Cross Layer Ad hoc Multiple channel Multicasting Protocol

Ravindra Vaishampayan
Department of Computer Science
University of California Santa Cruz
Santa Cruz, CA 95064
ravindra@soe.ucsc.edu

J.J. Garcia-Luna-Aceves
Department of Computer Engineering
University of California Santa Cruz
Santa Cruz, CA 95064
jj@soe.ucsc.edu

Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

***Abstract*— Capacity improvement is one of the major challenges in the design of mobile ad hoc networks. Use of multiple channels is a useful technique in order to achieve capacity improvement by allowing nodes that need to exchange data to operate on the same channel and nodes that do not need to exchange data to operate on different channels. In the recent past some research has been done in the area of improving channel capacity using multiple channels for unicast flows. However we are not aware of any prior work for improving the channel capacity for multicast flows.**

We present the Cross Layer Ad hoc Multiple channel Multicasting Protocol (CLAMMP) for mobile ad hoc networks (MANET's). CLAMMP is the first protocol for MANET's which uses multiple channels to increase capacity for multicasting flows. CLAMMP is a cross-layered protocol in the sense that it traverses both the routing as well as the MAC layers. Using simulations in Qualnet 3.5, we compare CLAMMP with PUMA (tree-mode) and ODMRP, which are representatives of mesh-based and tree-based multicast routing in ad hoc networks. The results from a wide range of scenarios show that CLAMMP improves network capacity by an order of magnitude compared to the other two protocols.

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

I. Introduction

[1] Using multiple channels for improving capacity of an ad hoc network has been the focus of research in recent times. Using multiple channels can improve the capacity of an ad hoc network because it allows a certain set of nodes to communicate with each other, even if another set of nodes is also communicating in the vicinity as long as both the communications are held on separate channels. This is illustrated in Figure 1 which has been taken from [1]. Assuming that nodes 1-6 are in each others transmission range, only one transmission is possible at a time. However if all three pairs of nodes transmit on different channels, then all three transmissions are possible simultaneously. So to improve capacity, the protocols need to ensure that nodes that are interested in communicating are on the same channel and nodes that do not wish to communicate are not on the same channel. When nodes possess the capability to operate in multiple channels, an accompanying question is whether nodes have multiple radios, which allows them to operate on multiple channels simultaneously, or only one radio which means that they can operate on only one channel at a time. We decided to focus on approaches which require nodes to have only one radio. In addition to increasing hardware cost, multiple radios also consume greater power which is still a significant constraint in mobile ad hoc networks.

[1]This work was supported in part by the National Science Foundation under Grant CNS-0435522, the US Army Research Office under grants W911NF-04-1-0224 and W911NF-05-1-0246, and by the Basking Chair of Computer Engineering

A. Prior Work

Two recent approaches which try and improve network capacity using multiple channels and assume nodes equipped with only one radio are SSCH [1] and MMAC [2]. Both the protocols divide time into fixed size slices. In SSCH they are called *slots* whereas in MMAC they are called *beacon intervals*. Both protocols assume the existence of some in-band or out-of-band solutions e.g. GPS in order to ensure synchronization between the starting and ending times of the time slices of various nodes. In simulation results presented in [1], SSCH for the most part assumes 13 orthogonal channels operating at 54 Mbps whereas MMAC in [2] assumes 3 orthogonal channels operating at 11 Mbps. SSCH's slot is 10 ms whereas MMAC's beacon interval is 100 ms. In SSCH each node generates a schedule for itself and transmits it every slot. Instead of transmitting a long list, it simply transmits a set of 4 (initial channel, seed) pairs, which serve as a rule specifying how the node will change its channel in the future. Nodes that wish to exchange data just need to ensure that they have the same (initial channel, seed) pair on a particular slot. The interesting property of an (initial channel, seed) pair is that nodes which have the same (initial channel, seed) pair are always on the same channel, whereas nodes which have a different (initial channel, seed) pair will be on the same channel exactly once in the entire sequence of hops that a node goes through on a particular slot.

In MMAC on the other hand, which is based on IEEE 802.11 PSM each beacon interval is divided into an ATIM window, and a window where actual data is exchanged. In the ATIM window all nodes operate in the default channel. In the ATIM window nodes that wish to exchange data negotiate a common channel and switch to that channel once the ATIM window ends, and exchange data. Although there is no simulation data comparing the two protocols, we intuitively feel that because a common channel is used for synchronization traffic, this could be a bottleneck when the number of nodes wishing to exchange data is high. Moreover lost packets in the ATIM window phase means that the channels are not properly assigned. In addition the

1-4244-0507-6/06/$20.00 ©2006 IEEE

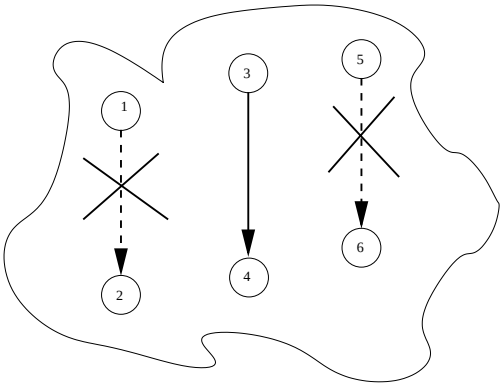


Fig. 1. Only one transmission is possible if only one channel exists

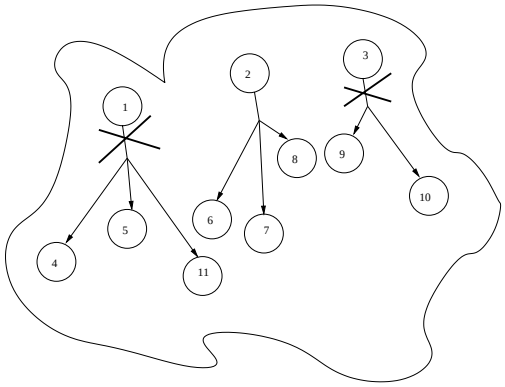


Fig. 2. Only one transmission is possible if only one channel exists

ATIM window is relatively large, i.e. 20% of the entire beacon interval. As a result a significant amount of bandwidth is used up in synchronization.

B. Contribution and problems solved by CLAMMP

Both SSCH and MMAC however consider only unicast flows, and not multicast flows. Multicast flows are fundamentally different from unicast flows, and as a result the techniques that are used to ensure greater capacity in terms of unicast flows cannot be directly applied to multicast flows. Moreover, the nature of the multicast flows is fundamentally related to the routing structure which is built by the routing protocol. As a result we have developed CLAMMP as a cross-layer protocol to handle both the media access as well as routing aspects. Although we feel that the media access part of CLAMMP could easily be modified to work with a different multicast routing protocol, we do not explore this issue exhaustively in this paper.

The challenges in improving throughput for multicast flows is significantly different from that for unicast flows. Multicasting packets are disseminated either over a tree or a mesh. In the design of CLAMMP we have assumed that they are disseminated over a tree although with minor modifications CLAMMP would also work with a mesh based protocol. Depending on the exact structure of the mesh, an algorithm would need to be designed to ensure that all nodes within the mesh have the same channel hopping sequence.

Figure 1 shows how nodes in unicast flows need to synchronize with the nodes they want to exchange data so that they are on the same channel. In multicast flows on the other hand, nodes may need to synchronize with multiple nodes depending on where they are in the multicast tree. To complicate matters further certain nodes may be part of multiple trees. This is illustrated in Figure 2. Assuming nodes 1 to 10 are in each others transmission range. Nodes 4, 5, and 11 are children of node 1 in a multicast tree. Nodes 6, 7 and 8 are children of node 2 in another multicast tree. Nodes 9 and 10 are children of nodes 3 in yet another multicast tree. If only one channel exists then the three multicast transmissions cannot occur at the same time. If however the nodes in the respective multicast trees are operating in different channels, then all three transmissions can occur simultaneously.

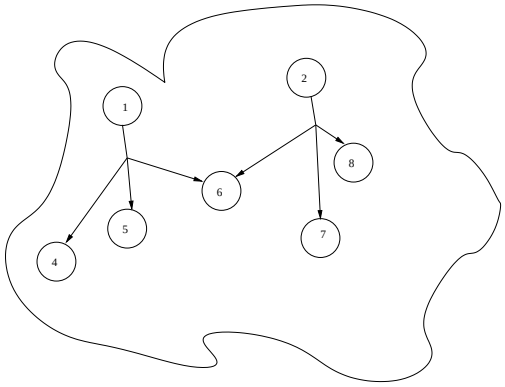


Fig. 3. Shared nodes reduce the capacity of the network

Another issue in the case of multicast flows is that if nodes are part of multiple trees then multiple transmissions transmissions cannot take place at the same time. This is illustrated in Figure 3. Assuming that node 1 transmits on channel A and node 2 transmits on channel B. Because node 6 is part of both trees transmissions from node 1 and node 2 cannot occur at the same time, as nodes have the ability to operate on only 1 channel at the same time. In order to minimize this problem the multicast routing protocol part of CLAMMP tries to generate trees with a minimum number of nodes which are part of multiple trees, or part of the fewest number of trees if the number of trees is very large. Requiring the construction of these kind of trees is one of the reasons why we decided to develop CLAMMP as a cross-layered protocol. This problem however would not occur if nodes were equipped with multiple radios.

Similar to MMAC and SSCH CLAMMP divides time into equal slices called slots. Similar to SSCH and MMAC, nodes are assumed to have some method of synchronization, so that each slot starts and ends at approximately the same time. Similar to SSCH the duration of each slot is set to be 10 ms. CLAMMP attempts to improve the capacity for multicasting flows by trying to ensure that nodes which are interested in exchanging data are on the same channel and the ones that do not want to exchange data are not. However as can be seen by comparing Figures 1 and 2 this synchronization is significantly more complicated for multicast flows than for unicast flows.

The remainder of the paper is organized as follows : Section II describes CLAMMP in detail. Section III provides detailed simulation results comparing CLAMMP to PUMA and ODMRP [3] using Qualnet 3.5 [4]. Section IV presents our conclusions and future work.

II. CLAMMP Description

A. Overview

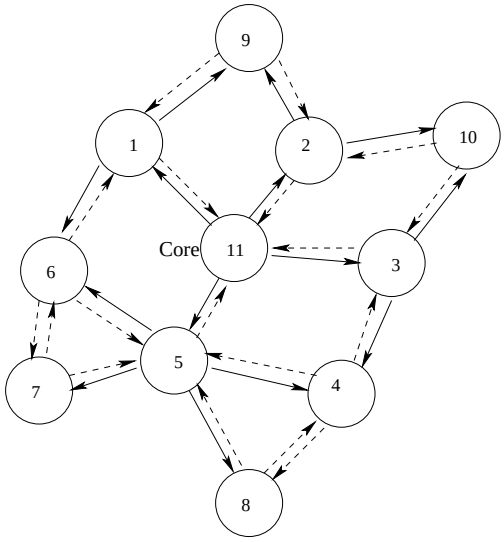
CLAMMP is a cross layered protocol encompassing the routing and MAC layers. CLAMMP is the first protocol which attempts to maximize throughput of multicasting flows by utilizing multiple channels to allow simultaneous transmission of data packets from nearby nodes without interference. The routing protocol of CLAMMP is a modified version of PUMA. The routing and MAC protocol parts of CLAMMP share information in order to achieve the overall objectives of CLAMMP.

In the MAC protocol part of CLAMMP, each node performs channel-hopping across slots based on 4 (initial channel, seed) pairs similar to SSCH. These 4 (initial channel, seed) pairs define the channel-hopping schedule of a node. Each node shares its schedule by broadcasting its (initial channel, seed) pairs once every slot. Nodes adapt their schedule to have a maximum overlap with nodes with whom they want to exchange data and minimum overlap with nodes with whom they do not want to exchange data.

B. CLAMMP Routing

Henceforth, we will call the routing protocol of CLAMMP as CLAMMP-Routing and the MAC protocol of CLAMMP as CLAMMP-MAC. CLAMMP-Routing is based on PUMA [5]. In this section we will focus on a brief description of CLAMMP-Routing and the main differences between it and PUMA. For a more detailed description of PUMA please refer [5].

1) Core Election: Like PUMA, CLAMMP-Routing elects a receiver of a particular group as the core of the group. A receiver participates in the election if it is not aware of any other core for the group. A receiver participates in a core election by broadcasting a multicast announcement to its neighbors. In order to broadcast a packet to all its neighbors CLAMMP-Routing simply passes the packet to CLAMMP-MAC. How CLAMMP-MAC handles the transmission of a broadcast packet is described in Section II-E.1. A multicast announcement has the following fields : *Core ID, Group ID, Distance to Core, Sequence Number, Tree Member, Parent*. A receiver participating in the core election for a particular group sets the Core ID to its own ID, and the Group ID to that of the corresponding group. The significance of the other fields will be explained shortly. When more than one receiver participates in an election, intermediate nodes will forward multicast announcements with the highest Core ID. Participants in the election will also accept another receiver as the core, once they receive a multicast announcement with a higher ID, and stop broadcasting multicast announcements. Eventually the participating receiver with the highest ID is elected core for the group. Once a node is elected core it periodically broadcasts multicast announcements which are flooded throughout the network. The core sets distance to core to zero, tree member to *receiving-member* (indicating that it is a member of the tree interested in receiving data packets) and parent to INVALID_ADDRESS (because it is the root of the tree and has no parent). Each consecutive multicast announcement has a higher sequence number.



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

Neighbor	Multicast Announcement	
	Distance To Core	Parent
5	1	11
1	1	11
7	2	5

Fig. 4. Dissemination of multicast announcements

2) Multicast Announcement Propagation: Every time a node receives a fresh multicast announcement (one with a higher sequence number than the last highest seen by the node), it waits for a certain time to collect all multicast announcements. Nodes ignore announcements with a sequence number other than the highest sequence number. All multicast announcements with the latest sequence number are stored along with the neighbor they were received from in a data structure called the *connectivity list*. Among all entries in the connectivity list the ones with the lowest distance to core are referred to as the *best multicast announcements*, and the neighbors from which they were received are referred to as *best neighbors*. Based on its connectivity list, a node generates its own multicast announcement, which has the same Core ID, Group ID and Sequence number of a best multicast announcement and adds 1 to the distance to core field (All best multicast announcements have to be same in these four fields).

This is illustrated in Figure 4. The solid arrows indicate the neighbors from which a node receives a best multicast announcement. Node 6 has three entries in its connectivity list for neighbors 5, 1, and 7. Nodes 5 and 1 qualify as *best neighbors* because they have the shortest distance to core i.e. 1. Node 6 generates its own multicast announcement, which specifies Core

ID = 11, Group ID = 224.0.0.1, Sequence Number = 79, distance to core = 2.

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 tree member status are forwarded immediately, without aggregation. This is to avoid delays in critical operations, like core elections and tree establishment.

3) Multicast Tree Establishment: The process of tree establishment starts with receivers and senders. All receivers (including the core as the core is also a receiver) set tree member to receiving-member in the multicast announcements they transmit, indicating they are members of the multicast tree that want to receive data packets. They set the *parent* field of the multicast announcement to the node they want to be the next-hop towards core. Once a node receives a multicast announcement from a receiving-member specifying the node as the parent, that node also considers itself as a receiving-member. Similarly, all senders set tree member to sending-member and similarly set their parent field. All parents of sending-members become sending-members on receiving the multicast announcement of their children. All receiving-members and sending-members set their tree member field appropriately in all future multicast announcements that they send. If a node qualifies to be both a sending-member as well as a receiving-member it prefers to be a receiving-member. The parent field for receiving-members and sending-members is chosen from among the *best neighbors*, in the following order :
a) If a node exists which is already a tree member (sending-member or receiving-member) of the group in question it is chosen.
b) Otherwise, if a node exists which is not a tree member of any group is chosen.
c) If however all best neighbors are tree member's of some group, none of which is the group in question, then the node which is the member of the minimum number of groups is chosen.

If more than one node exists which satisfies the above criteria then the node with the highest ID is chosen. The reason for these preferences is to avoid the situation described in Figure 3. The main idea is to restrict the number of groups in whose data forwarding a node is involved. This process of each tree member selecting a best-neighbor as its parent results in all nodes on a shortest path from each receiver and each sender to the core, being included in the multicast tree. This results in a multicast-tree rooted at the core.

This is illustrated in Figure 5. Nodes X, Y and Z are elected cores for groups 224.0.0.1, 224.0.0.2 and 224.0.0.3 respectively. Node V which is a receiver and hence a receiving-member has two best neighbors viz. nodes P and Q. It chooses node P as parent because node P is not a tree member of any other group whereas node Q is already a member of group 224.0.0.2 (Condition b above). However both nodes L and M have only one best neighbor which is node R, due to which both nodes select node R as parent. As a result node R is a receiving-member for 224.0.0.2 and 224.0.0.3. Node B has two best neighbors, nodes F and G. However it chooses node F as parent because node

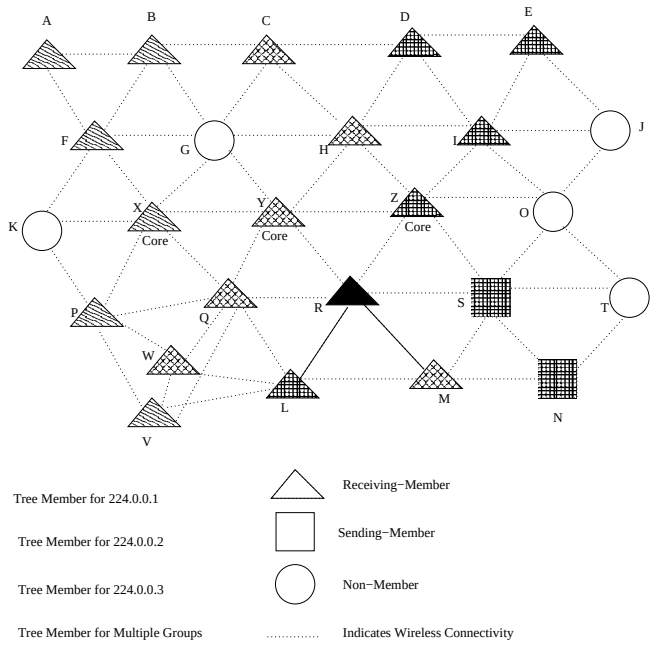


Fig. 5. Tree Establishment in CLAMMP

F is already a receiving-member of group 224.0.0.1 (Condition a above). The idea behind choosing parents which are already tree members is to leave greater number of nodes available for other groups. Node N is a sender for group 224.0.0.3 and hence a sending-member. Node N has only one best-neighbor node S which it chooses as parent, due to which node S also becomes a sending-member. Similarly node S also has only one best-neighbor node Z which is also the core, which it chooses as parent. Thus although node Z qualifies as a sending-member it prefers to remain a receiving-member.

C. Data Packet Forwarding

Once a data packet is transmitted by a sender it is retransmitted by its parent which is also a sending-member. Sending-members retransmit a packet only if they receive it from one of their children. In this way the packet moves hop by hop till it reaches a member of the inner-receiving-tree. The inner-receiving-tree comprises of those receiving-members which have at least one receiving-member as a child. The packets are then flooded within the inner-receiving-tree, and inner-receiving-tree members use a packet ID cache to detect and discard packet duplicates. The packet needs to be flooded only within the inner-receiving-tree because tree members which do not have any children simply need to receive the packet and not forward it. This is illustrated in Figure 5. A data packet sent by node N for group 224.0.0.3 will be forwarded by node S because S is a sending-member which receives the packet from its child. The packet sent by node S is received by nodes R and Z. As they are both inner-receiving-tree members both of them forward the packet because inner-receiving-tree members forward the packet irrespective of whether they are the parent of the sender or not. Node R will however not forward the packet for the second time when it is received from Z because it is present in R's packet ID cache.

Node I also forwards the packet once it receives it from Z as it is also an inner-receiving-tree member. Nodes L, D and E receive the packet but do not forward it as they are not inner-receiving-tree members.

Looking at Figure 5 again shows how CLAMMP increases network capacity for multiple multicast flows. The idea is for nodes to have channel hopping schedule so that nodes on the same tree are on the same channel most of the time and nodes on different trees are on different channels most of the time. Thus nodes like C and D can transmit data packets to nodes like H and I simultaneously.

D. CLAMMP-MAC

CLAMMP-MAC receives packets from CLAMMP-Routing and handles them as required, depending on whether they are broadcast packets or data packets for a particular multicast group. Similarly packets received from the network by CLAMMP-MAC are passed onto CLAMMP-Routing. As mentioned before in Section II-A, nodes in CLAMMP have a channel hopping schedule which they maintain and adapt so as to maximize the throughput of the existing multicast flows. As a result there are two main aspects of CLAMMP-MAC : 1) Representing the channel-hopping schedule and exchanging it with the neighbors 2) Adapting the schedule to coordinate with other nodes and maximize network capacity.

1) Representation and exchange of schedule information: The manner in which CLAMMP represents and exchanges schedule information is similar to SSCH. In this section we will describe it briefly for clarity. Time is divided into fixed size intervals called slots, and the slots of all nodes are assumed to by synchronized i.e. start and end at approximately the same time. The size of each slot is set based on the bandwidth. If 802.11a is being used which provides a bandwidth of 54Mbps, a slot length of 10 ms should suffice. Assume that from the time the network came up, slots are numbered 1, 2, 3, 4, 5 ... and so on. Slots numbered 1, 5, 9, 13 ... are referred to as SLOT 1, slots numbered 2, 6, 10, 14 ... are referred to as SLOT 2, slots numbered 3, 7, 11, 15 ... are referred to as SLOT 3 and slots numbered 4, 8, 12, 16 .. are referred to as SLOT 4. Please note that we use the uppercase *SLOT* to differentiate it from the lowercase *slot* which refers to any one unit of time in which the time is divided. Nodes use one (initial channel, seed) pair to represent the schedule of any one SLOT. Hence 4 (initial channel, seed) pairs represent the entire channel hopping schedule for the node. If there are 13 orthogonal channels as there are in 802.11a, then initial channel is represented as an integer in the range [0, 12] and seed is represented by an integer in the range [1, 12]. The value for the next channel in the slot is calculated based on the value of the present channel and the seed using the following formula :

$$newchannel \leftarrow (presentchannel + seed) \, mod \, 13$$

Hence if for a SLOT the (initial channel, seed) pair was (8, 4) the sequence of channels in that SLOT would be : 8, 12, 3, 7, 11, 2, 6, 10, 1, 5, 9, 0, 4 after which the schedule repeats. Nodes broadcast their 4 (initial channel, seed) pairs once every slot which allows other nodes to know about their channel hopping schedule. If two nodes have the same (initial channel, seed) pair for a particular SLOT, they will have the same channel on every slot in the SLOT. If two nodes have a different seed (it doesn't matter if they have the same initial channel) they will have the same channel in exactly one slot in every iteration for the SLOT. If however nodes have the same seed, but different initial channel they will never have the same channel in the SLOT. Hence in the unlikely case that two nodes have the same seed but different initial channel for each of the four SLOT's they will never share a common channel. To rectify this problem, after nodes have traversed through all channels on all slots, which is referred to as an *iteration*, they move to the *parity slot*. In the parity slot nodes have the channel which is equal to their seed for the first slot. This ensures that any two nodes will always share the channel on at least one slot per iteration.

2) Schedule selection and adaptation: As mentioned before nodes in CLAMMP select and modify their schedule based on the existing multicast flows in order to increase network capacity. We believe that this ability to adapt the channel hopping schedule to increase the throughput for multicasting flows is one of CLAMMP's most important contributions. For the remainder of this subsection we assume that multiple trees for multiple groups exist which have been constructed using CLAMMP-Routing. All the cores for all the trees randomly choose 4 (initial channel, seed) pairs, and broadcast them every slot. By default, children choose the same (initial channel, seed) pairs as their parents. However, although CLAMMP-Routing tries to minimize the number of nodes that are part of multiple trees as described in Section II-B.3 there will still be nodes that are. In that case nodes that are part of multiple groups will synchronize with different parents on different slots. This is illustrated in Figure 6. Assume node 5 is the core for Group 224.0.0.1 and node 6 is the core for group 224.0.0.2. Node 5 randomly picks four (initial channel, seed) pairs represented by [A, B, C, D] and node 6 picks four (initial channel, seed) pairs represented by [E, F, G, H]. As nodes choose the same (initial channel, seed) pairs as their parents by default, nodes 7, 10, 3 and 1 all pick (initial channel, seed) pairs [A, B, C, D] and nodes 2, 4, 9 and 13 all pick (initial channel, seed) pairs [E, F, G, H]. Node 8 however is a member of both trees hence has two parents, nodes 5 and 6, hence it synchronizes with node 5 on SLOT 1 and SLOT 3 and synchronizes with node 6 on SLOT 2 and SLOT 4. Hence the (initial channel, seed) pairs of node 8 are represented by [A, F, C, H]. Nodes 11 and 12 which are tree members for groups 224.0.0.2 and 224.0.0.1 respectively, synchronize with node 8 and have the same (initial channel, seed) pairs.

As mentioned in Section II-D.1 nodes broadcast their schedule information once every slot. This packet is called the *schedule packet*. A sample schedule packet that may be sent by node 8 in Figure 6 is shown in Figure 7. As shown in Figure 6 Nodes 11 and 12 will be on the same channel as node 8 in every slot because they are synchronized with node 8 in every SLOT. Nodes like node 8 in Figure 6 which are members of multiple groups can become a throughput bottleneck. As explained in Section II-E.2, a node like node 5 in Figure 6 cannot send a packet to node 8 on SLOT's 2 and 4 because both nodes are not on the same

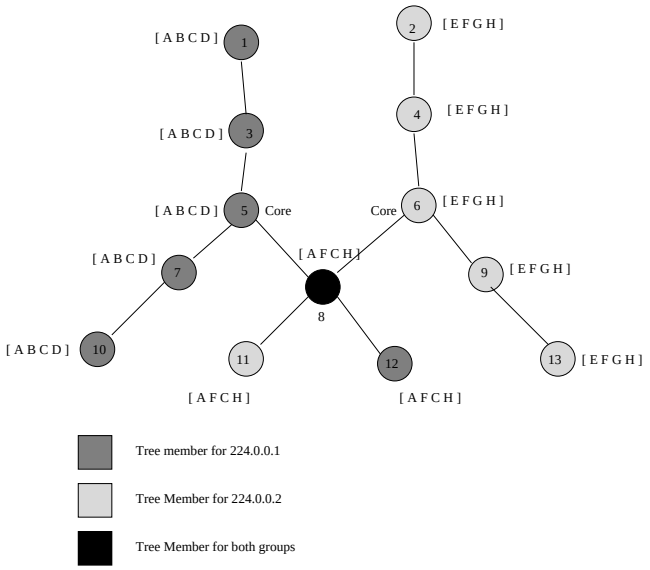


Fig. 6. Schedule Synchronization

Type	1
SLOT 1 Channel Seed Pair	(2, 5) = A
SLOT 2 Channel Seed Pair	(7, 9) = F
SLOT 3 Channel Seed Pair	(10, 2) = C
SLOT 4 Channel Seed Pair	(1, 8) = H

Fig. 7. Schedule Packet

channel. Thus packets for node 8 will be buffered at node 5 till the next slot when both of them share the same channel. If too many packets are received during this period then the buffer at node 5 may overflow and packets may be dropped. Thus node 8 synchronizes with node 5 and 6 on alternate SLOT's so node 5 is never desynchronized with any one parent for more than one slot at a stretch. Hence node 5 and 6 never have to buffer more than one slot worth of packets.

As mentioned previously in this section cores for different groups randomly choose different (initial channel, seed) pairs for each of their slots, and announce their schedule every slot using a schedule packet as shown in Figure 7. Children adopt the same (initial channel, seed) pairs as their parents. If a node is a member of multiple trees then it synchronizes with different parents on different slots as shown in Figure 6. A problem can occur if different cores which randomly choose the (initial channel, seed) pairs happen to choose the same (initial channel, seed) pairs on a few or all of the slots. As a result, nodes belonging to different trees and exchanging data for different multicast groups may end up interfering with each other. This is illustrated in Figure 8. Nodes 5 and 6 which are cores for groups 224.0.0.1 and 224.0.0.2 respectively, choose (initial channel, seed) pairs [A B C D] and [E B C F] respectively for the 4 SLOTS. This poses

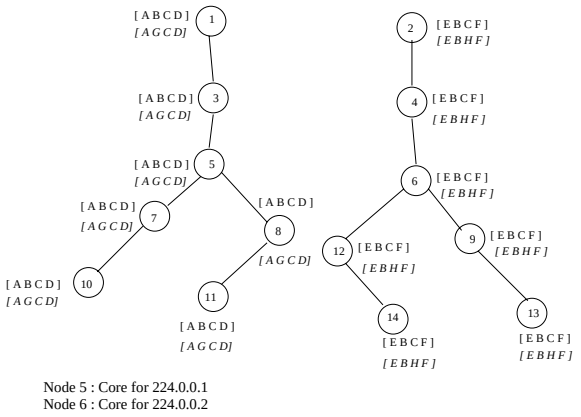


Fig. 8. Cores for different trees may choose the same (initial channel, seed) pairs

no problem as long as none of the tree members are within each other's radio range. However, if there are tree members from different groups which are in each others range like nodes 8 and 12 it will result in interference. For the purposes of explanation we say that nodes 8 and 12 *collide* on SLOTS 2 and 3. Thus if either node 5 or node 11 decided to send a data packet on SLOT 2 or 3, and node 12 also sent a data packet at the same time there would be a collision at node 8. Similarly there could be a collision at node 12 if node 8 and either of nodes 6 and 14 decided to send a data packet on SLOT's 2 and 3. If two nodes from different trees realize that they collide on an even numbered SLOT (which they will detect by examining each others schedule packets), then the node with the lower ID decides to pick a new (initial channel, seed) pair for that particular SLOT. Similarly, if two nodes from different trees collide on an odd numbered SLOT, then the node with the higher ID decides to pick a new (initial channel, seed) pair for that particular SLOT. This is to ensure that nodes with both higher as well as lower ID's equally share the burden of desynchronizing. Hence on SLOT 2 which is an even numbered SLOT node 8 would have the responsibility of desynchronization whereas on SLOT 3 which is an odd numbered SLOT, node 12 would have the responsibility of desynchronizing.

As described in Section I-B CLAMMP divides time into equal size units called slots. Whenever a node detects that it is synchronized with a neighbor belonging to a different multicast tree on the same SLOT, and it is the responsibility of the node to perform desynchronization, it picks a new (initial channel, seed) pair for that particular SLOT. It then sends out a Slot Reorganization packet informing neighbors about the group in question, the SLOT on which the collision occurs, and what the new (initial channel, seed) pair has been chosen as. This Slot Reorganization packet is sent on three consecutive slots to make sure it reaches all nodes in the tree in question. As Slot Reorganization packets (as well as schedule packets and multicast data packets) are sent unreliably, sending them with a random jitter on three consecutive slots ensures to a very high probability that the packet reaches all members of the multicast tree. Nodes receiving a Slot Reorganization packet will forward it

Type	2
SLOT	3
New Channel Seed Pair	H = (7, 9)
Group	224.0.0.2

Fig. 9. Slot Reorganization Packet

if they are members of the multicast tree of the group in question. If the newly selected (initial channel, seed) pair collides with yet another (initial channel, seed) pair of yet another multicast tree, then the node detecting this second collision chooses yet another (initial channel, seed) pair for the given SLOT, and floods another Slot Reorganization packet within the multicast tree. If three consecutive choices of (initial channel, seed) pairs lead to collisions, then the nodes in the tree stop making additional choices and revert back to the original (initial channel, seed) pair. However the probability of a failure is exceedingly small. The total number distinct (initial channel, seed) pairs in 156 in 802.11a (The channel can be chosen in 13 ways and the seed can be chosen in 12 ways). Even if we assume that 20 multicast trees corresponding to 20 multicast groups exist, the probability of a random choice being the same as an already selected (initial channel, seed) pair is just 12.8%. The probability of three random choices colliding with an already selected (initial channel, seed) pair is just 0.2%.

Suppose node 12 in Figure 8 detects that it collides with node 8 on SLOT 3, at slot number 30, based on the schedule packet it receives from node 8. As described above it has the responsibility of desynchronization as SLOT 3 is an odd numbered slot, and node 12 has a higher Id than node 8. Node 12 then randomly picks a new (initial channel, seed) pair H for SLOT 3. (If in addition to node 8, node 12 is aware of other neighbors belonging to different multicast trees, then node 12 needs to ensure that H is different from the (initial channel, seed) pairs on SLOT 3 of all those neighbors). Then on slots 31, 32, and 33 node 12 sends out a Slot Reorganization packet. The structure of a sample Slot Reorganization packet is shown in Figure 9. The Slot Reorganization packet is then flooded within the tree for group 224.0.0.2. If none of the nodes responds then (initial channel, seed) pair H, is chosen as the new (initial channel, seed) pair for SLOT 3 of group 224.0.0.2. However, if (initial channel, seed) pair H collides with some other multicast tree on SLOT 3, then the node detecting the collision chooses yet another (initial channel, seed) pair for SLOT 3. The process ends either when a new (initial channel, seed) pair is chosen or after three failures when nodes revert back to the original (initial channel, seed) pair, C. Node 8 similarly desynchronizes on SLOT2. Figure 8 illustrates the selection of new (initial channel, seed) pairs on colliding SLOT's in italics.

E. Interactions between CLAMMP-Routing and CLAMMP-MAC

CLAMMP-MAC passes all packets of protocol type CLAMMP in the IP header to CLAMMP-Routing that it receives from the network. CLAMMP-Routing passes two kinds of packets to CLAMMP-MAC which it needs to be transmitted to the network 1) Multicast Announcements which are broadcast packets and need to be transmitted to all neighbors and 2) Multicast data packets which need to be transmitted only to neighbors which are tree members for the group in question.

1) Transmission of Multicast Announcements: CLAMMP-Routing attempts to broadcast a multicast announcement to its neighbors once every *multicast announcement interval* i.e. 3 seconds. Thus every three seconds CLAMMP-Routing passes a multicast announcement to CLAMMP-MAC. Based on the schedule packet broadcast by every node every slot, as described in Section II-D.1 each node knows the channel hopping schedule of its neighbors. The multicast announcement is sent to each neighbor on the first slot that the node and the neighbor share a common channel. Please note that the multicast announcement passed by CLAMMP-Routing may be an aggregated multicast announcement representing multiple groups as described in Section II-B.1. For 13 channels and a slot time of 10 ms each node will share a channel with a neighbor at least once every 53 slots i.e. 530 ms for the extremely rare case described in Section II-D.1 and [1]. If a node is not synchronized with a neighbor on any of the four SLOTS without having the same seed as the neighbor on any particular SLOT it will share a common channel once every 13 slots i.e. 130 ms. If it is synchronized with a neighbor on 1 of the 4 SLOT's it will share a common channel once every 4 slots i.e. 40 ms. If however it is synchronized with a neighbor on all 4 SLOTS's it will always share a common channel with that neighbor and there will be no delay in sending the multicast announcement. The average delay in sending multicast announcements to neighbors in our simulations described in Section III-A was 87.56 ms.

2) Transmission of Data Packets: CLAMMP-Routing passes a multicast data packet to CLAMMP-MAC for transmission to the network only if the node is either a sending-member or a receiving-member. Under ideal situations, each tree member would have the same (channel, seed) pairs on all 4 SLOTS, due to which all tree member neighbors would always share a common channel. In that case CLAMMP-MAC could transmit a data packet as soon as it was received from CLAMMP-Routing because all the neighbors of the node which were tree members of the same group would also be in the same channel. The only exception to this would be when a node would be very close to the end of the slot and could not be sure that the recipient would be in the same slot by the time it received the packet. In that case the node would wait for the beginning of the next slot.

However as illustrated in Figure 6 all tree members of a particular group need not be synchronized due to nodes being tree members for multiple groups. In that case a node simply delivers the data packet to each neighbor that is a tree member individually other than the one from which it received the packet. A node delivers a data packet to a neighbor in the first slot that the node shares a common channel with that neighbor. This is possible because from the definition of CLAMMP-Routing which requires children to synchronize with parents on at least one SLOT. Please note that although all tree members neighbors may not be synchronized on all SLOT's most tree neighbors

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	54 Mbps
Data packet size	512 bytes

Fig. 10. Simulation environment

will be synchronized on most slots due to which a data packet sent by a node will generally reach multiple neighbors. In our simulations described in Section III-A each data packet was transmitted only 1.51 times on average.

III. Performance Comparison

We compared the performance of CLAMMP against PUMA [5] and ODMRP [3]. The MAC layer protocol for PUMA and ODMRP was the standard contention based 802.11a, whereas for CLAMMP it was CLAMMP-MAC, an extension to 802.11a as described in Section II-D. We modified PUMA to behave like a tree based protocol, so that nodes in PUMA include only one parent in the mesh as opposed to all parents as described in [5]. We did this because we wanted to compare CLAMMP against a mesh based (ODMRP) as well as a tree based protocol (PUMA).

For our experiments, we used the Qualnet [4] network simulator. Figure 10 lists the general parameters that characterize the simulation environment. The terrain size used is 1000 m X 1000 m for all experiments. Nodes moved at a constant speed (maximum speed was set equal to the minimum speed in the random waypoint model) as specified in the individual experiments.

In CLAMMP-MAC, all data packets as well as control packets (multicast announcements, Schedule Packets, Slot Reorganization Packets) were sent unreliably using only CSMA/CA. Each simulation was run for four different seed values. Multicast announcements in CLAMMP and PUMA as well as JOIN requests and JOIN tables in ODMRP were sent every three seconds.

As our main objective in this paper was to demonstrate how using multiple channels could improve the capacity of a network with respect to multiple multicast flows, our main performance metrics were **per receiver throughput** and **packet delivery ratio**. Per receiver throughput is defined as the average rate at which data traffic is received by any one receiver, which cannot be larger than the total traffic injected in the network by all senders combined. Packet delivery ratio is the ratio of data packets received to data packets that should have been received.

A. Simulation Scenarios

We carried out the following experiments:

- Experiment 1 : Multicast Groups varied from 1 - 20. Mobility = 5 m/s, Senders = 5 per group, Receivers = 20 per group. Each multicast group injected a traffic load of 250 packets/sec.
- Experiment 2 : Nodes varied from 50 - 200. Mobility = 5 m/s, Multicast groups = 20. Senders = 5 per group, Receivers = 20 per group. Each multicast group injects a traffic load of 250 packets/sec for a total traffic-load of 5000 packets/sec.
- Experiment 3 : Receivers varied from 5 to 30 per group. Mobility = 5 m/s, Multicast groups = 10. Senders = 5 per group. Each multicast group injects a traffic load of 250 packets/sec for a total traffic-load of 2500 packets/sec.
- Experiment 4 : Mobility varied from 0 to 20 m/s. Multicast groups = 10. Senders = 5 per group, Receivers = 20 per group. Each multicast group injects a traffic load of 250 packets/sec for a total traffic-load of 2500 packets/sec.
- Experiment 5 : Senders varied from 1 to 20 per group. Mobility = 5 m/s, Multicast groups = 10. Receivers = 20 per group. Each multicast group injects a traffic load of 250 packets/sec for a total traffic-load of 2500 packets/sec.

Experiment 1 basically measures the ability of the protocols to maintain a high throughput when the number of multicast flows is increased. Experiment 2 measures the ability of the protocols to exploit the added capacity of the network when the number of nodes in the network are increased. In Experiment 3, as the number of receivers per group is increased so does the number of nodes in each multicast tree. With a larger number of nodes involved in forwarding, this would lead to greater interference among the trees of different groups. Both the senders and receivers were chosen randomly from among the nodes. Experiment 4 checks how the protocols behave when mobility is increased. As described in Section II-E.2 and Section II-E.1, nodes in CLAMMP could experience a delay in the transmission of a data packet or a multicast announcement if the sending and receiving nodes are not on the same channel. By increasing the mobility we wanted to ensure that this problem would not be exacerbated in the face of high mobility. Experiment 5 tries to determine the effect of increasing the number of senders per group. Similar to increasing the number of receivers, as described in Experiment 3, increasing the number of senders would increase the number of sending members in each tree, potentially leading to more interference among nodes in different trees. Traffic load was equally distributed among all senders. Random allocation of nodes to groups could result in a single node being a member of multiple groups.

B. Analysis

As described previously in Section II-E.2 even though parents and children try to remain synchronized by adopting the same (initial channel, seed) pairs for their channel hopping schedule, this is not always possible because nodes may be members of different groups and may need to synchronize with different parents on different SLOT's. This problem becomes even more

Groups	Maximum Throughput	CLAMMP	PUMA	ODMRP
1	1.08	1.02	1.02	1.07
2	2.16	2.03	2.03	1.92
3	3.24	3.05	3.04	2.54
4	4.32	4.06	3.97	2.92
5	5.40	5.08	4.70	3.08
6	6.48	6.08	5.24	3.16
7	7.56	7.10	5.55	3.23
8	8.64	8.10	5.78	3.28
9	9.72	9.09	6.03	3.32
10	10.80	10.05	6.28	3.35
11	11.88	10.82	6.44	3.38
12	12.96	11.54	6.67	3.41
13	14.04	12.87	6.84	3.44
14	15.12	13.60	6.98	3.48
15	16.20	14.75	7.10	3.50
16	17.28	15.43	7.32	3.52
17	18.36	16.53	7.37	3.54
18	19.44	17.13	7.55	3.56
19	20.52	18.14	7.63	3.57
20	21.60	18.61	7.75	3.57

TABLE I

THROUGHPUT VARIATION WITH MULTICAST FLOWS

Protocol	50	100	150	200
CLAMMP	86.16	88.51	90.17	91.03
PUMA	35.88	35.99	36.09	34.22
ODMRP	16.53	16.19	15.86	15.50

TABLE II

PACKET DELIVERY RATIO VS NETWORK NODES

acute as the number of groups in the network is made greater than 10. In such a case, the sending node will have to postpone the sending of the data packet till the recipient node and the sending node share a common channel. For Experiments 1-5, the average delay experienced by a node in transmitting a data packet because the recipient was not on the same channel was 4.61 ms or around half a slot. In our simulations with 50 nodes and a terrain-size of 1000 m X 1000 m, this was not an issue as the average hop-count was only 2.76. However in larger networks this could be an issue. Similarly, due to a synchronization mismatch between parents and children a node may have to transmit a packet multiple times, so as to reach all neighbors. The average number of times a packet was transmitted in our simulations over all experiments was 1.51.

Similar to data packets, multicast announcements also need

Protocol	5	10	20	30
CLAMMP	93.15	92.85	93.06	92.91
PUMA	57.77	57.12	58.15	58.17
ODMRP	30.45	29.88	31.01	32.55

TABLE III

PACKET DELIVERY RATIO VS NUMBER OF RECEIVERS PER GROUP

Protocol	0	5	10	15	20
CLAMMP	97.79	93.06	90.83	88.67	85.43
PUMA	59.73	58.15	57.23	55.97	55.22
ODMRP	31.45	31.01	31.07	30.87	

TABLE IV

PACKET DELIVERY RATIO VS MOBILITY

Protocol	1	5	10	15	20
CLAMMP	93.69	92.84	91.98	91.29	90.47
PUMA	58.75	58.05	57.44	56.83	56.22
ODMRP	56.45	31.01	21.07	13.87	8.92

TABLE V

PACKET DELIVERY RATIO VS NUMBER OF SENDERS PER GROUP

to be sent multiple times and incur a delay because the node sending a multicast announcement and its neighbors are not on the same channel. The average delay incurred by a node sending a multicast announcement because its recipient was not on the same channel, was 87.56 ms or around 9 slots. The average number of times that a multicast announcement had to be transmitted was 4.32. The delay incurred and the number of times the packet is transmitted is much greater for a multicast announcement when compared to a data packet. This is because nodes interested in exchanging data packets amongst each other modify their channel hopping schedules so that they are on the same channel most of the time due to which delay and packet replication is low for data packets. Multicast announcements on the other hand have to be exchanged with all neighbors irrespective of whether they are synchronized or not, due to which nodes have to wait till they share a channel with the particular neighbor.

As mentioned previously, our main objective was capacity improvement with respect to multiple groups. As we can see from Table I the increase in throughput provides by CLAMMP with respect to the other two protocols viz. PUMA and ODMRP increases steadily as the number of groups in the network is increased. The main reason for this being that CLAMMP is able to ensure that nodes interested in exchanging data operate on the same channel, whereas nodes not interested in exchanging data operate on different groups. This drastically reduces the interference across multicast trees created for different groups. As we can see from Table I the throughput provided by CLAMMP continues to increase almost linearly even as the number of multicast flows is increased to more than 13. This may seem surprising because 802.11a has only 13 orthogonal channels. This occurs because even though the total number of groups in the network may be greater than thirteen, the number of groups any individual node is a member of would be much less because nodes try to build trees so that each node is a member of the least number of groups as possible as described in Section II-B.3.

On the other hand, protocols like PUMA and ODMRP the throughput provided tapers off much earlier, as increasing number of groups not only increases traffic load, but also leads to interference across the routing structures. PUMA performs better than ODMRP, because we have operated PUMA as a tree based protocol whereas ODMRP is a mesh based protocol. Meshes tend to have many more members than trees do because meshes provide multiple paths between senders and receivers. Having greater number of members means that the interference problem across meshes for different groups is more severe than that for trees. In addition the control overhead is ODMRP is greater than that for PUMA because every source performs a control packet flood.

Table II shows the behavior of the protocols as the number of nodes in the network is increased. ODMRP and PUMA register a slight drop in packet delivery ratio. Increasing the number of nodes increases control overhead and these two protocols are not able to effectively utilize the greater number of routes available due to an increase in the number of nodes. In CLAMMP on the other hand, there is a moderate increase in packet delivery ratio because CLAMMP is able to reduce the number of nodes which are tree members for multiple groups as shown in Figure 6. Having fewer overlapping nodes allows dissemination of data packets over different multicast trees to happen independently over different channels with minimum interference. In the future we plan to conduct tests with an even greater number of multicast groups, where we expect the impact of increasing the number of nodes in the network to be even greater.

Table III shows the performance of the three protocols on increasing the number of receivers per group. As we can see there is no discernible effect of increasing the number of receivers on the packet delivery ratio of the protocols. This is because on one hand the greater number of receivers leads to building of a larger tree/mesh, which leads to more interference. On the other hand having a larger tree/mesh offers greater redundancy of path. As these two effects tend to offset each other the overall packet delivery ratio does not increase very much.

Table IV shows the performance of the three protocols on increasing the mobility of nodes. All protocols have a reduction in packet delivery ratio with an increase in mobility. CLAMMP has the largest drop because as Section II-B.3 describes, CLAMMP attempts to build multicast trees where there is a minimum overlap across different groups. A consequence of this is that trees are built which have a minimum number of nodes, thus offering less redundancy of path. Lower redundancy leads to a larger packet drop in the face of mobility. ODMRP has the lowest drop in packet delivery ratio as it is a mesh based protocol. However the packet delivery ratio of ODMRP is still significantly lower than that of CLAMMP.

Table V shows the performance of the three protocols when the number of senders per group is increased. This has a major impact on the performance of ODMRP because in ODMRP each sender performs a network-wide control flood.

IV. Conclusions and Future Work

Cross Layer Ad hoc Multiple channel Multicasting Protocol provides a novel way of utilizing multiple channels in order to increase the throughput of multiple multicasting flows. Through simulations in Qualnet we demonstrate that CLAMMP effectively increases throughput of multiple multicast flows by building multicast trees and channel hopping schedules so that nodes that need to exchange data are on the same channel and nodes not interested in exchanging data are on different channels so as to minimize interference. In our simulations even for 20 multicast groups, with each multicast group injecting traffic of 1.08 Mbps, the throughput at each receiver was 18.61 Mbps out of a maximum of 21.60 Mbps (or a packet delivery ratio of 86.15%). The corresponding figures for PUMA and ODMRP were 7.75 Mbps and 3.57 Mbps respectively.

One drawback of CLAMMP is that it is able to increase network capacity only when multiple multicast groups exist. In future we plan to explore schemes so that network capacity is increased even when a single multicast group exists. Currently in the building of multicast trees in CLAMMP we consider connecting receivers to cores only on shortest paths. In the future we also plan to explore using longer paths if the nodes along those paths are sufficiently under-utilized. Yet another direction of future work is to make CLAMMP more robust in the face of mobility.

References

[1] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," in *Proceedings of Mobicom*, September 2004.

[2] J. So and N. Vaidya, "Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2004.

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

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

[5] R. Vaishampayan and J.J. Garcia-Luna-Aceves, "Protocol for unified multicasting through announcements (puma)," in *Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, October 2004.

[6] 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.

[7] 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.

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

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

[10] 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.

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

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

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

[14] 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.

[15] 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.

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

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

[18] 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.

[19] R. Vaishampayan and J.J. Garcia-Luna-Aceves, "Robust multicasting in ad hoc networks using trees(romant)," in *International Journal on Wirless and Mobile Computing(IJWMC)*.