Application Layer Multicasting for Mobile Ad-Hoc Networks with Network Layer Support

Jayanta Biswas and S. K. Nandy
CAD Lab, Indian Institute of Science
Bangalore, 560 012, India
{jayanta@cadl, nandy@serc} .iisc.ernet.in

Abstract

An ad hoc network is composed of mobile nodes without any infrastructure. Mobile nodes dynamically form temporary network to communicate with each other. Applications of mobile ad hoc networks require group oriented services. Hence multicast support is a critical feature for ad hoc networks. In this paper, we present an efficient application layer multicast solution suitable for medium mobility applications in MANET environment. We use network layer support to build the overlay topology closer to the actual network topology. We try to maximize Packet Delivery Ratio and to minimize control overhead. Through simulations we show that the control overhead for our algorithm is within acceptable limit and it achieves acceptable Packet Delivery Ratio for medium mobility applications. [1]

1. Introduction

Multicasting in a MANET environment faces many challenges because mobile nodes have limited bandwidth and battery power. The increased demand for multicast application in the mobile environment has resulted in the exploration of application layer multicasting with self organizing overlay topology as a viable alternative.

Many multicast routing protocols have been proposed for MANET [1] - [6]. For these protocols, even non member nodes actively participate in maintaining multicast state information and replicating multicast packets. If some non member nodes are fast moving, they affect all the involved multicast sessions.

Application Layer Multicasting is one of the possible solution for this scenario. In this case, nodes organize themselves to form an overlay topology and multicast communication between end systems is implemented by forwarding messages through the links between nodes in the overlay topology over unicast IP. By moving the multicast functionality to the end systems, we solve the problems associated with fast moving intermediate nodes in maintaining multicast state information but pay the penalty of increase in end to end latency due to duplication of packets flowing over underlying network. The work reported in [7], [8] and [9] are the only current available study in the literature for Application Layer Multicasting in MANET.

In this paper we propose a new scheme to solve the efficiency problem associated with application layer multicasting. In our proposal, we build overlay topology closer to the actual network topology with network layer support. The remaining sections are organized as follows. In section 2, we describe neighbor selection algorithm. In section 3, we describe the complete infrastructure of network layer support. In section 4, we describe simulation model and methodology and present comparative performance analysis of the proposed protocol with On Demand Multicast Routing Protocol (ODMRP) [1] in section 5. In section 6 we discuss related work and present the conclusion in section 7.

2. Neighbor selection algorithm

This algorithm creates a list of overlay neighbor nodes for each node in the topology graph. The list is sorted with increasing outdegree of each node in the topology graph. The first two entries in the list are selected while making the overlay neighbor list of the parent node in the topology graph. The first two entries are neighbors to each other and to all the list entries created for the current node. Hop count values between overlay neighbor nodes are also updated in this process. Once the algorithm runs for all the nodes, it outputs the neighborlist and the distance between two overlay nodes in terms of hop count. The pseudo code is presented in Fig. 1.

3. The proposed protocol

The disadvantage of overlay infrastructure is low packet delivery ratio and high end to end delay because of duplication of packets over the underlying network. In order to address the low efficiency of application layer multicasting,

1 This work was partially supported out of a research grant with ST Microelectronics.

```
compute−neighbors(int nodeid)
{

 list = NULL;
 for each child c of node for which visited[c] is FALSE
 {
  visited[c] = true;
  compute_neighbors(c);

  insert two neighbor nodes of c in the list;
 }

 if nodeid is an end node
 {
  nodeid is chosen as a;
  the node with least outdegree from the list is
  chosen as b;
 }
 else
 {
  the two nodes with least outdegree from the list is
  chosen as a and b;
 }

 all end nodes are assigned as neighbors of a and b
 unless already assigned;

 node a and node b are assigned as neighbor to each
 other unless already assigned;

 hop count between the neighbors is also registered
 in the neighborlist database;

 associate a and b as two end node neighbors with nodeid;

}
```

Figure 1. Neighbor selection algorithm running at each server.

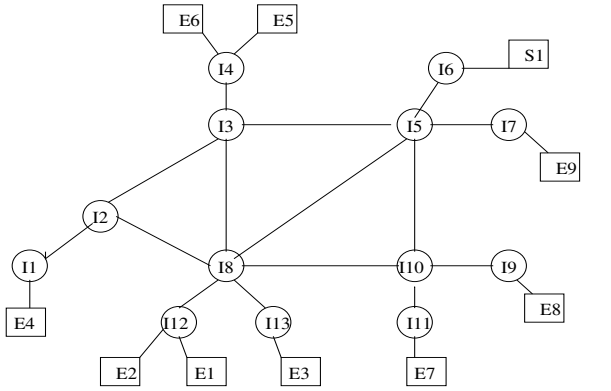

Figure 2. Example network topology.

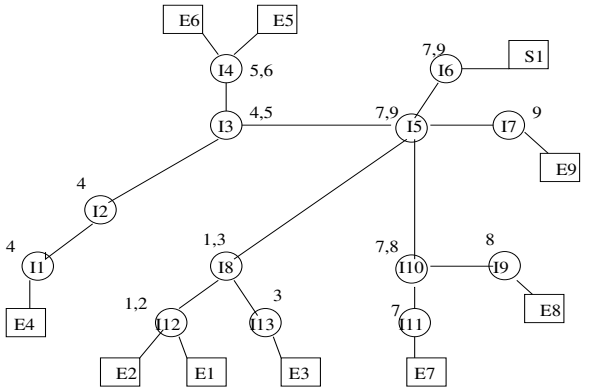

Figure 3. Topology view at the server.

we propose to follow the underlying network topology. We use network layer support to create the overlay topology. Only unicast capability between any two nodes is assumed. We assume there are at most n servers out of which one is always reachable. Nodes with the least identity value are chosen as server. Nodes chosen as servers can also act as end nodes at the same time. Here least identity value refers to least MAC address of the nodes in the overlay topology. The number n is selected by optimizing the performance of the proposed algorithm in terms of packet delivery ratio and control overhead using detailed simulation.

Each node periodically sends poll message to the next hop node to reach the server every t_{poll} time. Each node follows the procedure for all the servers. Each poll message contains network topology subgraph corresponding to the destination server node. So when the poll packet reaches the server, it carries underlying network topology spanning all the end nodes. It is a partial network topology as only nodes through which poll packets have traveled are included in the topology graph. Each server collects this information and updates its view of network topology graph. Each server periodically computes neighbor information of all end nodes from its view of network topology graph every t_{server} time. Neighbor selection algorithm is described in section 2. After computing neighborlist for each end node, each server unicasts the neighborlist information to all the one hop children nodes in the network topology graph. On receipt of the neighborlist information each node unicasts the same information to all the one hop children nodes in its network topology graph view except the node from which it receives the information. Thus the neighborlist information gets propagated to all the end nodes in the overlay topology. Each node in the overlay topology maintains multicast group membership information. Each end node calculates its routing table from its topology view. Each source node sets up and maintains source rooted multicast tree with overlay nodes.

3.1. Group management

Fig. 2 depicts one sample topology to illustrate the neighbor selection algorithm and creation of multicast data delivery tree. In this example we consider one server node and nine end nodes participating in the over-

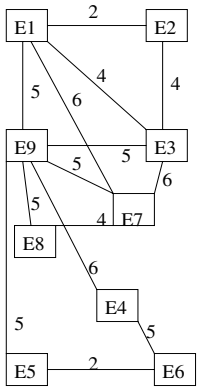

Figure 4. Overlay topology view at each node.

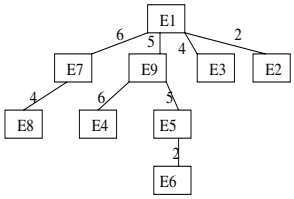

Figure 5. Overlay data delivery tree.

lay topology. There are 13 intermediate nodes. All these nodes constitute the network topology. Edge connectivity from node i to node j indicates that j is the next hop node from i to reach the server S1. All end nodes belong to one multicast group. Each end node sends *poll* message for the server S1 to the next hop node. The network topology graph maintained at the server is shown in Fig. 3. Fig. 3 also shows the end nodes selected for each intermediate node by the neighbor selection algorithm. For example at node I5, E7 and E9 are selected. It means E7 and E9 are neighbors to each other and also neighbors to E1, E3, E4, E5 and E8. The overlay topology created is shown in Fig. 4. Overlay data delivery tree is shown in Fig. 5. Edge weights in Fig. 4 and Fig. 5 represent distance between two end nodes in overlay topology in terms of hop count.

New Node Join: Prior to joining the overlay topology each node collects active server information during boot strapping and sends *node join request* to each server. Each server sends back *node join response* to the source node. The node keeps on sending *join request* to each server every $t_{joinsent}$ time until it receives node *join response* from that server. Once the node receives *node join response* from any server it transits to *joined* state. In the sample topology in Fig. 2, E9 sends *node join request* to the server S1. The server sends back *node join response* message to E9. E9 enters *joined* state and starts sending *poll* message to next hop node I7 for server S1. I7 is the next hop node from E9 to reach server S1.

Node Leave: When a node leaves, it sends *node leave request* to all nodes which is a next hop entry in its replication list.

Mesh partition repair: Whenever a source node detects partition of overlay topology view while trying to establish multicast connection to the nodes who are members of the same group, it puts the destination member node as the child of the source node and waits for *server response* message to arrive. By doing this we make sure the data connection to the member node is not disrupted completely. Once the source node receives *server response* message and connection setup timer expires, it recomputes its routing table.

In the sample topology, if node E4 and node E5 leaves together, then the overlay topology gets partitioned. Node E1 sends *connection delete* message to node E9 for member node E6. E9 clears replication list entry for source E1 for the multicast group to which node E6 belongs. E1 puts E6 as its child in the overlay data delivery tree and waits for *server response* message to come from S1. By doing so connection between E1 and E6 remains active. Once E1 receives *server response* message, it recomputes route to E6 from E1 and sends connection setup message again to node E6 to setup the data delivery tree.

If node E5 leaves, E5 sends *node leave request* to node E1 as E5 contains replication list entries corresponding to E1 source node. E1 receives *node leave request* message sent by node E5 and computes the new routes for affected active connections. E1 sends *connection setup* message to node E6. On receiving *connection setup* message E6 sends *connection confirm* message to E1. In this process data delivery tree is modified and E6 gets connected to E4.

Data delivery tree: The maximum degree of a node in a data delivery tree is restricted to 5 to avoid performance bottleneck at the end nodes. While sending *connection setup* message, if a node is unable to find a route to create data delivery tree satisfying the above condition, it violates this restriction in creating data delivery tree.

Multicast connection setup: Whenever neighbor table entry or multicast group membership changes and connection setup timer expires, multicast connections are reestablished if two consecutive end nodes in the path to the destination node are no more neighbors to each other. If the cost of the multicast connection from the source node to destination node decreases more than a fixed percentage value $p\%$, route to the destination is recalculated and multicast connection is reestablished. Cost of the multicast connections are calculated in terms of hop count.

Intermediate node leave: *Poll* message updates the network topology graph in the server if any intermediate node leaves. Old path information in the topology graph times out gradually.

Server node failure: A node is said to detect a dead server if it does not receive *server response* message from the server for $3 \times t_{server}$ time.

4. Performance Evaluation

This section presents the simulation results of the proposed overlay multicast protocol APPMULTICAST. We compare the results of APPMULTICAST with ODMRP. We choose ODMRP for comparison because this is the best performing multicast routing protocol available in the literature. This is shown in [11]. An overlay multicast scheme cannot provide performance comparable to ODMRP but we want to show merits and demerits of the proposed scheme compared to ODMRP.

The metrics are packet delivery ratio, number of data packets transmitted per data packet received, number of control bytes transmitted per data byte received and number of total packets transmitted per data packet received.

4.1. Experimental Setup

We develop a simulator with Glomosim library [12] for comparing the performance of ODMRP and APPMULTICAST. We use ODMRP code which is freely available with Glomosim 2.03 release. We fix bugs in *join reply* handling and doubly linked list deletion in ODMRP code and use for comparison purpose with APPMULTICAST. Mobility prediction is not present in this ODMRP implementation. ODMRP parameter values used are shown in Table I.

4.2. ODMRP protocol functionality

ODMRP forwards multicast packets using mesh based forwarding group concept. It is an on demand protocol and uses soft state approach in maintaining group membership. Group membership and multicast routes are updated by the source on demand. When multicast sources have data to send but no route to the multicast group, it broadcasts *join query* message to the entire network periodically to refresh the membership information and multicast routes. When an intermediate node receives *join query* message, it stores source address and sequence number in the message cache to prevent duplicate processing of the received packet. The routing table is updated with source node id. If the received packet is not a duplicate and *time to live* is greater than zero, node broadcasts the *join query* message again.

Once *join query* message reaches the node which belongs to the same multicast group, the node broadcasts *join reply* message. On receipt of *join reply* message, each node checks whether the next hop address matches its own id. If it matches, this node sets the forwarding group flag for the multicast group. It then rebroadcasts *join reply* message based on the match in its routing table. Thus join reply message gets propagated to the multicast source *via* the shortest path. This process creates multicast routes from source to receivers.

Parameter	*Value*
JOIN DATA refresh interval	3 sec
Acknowledgment timeout for JOIN TABLE	25 msec
Maximum JOIN TABLE retransmission	3

Table 1. Parameter Values for ODMRP.

After setting up forwarding nodes, source can multicast packets to receivers. When a node receives data packet, it forwards the packet if it is a forwarding node and the packet is not a duplicate.

Our simulation modeled a network of 50 mobile hosts placed randomly within a 1000 m $\times$ 1000 m. Radio transmission range used is 250 meters and channel capacity is 11 Mbps. A free space propagation model with a threshold cutoff is used in our experiments. All nodes share the same physical channel.

The IEEE 802.11 MAC with Distributed Coordinated Function (DCF) is used as the MAC protocol. Fisheye Routing [10] is used as unicast network routing protocol. We use default MAC code and Fisheye Routing code from the Glomosim distribution.

There is no network partition throughout the simulation and each simulation is conducted for 600 seconds. Multiple runs are taken with different seed values for each scenario and collected data are averaged over those runs.

4.3. Traffic Pattern

CBR source: Constant bit rate traffic with application data size of 512 bytes is generated. Rate of the traffic is an adjustable parameter which is varied for different scenarios. The senders are chosen randomly from group members and group members are chosen randomly among all nodes in the network. The member nodes join the multicast group at the start of the simulation and continue to be a member till the end of the simulation. Start time for data transmission for each source starts with a random time uniformly distributed over [5, 25] sec. Data is transmitted till the end of the simulation time.

5. Simulation Results

We conduct several experiments to show the performance of our proposal for different type of network configurations.

5.1. Mobility Speed

In this experiment, each mobile node stays stationary for 10 seconds and starts moving to a randomly selected location with a random speed uniformly distributed over [0, maxvalue] m/sec. Value of maxvalue is varied from 0 to 20. 20 nodes are configured as multicast members of the same group and 5 sources transmit packets at 2 pkt/sec each.

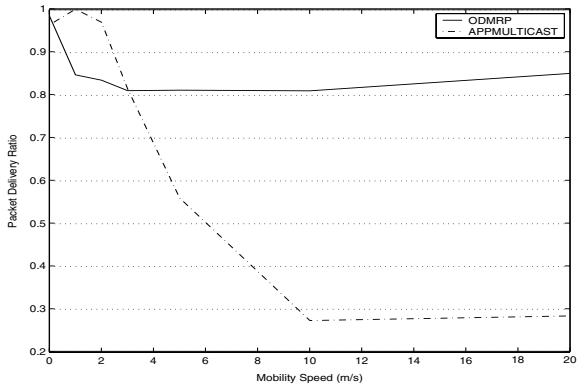

Figure 6. Packet delivery ratio for various mobility speed.

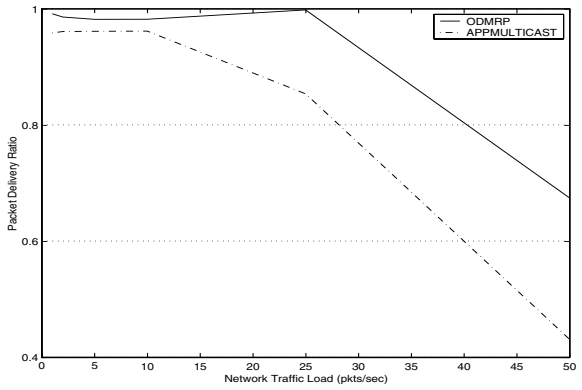

Figure 7. Packet delivery ratio for various network traffic load.

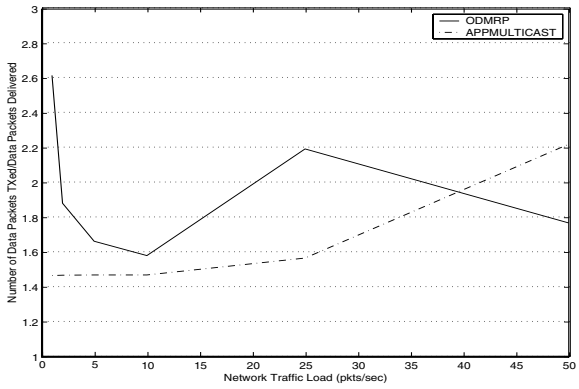

Figure 8. Number of data packets transmitted per data packet delivered for various network traffic load.

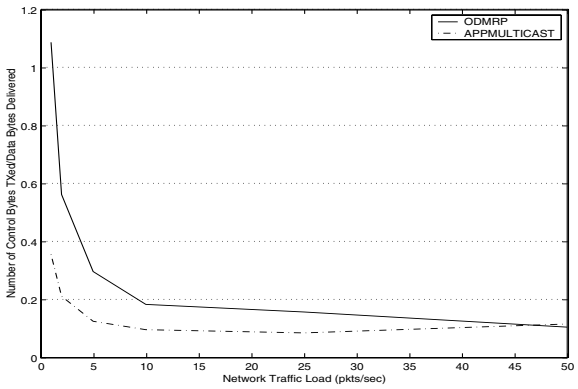

Figure 9. Number of control bytes transmitted per data byte delivered for various network traffic load.

Fig. 6 depicts packet delivery ratio of APPMULTICAST and ODMRP under different mobility speeds. ODMRP provides high packet delivery ratio even at high mobility. ODMRP uses redundant mesh forwarding logic and is able to deliver packets to destination even when primary routes are unavailable. Hence ODMRP is robust to mobility. APPMULTICAT provides good performance upto maximum mobility speed of 2 m/sec. As the mobility speed increases beyond 2 m/sec, packet delivery ratio drops sharply to 0.28. APPMULTICAST cannot update its neighbor table fast enough to track all the nodes at high mobility and it triggers lot of neighbor information updates at high mobility. Hence performance of APPMULTICAST is good only at low mobility.

5.2. Network Traffic Load

In this experiment the impact of network traffic load on the multicast routing protocols is studied. Each node stays stationary and 20 nodes are configured as multicast members of the same group. 5 sources transmit packets and total network traffic load is varied from 1 pkt/sec to 50 pkt/sec.

Fig. 7 illustrates packet delivery ratio of the two protocols under different network load scenarios. Both the protocol shows similar trend in packet delivery ratio as the network load increases but ODMRP performs better. For high network traffic load some nodes become bottleneck for APPMULTICAST and they start loosing data packets. As APPMULTICAST uses tree forwarding logic, this leads to sharp drop in packet delivery ratio for network traffic load beyond 25 pkt/sec.

Fig. 8 depicts number of data packets transmitted per data packet delivered for the two protocols. APPMULTICAST provides better performance than ODMRP until network load attains value of 40 pkt/sec based on this parameter. ODMRP provides better result beyond network traffic load of 40 pkt/sec.

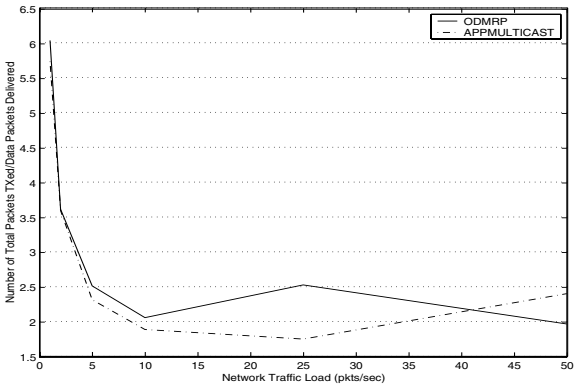

Figure 10. Number of all packets transmitted per data packet delivered for various network traffic load.

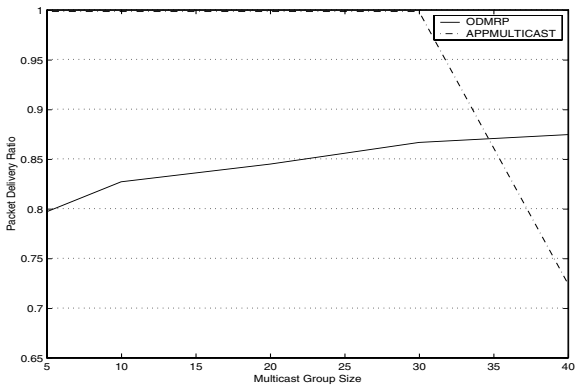

Figure 11. Packet delivery ratio for various multicast group size.

The value of this ratio increases for APPMULTICAST as some nodes start loosing data packets with the increase in network traffic load. The value of this ratio decreases with high network traffic load for ODMRP as it uses mesh forwarding logic and achieves high packet delivery ratio.

Fig. 9 shows number of control bytes transmitted per data bytes delivered. The value of this metric decreases with increased network traffic load. Increased traffic load does not trigger neighbor updates for APPMULTICAST. So this metric remains at a low value for APPMULTICAST at high network load.

Fig. 10 depicts total number of packets transmitted per data packet delivered for the two protocols. Both the protocols show similar trends. APPMULTICAST provides little better performance based on this metric. For APPMULTICAST the value of this metric increases slightly beyond network load of 25 pkt/sec because some nodes start loosing data packets during data replication.

APPMULTICAST provides better performance than ODMRP upto network load of 40 pkt/sec, beyond network load of 40 pkt/sec ODMRP performs better.

5.3. Multicast Group size

To study the impact of various multicast group sizes on multicast routing protocols, we varied multicast group sizes. Multicast group size is varied from 5 to 40 members. 5 sources transmit packets and total network traffic load is fixed at 10 pkt/sec. Each node stays stationary for 10 seconds and starts moving to a randomly selected location with a random speed uniformly distributed over [0, 1] m/sec.

Fig. 11 shows packet delivery ratio of the two protocols with different multicast group sizes. APPMULTICAST provides very high packet delivery ratio till the multicast group size is less or equal to 30. As the group size increases beyond 30, packet delivery ratio drops sharply.

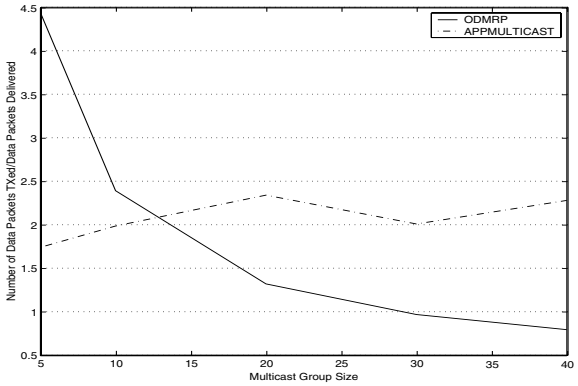

Figure 12. Number of data packets transmitted per data packet delivered for various multicast group size.

For ODMRP, packet delivery increases as multicast group size increases. As the density of the member nodes increases with the increase in multicast group size, *join reply* packets are lost less often and hence ODMRP performs better.

Fig. 12 depicts number of data packets transmitted per data packet delivered for the two protocols. For ODMRP, the value of this ratio decreases as *join reply* packets are lost less often with the increase in multicast group size. ODMRP provides better performance considering this parameter for group sizes larger than 15.

Fig. 13 shows number of control bytes transmitted per data bytes delivered. For smaller group size APPMULTICAST provides better performance based on this parameter. ODMRP provides better performance for group size beyond 35.

Fig. 14 shows number of total packets transmitted per data packet delivered. APPMULTICAST perfor-

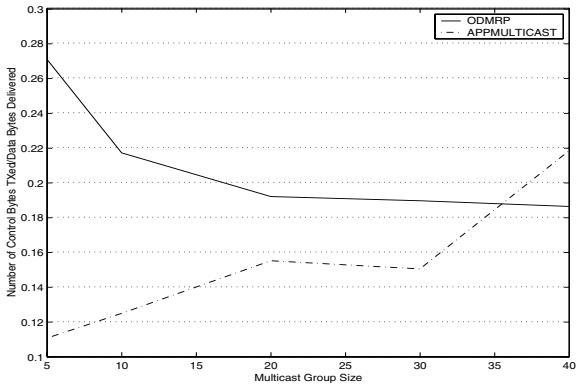

Figure 13. Number of control bytes transmitted per data byte delivered for various multicast group size.

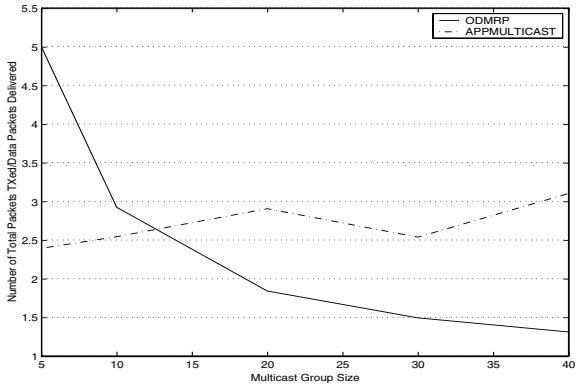

Figure 14. Number of all packets transmitted per data packet delivered for various multicast group size.

mance is good based on this parameter for smaller group size. ODMRP provides better performance for group size beyond 10.

Figs. 11-14, show that APPMULTICAST provides better performance than ODMRP till the multicast group size is less or equal to 30. ODMRP performs better for group sizes larger than 30. This is because ODMRP uses mesh forwarding logic and the number of forwarding nodes increases with the increase in multicast group size. APPMULTICAST replicates a new copy of the data packet for each destination node and the number of times a packet is replicated at a node increases with the increase in group size. Also increase in group size increases network load due to neighbor update traffic for APPMULTICAST.

5.4. Number of Senders

In this scenario, we studied the impact of number of senders on multicast routing protocols. This gives a measure of scalability of the protocol. 20 nodes are configured as multicast members of the same group and total network traffic load is fixed at 10 pkt/sec. Node mobility is kept at a low value. Each node stays stationary for 10 seconds and starts moving to a randomly selected location with a random speed uniformly distributed over [0, 1] m/sec. Number of senders is varied from 1 to 20.

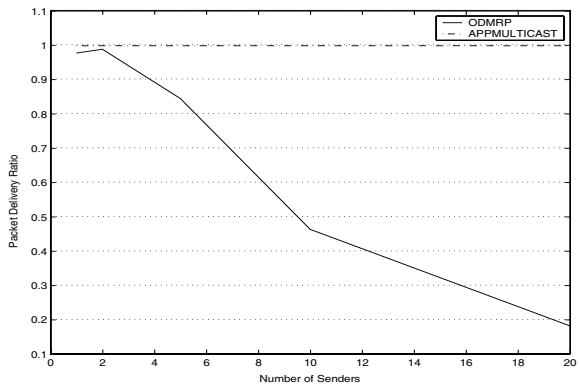

Figure 15. Packet delivery ratio for various number of senders.

Fig. 15 depicts packet delivery ratio of the two protocols with different multicast group sizes. APPMULTICAST provides much better performance than ODMRP considering this parameter. The value of this ratio stays at maximum for APPMULTICAST even with the increase of number of senders. For ODMRP the value of this ratio decreases with the increase of number of senders.

Results for the next three metrics cannot be shown due to space constraint. However, we describe the result. Number of data packets transmitted per data packet delivered decreases slowly with the increase of number of senders for APPMULTICAST. The value of this ratio increases with the increased number of senders for ODMRP.

Number of control bytes transmitted per data bytes delivered stays same for APPMULTICAST while it increases linearly for ODMRP with the increased number of senders. APPMULTICAST provides better performance than ODMRP based on this parameter.

Number of all packets transmitted per data packet delivered increases very little for APPMULTICAST while it increases linearly for ODMRP with the increased number of senders. APPMULTICAST provides better performance than ODMRP based on this parameter.

The observation for this experiment is explained here. The number of times a packet is replicated at a node does not increase and packet replication gets more distributed with the increase of number of senders for APPMULTICAST. Hence performance of APPMULTICAST improves with increased number of senders. Number of *join query*

and *join reply* packets flowing in the network increases as the number of senders increases. This leads to more contention of the shared channel and some nodes start dropping *join reply* packets. Hence ODMRP performance degrades with the increase of number of senders.

In sum, APPMULTICAST provides better performance than ODMRP for multicast groups with large number of senders.

6. Related Work

AMRoute [7] is a application layer multicasting proposal which connects multicast group members with bidirectional unicast tunnels and thus forms a shared mesh. Then it selects a shared data delivery tree from the mesh. One of the member is selected as logical core who is responsible for maintaining the shared tree. The protocol is simple and scalable at zero mobility. But as the mobility slightly increases, packet delivery ratio drops sharply. Other drawbacks of AMRoute are the existence of loops and inefficient formation of trees.

The work reported in [8] builds Location Guided Tree with the help of GPS device at each node. Two tree construction algorithms are proposed: *k-ary tree* construction (LGK) and *steiner tree* construction (LGS). Geometric length between two nodes is considered as the heuristic of link cost. LGS constructs the tree with this heuristics. LGK finds k nearest nodes and selects those nodes as one hop children. Other nodes are partitioned based on the distance from the children nodes. The disadvantage of this proposal remains with the fact that all nodes which are located nearby physically are not always one hop neighbors to each other.

Overlay Multicast[9] uses expanded ring search and link state exchange technique to create a virtual overlay mesh. It also uses stateless approach by appending multicast tree information with the data packet. The maximum degree of the virtual topology is also controlled. Then it creates multicast data delivery tree using source based *steiner tree* algorithm. This proposal is not likely robust to high node mobility.

However, with proposals [8] and [9] detail simulation methodology is not followed. Both of the proposal concentrates on the quality of the multicast data delivery tree in terms of different metrices. But none of them provides results with application data traffic. For example, loss of a control packet due to collision could alter the result of the simulation. So we cannot really comment on the actual performance of the above two schemes in ad hoc networks.

In our proposal each node in the overlay topology stores the overlay topology mesh view. We achieve comparable results for metrics such as control overhead and packet delivery ratio with other efficient multicast routing proposals available in the literature. Our proposal is suitable for applications with small and medium sized groups.

7. Conclusion

We have presented a novel overlay multicast solution for ad hoc networks using network layer support. We have also presented minute details of application layer multicast infrastructure using this topology. This algorithm is suitable for small and medium sized groups in a medium size overlay topology with around 200 nodes. To our knowledge, this is the first complete simulation study for overlay multicast in ad hoc networks. This algorithm is adaptable to network dynamics. Through simulations we show that our proposal is suitable for low traffic load network or with multicast groups with large number of senders for medium range mobility applications.

References

[1] S.J. Lee, M. Gerla, and C. C. Chiang, "On Demand Multicast Routing Protocol", *In Proceedings of IEEE WCNC'99*, pp. 1298-1302, September 1999.

[2] C. C. Chiang, M. Gerla, and L. Zhang, "Forwarding Group Multicasting Protocol for Multihop, Mobile Wireless Networks", *ACM-Baltzer Journal of Cluster Computing: Special Issue on Mobile Computing*, vol. 1, no. 2, pp. 187-196, 1998.

[3] J.J. Garcia-Luna-Aceves, and E. L. Madruga, "The Core-Assisted Mesh Protocol", *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1380-94, August 1999.

[4] E. M. Royer, and C. E. Perkings, "Multicast Operations of the Ad-hoc On-Demand Distance Vector Routing Protocol", *In Proceedings of ACM/MOBICOMM*, August 1999.

[5] L. Ji, and M.S. Corson, "A Lightweight Adaptive Multicast Algorithm", *In Proceedings if IEEE/GLOBECOMM'98*, November 1998.

[6] S.K. Das, B.S. Manoj, and C.S.R. Murthy, "A Dynamic Core Based Multicast Routing Protocol for Ad hoc Wireless networks", *In Proceedings of ACM/MOBIHOC*, June 2002.

[7] E. Bommaiah, M. Liu, A. MvAuley, and R. Talpade, "AMRoute: Ad hoc Multicast Routing Protocol", Internet Draft, draft-manet-amroute-00.txt.

[8] K. Chen and K. Nahrstedt, "Effective Location - Guided Tree Construction Algorithm for Small Group Multicast in MANET", *In Proceedings of IEEE/INFOCOM'02*, May 2002.

[9] C. Gui and P. Mahapatra, "Efficient Overlay Multicast for Mobile Ad Hoc Networks", *In Proceedings of IEEE*, 2003.

[10] G. Pei, M. Gerla, and T. W. Chen, "Fisheye State Routing: A Routing Scheme for Ad Hoc Wireless Networks", *In Proceedings of IEEE/ICC'00*, June 2000.

[11] S. Lee, W. Su, J. Hsu, M. Gerla, and R. Brigodia, "A Performance Comparison Study of Ad hoc Wireless Multicast Protocols", *In Proceedings IEEE/INFOCOM'00*, March 2000.

[12] UCLA Parallel Computing Laboratory and Wireless Adaptive Mobility Laboratory, "GloMoSim: A Scalable Simulation Environment for Wireless and Wired Network Systems", *http://pcl.cs.ucla.edu/projects/domains/glomosim.html.*