A Scalable Explicit Multicast Protocol for MANETs

Hrishikesh Gossain, Kumar Anand, Carlos Cordeiro, and Dharma P. Agrawal

Abstract: **Group oriented multicast applications are becoming increasingly popular in mobile ad hoc networks (MANETs). Due to dynamic topology of MANETs, stateless multicast protocols are finding increased acceptance since they do not require maintenance of state information at intermediate nodes. Recently, several multicast schemes have been proposed which scale better with the number of multicast sessions than traditional multicast strategies. These schemes are also known as explicit multicast (Xcast; explicit list of destinations in the packet header) or small group multicast (SGM). In this paper, we propose a new scheme for small group multicast in MANETs named extended explicit multicast (E2M), which is implemented on top of Xcast and introduces mechanisms to make it scalable with number of group members for a given multicast session. Unlike other schemes, E2M does not make any assumptions related to network topology or node location. It is based on the novel concept of dynamic selection of Xcast forwarders (XFs) between a source and its potential destinations. The XF selection is based on group membership and the processing overhead involved in supporting the Xcast protocol at a given node. If the number of members in a given session is small, E2M behaves just like the basic Xcast scheme with no intermediate XFs. As group membership increases, nodes may dynamically decide to become an XF. This scheme, which can work with few E2M aware nodes in the network, provides transparency of stateless multicast, reduces header processing overhead, minimizes Xcast control traffic, and makes Xcast scalable with the number of group members.**

Index Terms: **Explicit multicast, multicast, small group multicast, wireless ad hoc networks.**

I. INTRODUCTION

Multicasting [1] is the transmission of datagrams to a group of hosts identified by a single destination address and hence is intended for group-oriented computing. To support multicast in a network, intermediate nodes need to maintain state information for a given multicast session. This approach, when applied to sparsely distributed groups, becomes extremely inefficient and suffers from scalability problems. Recently, several multicast schemes have been proposed which scale better with the number of multicast sessions than traditional multicast schemes. These schemes are also known as explicit multicast (Xcast; explicit list of destination in the packet header) or small group multicast (SGM) [3], [4]. In Xcast, the multicast source includes the list of all group members explicitly in the extended packet header and assumes the underlying routing protocol to deliver the packet to all the destinations. This approach is also known as stateless multicast, since none of the intermediate routers need to maintain any state information related to any ongoing session.

In this paper, we have considered the problem of providing SGM in a mobile ad hoc network (MANET). A MANET is an infrastructure-less, dynamically reconfigurable wireless network, wherein the mobility of any node results in rapid and unpredictable changes in network topology. As a result, the issues related to multicast routing in MANETs prove to be different and harder to address than those in fixed wired networks [5]. There is a plethora of work to provide traditional multicast in MANETs. A recent survey of multicast routing protocols for MANETs can be found in [5] and a performance comparison of some of these protocols has been discussed in [6]. These protocols (e.g., [7]–[10]) follow the traditional multicast approaches meant for wired networks, i.e., they maintain multicast state information in each and every node for a given session, and hence have limited scalability with number of different multicast sessions.

In a MANET, due to node movement and frequent changes in network topology, it is difficult to maintain correct multicast state information. Hence, there is a recent trend towards designing stateless multicast routing schemes wherein an intermediate node (which also serves as a router here) need not maintain any state information. Generally, these schemes are meant for small groups and some of the examples include differential destination multicast (DDM) [11], location guided tree (LGT) [12], and route driven gossip (RDG) [13].

There are some proposals to enhance the basic Xcast scheme and make it scalable with number of users and reduce the overhead involved in processing the Xcast header at intermediate nodes [4], [14]. However, these proposals have been fine-tuned for wired network and are not suitable for MANETs. In this paper, with similar objectives we propose a new scheme for MANETs, named extended explicit multicast (E2M), which is built on top of Xcast and introduces novel schemes to make it scalable with the number of group members. E2M employs the concept of Xcast forwarder (XF), but unlike other schemes designed for wired networks, the selection of XF is dynamic and among other things, depends on the number of group members. E2M does not make any assumptions related to network topology or node location, which is important given the rapid and unpredictable change in MANET topology due to node movement. In short, E2M provides the transparency of Xcast, reduces header processing overhead at intermediate nodes, minimizes control traffic, and makes Xcast scalable with number of group members in a given multicast session.

Manuscript received January 3, 2004; approved for publication by Ekram Hossain, Division II Editor, February 6, 2005.

H. Gossain is with the Mesh Networks Product Group, Motorola, Inc., Maitland, FL 32751, email: Hrishikesh.Gossain@motorola.com.

K. Anand is with Qualcomm, San Diego, CA, email: kanand@qualcomm.com.

C. Cordeiro is with the Wireless Communications and Networking Department, Philips Research, Briarcliff Manor, NY 10510, USA, email: carlos.cordeiro@philips.com.

D. P. Agrawal is with the OBR Center for Distributed and Mobile Computing, Department of ECECS, University of Cincinnati, Cincinnati, OH 45221, email: dpa@ececs.uc.edu.

An earlier version of this paper has appeared in ICC 2004 [2]. This paper contains the complete description of E2M protocol and comprehensive simulation results.

The outline of rest of the paper is as follows. In Section II, we first provide an overview of Xcast, followed by a discussion of some of the related existing Xcast schemes in Section III. Section IV describes the proposed E2M protocol while in Section V we compare the performance of E2M with other existing schemes. Section VI describes how E2M can be extended for IP network, and in Section VII some of the routing and medium access control (MAC) layer issues related to implementing Xcast in MANETs are discussed. Finally, Section VIII concludes the paper.

II. XCAST IN MANETs

Xcast [3], [4] is source-based multicast scheme wherein the multicast source explicitly puts the list of destination addresses in the extended packet header, and assumes that the underlying routing protocol will deliver the packet to all of its destinations (for example, the option header in IPv4 can be used to put the list of destinations for an Xcast packet). In a wired network, if a source wants to send an Xcast packet, it will group the list of destination addresses according to their next hop and then repeatedly unicast the multicast packet to each of these next hops with the list of their respective destination addresses. When a next hop node receives a packet, it extracts the destination list meant for it and employs a similar approach as the source to forward the packet. Fig. 1 illustrates this basic Xcast scheme in detail. Here, node A is the source of an Xcast session and nodes B, C, D, E, F, and G are the prospective recipients. Thus, the extended header at the source node A will have N1: B, C, D, E, F, G where N1 is the next hop (downstream) node. Upon receiving the packet, node N1 determines that node N2 is the next hop for all the destinations contained in the packet header and modifies the extended header as N2: B, C, D, E, F, G. When node N2 receives this packet, it regroups the list of destinations based on next hops, namely, nodes N3 and N4. In the case of Fig. 1, node N2 modifies the extended header to be N3: B, C; N4: D, E, F, G. Subsequent nodes follow similar steps until the packet reaches all the destinations.

There are some schemes which extend this idea of Xcast for MANETs. For example, DDM uses an extended header to include the list of destinations and their next hop. Source node encodes the entire next hop list and the corresponding destination IPs in the packet header and broadcasts the Xcast packet to all its neighbors. All nodes in the neighborhood who receive it process the extended header to check if they are present in the next hop list.

Some points about basic Xcast scheme in MANETs are worth noticing here. First of all, all the nodes in the neighborhood who receive an Xcast packet need to process the extended header to check if they are present in the next hop list. This is a major overhead both in terms of processing and delay associated with forwarding an Xcast packet. The other option is to separately send the Xcast packet to each next hop as in a wired network, but this scheme does not utilize the broadcasting property of wireless links and results in duplicate transmissions of the same packet. Besides, maintaining the entire Xcast group membership at the source may create its own problem. For example, if members send periodic membership update to the source, it may result into a "join implosion" problem at the source, even for a moderate size Xcast session.

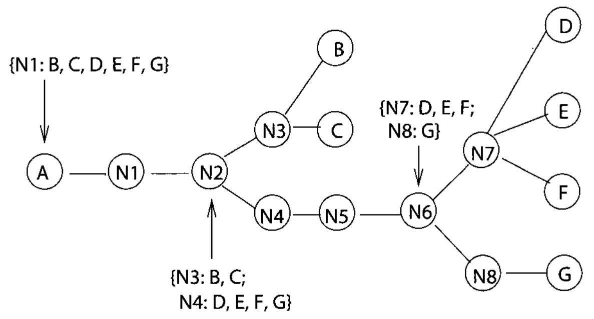


Fig. 1. Xcast packet delivery.

III. RELATED WORK

As discussed above, the self-routing nature of the Xcast scheme eliminates the need to maintain any state information at intermediate nodes, and hence makes it a suitable choice for small groups. Because of these features, there is a recent shift towards stateless multicasting in MANETs and this is reflected by protocols such as DDM [11], LGT [12], and RDG [13].

LGT is a small group multicast scheme based on packet encapsulation. It builds an overlay multicast packet distribution tree on top of the underlying unicast routing protocol. Multicast data is encapsulated in a unicast packet and transmitted only amongst the group nodes. One of the assumptions LGT makes is that every member of the multicast group is aware of other members of the group. Also, each node is aware of its own geometric location as well as the location information of all the other members in the group. On the other hand, RDG uses a probabilistically controlled flooding technique, termed as gossiping, to deliver packets to all group members. In DDM, the source encodes multicast receiver addresses in multicast data packets using a special DDM Data Header. This variable length destination list is placed in the packet headers, resulting in packets being self-routed towards the destinations using the underlying unicast routing protocol. It eliminates maintaining per-session multicast forwarding states at intermediate nodes and is thus scalable with respect to the number of sessions.

It is worthwhile to note that though the development of Xcast schemes as a stateless multicast protocol is intended for small groups, the size of a "small group" is still not well defined. For example, if the number of destinations which map to same next hop is more than 9, and assuming Xcast protocol uses the option header in IPv4, a source can include at most 9 destinations in the packet header [4], and will have to perform multiple transmissions of the same packet so as to reach more than 9 destinations.

When the group size is moderate or large, placing the addresses of all the members into the packet header turns out to be extremely inefficient. Although protocols such as DDM offer a caching mode wherein only the difference between the previous headers is kept in the packet header, the support of caching in each and every intermediate node is a tough task in MANETs where nodes can move arbitrarily. In addition, DDM requires every node (router) in the network to cooperate. The proposed

E2M protocol is designed keeping these constraints in mind, while at the same time making the Xcast scheme more scalable with number of group members, and incurring lower network overhead.

IV. THE PROPOSED E2M PROTOCOL

A. Overview of Proposed Approach

The E2M protocol aims to overcome the limitations in the existing Xcast schemes for MANETs by utilizing a new combination of adaptive mechanisms. E2M employs Xcast forwarders (XFs) as hierarchical forwarders, which are selected dynamically during the message forwarding procedure. Similar schemes to tackle the overhead involved in supporting Xcast have been proposed for wired network [4], [14], wherein an edge router, also known as designated router (DR), joins the Xcast session after receiving a join message from one of its downstream members. In this case, even if the DR serves more than one member, the source needs to put only the address of the DR in the list of destinations. This approach considerably reduces the Xcast packet header size and the overhead involved in processing the packet at intermediate nodes.

When applied to MANETs, these schemes are observed to create significant problems. In wired networks, members are typically located at the network edge which makes it relatively simple to locate an edge router and designate it as DR. However in a MANET, since member nodes can be located anywhere (not necessarily at the edge) and each node serves as a router as well, the task of selecting a DR becomes difficult. Besides, as the source has no knowledge of the members served by a DR, movement of a DR will very likely result in disruption of packet delivery to all group members served by this DR. Also, due to frequent change in network topology, it is hard to select a well-defined DR.

The proposed E2M protocol is designed by taking mobility into consideration and is particularly suited for MANETs. To explain the difference between E2M and the aforementioned schemes, please refer to Fig. 2. In E2M, the multicast source node A puts [N1: B, C, N6], while the list would be [N1: N3, N7, N8] for the schemes in [4], [14]. In Fig. 2 and for the case of E2M, only node N6 needs to be a XF as the number of nodes served by it is greater than a threshold (in this case, 3). The selection of XF is dynamic, helps in reducing the packet header size and reduces the processing overhead and delay. Another important advantage of E2M is that it minimizes the MEMBER_JOIN implosion problem at the source. In the basic Xcast scheme, the source node will be flooded with MEMBER_JOIN messages at periodic intervals due to member nodes trying to update their session membership. However, in E2M a XF combines all the MEMBER_JOIN messages it receives form its downstream member nodes and sends a single XF_JOIN message to the source. This reduces the control traffic overhead and also addresses the problem of "join implosion" at the source. E2M can work with only a few intermediate nodes which are E2M aware. In the following subsections we describe the E2M protocol in detail.

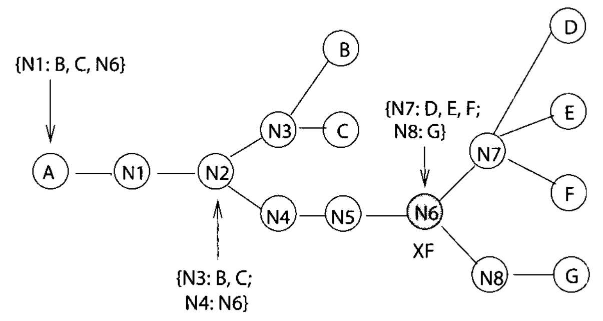


Fig. 2. E2M packet delivery.

B. Membership Management

Membership management in E2M is similar to the basic Xcast scheme. A source works as an admission controller and plays an important role in the membership management. The protocol proceeds independently for each source sending data to an Xcast session. E2M employs two types of packets: Control and data. Data packet is an Xcast packet with extended header. The extended header in the Xcast packet includes the list of destinations (members who do not belong to a XF) and the list of current XFs (Xcast forwarders). There are four types of control packets: MEMBER_JOIN, MEMBER_ACK, XF_JOIN, and XF_ACK.

If a node is interested in joining a particular Xcast session, it unicasts MEMBERS_JOIN message to the source of the session. The format of MEMBER_JOIN is as follows.

<sessionSource, sessionID, joinerID>

Underlying assumption here is that the node knows the session ID and the ID of the source. Also, the sessionSource and sessionID uniquely identify an Xcast session. The source, upon receiving a MEMBER_JOIN message, performs necessary checks to see if the member can join the multicast session. If the joiner passes the admission requirements, the source adds it to session membership table (SMT) and acknowledges the MEMBER_JOIN message by unicasting a MEMBER_ACK message to the joiner. The admission policies are beyond the scope of this paper.

A joiner after sending a MEMBER_JOIN message waits for JOIN_WAITING_PERIOD to receive a MEMBER_ACK from the source. If it does not receive a MEMBER_ACK during this period, it resends the MEMBER_JOIN message till a MEMBER_ACK is received or MAX_MEMEBER_JOIN_RETRY join messages have been sent, after which it concludes that the source is unreachable.

Due to dynamic nature of MANETs, node reachability may change unpredictably. Hence, the source needs to update its SMT from time to time. In E2M, the primary membership update mechanism is source initiated. Source initiated membership update is preferred over receiver initiated update because it eliminates synchronization issues [11]. Nodes could be joining the session on an ad-hoc basis at different time intervals. In order to send periodic refresh messages to the source they will have to synchronize their clock with the source if update is receiver initiated. However, if the update is source initiated, they do not have to keep track of when they should be sending their membership refresh message. Once every MEMBER_REFRESH_PERIOD, the source sets a POLL flag in the next outgoing Xcast data packet. Upon receiving such a data packet, all members need to send a MEMBER_JOIN message to the source and update their membership. If a source does not receive an update from any particular member, it marks the member inactive but does not remove it from the SMT. This is because the source has no knowledge to determine if the Xcast packet (with the set POLL flag) reached the member node, or if corresponding MEMBER_JOIN update itself was lost due to collision. Through our simulation work, we observed the occurrence of packet losses very frequently. Hence, after missing an update message from a member, the source waits for MAX_UPDATE_MISS times before deleting the member from the SMT. In addition, if a node detects its movement by missing MAX_SEQUENCE_GAP number of Xcast packets; it sends a MEMBER_JOIN message again to the source (Section VII). This is our secondary way of membership update. This helps the source to update its SMT, as well as the route to this member.

The initial packet forwarding in E2M works similar to the Xcast scheme by encoding the list of all possible destinations in the extended Xcast packet header. A XF_JOIN message is sent only when a node decides to become an XF for the ongoing Xcast session. XF_ACK is sent from the source in response to XF_JOIN, and confirms the originator of XF_JOIN that it has been added as a XF at the source. Following subsection will elaborate more on these two messages and the XF selection procedure.

C. XF Selection and Removal

Once a node starts receiving Xcast session packets, depending on the number of destinations it is serving and overhead involved in processing the Xcast header, it may decide to become an XF for its downstream group members. In the present version of E2M, we have used the number of destination IDs in the received Xcast packet header and the corresponding number of next hop branches to decide if a node should become an XF. If the number of destination IDs is more than a threshold (for example, it may be 9 if destination IDs are encoded in the options field of IPv4, so as to prevent multiple transmission of same packet) and if the node is a branching point it may decide to become an XF. If an intermediate node X decides to become an XF for a particular session, before sending a XF_JOIN message to the source it performs two more operations. First, to prevent multiple nodes trying to send XF_JOIN towards the source, each prospective XF sends the join message only after a specific delay whose duration is given by

- T = (ESTIMATED_NETWORK_RTT)/(HOP_DISTANCE),
- ESTIMATED_NETWORK_RTT = estimated round trip time of a packet to traverse the network width,
- HOP_DISTANCE = number of hops the node is away from the source of the session.

This delay partially ensures that XF_JOIN messages from downstream nodes will reach the session source before another upstream node along the same path tries to send another XF_JOIN. If an upstream node receives an XF_JOIN from any of its downstream nodes, it simply aborts its attempt to become an XF. Secondly, before sending a XF_JOIN, a node also ensures that it has received SEQ_GAP consecutive Xcast packets in which it is eligible to become an XF. In case of considerable changes in the downstream destination list during this interval, the node concludes that the network is too dynamic and aborts its attempt to become an XF for its downstream members. This will happen frequently in case node mobility is very high. As suggested in [11], in such scenarios, the ideal way to support Xcast is through flooding.

In case both the above criteria are satisfied, the node sends a XF_JOIN to the source which contains the present list of destination members it can serve. The format of XF_JOIN is as follows.

<sessionSource, sessionID, xfID, destination_list, incr_list, decr_list, path_list>

XF_JOIN message sent by a node X for the first time contains only the list of destination members (encoded in destination_list) and the incr_list and decr_list are both initialized to NULL. As the XF_JOIN message is forwarded to the session source, each intermediate node appends its ID to the path_list. The path_list thus records the complete route from the XF to the source. After receiving a XF_JOIN from a node X, the source checks to see if there is a XF present in the recorded path_list. If not, the source updates its Xcast forwarding table (XFT) by inserting the node X and the corresponding list of destinations served by node X (Fig. 3), and sends a XF_ACK to node X. From this point onwards, the source only specifies node X in its extended packet header instead of putting all the corresponding group members served by node X. From now on, extended header of all Xcast packets sent by the source will have the ID of the new XF in the list of XFs, and serves as an implicit acknowledgement of previous XF_JOIN message sent from node X.

In the present version of E2M, to keep things simple and avoid formation of hierarchy of XFs along a path, a node before sending a XF_JOIN also checks if there are no XFs already present in the destination list passing through it. This is possible since each Xcast data packet also contains the Xcast forwarder along this path. A check for the presence of XF ensures that there is no XF already present in the downstream path. The likelihood of such a scenario occurring though is minimal, because if a XF is already present in the downward path, the list of destinations passing through this node will likely be small, and this node will not meet the criteria for becoming a XF.

Similar to MEMBER_JOIN message used by session members to update their membership at the source, node X also takes care of sending periodic XF_JOIN refresh messages to the source so as to notify any change in the membership of its downstream destinations. Any change in the membership is conveyed through incr_list and decr_list. Through incr_list, node X tells the source about new members which are interested in the group or have just moved in under this XF, whereas decr_list is used to inform the source about the members which are no longer interested in the session or have moved out of this XF.

XFs also take care of handling the MEMBER_JOIN for their downstream members. As already mentioned, membership update is initiated by the session source. All members send a MEMBER_JOIN message to the session source when they receive an Xcast packet with the POLL flag set. When a XF re-

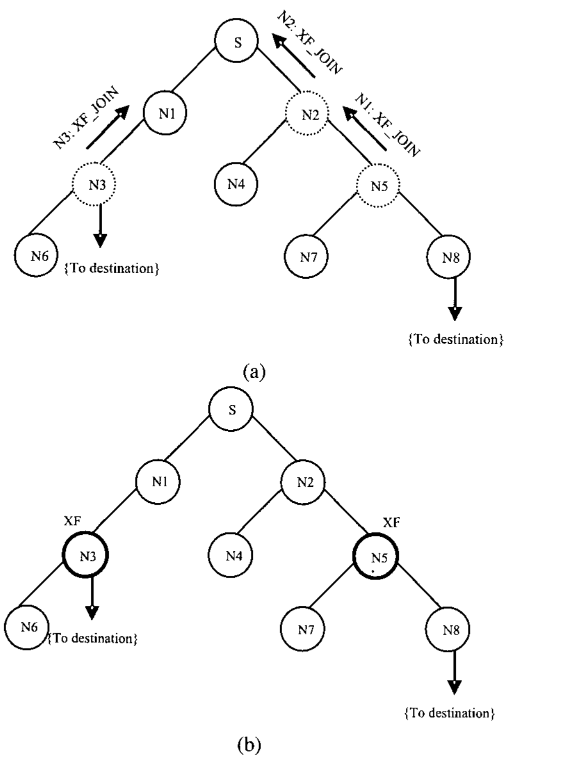


Fig. 3. XF-selection: (a) XF_JOIN propagation, (b) N3 and N5 as XF.

```
Procedure RECEIVE_MESSAGE_AT_SOURCE ()
Input: Mpkt // received packet
Begin
  If (MPkt->type == MEMBER_JOIN) Then
    If isNewMemberJoin(MPkt->source) Then
      Insert (MPkt->source) in SMT //new member has joined the session
    Elseif !isNewMemberJoin(MPkt->source) Then
      Update (MPkt->source) in SMT and XFT //received a member refresh
    Endif
  Elseif (Mpkt->type == XF_JOIN) Then
    If isNewXFJoin(Mpkt->source) Then
      If isXFPresentInPathList(Mpkt->path_list) Then
        removeXF (Mpkt->path_list->XFId)
        updateXFT(Mpkt->Source, NextHop, NumHops, Dest-List)
      Else //No XF is present in the path list
        updateXFT(Mpkt->Source, NextHop, NumHops, Dest-List)
      Endif
    Else //this XF is already there
      update(Mpkt->source) in XFT and SMT
    Endif
  Endif
FreePacket(Mpkt)
End
```

Fig. 4. Message handling at the source.

ceives a MEMBER_JOIN message from its downstream node, instead of forwarding it to the source, it updates its membership table. If the member does not belong to it, it adds the member to the incr_list. If it does not receive a MEMBER_JOIN from any of the members it is already serving, it adds the node to the decr_list assuming member has either moved out of this XF or is no longer part of the session. As a result, the source node gets only a single XF_JOIN for all the members currently served by a given XF, and also for all the members who do not belong to this XF but have moved under this XF due to their mobility. The source node, on receiving a refresh XF_JOIN, adds all the IDs in incr_list to the list of destinations being served by this XF. The list of destination IDs present in decr_list are removed from those being served by this XF and will be encoded explicitly in the packet header thereafter. However, if a node in the decr_list moved under another XF, it will be present in the incr_list of the new XF. The source will, accordingly, update the XFT and subsequently the new XF will take care of forwarding the Xcast packet to this node. It is to be noted that if a node no longer wants to serve as an XF, it simply needs to stop sending periodic XF_JOIN message to the source.

On the other hand, if the source discovers that there is a XF already present in the recorded path_list of the XF_JOIN message, it removes this XF from the list of Xcast forwarders. This is done not only to ensure that there is no hierarchy of XF's, but is also based on the assumption that the node which sent the new join message is more suitable to become a XF since it is located at greater hop distance from the source. XF removal at the source is therefore dynamic and is triggered if a new node along the same path and located at greater hop distance tries to become a XF. To remove a XF, the session source starts by putting the destination IDs of all the members served by this XF in regular Xcast packet and removes its entry from the XFT. From now on the Xcast data packets will not contain the ID of the removed XF in the XF header along the path. The removed XF, on receiving such an Xcast data packet, will know that it has been removed as a XF. It will then clear its local membership table and will thereafter forward Xcast packets simply based on the destinations encoded in the extended packet header. Please refer to Fig. 4 for the algorithm to select an XF at a source.

D. Packet Forwarding

In E2M, packet forwarding at any node is based on destination list received and whether the node is serving as an XF or is just a simple next hop node. Each Xcast packet is built with a payload and list of destinations grouped according to their next hops. The destination list explicitly specifies the node IDs (not under any XF) who should receive this Xcast packet and also includes the list of XFs along that path. When a node receives an Xcast data packet from an upstream neighbor, it first tries to locate its own destination list by looking at the next hop block. If no such block is present, the packet is simply discarded and forward processing stops (please refer to Fig. 5). After finding its own next hop block, the receiving node compares the sequence number of the data packet with its own local copy of the sequence number in its cache for this Xcast session. If this packet has been forwarded before, the message is discarded.

In case this is a new packet, the node checks if it is presently serving as a XF. If not, it checks if it should try to become a XF for the session. On finding itself eligible to become a XF (refer to Section IV-B), it creates a XF_JOIN message with the list of destination IDs in its member list, and sends it to the source of the session with incr_list and decr_list set to NULL. Till the time it receives an explicit XF_ACK from the source or receives an Xcast packet with its ID present in the XF list (implicit ACK), the node keeps on forwarding Xcast packets like a simple Xcast forwarding node. The node declares itself XF and activates its member list only when it receives an XF_ACK (explicit or implicit) from the source.

XF_JOIN messages might be lost due to packet colli-

```
Procedure RECEIVE_PACKET ()
Input: MPkt //received Packet
Begin
  If (MPkt->type == XCAST_DATA) Then
    If isInNextHopList(Mpkt) Then
      If (Mpkt->seq_no > Cache->seq_no) Then //received a new packet
        UpdateCachePktSequence(MPkt->seq_no)
        If !isXF(Mpkt->sessionID)
          If isXFSendEligible(Mpkt) Then
            sendXFJoin(Mpkt->source, dst_list)
            // Start a timer to resend the XF_JOIN if XF_ACK not received
          Endif
        Endif
        If isXF(Mpkt->SessionID) Then
          XF-DST-LIST = getMemberListofDst(SessionID)
          Mpkt-> Dst-list = Aggregate (XF-DST-LIST, Mpkt->Dst-list)
        Endif
        If isSetPOLL(Mpkt) Then
          If isMember() Then
            sendMemberJoin(Mpkt->source)
          Elseif isXF(Mpkt->SessionID)
            //start a timer to receive MEMBER_JOIN from downstream nodes
          Endif
        Endif
        getNextHop(Mpkt->Dst-List)
        Sort-Dst(NextHop, Mpkt->Dst-List)
        forward (MPkt)
      Endif
    Endif
  Endif
  If (MPkt->type == MEMBER_JOIN) Then
    If isXF(MPkt->sessionID) Then
      If isInMemberTable(Mpkt->senderID) Then
        updateMemberTable(Mpkt->senderID)
      Else
        addNodeToIncrList(Mpkt->incr_list)
      Endif
    Else
      forward (Mpkt)
    Endif
  Endif
  If (MPkt->type == XF_JOIN) Then
    If isSource(Mpkt->sessionID) Then
      RECEIVE_MESSAGE_AT_SOURCE(Mpkt)
    Else
      forward (Mpkt)
    Endif
  Endif
  FreePacket(Mpkt)
End
```

Fig. 5. Message handling at an intermediate node/XF.

sion at MAC layer and therefore, if the node does not receive an XF_ACK, it resends the XF_JOIN to the source till MAX_XF_JOIN_RETRY is reached, after which it concludes that either source is no longer reachable or the source has already designated another node as a XF along this path. In the latter case, this node should stop resending XF_JOIN.

In case the node is already an XF, it extracts the list of members currently served by it and adds them to the list of destinations already encoded in the extended packet header of the received Xcast packet. Then, it regroups the IDs according to their next hops and forwards the packet.

E. Data Structures

It is important to mention the data structures involved in implementing the E2M protocol, especially at the source node and at the XF. For a given session, a source maintains following tables.

- SMT (session membership table): Maintains the current membership for a given session.
- SFT (session forwarding table): Maps destinations to their next hop. Nodes with the same next hop are grouped together.
- XFT: A set of nodes working as a XF for the multicast group and their associated downstream session members.

A source node maintains a SMT for each multicast session it is working as a source. The XFT stores, for a given session, the mapping of XFs to their corresponding list of destination addresses. Also, SFT maps destination nodes to their respective next hops. The SFT is further divided into smaller subsets which groups the set of destinations whose next hop is the same. If an intermediate node wants to serve as a XF, it needs to maintain both a SMT and a SFT. As mentioned earlier in our discussion, a XF need not maintain a XFT if we assume group membership not to be very large.

F. Handling Node Movement

MANET is a dynamically reconfigurable wireless network where nodes are mobile resulting in variable network topology. Thus, the Xcast delivery in MANETs is far more challenging than its wired counter part. Fine-tuned for wired networks, the basic Xcast scheme seems to be immune to change in network topology by having the source node explicitly mention the list of destinations in the extended header, while the routing protocol is responsible to deliver the packets to the required destinations. There may be issues regarding packet delivery in MANETs and an interesting scenario comes into picture when the XF or one of its members moves out. Movement of a member node that does not belong to any XF is handled by the underlying routing protocol, since the ID of this node would always be explicitly encoded in the packet header.

In case of XF movement, as soon as the source comes to know about it (e.g., route error packet generation in AODV and DSR, or the neighbor discovery phase through hello packets), the source node obtains the list of destinations served by this XF from its XFT and includes this list of destinations explicitly in its extended packet header for all Xcast packets thereafter. The source node also removes the corresponding XF entry from it XFT. The movement of a member node is also transparent and is handled through MEMBER_JOIN message sent to the source. For example, if a node X moves to a new point of attachment where it has a different path to the source than its previous path, its periodic MEMBER_JOIN will either directly reach the source or will pass through an existing XF in that area. If the source receives this join message directly, it adds the address of node X in subsequent extended packet headers explicitly and also updates the XFT accordingly. As for the old XF of node X, it will come to know about node X's movement when it fails to receive node X's periodic MEMBER_JOIN. As a consequence, in the next XF_JOIN to the source, the XF will mention this in its decr_list in its packet. However, it may also happen that MEMBER_JOIN message from node X passes through any of the serving XF. In such a scenario, the new XF will add the node to its local SMT and then update the session source about this change through its XF_JOIN incr_list.

V. PERFORMANCE EVALUATION

A. Simulation Environment

We have implemented Xcast module in NS (version 2.26) [15]. The simulation environment models a MANET comprised of 75 mobile nodes. We have simulated Xcast session for group membership size of 10, 20, 30, 40, and 50 nodes. At the beginning of simulation, 75 nodes are randomly placed in a grid of size 1000 m by 1000 m. When the simulation starts, each node randomly picks a destination and moves towards it with a random constant speed ranging from 2 m/s to 10 m/s. We vary the mobility with different pause times as 0, 150, 300, 400, and 500 seconds. After a node reaches its destination, it randomly picks another destination and starts moving towards this new direction with a newly selected random speed. The network stack of each mobile node consists of link layer, an ARP module, interface priority queue, IEEE 802.11 MAC layer with 250 meters transmission range, and a network interface. Table 1 outlines different simulation parameters used in the simulation. For radio propagation model, our simulation uses free space model for distance less than cross-over distance [15] (also known as Fresnel Breakpoint). For distance larger than cross over distance, simulation uses two ray ground propagation model. In NS, the value of cross-over distance is approximately 80 meters.

For the simulations that follow, we have considered CBR traffic with payload size set to 512 bytes. Data packets are generated at the source at a rate of 2 packets per second. Each simulation runs for 500 seconds and all results are averaged over 10 different seeds. There is no network partition during the course of simulation. We have compared the performance of E2M with basic DDM (with no caching mode) scheme and flooding. Our simulation is basically divided into two groups. In the first group of simulations, we vary the group size with no mobility, whereas in the second group of simulation we keep group size fixed to 40 nodes and vary the node mobility. This is done to clearly identify the impact of changing group size and mobility on different protocols.

B. Protocols

Both E2M and DDM run over the MANET unicast routing protocol AODV (ad hoc on-demand distance vector) [16]. NS already provides the AODV routing agent, while we had to modify the AODV agent to support Xcast over it. However, it should be noted that the E2M is independent of underlying routing protocol and it can run over any of MANET routing protocols. Only reason of choosing AODV is its ease of implementation in NS. To facilitate the interaction between the Xcast agent and the AODV agent, we have adopted a cross-layer design approach, wherein an Xcast agent can access the AODV routing table to group the nodes which are reachable through the same next hop. In case a route is not available, the Xcast agent simply sends a unicast packet to the AODV agent, which triggers a route request for this destination. When the AODV agent receives a route reply from the required destination, it updates its route table accordingly. An Xcast agent can now directly obtain the next hop information from the routing layer.

Table 1 outlines some of the AODV parameters selected for our simulation work. We have enabled the gratuitous mode of AODV. This is because in AODV when a node receives a route request and responds with a route reply, it does not forward the route request any further. If all route requests generated by the source are replied to by intermediate nodes, the destination member does not receive any copies of the route request. Hence, it does not learn of a route to the source node. This may create problem for the case when a member node is trying to send a periodic MEMBERSHIP_JOIN update towards the source, and it may force a member to do a route request for the source. By enabling gratuitous mode, intermediate nodes after receiving a route request and responding with a route reply (incase it has a route), also unicasts a gratuitous route reply to the destination member. This helps in reducing the route request generated by different session members.

Table 1. Simulation parameters.

	Parameters	Value
Simulation	Number of nodes	75
	Packet size	512 bytes
	Simulation time	500 s
	X-dimension of motion	1000 m
	Y-dimension of motion	1000 m
	Transmission range	250 m
	Bandwidth	1 Mbps
	Node placement	Random
	Radio propagation model	TwoRayGround
	MAC protocol	IEEE 802.11
	Transport protocol	CBR
XCAST	MEMBER_JOIN_RATE	50 s
	XF_JOIN_RATE	50 s
	XF_SELECTION_THRESH	9
	MAX_XF_JOIN_RETRY	3
	MAX_UPDATE_MISS	3
	MAX_SEQUENCE_GAP	3
	ESTIMATED_NETWORK_RTT	2 s
AODV	ACTIVE_ROUTE_TIMEOUT	10 s
	HELLO_INTERVAL	1 s
	GRATUITOUS MODE	ON

C. Performance Metrics

The following metrics are used to compare the performance of different schemes.

- Average Xcast packet header size: Average size of the Xcast packet header needed to include the list of all destinations. This metric does not include payload size.
- Data packet delivery ratio: Ratio of the number of data packets actually received by group members to the number of data packets which should have been received.
- Xcast routing protocol overhead: It includes
  - relative control message overhead: Control overhead involved in supporting different Xcast schemes,
  - relative control byte overhead: Control byte overhead involved in supporting Xcast. This includes both the Xcast header as well as Xcast control message overhead.
- Forwarding efficiency: This is a measure of the number of data packets transmitted per data packet delivered.
- Member refresh overhead at source: Ratio of the number of join refresh messages received at the sources to the number of Xcast packets transmitted by it.

D. Simulation Results

D.1 Average Xcast Packet Header Size

One of the main points of interest of our simulation work was to calculate the average Xcast packet header size transmitted in the network. It refers to the extended header needed to explicitly encode the address (4 bytes each) of group members in the Xcast packet. The significance of this result is that as the value of header size increases, there is higher number of destination IDs mentioned in the Xcast packet, which increases the time needed to process the packet at intermediate nodes. Fig. 6 compares the average packet header size (in bytes) for E2M and DDM schemes. For small group size, E2M behaves similar to DDM scheme since there are no XFs selected. As the group size increases (Fig. 6(a)), average Xcast packet header size increases for both DDM and E2M. However, in E2M, the selection of XF helps in reducing the list of destinations IDs needed to be put in the extended header and hence it outperforms DDM. Interestingly, with the introduction of mobility, the average packet size for both E2M and DDM goes down (Fig. 6(b)). This is because of increased number of miss of membership update from the member nodes due to node movement, which makes source to put less number of destinations in the packet header. But in this case also E2M outperforms DDM. For Flooding, the size of Xcast extended header is always zero.

D.2 Data Packet Delivery Ratio

From Fig. 7(a), we can see that the packet delivery ratio of flooding is good for both varying group size as well as mobility. This is because of the redundant packet transmission involved in flooding. The packet delivery ratio for E2M and DDM is comparable to flooding in Fig. 7(a), but the delivery ratio decreases rapidly with the increase in node mobility (Fig. 7(b)). There are several factors which contribute to the poor performance of basic DDM and E2M when mobility is introduced. Some of them include packet collision at MAC layer, since packets are transmitted without RTS/CTS and increased routing overhead due to changing network topology. We have elaborated more on these issues in Section VII. However, it is worthwhile to note that, as compared DDM, E2M has a comparable packet delivery ratio even with reduced extended header.

D.3 Xcast Routing Protocol Overhead

We have studied the Xcast routing protocol overhead involved in supporting Xcast. It is to be noted that this is the pure control traffic generated by an Xcast agent, and we have not considered the control overhead introduced because of the routing layer. This study is performed in two perspectives, control packet overhead which is the control packets generated by an Xcast layer to maintain the group membership (MEMBER_JOIN, MEMBER_ACK, XF_JOIN, XF_ACK), and control byte overhead which, in addition to above messages, also includes the number of bytes used to mention the list of destinations.

Although both of the above metrics contribute to the network overhead, the former is especially significant since the cost of accessing wireless link is very high. For flooding, the control message overhead is clearly zero.

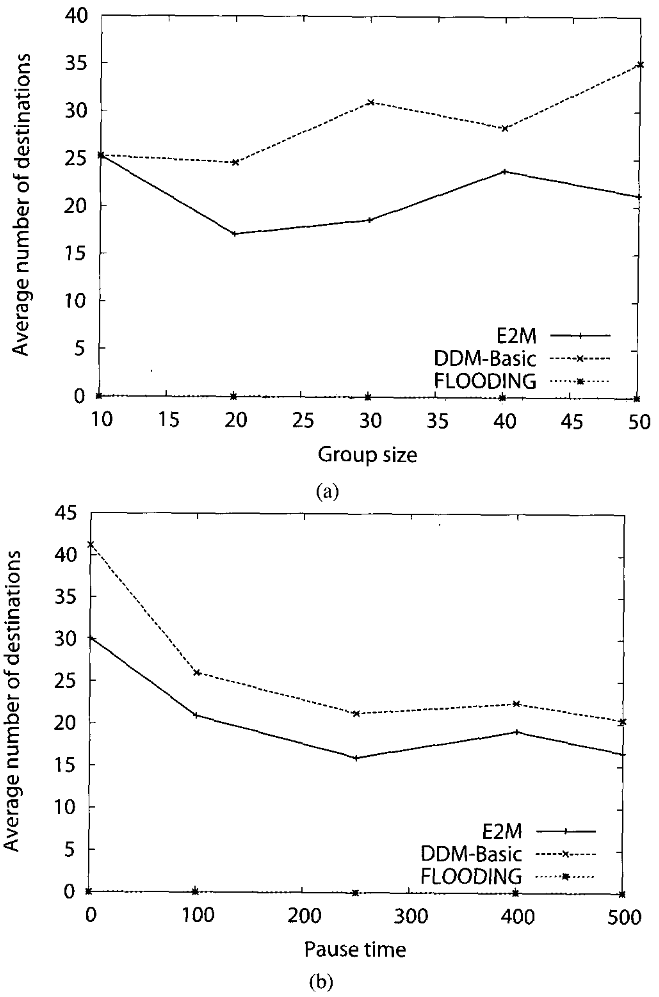


Fig. 6. Average Xcast packet header size for group size and pause time: (a) Varying group size, (b) varying pause time.

- Xcast control packet overhead: It is defined as the ratio of number of control packets transmitted to the number of data packets sent from the source. For DDM, it includes MEMBER_JOIN and MEMBER_ACK packets, whereas for E2M it includes MEMBER_JOIN, MEMBER_ACK, XF_JOIN, and XF_ACK. These join refresh messages are sent periodically at MEMBER_JOIN_RATE and XF_JOIN_RATE. These are pure control packets and do not carry any user payload. Fig. 8 gives the control packet overhead associated with each of the Xcast schemes. In DDM, the membership update messages are sent at periodic interval of time to the source. This messages travel all the way to the source. On the other hand, in E2M, some of the periodic membership update messages are handled by XF itself and hence the total number of control traffic injected into the network goes down.
- Xcast control byte overhead: Another interesting metric to compare the performance of the Xcast schemes is to analyze the control bytes transmitted by each of them. This includes the bytes of control packets, and embedded control information in data packets. Fig. 9 gives the ratio of control bytes transmitted to the number of data bytes sent from the source. As before, flooding does not have any control information. On the other hand, the majority of control bytes in DDM and

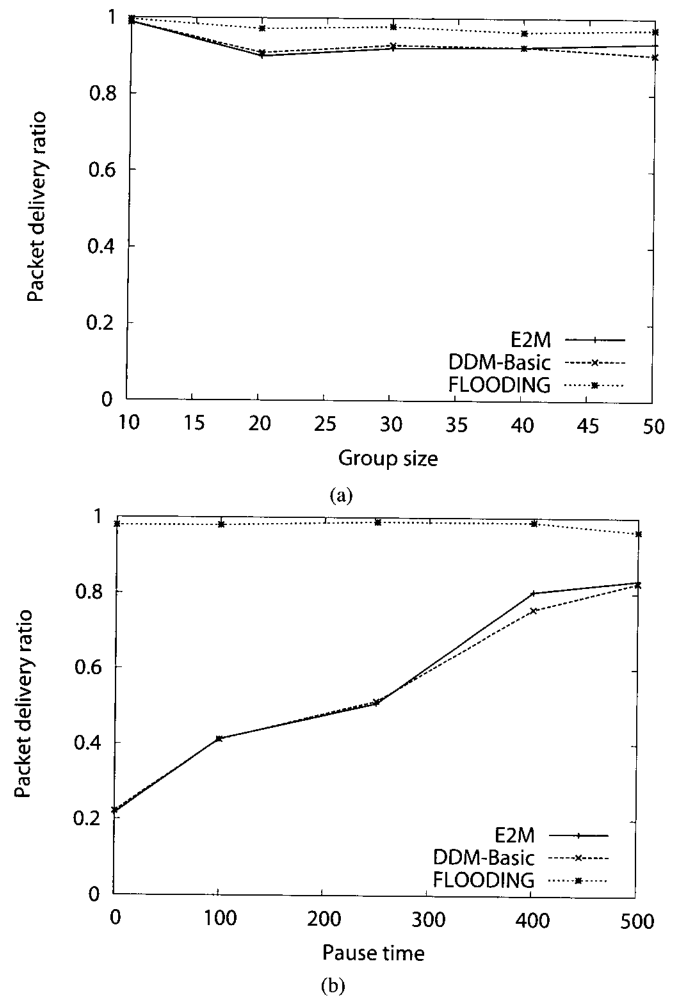


Fig. 7. Average Xcast packet header size for group size and pause time: (a) Varying group size, (b) varying pause time.

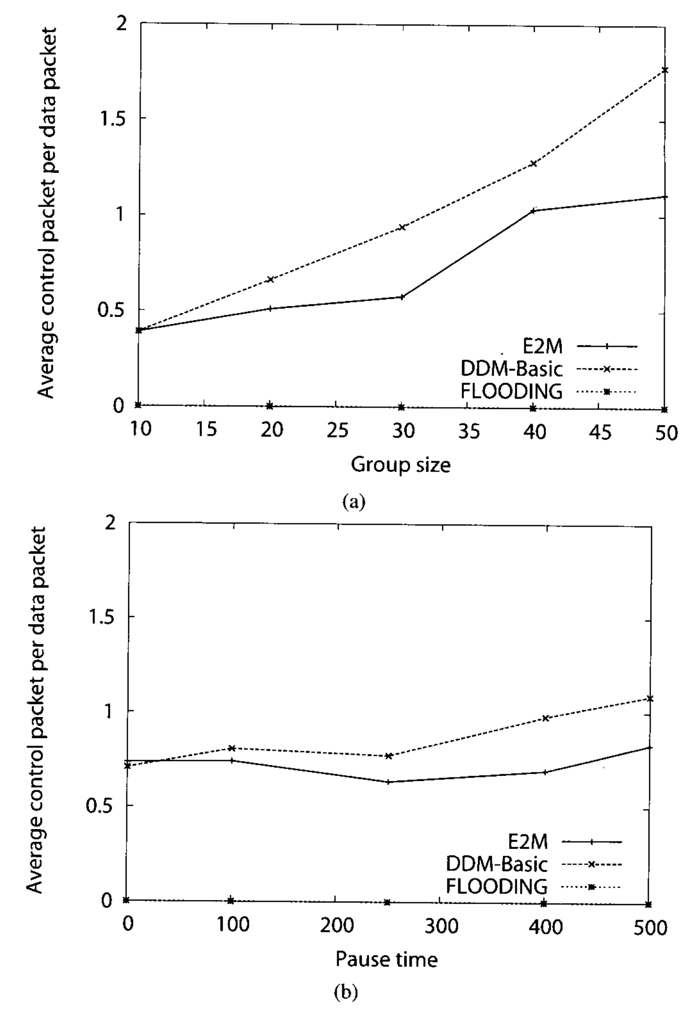


Fig. 8. Average Xcast packet header size for group size and pause time: (a) Varying group size, (b) varying pause time.

E2M are due to the extended header. E2M performs better than DDM because of lower number of refresh messages injected into the network. However, it is to be noted that the performance of DDM and E2M is same for 10 members list, as there is no XF selected and E2M behaves like the basic Xcast scheme.

D.4 Forwarding Efficiency

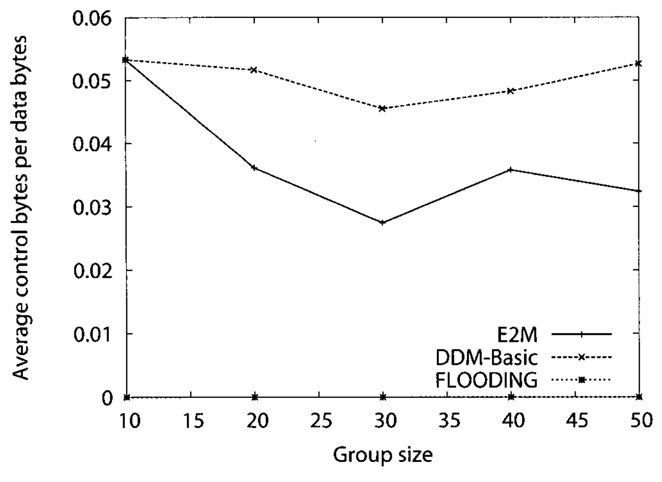
A higher value of forwarding efficiency indicates a less efficient protocol, as it needs a higher number of transmissions to reach the same number of destinations. This is also a good measure of bandwidth efficiency of a protocol, as the size of data packets is much larger than control packets. The respective plots for this metric are shown in Fig. 10.

The data forwarding efficiency of flooding is very poor as compared to E2M and DDM. This is because of the redundant transmissions involved in forwarding the Xcast packet. The forwarding efficiency of flooding improves as the number of group members increases since more and more nodes are included in the forwarding topology, whose size does not change. Both E2M and DDM are relatively immune to the group size change. Interestingly, the forwarding efficiency improves slightly with increase in group size for both of them, since it increases the chances of path overlap between source and destination. The forwarding efficiency of E2M is slightly better than DDM in mobile scenarios. However, the increased node speed has a little impact on forwarding efficiency, this is mainly because of the presence of hidden terminal problem. A similar result has also being reported in [11].

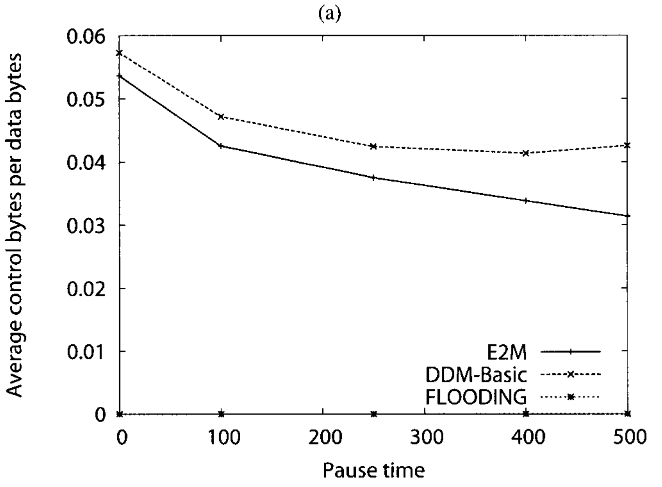
D.5 Member Refresh Overhead at Source

Members refresh overhead at the source measures the number of member join packets received at the source per data packet transmitted by it. This is an important metric as it indicates the extra load generated on the source because of member refresh. A higher number of join refresh may cause congestion at the source, and hence may result into more collisions. It is to be noted that a packet dropped near the source has greater negative impact on delivery ratio than a packet dropped farther away. All these factors reduce the data forwarding efficiency of a protocol. In short, it is necessary to reduce the traffic flowing towards the source. Fig. 11 gives the member refresh overhead at the source. For DDM, it includes MEMBER_JOIN, whereas for E2M it includes both MEMBER_JOIN and XF_JOIN.

Obviously, there is no member refresh overhead in flooding. For DDM, this overhead increases with the increase in group membership, as more and more nodes will send join messages towards the source. In E2M, interception and aggregation of

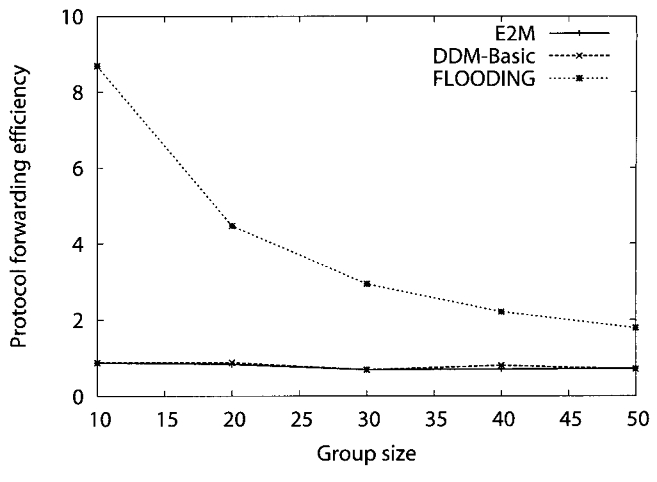


(a)

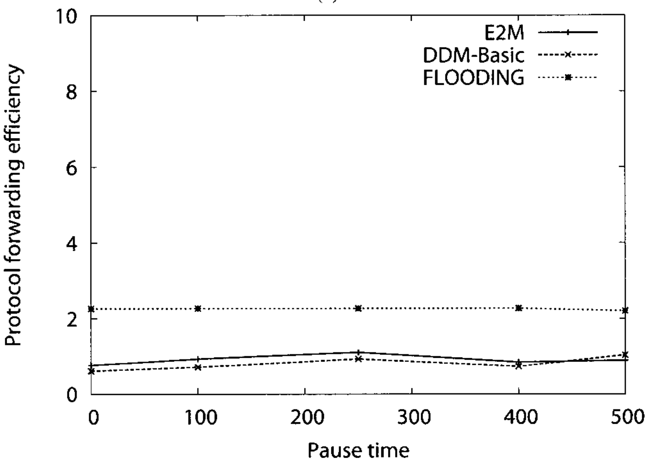


(b)

Fig. 9. Average Xcast packet header size for group size and pause time: (a) Varying group size, (b) varying pause time.



(a)



(b)

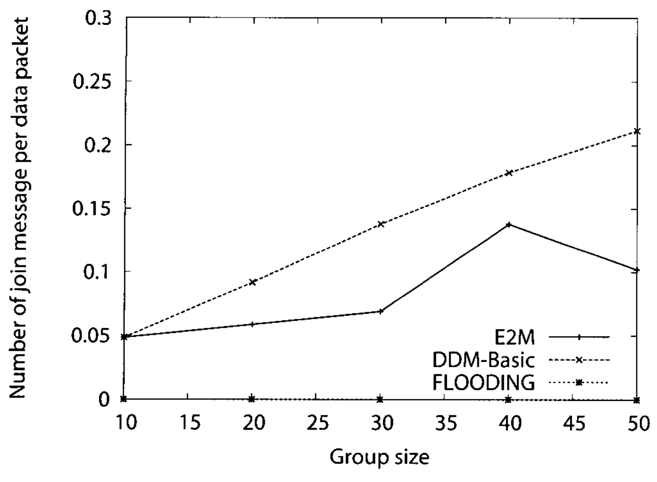
Fig. 10. Average Xcast packet header size for group size and pause time: (a) Varying group size, (b) varying pause time.

MEMBER_JOINs at an XF, and transmission of a cumulative XF_JOIN for all the members served by the XF helps in reducing the number of refresh messages going towards the source. In Fig. 11(a), a drop in "join implosion" above 40 users is mainly contributed by the selection of additional XF for the same session. However, as shown in the figure, the "join implosion" problem becomes severe for DDM as the group size increases.

VI. E2M FOR IP NETWORKS

One of the primary motivations in designing the E2M protocol was to come up with an algorithm, which could be extended to an IP network. To do so, one has to consider how one can implement Xcast in the IP protocol stack. Do we need to have a new header for Xcast packet (like DDM) on top of IP or we should extend the IP header to support an Xcast session. In the later case, the option header (IPv4) or next header (IPv6) can be used to encode the list of destination for an Xcast session, together with a flag to identify if it is an Xcast packet. As discussed before, this approach may result into duplicate transmission of Xcast packet if the number of users passing through a next hop is more than what extended header can support. This is true for both infrastructure and infrastructureless network as long as we are extending IP to support basic Xcast scheme.

The initial implementation of E2M in MANETs follows the scheme similar to DDM to support Xcast through extended header. This form of Xcast implementation prevents the duplicate transmission of the same packet to reach different next hops. But since the packets are transmitted through one hop broadcast, and due to MAC layer collision, packet delivery ratio is reduced significantly. It is worthwhile to mention here that E2M can also work in IP-mode, wherein a node groups list of all destinations served by a particular next hop and unicast the packet to this particular next hop. Though this can lead to duplicate transmissions, this scheme has the advantage that the MAC layer (802.11) will deliver the packet, through RTS and CTS, and hence will have far better packet delivery ratio than trying to send it through one hop broadcast. We consider this approach as our future research work. In the next section, we explain some of the reasons behind the poor delivery ratio in the broadcast approach. This observation is based on our simulation work and we try to point out some of the MAC layer and routing layer related issues when we try to provide Xcast in a MANET scenario.



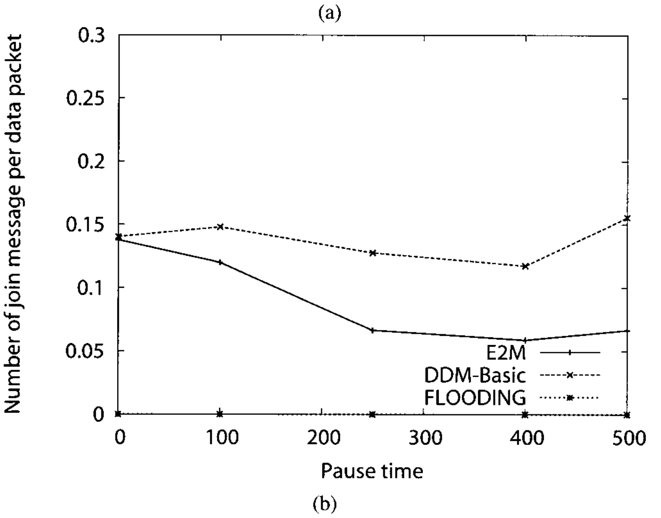


Fig. 11. Average Xcast packet header size for group size and pause time: (a) Varying group size, (b) varying pause time.

VII. XCAST IN MANET: ROUTING & MAC LAYER ISSEUS

As discussed before, present schemes to implement Xcast over MANETs try to utilize the broadcast property of the network to forward an Xcast packet. This is carried out by making the destination of the packet as broadcast and making the time-to-live (TTL) field to one, so that the packet should not be propagated to the whole network. When MAC layer (IEEE 802.11) receives such a packet, it treats the packet as a simple broadcast and sends it without RTS, CTS, and ACK. So, there is no mechanism employed by the MAC layer to tackle the hidden terminal problem. For example, in the following Fig. 12, if both nodes N1 and N3 try to forward the Xcast packet at the same time, there will be collision at N2, and hence packets will be lost. One way to minimize this effect is it to introduce a jitter while forwarding a multicast packet, but as found in our simulation study, its effect on minimizing the packet collision is less than desired. We have observed by simulation that this effect is much more severe as the rate of packet generation increases.

The packet loss due to MAC layer collision has a cascading effect on the performance of the routing layer, and hence on Xcast which is closely dependent on it. First of all, due to adopting a broadcast approach for packet forwarding, a node has no

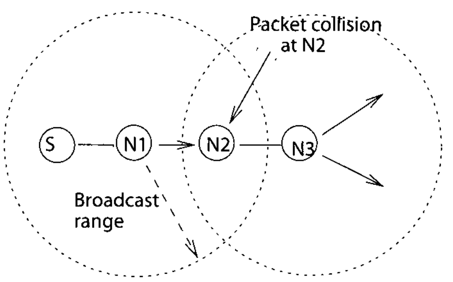


Fig. 12. Hidden terminal.

way of knowing if a particular next hop has received any packet. In addition to this, if a particular node which was serving as a next hop during an Xcast session moves out, the routing layer will not be aware of this movement for some time. This is because the MAC layer will not trigger any link failure to the routing layer, which, in turn will not send any route error towards the source. Although nodes will eventually learn about node movement of their neighbors through the neighbor discovery phase of some on demand protocols (hello packet in AODV), but neighbor discovery is performed only at periodic intervals and there is a time gap between them. This implies that there is always a potential problem of packet loss. In short, the routing layer will not come to know about the movement of the node till the next hello phase. The other way for a source to know about the possible movement is through a MEMBER_JOIN message sent from a particular member. But one has to keep in mind that this MEMBER_JOIN message is also sent only at regular intervals. A source initiates a route request for a node only when it misses the MEMBER_JOIN message from this node.

It is to be noted that a route request is broadcasted throughout the network, and hence generates a lot of overhead [17]. In MANETs, this becomes severe if the nodes are mobile and a source, after missing a MEMBER_JOIN, does a route request for most of them at one time after MEMBER_REFRESH_PERIOD. To prevent this from happening, in E2M, if a member misses MAX_SEQUENCE_GAP number of packets from the source, rather than waiting for the source to time out, it again sends a MEMBER_JOIN message to the source. This helps not only in minimizing the time between a node moving out and the source coming to know about it, but also prevents the source from doing multiple simultaneous route requests.

Another important observation in implementing E2M is the expiry of a route entry for destinations served by a particular XF at a forwarding node located between the source and the XF. This may happen since a source, after selecting a XF puts only XF's address in the destination list field. For AODV (or any on-demand protocol), which initially performed a route request for all the member nodes, the route entries for nodes which have moved under a XF are no longer updated, and only entry which is updated in the routing cache of intermediate nodes is that of the new XF. So, after ACTIVE_ROUTE_TIMEOUT, the source and the intermediate nodes will remove the entries for all such member nodes. Now, in E2M, if a periodic XF_JOIN refresh message is lost, the source will simply remove this XF from its XFT and encode the list of destination nodes served by this

XF explicitly in all its future packet headers. Since the route for this packet is already expired, it will trigger a route request for all these destinations at the source. To prevent this from happening, we have used XF_ACK for periodic XF_JOIN sent from an XF. If an XF misses an ACK, it resends the XF_JOIN to the source till MAX_XF_JOIN_RETRY is reached, when it concludes that the source is no longer reachable. Other option is to put the complete list of destinations in the packet header (similar to basic DDM scheme) at least once in each ACTIVE_ROUTE_TIMEOUT period. But, as observed through simulation, the chance of this packet being lost near the source is very high. Also, periodic mentioning of complete destination list will increase the Xcast header size.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we propose a new scheme for small group multicast in MANETs named extended explicit multicast (E2M), which is built on top of explicit multicast (Xcast) and makes it scalable with the number of group members for a given multicast session. E2M is based upon the novel concept of dynamic selection of Xcast forwarders (XFs) between a source and possible destinations. Unlike other schemes, E2M does not make any assumptions related the to network topology or node location, which is important for MANETs given its rapid change in topology. The selection of XFs is done based on group membership and processing overhead involved in supporting Xcast at a given node. This scheme can work with only a few E2M aware nodes in the network and provides the transparency of stateless multicast, reduces header processing overhead, minimizes control traffic, addresses the "join implosion" problem at source, and makes Xcast scalable with the number of session members without compromising the throughput. Through our extensive performance evaluation, we have observed that E2M performs better than basic Xcast scheme and existing schemes such as DDM.

As future work, we plan to investigate the issue of multiple sources for the same multicast session. Also, we intend to look into the issue of hierarchical XFs wherein a XF can itself be a parent of some other XF. Support of reliable Xcast in MANETs will also be considered.

ACKNOWLEDGEMENT

This work has been supported by the Ohio Board of Regents Doctoral Enhancement Funds and the National Science Foundation under grant CCR-113361. We would like to thank the reviewers for their valuable comments. We also like to thank Qihe Wang who reviewed the final version of this paper.

REFERENCES

[1] H. Gossain, C. M. Cordeiro, K. Anand, and D. P. Agrawal, "E2M: A scalable explicit multicast protocol for manets," in *Proc. IEEE ICC 2004*, France, June 2004.

[2] H. Gossain, C. M. Cordeiro, and D. P. Agrawal, "Multicast: Wired to wireless," *IEEE Commun. Mag.*, vol. 40, no. 6, pp. 116–123, June 2002.

[3] R. Boivie, "A new multicast scheme for small groups," IBM Research Tech., Rep. RC21512 (97046), June 1999.

[4] D. Ooms and W. Livens, "Connectionless multicast," *IETF Internet Draft*, draft-ooms-cl-multicast-01.txt, work in progress, Oct. 1999.

[5] C. M. Cordeiro, H. Gossain, and D. P. Agrawal, "Multicast over wireless mobile ad hoc networks: Present and future directions," *IEEE Network, Special Issue on Multicasting: An Enabling Technology*, vol. 17, no. 1, Jan./Feb. 2003.

[6] S. J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A performance comparison study of ad hoc wireless multicast protocols," in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000.

[7] J. J. Garcia-Luna-Aceves and E. L. Madruga, "The core-assisted mesh protocol," *IEEE J. Select. Areas Commun.*, pp. 1380–1394, Aug. 1999.

[8] M. Gerla, S.-J. Lee, and W. Su, "On-demand multicast routing protocol (ODMRP) for ad hoc networks," *IETF Internet Draft*, draft-ietf-manet-odmrp-02.txt, work in progress, 2000.

[9] E. M. Royer and C. E. Perkins, "Multicast operation of the ad hoc on-demand distance vector routing protocol," in *Proc. ACM MOBICOM'99*, Aug. 1999, pp. 207–218.

[10] L. Ji and M. S. Corson, "A lightweight adaptive multicast algorithm," in *Proc. IEEE GLOBECOM'98*, 1998, pp. 1036–1042.

[11] L. Ji and M. S. Corson, "Differential destination multicast—a MANET multicast routing protocol for small groups," in *Proc. IEEE INFOCOM 2001*, 2001, pp. 1192-1202.

[12] K. Chen and K. Nahrstedt, "Effective location-guided tree construction algorithms for small group multicast in MANET," in *Proc. IEEE INFOCOM 2002*, 2002, pp. 1180–1189.

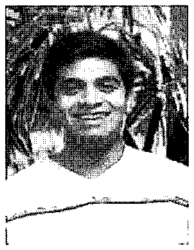
[13] J. Luo, P. T. Eugster, and J.-P. Hubaux, "Route driven gossip: Probabilistic reliable multicast in ad hoc networks," in *Proc. IEEE INFOCOM 2003*, 2003.

[14] M.-K. Shin, Y.-J Kim, K.-S. Park, and S.-H Kim, "Explicit multicast extension (Xcast+) for efficient multicast packet delivery," *ETRI J.*, vol. 23, no. 4, Dec. 2001.

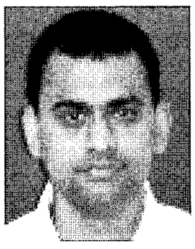
[15] NS-2 network simulator (online), available at http://www.isi.edu/nsnam /ns/index.html.

[16] C. E. Perkins, E. M. Belding-Royer, and S. Das, "Ad hoc on demand distance vector (AODV) routing," *IETF Internet Draft*, draft-ietf-manet-aodv-12.txt, work in progress, Nov. 2002.

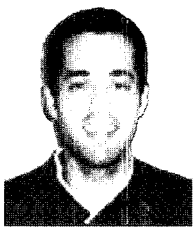
[17] Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc networks," in *Proc. ACM/IEEE Int. Conf. Mobile Computing Networking*, Aug, 1999, pp. 152–162.



Hrishikesh Gossain is a Senior Systems Engineer in Mesh Products Development Group in Motorola Inc. He received his M.S. and Ph.D. from the Department of ECECS, University of Cincinnati and B.E. in Electronics Engineering from Motilal Nehru Regional Engineering College, India, where he was undergraduate Gold-Medalist of the College. He has several approved and pending patents in the areas of wireless and mobile computing, access network design, QoS, and e-media. He has previous work experience in Nortel Networks in Richardson, Texas and Center for Development of Telematics (C-DoT), India.



Kumar Anand is an Engineer in Corporate R&D group at Qualcomm, San Diego, CA. He received his Bachelors degree in Electrical Engineering from University of Roorkee, India and completed his Masters in Computer Engineering from University of Cincinnati, Ohio. His interest areas are wireless cellular systems, multicasting in mobile ad-hoc networks, IEEE 802.11, TCP/IP, protocol design for sensor networks, and embedded systems development.



Carlos Cordeiro is a Senior Member Research Staff of Philips Research USA, Briarcliff Manor, NY. Before joining Philips Research, he was a Senior Research Engineer at Nokia Research Center. In his current capacity at Philips Research, Dr. Cordeiro is involved with research of PHY and MAC aspects in the area of cognitive radios. He actively participates in the IEEE 802.22 standardization effort, and amongst other responsibilities serves as the Chair of the MAC subcommittee in IEEE 802.22. Dr. Cordeiro received his Ph.D. in computer science and engineering in 2003 from the University of Cincinnati, OH, USA, where he won the honorable Outstanding Doctoral Dissertation Award and the prestigious 2003/2004 National Dean's List Award. He is also listed in the 2005 Edition of Marquis Who's Who in America. His research interests include MAC protocol analysis and design, IEEE 802.11/15/16/22, cognitive radios, power control, and ad hoc and sensor networks. Dr. Cordeiro has served as TPC member of various meetings, has published numerous papers in the wireless area, and in the past was the recipient of best paper awards from refereed networking conferences.

Dharma P. Agrawal is the Ohio Board of Regents Distinguished Professor of Computer Science and Computer Engineering and the founding director for the OBR Research Center for Distributed and Mobile Computing in the Department of Electrical & Computer Engineering and Computer Science, University of Cincinnati, OH. His current research interests includs the various aspects of wireless and mobile networks. Dr. Agrawal is an editor for the Journal of Parallel and Distributed Systems and the International Journal of High Speed Computing. He has served as an editor of the IEEE Computer Magazine, IEEE Transactions on Computers, International Journal on Distributed Sensor Networks, International Journal of Ad Hoc and Ubiquitous Computing, and International Journal of Ad Hoc and Sensor Networks. His recent co-authored book on "Introduction to Wireless and Mobile Systems" has been adopted worldwide. He has been the Program Chair and General Chair for numerous international conferences and meetings. He was selected for the "Third Millennium Medal" by the IEEE for his outstanding contributions. Four of his patents in wireless networking area have also been approved recently. Dr. Agrawal is a fellow of the IEEE, ACM, and AAAS.