around 234 multicast announcements. This shows that the average number of multicast announcements triggered by a change in mesh member status is less than 26 per node for a 700 second simulation or one in 27 seconds. 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.

IV. Conclusions and Future Work

The Protocol for Unified Multicasting through Announcements (PUMA) is based on the novel idea of using simple multicast announcements to elect a core for the group, inform all routers of their distance and next-hops to

Exp No	1	2	3	4	All
PUMA Avg	255.6	253.6	250.6	246.4	250.4
PUMA Std	9.2	0.4	4.4	12.2	8.4
ODMRP Avg	1813.8	2809.4	1879.5	1798.3	2062.1
ODMRP Std	40.0	3006.9	921.0	460.3	1488.5
MAODV Avg	5003.7	252.1	3425.0	4098.5	3237.9
MAODV Std	6544.8	8.9	7098.5	6500.9	5696.4

TABLE II

CTRL OVERHEAD AVERAGE AND STANDARD DEVIATION PER NODE

the core, join, and leave the multicast group. In addition to providing the lowest control overhead compared to ODMRP and MAODV, PUMA provides a very tight bound for the control overhead; the standard deviation of control overhead of all experiments carried out is less than 3.4% of the mean. In other words, the control overhead of PUMA is almost constant when mobility, number of senders, number of members, or traffic load are changed. PUMA also provided the highest packet delivery ratio for all scenarios. The mesh constructed by PUMA restricts redundancy to the region containing receivers, thus reducing unnecessary transmissions of multicast data packets. PUMA 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. PUMA was also more scalable in terms of number of multicast groups compared to the other two protocols.

In PUMA we have reduced the control packet overhead to a very small fraction of the total packet overhead, even for moderate loads, e.g., 5 pkts/sec in a 50 node network. Our current research focuses in two directions: (a) The effect of redundancy on protocol performance, i.e., mechanisms that adjust the amount of redundancy depending on network conditions; and (b) 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. We also to intend to compare the packet delay characteristics of PUMA with other multicasting protocols.

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

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

[7] 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.

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

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

[17] 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.

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

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

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