

**LOCALIZED TREE CHANGE MULTICAST
PROTOCOL FOR MOBILE AD HOC NETWORKS**

A PROJECT REPORT

Submitted by

G SRIKRISHNAN

SUDHARSAN RANGARAJAN

S VIJAY

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

SRI VENKATESWARA COLLEGE OF ENGINEERING,

PENNALUR

ANNA UNIVERSITY :: 600025

MAY 2005

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report “LOCALIZED TREE CHANGE MULTICAST PROTOCOL FOR MOBILE AD HOC NETWORKS” is the bonafide work of G Srikrishnan, Sudharsan Rangarajan and S Vijay, who carried out the project under my supervision.

R Ramachandran

HEAD OF THE DEPARTMENT

Department of computer science and
engineering

Sri Venkateswara College of engineering

Pennalur

Sriperumbudur – 602105

Tamil Nadu, India

T Srinivasan

SUPERVISOR

Assistant Professor

Department of computer science and
engineering

Sri Venkateswara College of engineering

Pennalur

Sriperumbudur – 602105

Tamil Nadu, India

ABSTRACT

Multicasting efficiently supports a wide variety of applications characterized by a close degree of collaboration, typical of many ad-hoc applications currently envisioned. Within wireline networks, well-established routing protocols exist to offer an efficient multicasting service. As nodes become increasingly mobile, these protocols need to evolve to similarly provide an efficient service in the new environment. In our project, we present a new source based multicast routing protocol - The Localized Tree Change Multicast Protocol (LTMP). We compare the performance of the protocol with existing source based multicast schemes: The Forwarding Group Multicast Protocol (FGMP), Differential Destination Multicast (DDM). We also propose a variant of FGMP named Reverse FGMP, which reduces the control overhead of FGMP. The protocols differ in the nature of the control traffic generated to maintain the multicast tree and the state information maintained for each multicast group. Our results show that in many scenarios LTMP offers better performance than the existing schemes; a consequence of the reduced control overhead of our protocol and its ability to perform localized tree changes.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF SYMBOLS	
1	INTRODUCTION TO MOBILE AD HOC NETWORKS	1
	1.1 MOBILE AD HOC NETWORKS	1
	1.2 THE MULTICAST CAPABILITY	2
	1.3 MULTICAST ROUTING ISSUES FOR MANETS – PROBLEMS WITH THE DVMRP APPROACH	4
2	EXISTING MULTICAST PROTOCOLS	6
	2.1 FGMP AND RFGMP	6
	2.1.1 FG maintenance using receiver advertising	8
	2.1.2 Reverse FGMP	15
	2.2 DDM	19
	2.2.1 Soft State Operation	25
	2.3 CONSTRAINED FLOODING	32
	2.4 LOOP FREEDOM OF A ROUTING PROTOCOL	34
3	LOCALIZED TREE CHANGE MULTICAST PROTOCOL	36
	3.1 THE LTMP PROTOCOL DESIGN	38

	3.2 LOOP FREEDOM OF THE LTMP PROTOCOL	45
4	PERFORMANCE ANALYSIS	47
	4.1 MULTICAST DELIVERY	47
	4.2 MULTICAST EFFICIENCY	53
	4.3 FORWARDING GROUP SIZE	56
	4.4 CONTROL OVERHEAD	59
5	CONCLUSION	64
	REFERENCES	66

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1	Group Join in FGMP	10
2	Group Creation in FGMP	10
3	FGMP Group Maintenance	11
4	Group Join in Reverse FGMP	15
5	Handling Group Requests in Reverse FGMP	16
6	Group Join in DDM	22
7	Group Creation in DDM	22
8	Creation of FS and DS in DDM	23
9	Group Maintenance in DDM	24
10	Join Request and Group creation in LTMP	41
11	Group Maintenance in LTMP	41
12	Group LEAVES in LTMP	42
13	Multicast Delivery vs. Number of Groups	49
14	Multicast Delivery vs. Node Mobility	51
15	Multicast Efficiency vs. Number of groups	54
16	Multicast Efficiency vs. Node mobility	56
17	Average FG size vs. Number of Groups	57
18	Average FG size vs. Node mobility	59
19	Control overhead vs. Number of groups	60
20	Control overhead vs. Node mobility	62

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

MANET – Mobile Ad Hoc Network

FGMP – Forwarding Group Multicast Protocol

DDM – Differential Destination Multicast Protocol

RFGMP – Reverse FGMP

LTMP – Localized Tree Change Multicast Protocol

DVMRP – Distance Vector Multicast Routing Protocol

CBT – Core Based Tree

DIS – Distributed Interactive Simulation

FG – Forwarding Group

MAC layer – Medium Access Control layer

AODV – Ad Hoc On Demand Distance Vector Routing protocol

DSR – Dynamic Source Routing protocol

CHAPTER 1

INTRODUCTION TO MOBILE AD HOC NETWORKS

In this chapter we provide a gentle introduction to ad hoc networks, we look at their applications, the different mechanisms of establishing multicast trees in terms of their pros and cons. We then show by means of an example why multicast protocols for wireline networks need to evolve for the new environment.

1.1 Mobile Ad Hoc networks

A mobile Ad hoc Network (MANET) is a collection of autonomous mobile nodes capable of communicating with each other via wireless links. Nodes in a MANET have limited transmission range; communication is achieved by making use of nodes to forward packets to other nodes, which thereby have to operate as routers. Finding a path between two communication end points in an ad hoc network is non trivial; node mobility results in highly dynamic network topologies. These networks are rapidly deployable as they do not require any infrastructure in place. MANETs are highly desirable in a variety of scenarios: disaster recovery - where the entire communication infrastructure might have been destroyed, business meetings - where a group of people have to share resources and communicate with each other, communication over rugged terrain – where establishing infrastructure is not cost effective. Ad hoc networks can also be used to deploy multimedia services; however efficient routing protocols have to be developed before this can be realized.

The high node mobility, low bandwidth wireless interfaces, limited battery power and contention for the shared wireless medium makes designing routing protocols for ad hoc networks difficult; any new routing protocol must take note of these factors critically. Several unicast routing protocols for ad-hoc networks have already been proposed [1]. Our project addresses multicast routing in MANETs.

1.2 The Multicast capability

Multicasting is the ability to send packets to and receive packets directed at a subset of nodes in the network. Multicast group communication has many applications, including video and audio conferencing and distributed interactive simulation (DIS). Another interesting application is video-on-demand, wherein a set of multicast receivers are able to receive (when required) video data multicast by a single video server. Establishing multicast trees are definitely more efficient than the brute force approach of Unicasting multicast packets to all the members of a group. In general, multicast routing is achieved using either source based or core based trees. In a source based mechanism, each sender in the multicast group constructs a shortest-path tree spanning all multicast members. The advantage of a source based mechanism is that multicast packets are delivered with relatively low latency, because of the use of shortest paths for communication between the sender and the multicast receivers. The disadvantage, however, is that this scheme does not scale very well as the number of multicast senders in a group increases, because of the need to maintain tree information with respect to every sender in the multicast group. Clearly, such a scheme is well suited to group communication

involving few senders, such as the video-on-demand application mentioned above. However ensuring a shortest path tree at all times is impossible and if attempted would incur excessive control overhead. Our proposed multicast protocol for MANETs - LTMP, is a source based multicast protocol.

The core based multicast mechanism [2] uses a multicast tree shared by all members of the multicast group. The CBT (Core Based Tree) mechanism was designed primarily to address the topic of scalability; shared tree architecture offers an improvement in scalability over source tree architectures by a factor of the number of active multicast sources. The CBT is optimized with respect to a common 'core' i.e. the CBT is a shortest path spanning tree from the core spanning all multicast members. However, source-based tree algorithms are clearly robust; a sender simply sends its data, and intervening routers "conspire" to get the data where it needs to, creating state along the way. This is the so-called "data driven" approach -- there is no set-up protocol involved. It is not as easy to achieve the same degree of robustness in shared tree algorithms; a shared tree's core router maintains connectivity between all group members, and is thus a single point of failure. Protocol mechanisms must be present that ensure a core failure is detected quickly, and the tree reconnected quickly using a replacement core router. Any CBT mechanism for a MANET environment is complicated by the fact that selecting a core is non trivial and a poor selection can result in large latencies.

1.3 Multicast routing issues for MANETs – Problems with the DVMRP approach

In DVMRP (Distance Vector Multicast Routing Protocol) [3], the protocol used in the internet (Mbone), each multicast sender uses flooding to direct the multicast packets to all nodes within a specified range (defined by TTL). Multicast packets are selectively forwarded according to the reverse shortest path forwarding protocol [8]. Non-member leaf nodes and nodes without any downstream members send prune messages upstream to prune off branches to non-member nodes. After timeout, pruned branches become alive again and get flooded with messages. A new receiver member can also send a graft message to upstream nodes in order to speed up the connect process.

To understand some of the multicast routing issues in a MANET environment, it would be useful to analyze the problems that occur when DVMRP is extended to a MANET environment. One problem is the data flooding overhead. Because upstream nodes may change or be disconnected due to node mobility, it is necessary to reflood by exploring pruned branches after timeout in order to reestablish the upstream information, reconnect lost members, or allow new members to join. In addition, reflooding is needed to confirm the existence of the sender source. A sender is declared non-existent after timeout. This periodical reflooding of data causes very large transmission overhead for the low bandwidth wireless channel that exists in a MANET. Another problem is the Reverse shortest Path Forwarding (RPF). Since DVMRP uses the RPF mechanism, packets are accepted only from the shortest path. If the shortest path changes and there are no packets to be

forwarded in the new path, the node will be disconnected from the tree (since it will not accept packets from the old path).

Furthermore, in a wireless environment, there is no notion of explicit link interface like in a wired point to point channel. Multicast forwarding is based on nodes (routers) which are going to accept multicast packets rather than on links on which multicast packets are forwarded. Traditional multicast protocols based on upstream and downstream links (like DVMRP) are not suitable here because creating and maintaining upstream and downstream link status in a wireless network is not efficient. The FGMP and other protocols (described in the next chapter) are explicitly designed to work in a MANET environment and overcome some of the problems that occur when DVMRP is extended to a wireless environment. Our proposed protocol, LTMP, is modeled on FGMP and is an improvement over several existing MANET protocols.

CHAPTER 2

EXISTING MULTICAST PROTOCOLS

In this chapter, we briefly describe some existing multicast routing protocols for MANETs, with which we have compared the performance of the LTMP protocol. We also propose a variant of FGMP called Reverse FGMP and prove that the protocol is loop free.

2.1 FGMP and RFGMP

Unlike DVMRP, FGMP (Forwarding Group Multicast Protocol) keeps track not of links but of groups of nodes which participate in multicast packets forwarding. To each multicast group G is associated a forwarding group, FG. Any node in FG is in charge of forwarding (broadcast) multicast packets of G . That is, when a forwarding node (a node in FG) receives a multicast packet, it will broadcast this packet if it is not a duplicate. All neighbors can hear it, but only neighbors that are in FG will first determine if it is a duplicate and then broadcast it in turn. This scheme can be viewed as “limited scope” flooding. That is, flooding is contained within a properly selected forwarding set. It is interesting to note that with proper selection of the forwarding group, the FG scheme can emulate any of the existing schemes. For example, to produce global flooding, the FG must include all nodes in the network. For CBT, the FG is restricted to the nodes on the shared tree except the leaf nodes. In DVMRP, FG includes all the non-leaf nodes on the source trees. Only one flag and a timer are needed for each forwarding node. Storage overhead, a major problem in traditional multicast

Protocols, is low, thus increasing the scalability. When the forwarding flag is set, each node in FG forwards data packets belonging to G until the timer expires. Timer is refreshed by the forwarding group updating protocol, described below. The soft state approach of using a timer works well in dynamic changing environments.

To reduce the overhead, FGMP requires group members to flood explicit sender or receiver membership (rather than data packets as in DVMRP). Namely, instead of flooding data packets like DVMRP, we only flood small size control messages and with less frequency. In DVMRP, flooding is needed to refresh senders' status and reestablish multicast states (upstream and downstream). To adapt to the changing environment (mobility), the flooding frequency must be increased as mobility increases. Thus, high mobility needs high frequency flooding to maintain and reestablish the multicast status, but this will increase the channel overhead and degrade the performance. Our proposed membership advertising scheme only refreshes the membership. Channel overhead is much lower than DVMRP, making FGMP effective in a mobile wireless environment. In the proposed scheme, the decision to forward multicast packets depends on a forwarding flag. The forwarding flag is associated with a timer ("soft state"). Unlike DVMRP, which requires each tree node to store information about upstream and downstream links, FGMP requires the use of only one forwarding flag and a timer at each forwarding node.

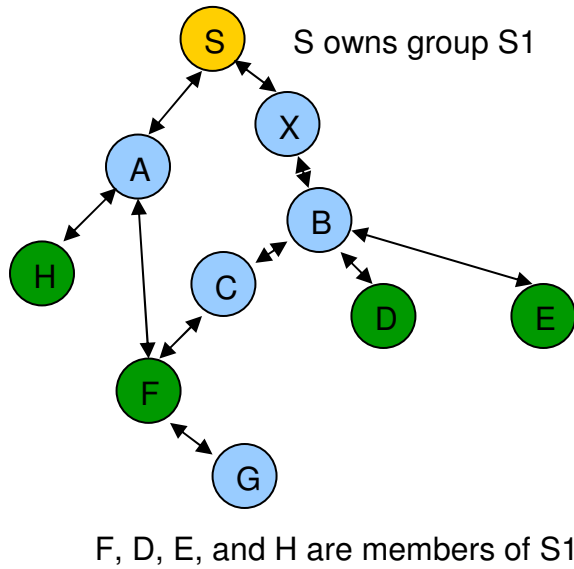
The FGMP forwarding group maintenance process can be carried out using either Receiver advertising or Sender advertising. We chose Receiver advertising in our implementation. This method has been explained below with an illustration.

2.1.1 FG maintenance using receiver advertising

Each multicast receiver periodically and globally floods its member information formatted in a `join_request` packet. The `join_request` packet flooded by each receiver is a four tuple \langle Multicast group ID, Receiver ID, Sequence no, TTL \rangle . Each multicast source maintains a member table that contains the set of receivers associated with each multicast group in which the node is a multicast sender. Each entry in the table is of the form \langle Multicast group ID, Refresh Timer, Receiver Set \rangle , where Receiver Set is a set of ordered pairs of the form \langle Receiver ID, receiver refresh timer \rangle listing the receivers associated with the group identified by Multicast group ID. Here, receiver refresh timer represents the amount of time that can elapse before which the source must receive a `join_request` from the particular receiver identified by Receiver ID, failing which the receiver is removed from the Receiver Set. Refresh Timer represents the amount of time that can elapse before which the source must refresh its status as a member of the group identified by Multicast Group ID. The sender will broadcast multicast data packets only if the member table is not empty. When a sender receives the join request from receiver members, it updates the member table. After updating the member table, the sender creates from it the `forwarding_table`. The format of the `forwarding_table` packet is as follows: \langle Multicast Group ID, Forwarder set \rangle where Forwarder set is a set of ordered pairs of the form

<Receiver ID, Next hop>. Each forwarder in the Forwarding group FG associated with a multicast source maintains a Forwarding Flag and an associated forwarding_flag_timer that must be refreshed periodically by the reception of forwarding_table packets with the forwarder listed as a next hop in the Forwarder set.

Next hop information is obtained from preexisting routing tables. The forwarding table FW is broadcast by the sender to all neighbors; only neighbors listed in the next hop list (next hop neighbors) accept this forwarding table (although all neighbors can hear it). Each neighbor in the next hop list creates its forwarding table by extracting the entries where it is the next hop neighbor and again using the preexisting routing table to find the next hops, etc. After the FW table is built, it is then broadcast again to neighbors and so on, until all receivers are reached. The forwarding group is created and maintained via the forwarding table FW exchanges. At each step nodes on the next hop neighbor list after receiving the forwarding table enable the forwarding flag and refresh the forwarding_flag_timer. Soft state dynamic reconfiguration provides the ability to adapt to a changing topology.



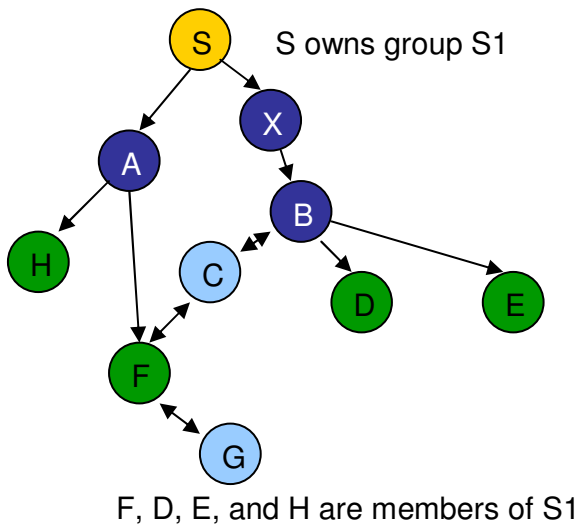
Join Request

Join Requests

My Address (F)	Group Address (S1)
My Address (H)	Group Address (S1)
My Address (D)	Group Address (S1)
My Address (E)	Group Address (S1)

Figure 1 - Group Join in FGMP

The Members of the group (F, D, E, and H) periodically flood a membership request. The interval between two successive floods is varied in accordance with the node mobility. The source maintains a list of nodes which have requested to become members.



Group Creation

Forwarding table for S1 by S

Member Address	Next Hop
F, H	A
D, E	X

Forwarding table for S1 by X

Member Address	Next Hop
D, E	B

Figure 2- Group Creation in FGMP

The source periodically refreshes the group of forwarders for the group by periodically multicasting forwarding tables. It partitions the member list based on the next hop, obtained from its unicast routing table. Nodes which receive the broadcast check if their IP address is in one of the next hops. If so, the node becomes a forwarder, it obtains the list of nodes for which it is to forward from the packet, partitions them based on the next hops and so on till the destinations are reached. A timer is set for the forwarding flag; if it is not refreshed before the timer expires; the node is no longer a forwarder for this group

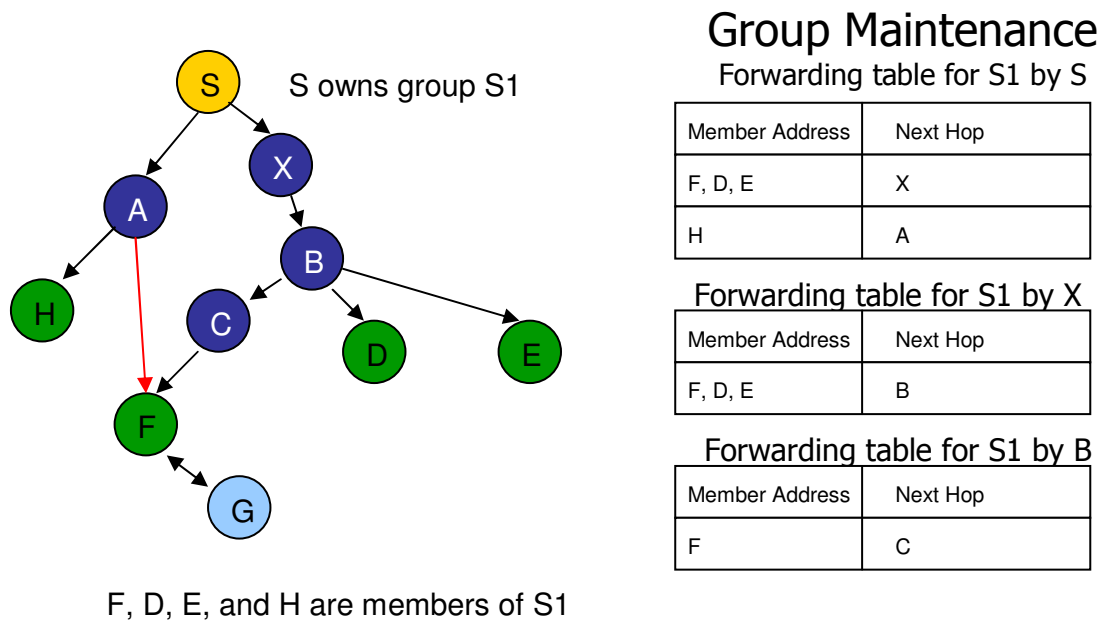


Figure 3– FGMP Group Maintenance

The red link indicates a link failure. In the next refresh interval, the source switches to X as its next hop for F. B introduces a new forwarder C for the group. In this way failures are handled, i.e. new forwarders are added; existing nodes may also be used to forward the traffic in accordance with the unicast routing table. Group leaves are handled similarly.

Receivers periodically flood join request packets (using a constrained flooding algorithm using sequence numbers). Senders then compute and maintain forwarding groups by periodically broadcasting forwarding tables to refresh forwarder status.

The multicast senders periodically sends forwarding tables to refresh the multicast tree.

Procedure SendForwardingTable (n, G)

Parameters

N: Node sending Forwarding Table

G: Group identifier

Begin

R_G : Receiver set for group G obtained from Member List

P: Forwarding table packet created

Add to P <G, N>

For each R in R_G

M: Receiver id in R

N: Next Hop for M from routing table

Add to P <M, N>

Broadcast P

End

The multicast receivers periodically – frequency depends on receiver_refresh_timer, send join requests for the group they would like to be members.

Procedure flood_join_request (n, G)

Parameters

N: Node sending join request

G: Group identifier

Begin

S: Increment a locally maintained counter

Create the Join Request packet <G, N, S>

Flood Join Request packet

End

The following procedure is invoked when a forwarding table is received

Procedure ProcessForwardingTable (N, F)

Parameters

N: Node that received Forwarding Table

F: Forwarding table received

Begin

G: Group id in forwarding table

FS_G: Forwarding set in forwarding table

ML: Member List created

P: Forwarding table packet created

For each FG in FS_G

 If N is a next hop in FG

 M: Receiver id in FG

 Add to ML <M>

```

        Set forwarding flag for G once
    If ML is not null
        Add to P <G, N>
        For each R in ML
            M: Receiver id in R
            N: Next Hop for M from routing table
            Add to P <M, N>
        Broadcast P
    Else
        Drop packet
End

```

The following procedure is invoked when a node receives a join request.

Procedure ProcessJoinRequest (N, J)

Parameters

N: Node that received the join request

J: Join Request received

Begin

G: Multicast group id in join packet

R: Receiver id in join packet

S: Sequence number in join packet

If <R, S> is not JREQ_{cache}

 If N owns G

 Add R to MemberList_G

 Else

 Rebroadcast J

```

Else
    Drop packet
End

```

2.1.2 Reverse FGMP

In this variant of FGMP, multicast sources use the reverse path set up by the join_request packets rather than relying on the unicast routing protocol to create forwarding groups. To accommodate the use of join requests to form the unicast routing table, the join request packets format was changed to

<Multicast group ID, Receiver ID, Sequence no, Hop Count, Next Hop Id>

To a node, next hop represents the node via which the join request from Received ID was received. The Routing table is a list of entries of the form

<Destination, Next Hop, Sequence Number, Hop Count>.

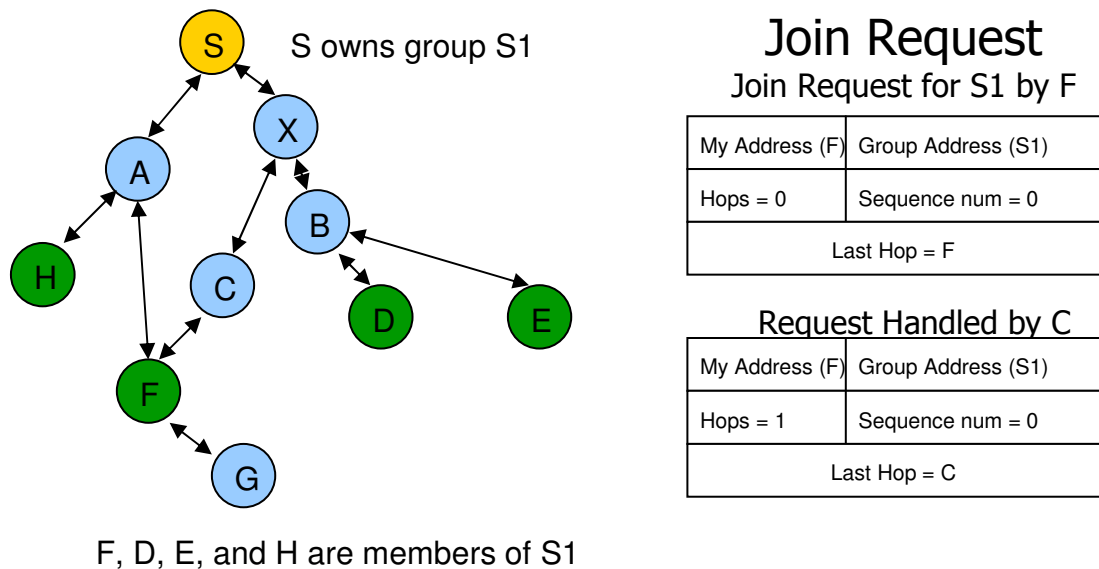


Figure 4 – Group Join in Reverse FGMP

By including additional fields (Hops, Sequence number) RFGMP does away with the underlying unicast protocol required by FGMP. When a node receives a join request, it updates and rebroadcasts the request packet if either of the below hold good

- (i) The sequence number in the packet is greater than the highest known of the requesting node
- (ii) The sequence number in the packet is no worse than what is known and the hop count of the packet is less than what is maintained

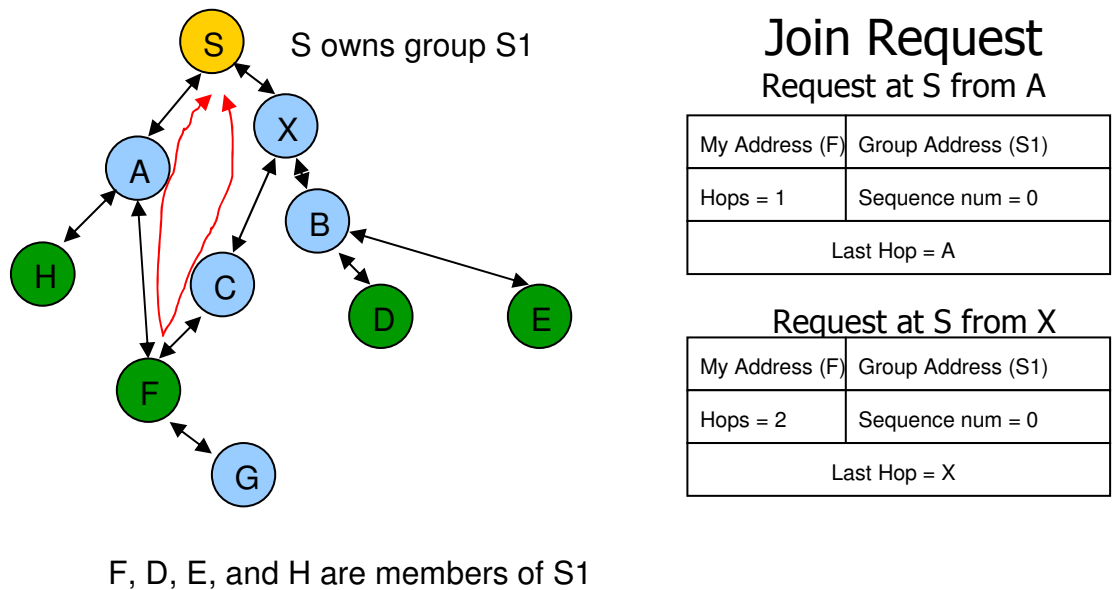


Figure 5 – Handling Group Requests in Reverse FGMP

Source S receives the request by F via two nodes: A and X. Since the sequence numbers are equal, and initially S knows of no route to F, S chooses A as its next hop to F and updates its routing table. F on its next flood would increment its locally maintained counter; hence the packets' sequence number would be greater than what S knows.

This modified version of FGMP is basically the same as FGMP except for the use of the join_request packet to set the reverse path to the multicast receivers. In the previous version, forwarding tables were created using the unicast routing table created by the unicast routing protocol. In this case, the flooding of join_request packets is effectively made use of in the actual computation of routes to the receivers thereby making the process more efficient. Only two procedures of FGMP need to be modified. The multicast receivers periodically (frequency depends on receiver_refresh_timer), send join requests for the group they would like to be members.

Procedure FloodJoinRequest (n, G)

Parameters

N: Node sending join request

G: Group identifier

Begin

S: Increment a locally maintained counter

Create the Join Request packet <G, N, S, 0, N>

Flood Join Request packet

End

The following procedure is invoked when a node receives a join request.

Procedure ProcessJoinRequest (N, J, RT)

Parameters

N: Node that received the join request

J: Join Request received

RT: Routing table maintained at N

Begin

G: Multicast group id in join packet

R: Receiver id in join packet

S: Sequence number in join packet

H: Hop count in join packet

N: Next Hop in join packet

RT_R : Routing table entry with Destination field set to R

If RT_R is null

 If N owns G

 Add R to $MemberList_G$

 Add Routing table entry $\langle R, S, H, N \rangle$

 Else

 Add Routing table entry $\langle R, S, H, N \rangle$

 Rebroadcast J

 End if

Else

S_{RT} : Sequence number in RT_R

H_{RT} : Hop count in RT_R

 If $S > S_{RT}$

 Update Routing table entry $\langle R, S, H, N \rangle$

 Else if $S = S_{RT}$ and $H < H_{RT}$

 Update Routing table entry $\langle R, S, H, N \rangle$

 End if

 If N owns G

 Add R to $MemberList_G$

 Else

 Rebroadcast J

End if

End if

End

2.2 DDM

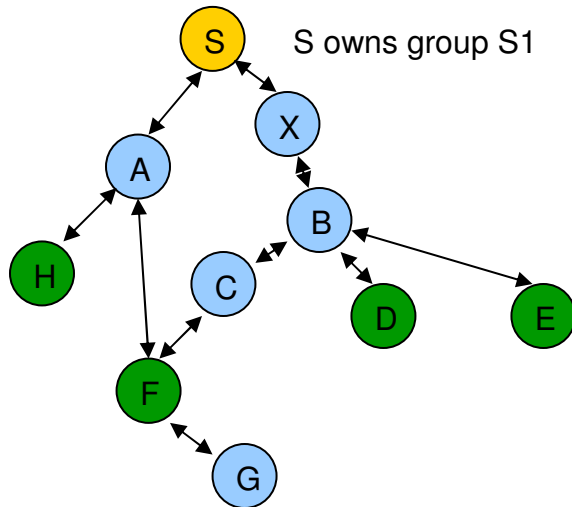
The Differential Destination Multicast (DDM) [5] protocol is motivated, in part, by the approach to unicast routing of Dynamic Source Routing (DSR) [7]. In DDM (Differential Destination Multicast), the sources control multicast group membership to ease certain aspects of security administration. More importantly, and a departure from other proposed MANET multicast protocols, DDM encodes the destinations (i.e. the multicast group members to whom the data needs to be delivered) in each data packet header. This in-band information can be used to establish soft state routing entries if desired. Using such an approach has two advantages for MANETs. Firstly, there is no control overhead expended when the group is idle; a characteristic shared with the DSR. Secondly, it is not necessary for the nodes along the data forwarding paths to maintain multicast forwarding state if it chooses to run under stateless mode. When one intermediate node receives a DDM data packet, it only needs to look at the DDM header to decide how to forward the packet; another similarity to DSR. Assuming that routers can handle this processing cost, this stateless mode can be very reactive and efficient. This stateless approach also avoids loading the network with pure signaling traffic; a third trait shared with DSR. In so doing, the hope is that the unicast algorithm can converge much faster, with DDM then making immediate use of new unicast routing knowledge.

While the fixed networks and MANETs can both benefit from stateless, explicit multicasting because of its savings in storage complexity, it is even more desirable to follow this approach in MANETs due to the special characteristics of the MANETs. Firstly, in wired Internet, network topology change rarely occurs. Therefore if group membership is stable, after a multicast tree is constructed, the maintenance effort is small and tree links seldom need repair. On the other hand, MANET topology is subject to constant and sometimes dramatic changes. Multicast forwarding topology built in MANETs needs constant repair even rebuild. Maintaining multicast routing state is a much more expensive operation in MANETs than in wired networks. This is even worsen by the tight bandwidth constrains of MANETs. Secondly, the cost of medium access is high in wireless broadcast networks due to the MAC mechanism and broadcast transmission's blocking effect to neighboring nodes. Although packing routing information together with data traffic will enlarge data packet size, it reduces the total number of channel accesses because it reduces the number of pure control packets generated by the protocol. Therefore such approach can be more efficient overall in many scenarios. Thirdly in broadcast networks, when a node needs to send to more than one neighbor, it only needs to broadcast the packet once. So this approach has better bandwidth consumption and channel access properties in broadcast networks than in point-to-point networks.

In scenarios where the stateless approach is not favorable, DDM may also operate in a "soft-state" mode. In this mode, as data packets with in-band control information are routed through the network, each node along the forwarding path remembers the destinations to which it forwarded the last time and how the data was forwarded (i.e. which next hop was used for

each destination). By caching this information, the protocol no longer needs to list all the destinations in every data packet header. When changes occur in the underlying unicast routing or destination list, an upstream node only needs to inform its downstream neighbors (i.e. its next hops) regarding the differences in destination forwarding since the last packet; hence, the name “Differential Destination” Multicast. Reporting only these differences significantly reduces DDM header sizes. Ideally, in a stable network where the topology and membership remain unchanged, only the first data packet needs to contain destination addresses and all subsequent packets would contain no destination information. In practice, the state kept at each node along the forwarding paths is “soft”. Each time data forwarding occurs, this state is refreshed. Stale state eventually times-out and is removed. This mode is better suited for applications that generate small data packets at relatively high rate.

DDM is not a general purpose multicast protocol in the conventional sense. The header-encoded destination mechanism does not scale well with group size. The stateless mode, however, does scale well with the number of multicast groups, as no per-group state is required in any routers. In MANETs, if the number of multicast groups is small enough to permit state storage, then the soft-state version (which we have implemented) with differentially encoded header processing can be used to reduce average packet size and save bandwidth. This approach, however, has little applicability to fixed networks as the complexity of the differentially-encoded header processing would significantly slow down forwarding rates in high-speed networks.



F, D, E, and H are members of S1

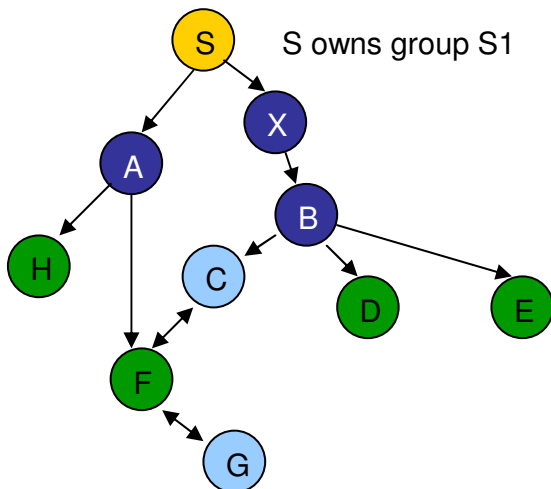
Join Request

Join Requests

My Address (F)	Group Address (S1)
My Address (H)	Group Address (S1)
My Address (D)	Group Address (S1)
My Address (E)	Group Address (S1)

Figure 6 – Group Join in DDM

The members of the group (F, D, E and H) periodically unicast a membership request making use of an underlying unicast protocol. The source maintains a list of nodes which have requested to become members.



F, D, E, and H are members of S1

Group Creation

Packet created by S

Sender address (S)	Group address (S1)	Next Hop=A	Block Type = Refresh
Block Sequence=0	Receiver address =H	Receiver address =F	Next Hop=X
Block Type= Refresh	Block sequence = 0	Receiver address =D	Receiver address =E

Packet created by X

Sender address (X)	Group address (S1)	Next Hop=B	Block Type = Refresh
Block Sequence=0	Receiver address =D	Receiver address =E	

Figure 7- Group Creation in DDM

The source whenever it needs to send data, partitions its member list by next hop, obtained from its unicast table. Nodes which receive the broadcast check if their IP address is in one of the next hops. If so, the node becomes a forwarder, it obtains the list of nodes for which it is to forward from the packet, partitions them by next hop and so on till the destinations are reached. A timer is set for the forwarding flag; if it is not refreshed before the timer expires the node ceases to be a forwarder. The partition forms the Direction set for this group, which forms the downstreams forwarding set for this group. The DS is set to expire before FS.

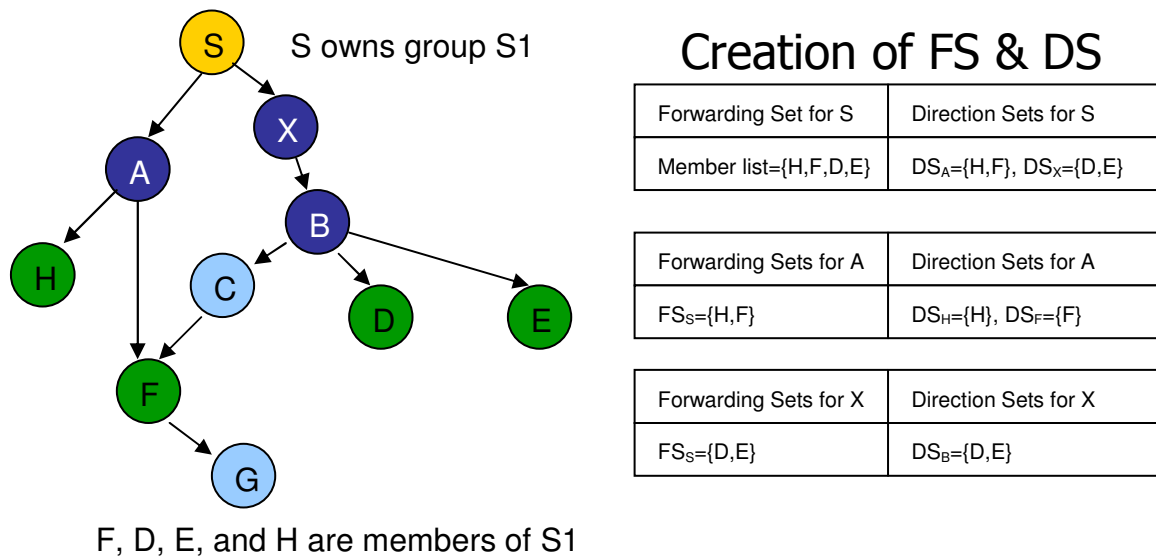
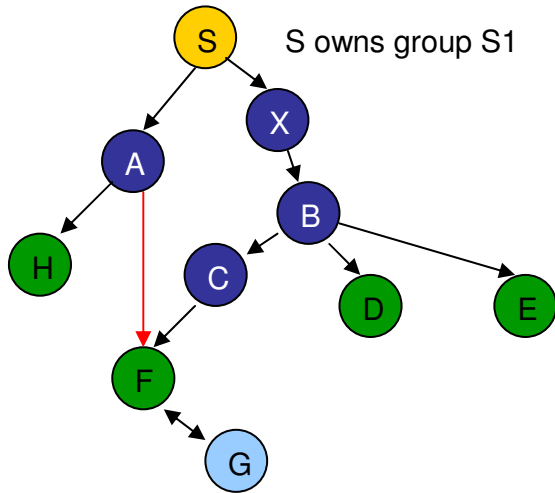


Figure 8 – Creation of FS and DS in DDM

Consider the first time creation of DS and FS. For the source S, the Member list for the group S1 forms its FS for S1. Note that the DS_A maintained by S for S1 forms a particular FS for S1 for the node A. In general, a node could have multiple predecessors and FS is a union, like DS. Storing FS and DS allows incremental packets to be sent



Maintenance

Sender address (S)	Group address (S1)	Next Hop=A	Block Type = Decrement
Block Seq = n	Receiver address = F	Next Hop=X	Block Type = Increment
Block Seq = m	Receiver address =F		

F, D, E, and H are members of S1

Figure 9 – Group Maintenance in DDM

The red link represents a failed link. When source S has data to send, it creates a new DS. It compares the new DS with the old DS. If a particular new DS has fewer members than the old DS then a decrement packet (D) is sent (The DS for A, no F in the new DS). If a particular new DS has more members than the old DS, then an incremental packet (I) is sent. If such a DS did not exist, then a refresh packet (R) is sent. If there is no change an empty packet is sent. Whenever R, I and D packets are sent, the sequence number is incremented. Empty packets (E) are sent with the old sequence number. Hence if a I, R or D packet is lost, an E packet sent would cause a rsync packet to be sent. Group leaves are handled as followed: if a forwarder is not on the path to any receiver, it is not listed in any block generated for this group anymore. The flag eventually expires and the node is no longer a forwarder for this group.

2.2.1 Soft State Operation

Member list ML: The multicast source maintains a list of receivers in a list called ML. Each receiver entry is associated with a timer that indicates the amount of time within which a `join_request` must be received from the receiver.

Each node that is a forwarder maintains two sets FS and DS with respect to a multicast group.

-- FS is a set of elements of the form FS_k

Where FS_k is the set of IDs of all receivers that have been assigned to the node by upstream k

The union of all receiver IDs in each $FS_k \in FS$ is used by the node to compute DS.

-- DS is a set of elements of the form DS_k .

Where DS_k is the set of IDs of all receivers that have been assigned to downstream k by the node (using the unicast routing table).

Furthermore, with each DS_k and FS_k are associated a sequence number (used for synchronization purposes) and a timer, which indicates the amount of time for which the respective set is to be maintained by the node. The `join_request` packet is used by a receiver to join a multicast group, and a `leave_request` packet is used to leave it.

Each `ddm_multicast_data_packet` sent by the source or an intermediate node contains the following information:

- The multicast payload along with the multicast group id, source id and sequence number (to avoid retransmission of seen packets)
- The DS computed by the node, along with the sequence numbers associated with each DS_k
- Each DS_k block can be of the following types

Refresh: This block type indicates a re-initialization of the DS_k set. All the elements in DS_k are included.

Empty: This block type indicates that no changes need to be made to the DS_k . No elements are included

Increment/Decrement: These two block types indicate an increase or decrease in the number of receivers in set DS_k . Only the changes (such as new entries or deletions to/from the old DS_k) are included.

Initiated by multicast receiver periodically to refresh status with the source

Procedure JoinRequest (n, G)

Parameters

N: Node sending join request

G: Group identifier

Begin

Unicast `join_request` packet to the source

End

Initiated by multicast receiver to leave a group

Procedure LeaveRequest (n, G)

Parameters

N: Node sending leave request

G: Group identifier

Begin

Unicast leave_request packet to the source

End

When a node has multicast data to send or a node receives a multicast data packet the following procedure is called. Intermediate nodes use the information in the packet for this procedure

Procedure HandleMulticastDataPacket (n, G, P)

Parameters

N: Node that has to handle the data

G: Group identifier

P: Data , which includes DDM block for intermediate nodes

Begin

Update/Initialize set FS

//Update set FS_k (or initialize FS_k , if such a set does not exist) using

//information from received DS_j block (Where k is the ID of the

//upstream that sent ddm_multicast_data_packet and j is my ID) and

//previously existing set FS_k (if present) using the DDM block type
//as follows

For each DDM block associated with node j

Switch 'block type'

Case 'refresh'

Re-initialize FS_k with the IDs in received DDM block

Assign sequence number of DS_j to FS_k and reset its timer

Case 'increment'

If DDM block's sequence number is (sequence number of $FS_k + 1$)

Add to FS_k the IDs in DDM block

Increment FS_k 's sequence number and reset its timer

Else

Invoke resynchronize_with_upstream (k)

Case 'decrement'

If DDM block's sequence number is (sequence number of $FS_k + 1$)

Remove from FS_k the IDs in DDM block

Increment FS_k 's sequence number and reset its timer

Else

Invoke ResynchronizeWithUpstream (k)

Case 'empty'

If block's sequence number is greater than the
sequence number of FS_k

Invoke ResynchronizeWithUpstream (k)

Else

Reset FS_k 's timer

End switch case

End loop

//The source does it from the member list ML

//There is no ' FS_k ', for the source. Assume FS contains a single set

// F_0 containing all receiver IDs in ML

Compute Receiver set RS, which is the union of all receiver IDs in
each $FS_i \in FS$

Compute set DS, consisting of elements DS_i where DS_i denotes the set
of receiver IDs in RS which have i as the next hop in the unicast
routing table.

Refresh the timers for each DS_i

If the multicast data is new

//Compare each newly computed/updated DS_i block with
//previously existing block.

//The first case could occur if either the previous DS_i timed out

//or this is the first such DS_i received

If there previously existed no DS_i

Create DDM block for set DS_i
Set block type to 'refresh'
Include all elements in DS_i as part of ddm multicast data packet
Assign new sequence number

Else

Compare newly updated DS_i with the old DS_i
If there are no changes

Create DDM block for set DS_i
Set block type to 'empty'
Retain the same sequence number as before

End if

If there are additional elements

Create DDM block for set DS_i
Set block type to 'increment'
Increment sequence number by one
Include additional elements in DS_i as part of the block in ddm multicast data packet

End if

If there are deletions

Create DDM block for set DS_i
Set block type to 'decrement'
Increment sequence number by one
Include deletions as part of the block in ddm multicast data packet

```

        End if
        Create a ddm_multicast_data_packet by including DDM blocks
        for each  $DS_i$  in DS. Also include the multicast payload, group
        ID etc
        Broadcast ddm_multicast_data_packet with hop limit of one
        End if
    End if
End

```

In the DDM protocol, the downstream-upstream pair ensures that they both maintain the same set of receiver IDs in their respective FS/DS sets by using sequence numbers and a Resync packet that is sent by the downstream when there is a mismatch in sequence numbers.

Procedure ResynchronizeWithUpstream (N, G, U)

Parameters

N: Node that has to resynchronize with U

G: Multicast group Id

U: Upstream to synchronize with

```

Begin
    Unicast Resync packet <N, G, U>
End

```

2.3 Constrained Flooding

In this simple protocol, the multicast source globally floods all multicast packets using a constrained flooding algorithm. All multicast data packets have the following format: flood_packet <Multicast group ID, Source ID, Sequence number, Multicast payload>. To prevent nodes from reforwarding duplicate multicast data packets, sequence numbers are used. The pseudo code for this protocol is shown below. DATA_{CACHE} is a list of entries of the form < Source ID, Sequence number>

When a multicast source has data to send it initiates the following procedure

Procedure SendMulticastData (N, G, P)

Parameters

N: Node sending packet

G: Multicast group Id

P: Payload

Begin

S: increment a locally maintained counter

Broadcast Packet <G, N, S, P>

End

An intermediate node when it receives a flood packet initiates the following procedure

Procedure ReceiveMulticastData (N, P)

Parameters

N: Node receiving multicast data packet

P: Packet received

Begin

SC: Source of the packet

S: Sequence number of the packet

D: Entry in $DATA_{CACHE}$ with Source ID set to SC

If D is null

 Add $\langle SC, S \rangle$ to $DATA_{CACHE}$

 Rebroadcast P

Else

S_D : Sequence number in $DATA_{CACHE}$

 If $S_D < S$

 Update entry $\langle SC, S \rangle$

 Rebroadcast P

 Else

 Drop Packet

 End if

End if

End

Flooding is likely to be the most robust of all protocols, because the multicast data packet is flooded to the entire network. It is also the least efficient among the above mentioned protocols because there is an excessive wastage of bandwidth and energy. (Each node in the network remains a forwarder for all multicast groups)

2.4 Loop Freedom of a multicast routing protocol

The protocols mentioned above, namely DDM and FGMP, ensure loop free operation. This is because the underlying unicast routing protocol, AODV, used in both these multicast routing protocols has the loop freedom property. The proof that any distance vector protocol using either newer sequence numbers or shorter routes to update next hop information is loop free is given in [6]. Because the underlying unicast protocol is loop free, the unicast path from the multicast source to each of its receivers is free of loops. As a result, the multicast protocol, which simply uses these loop free paths to deliver multicast packets, is also loop free.

The Reverse FGMP protocol does not use any unicast protocol for its operation. Paths between the multicast source and its receivers are setup by the flooding process used to advertise receiver membership information. If the paths thus created are loop free, then Reverse FGMP is also loop free. The route discovery and update process in Reverse FGMP is such that all paths from the multicast source to its receivers are loop free. This is the same route update process used in the AODV protocol. Therefore, the proof in [6] also applies here. Multicast receivers frequently flood membership information in Reverse FGMP, using which intermediate nodes update their next hop to the receiver. In Reverse FGMP, there are two situations in which a node A can switch its next hop to receiver C. In both cases, there is no possibility of loop formation.

(I) Node A can switch to Node B as its next hop for receiver C, if their sequence numbers of C are equal and the distance s_B^C of node B to node C is less than the distance s_A^C of node A to node C. By choosing B as a next hop, A can close a loop only if A is on B's path to C. This would mean that $s_A^C < s_B^C$. This contradicts our initial assumption that $s_B^C < s_A^C$.

(II) Node A can switch to node B as its next hop for receiver C, if B's sequence number for C, is greater than A's sequence number for C. By choosing B as a next hop, A can close a loop only if A is on B's path to C. If A is on B's path to C, then A's sequence number of C can be no worse than B's sequence number, i.e. A's sequence number would be greater than or equal to B's sequence number for C. This is because of the manner in which sequence numbers are propagated – a more rigorous proof of the fact that using newer sequence numbers results in loop free operation is given in [6]. This contradicts our initial assumption that B's sequence number is greater than A's sequence number for C.

As a result, all the protocols mentioned above, including Reverse FGMP, ensure loop free operation.

CHAPTER 3

LOCALIZED TREE CHANGE MULTICAST PROTOCOL

The previous chapter provided a brief description of some of the existing source based multicast routing protocols for ad hoc networks. Although these protocols were designed with the aim of coping with some of the challenges posed by a MANET environment and are hence an improvement over the DVMRP approach, there remain some issues that need to be addressed.

Consider for instance, the FGMP protocol. Although FGMP improves upon DVMRP by reducing unnecessary storage overhead, it still requires group senders or receivers to *periodically flood* membership information *globally*. Such periodic global flooding can increase contention for the shared wireless medium. Clearly, such a mechanism is not well suited to a scenario in which the sender or receiver membership status changes infrequently. A better method would be to require senders or receivers to globally update their membership information, only when there are changes to their current status. Also, in most multicast protocols, including FGMP and DDM, nodes use the MAC layer *broadcast* facility to forward multicast packets. Since MAC layer protocols such as 802.11 provide acknowledge messages only for *unicast* transmissions, multicast data packet collisions at the MAC layer remain undetected, i.e. collisions result in multicast data packet loss. The increased contention for the shared wireless medium (due to the *periodic global flooding* in FGMP) only increases the probability of multicast data packet loss due to collisions at the MAC layer. The DDM protocol, also described in detail in the previous chapter, suffers from

excessive complexity and requires forwarder nodes to possess good processing power in order to ensure fast delivery of multicast packets. Also, both these protocols are heavily dependent on an underlying unicast protocol with *good* convergence characteristics.

The LTMP protocol has been designed to address the issues mentioned above and is modeled on FGMP, in that a forwarding group is maintained by enabling a forwarding flag at each of the forwarder nodes. Such an approach, as explained in chapter 1, makes better sense in a MANET environment. Unlike FGMP, however, LTMP does not require group members to periodically flood their membership status. This reduces the control overhead considerably in our protocol when compared with that of FGMP. Group receivers that wish to *receive* data from a multicast sender use a Request Response procedure to join the group. Such a scheme allows for end-to-end signaling between the source and the receiver. End to end signaling between the multicast source and the receiver has several useful benefits, as illustrated in [5]. In particular, it is possible for the source to authenticate potential multicast receivers. When a receiver wants to *leave* a group, it simply stops transmitting group hello packets (used in group maintenance), thus negating the need for any explicit GROUP LEAVE packet, resulting in a further reduction in control overhead. We believe that this reduced control overhead in our protocol is one of the main reasons behind its excellent overall performance. Also, conceptual multicast tree changes are localized and no upstream information is maintained. We make use of the broadcast nature of the wireless medium to encapsulate upstream information in group hello packets meant for downstream nodes. This ability for localized tree change allows our protocol to ensure stable multicast

delivery with as few additions to the forwarding group as possible, thus improving multicast efficiency and reducing the channel overhead. Also, unlike DDM and FGMP, the LTMP protocol does not depend on any unicast protocol for its operation. And unlike DDM, LTMP is a simple protocol, with very little processing cost at the forwarder nodes.

3.1 The LTMP protocol design

The group hello packet used in forwarding group maintenance is a four tuple containing the following entries {Sequence number, Group address, Multicast Source Address, My address, My upstream's address}. The sequence number is used to ensure freshness of the group hello packet. As mentioned earlier, the upstream address information is encapsulated in the group hello packet meant for downstream nodes. As in FGMP, a forwarding flag is set at each node to indicate that it is a forwarder for a particular multicast source. Associated with each such flag are two timers described below. If either of these time out, the forwarding flag is reset and the node ceases to be a forwarder.

The `Timer_group_hello` timer is set to indicate the maximum amount of time that can elapse before a FRESH group hello packet, with a sequence number higher than that of the Group hello last received from a particular multicast source, **MUST** be received, failing which the node ceases to be a multicast forwarder for that multicast source. The `Timer_group_hello_upstream` timer is set to indicate the maximum amount of time that can elapse before a multicast forwarder **MUST** receive a FRESH group hello packet from one of its downstream nodes, failing which the node

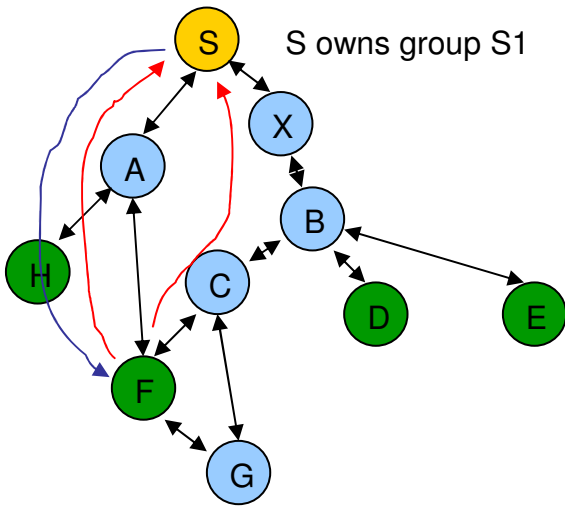
ceases to be a multicast forwarder for that multicast source. The functions of these timers become clear when the group maintenance process is understood.

The forwarding group with respect to a particular multicast source is first initialized using a request response procedure. A group JOIN request is sent by a receiver when it wants to receive multicast packets from a multicast source. This JOIN request packet is flooded and is handled by intermediate nodes in a manner similar to the RREQ (Route Request) packet in the AODV (Ad Hoc on Demand Distance Vector) protocol [4]. An important difference, however, is that only the source can respond with an RREP packet, because of the need for end to end signaling (unlike in AODV, where an intermediate node with a fresh enough route to the source can respond with an RREP packet). End to end signaling between the source and the receiver has several useful benefits, as illustrated in [5]. In particular, it is possible for the source to authenticate potential multicast receivers. Another difference in the way JOIN request packets are handled in LTMP (as opposed to the RREQ packet in AODV) is that once a JOIN request packet reaches a group forwarder, a flag in the packet is enabled and the packet is then forwarded only by group forwarders. This is an improvement over rigorous flooding, which requires every node in the network to forward the packet. The source responds to the first JOIN request it receives with an ACCEPT packet similar to the RREP in AODV. All nodes on the intermediate path are made forwarders with respect to that multicast source to which the JOIN request was sent.

As mentioned earlier, group maintenance is achieved by multicasting group hello packets. The manner in which a forwarder handles group hello packets is explained in the pseudo code. First, the node checks to see if the packet received is not a duplicate, using the sequence number field. If the packet is fresh enough, it inserts upstream information into the packet and forwards it after refreshing `Timer_group_hello`. If the packet is a duplicate, then the node checks to see if it has been made an upstream by the node forwarding the packet. If so, it resets `Timer_group_hello_upstream`. From this procedure, it is clear that a node ceases to be a forwarder under two conditions.

(I) A forwarder hasn't received a fresh enough group hello packet from the source for a length of time defined in `Timer_group_hello`. This could be due to node failures on the intermediate path between the source and the forwarder, due to node mobility or simply due to packet loss. Or it could be that the source just stopped sending group hello packets, because it has no more data to send.

(II) A forwarder hasn't received a group hello packet in which it has been specified as an upstream. This could be due to packet loss or it could mean that there are no forwarders downstream of the forwarder with respect to the multicast source. It must however, be noted that only condition I applies to a multicast receiver. All multicast receivers are, by default, forwarders, even if they do not have a downstream. This is to ensure that nodes upstream of the receiver refresh `Timer_group_hello_upstream`.



F, D, E, G, and H are members of S1

Join Request

Join Request for S1 by F

My Address (F)	Group Address (S1)
RREQ	

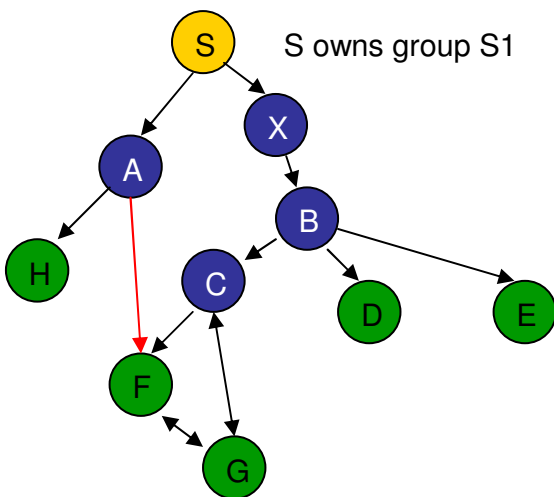
Group Creation

RREP

- RREP traverses reverse path
- Forwarding flags are set along the path for this group
- Node A therefore becomes a forwarder to F for group S1

Figure 10– Join Request and Group creation in LTMP

Join requests for a particular group are made by means of RREQs during which reverse paths to the destination are set up. The source acknowledges the first arriving RREQ, the RREP traverses the pre established reverse path setting up forwarding flags for the group on its way. Source S receives RREQs for S1 from F via 2 paths. It chooses the path S-A-F.



F, D, E, G, and H are members of S1

Group Maintenance

Periodic group hello for S1 by S

Group address(S1)	Sequence number(0)
Parent address (S)	My address (S)

Group Hello by F before failure

Group address(S1)	Sequence number(0)
Parent address (A)	My address (F)

Group Hello by F after failure

Group address(S1)	Sequence number (0)
Parent address (C)	My address (F)

Figure 11 – Group Maintenance in LTMP

Node F before the link failure would have rebroadcast the group hello with parent address set to A. However once it fails to get a group hello from A, it switches to C as its upstream. This is the localized tree change; we resort to flooding only when absolutely necessary. C learns by means of F's rebroadcast that there exists at least one child node and therefore should remain a forwarder. If C was not a forwarder for S1, F would not have received any group hello and its timer would have expired. It would have ceased to be a forwarder and a local tree change would have been attempted at levels lower to F, i.e. its downstream nodes.

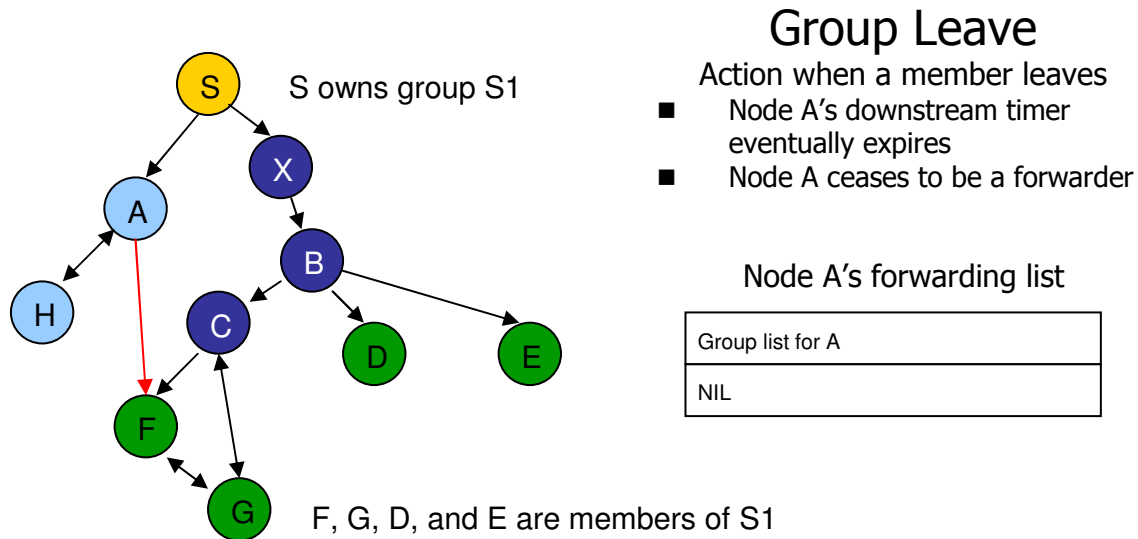


Figure 12 – Group LEAVES in LTMP

A node needs to remain as a forwarder only as long as there remains at least one node downstream to it. Consider the case of A, node F switched predecessors and node H quit the group. A fails to receive at least one group hello with parent address set to itself. Eventually its timer expires and A ceases to be a forwarder for A.

The source of a multicast group periodically sends group hellos.

Procedure SendGroupHello (n, G)

Parameters

N: Node that owns group G

G: Group identifier

Begin

S: increment locally maintained counter

Broadcast group_hello <S, G, N, N>

End

Every node periodically calls this procedure to refresh their status for the group.

Procedure MaintainState (n, G)

Parameters

N: Node that calls this procedure

G: Group identifier

Begin

If Timer_group_hello expires

 If N is a member of G

 Rebroadcast a RREQ to the source

 Else

 Cease to be a forwarder

 End if

Else if timer_group_hello_upstream expires

 Cease to be a forwarder

End if
End

A node initiates the following procedure for a group hello received.

Procedure ProcessGroupHello (n, P)

Parameters

N: Node that has received P

P: Group Hello Packet

Begin

G: Group Id in P

U: Parent Id in P

M: My id in P

S: Sequence number in P

If N is a member of G

T: Group Table Entry for Group G

If T is null

Drop packet

Else

S_G: Sequence number in T

If S > S_G

Update T as <G, S, now, ->

Broadcast <G, S, N, U>

Else If S_G = S and U = N

Update T as <G, -, -, now>

End if

End if

End

Multicast group members do not use an explicit LEAVE packet to indicate that they no longer wish to be part of a group. A multicast receiver, when it wishes to leave the group, simply stops forwarding group hello packets. A multicast sender also stops sending group hello packets when it wants to leave a group. The use of timers to maintain forwarder status results in correct synchronization of the forwarding group to reflect topology changes and member ship changes.

3.2 Loop Freedom of the LTMP protocol

The LTMP protocol ensures loop free paths for a multicast packet from a multicast source S to its receivers i.e. the conceptual multicast tree rooted at source S is indeed a tree and is devoid of any loops. To prove this, we follow the style of proof used in [6]. Consider the directed graph G defined by nodes i and arcs (U_i^S, i) , where U_i^S denotes the upstream for node i with respect to the multicast source S , where each node i is either a multicast group member or a forwarder. The LTMP protocol ensures that G is loop free i.e. the conceptual multicast tree rooted at S is indeed a tree and is loop free. Potentially, a loop may form when a node i changes its upstream. This can happen in two cases. The first case is when node i detects that the link to U_i^S is broken and sets U_i^S to null. Clearly, this cannot result in a loop. The second case is when node i receives a group hello packet from one of its neighbours k , with sequence number X_k^S that is greater than X_i^S . Here, X_i^S denotes the sequence number corresponding to multicast source S

stored at node i . By choosing node k as its next hop, node i cannot close a loop. This can be deduced from the following observation. A node i propagates sequence number X_i^S only after receiving it from its current upstream. Therefore, at all times, the sequence number stored at the upstream is always greater than or equal to the sequence number stored at node i . Starting from node i , if we follow the chain of upstreams, the sequence number values corresponding to S stored at the visited nodes would form a nondecreasing sequence. Now assume node i closes a loop by choosing node k as its upstream. This would imply that $X_k^S \leq X_i^S$. But this contradicts our initial assumption that $X_i^S < X_k^S$. Thus, loops cannot be formed if newer sequence numbers are used to pick upstream nodes.

CHAPTER 4

PERFORMANCE ANALYSIS

For evaluation purposes, we have formulated and implemented our protocol in the MANET simulator, Glomosim. We have also implemented four other existing protocols for comparison purposes, namely FGMP, Reverse FGMP, DDM and Constrained Multicast flooding. From the simulation results, it is clear that LTMP outperforms most of the protocols under analysis with regard to a number of performance parameters. We believe that this quantum improvement in performance is due to the reduced control traffic in our protocol and its ability for localized tree change.

To evaluate the performance of the LTMP protocol and to enable comparison with the existing protocols, we have used four performance parameters, namely multicast delivery (throughput), control overhead, multicast efficiency and average forwarding group size. The plots shown below illustrate how each of these parameters varies with the number of groups in the network and with node mobility. A large number of simulation trials were carried out, and the graphs shown represent the mean values.

4.1 Multicast Delivery:

To measure multicast performance, we define a parameter called the multicast delivery ratio, as follows:

Multicast delivery ratio = Total number of multicast packets received at all multicast receivers

$$\sum_i R_i \times P_i$$

Where R_i is the number of receivers in group i

P_i is the total number of multicast packets sent by all senders in the group i over the simulation period

The multicast delivery ratio is an indication of the effectiveness of the multicast protocol in delivering packets in a reliable manner. Multicast delivery is affected primarily by the loss of a multicast route due to mobility and by packet loss due to collisions at the MAC layer.

The graph Figure 13 shows how the multicast delivery (expressed in percentage) varies with the number of multicast groups. The simulation environment was set up in Glomosim such that each multicast group had five receivers and one sender. Each sender used a CBR traffic generation model to generate multicast traffic for the experiment. This configuration corresponds roughly to a broadcast service such as video on demand. For a particular N value, simulation trials were carried out at various member configurations and node speeds. The mean multicast delivery is plotted.

As expected, flooding has the best delivery characteristics due to the high level of redundancy in multicast routes (Every node is a forwarder). There is, however, a dip in delivery rates as the number of groups increase. This is because occasional packets remain undelivered as a result of packet loss due to collisions at the MAC layer. (The higher amount of traffic in the network, for larger N values, increases the likelihood of there being packet loss due to collisions at the MAC layer.)

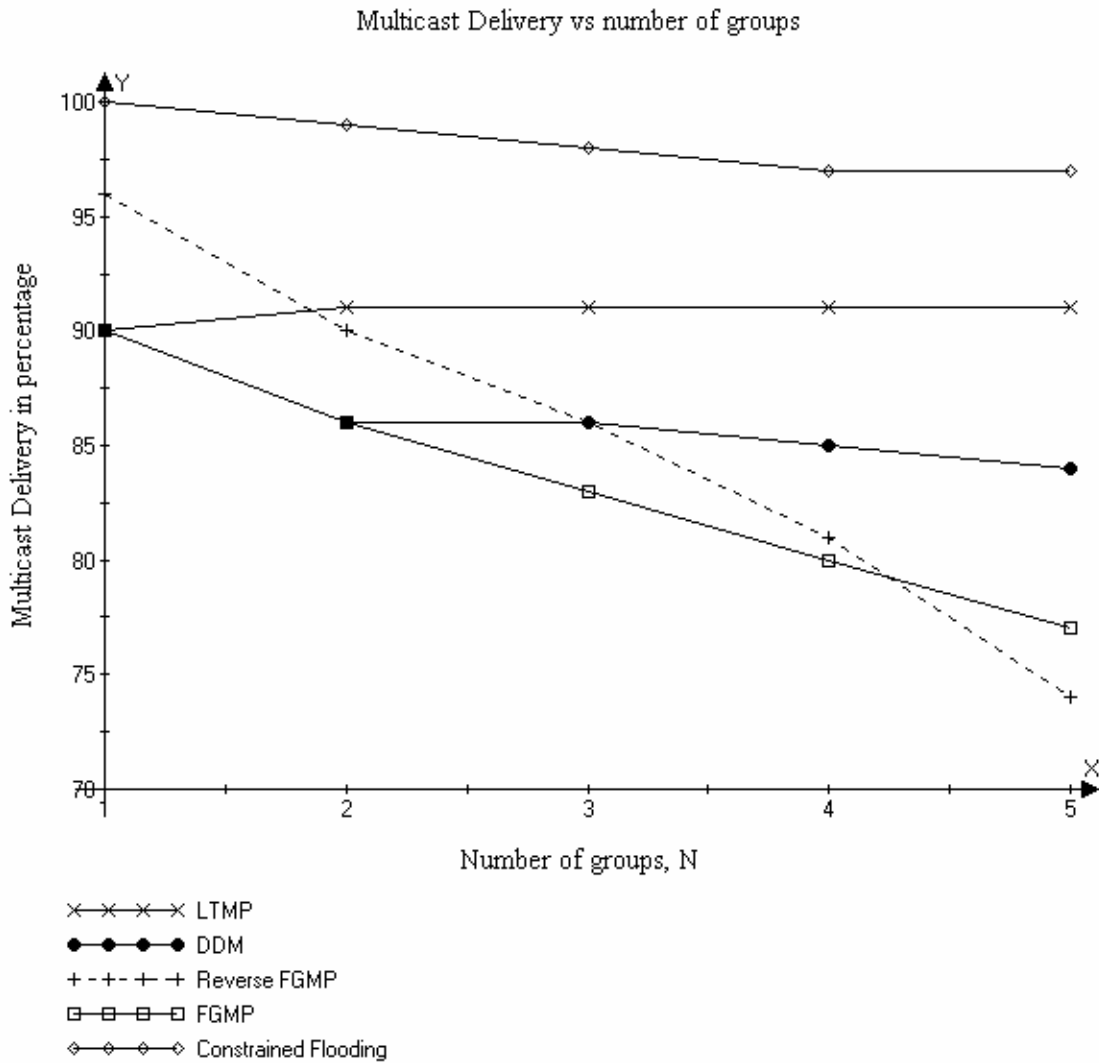


Figure 13 – Multicast Delivery vs. Number of Groups

The LTMP protocol clearly outperforms the other protocols for network configurations where the number of groups is greater than two. It can be seen from the graph that the multicast delivery of the FGMP and Reverse FGMP protocols degrades much more than LTMP's as the number of groups, N increases. This is due to the excessive amount of *control traffic* involved in these protocols, as for example is generated by receivers who

frequently advertise their membership by *globally flooding* membership packets. This *control traffic* increases as N increases and, in combination with the already high multicast traffic for large N values, results in an increased probability of packet loss due to collisions at the MAC layer. Unlike flooding, FGMP and Reverse FGMP do not have much redundancy in multicast routes. As a result, the effect of packet loss on multicast delivery is magnified.

The LTMP protocol does not employ as much unnecessary *control traffic* as the other protocols, with the result that its multicast delivery ratio is relatively higher for larger N values. It is also interesting to note that LTMP's delivery ratio is stable across varying N. This can be explained as follows; even though an increase in control traffic is expected as the number of groups increases, the LTMP protocol has been designed in such a way (with minimal inherent control traffic) that even this increase does not result in appreciable data packet loss due to collisions at the MAC layer. DDM's delivery ratio, although stable, is much lower than LTMP's.

Figure 14 shows how the multicast delivery varies with node mobility. The multicast delivery ratio (in percentage) is plotted against various values of node speeds. We used the random waypoint model in our simulation experiments. For each value of node speed, simulation trials were carried out at various member configurations and for network configurations with varying number of groups. The mean multicast delivery is plotted.

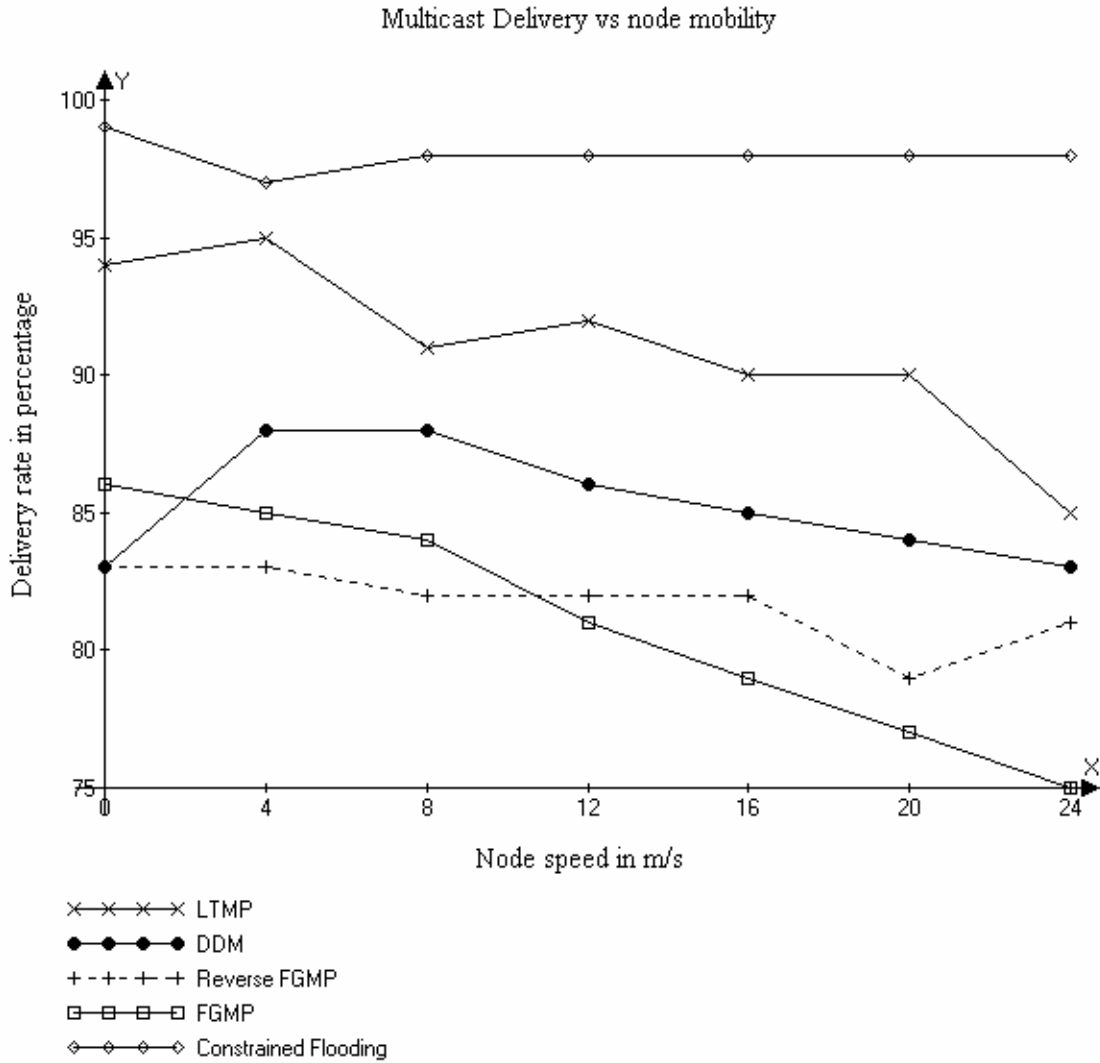


Figure 14 – Multicast Delivery vs. Node Mobility

Flooding has the best delivery characteristics among the protocols under analysis. This is because of the high redundancy in multicast routes. Also, node mobility has no effect on multicast delivery because node movement does not result in any *loss of multicast routes*, unlike in the other protocols.

LTMP outperforms other protocols at all node speeds. We believe that one reason for this could be that tree changes are *localized* i.e. a node can change its upstream dynamically and start obtaining multicast data from the sender. LTMP will thus resort to flooding only if a receiver cannot find an upstream for the group. Global Flooding results in more packets being broadcast and only increases contention for the channel (which is shared by adjacent nodes). The reduced flooding requirements of our protocol thus results in reduced packet loss due to collisions and improves multicast delivery. Also, the LTMP protocol does not generate unnecessary control traffic unlike the other protocols, and this again reduces the likelihood of packet loss due to collisions.

The other protocols, however, do frequently resort to flooding to re-establish multicast routes. Protocols such as FGMP and DDM use the underlying unicast protocol (AODV in our simulation experiments) to find routes to multicast receivers. Increased mobility would mean frequent link failures. This, in turn would require nodes to recompute routes to the receivers by *flooding* route request packets. Delivery is thus affected by the frequent absence (at least until the route discovery process is finished) of a valid route to the receiver. Also, the *flooding* requirement, coupled with other unnecessary control traffic results in increased packet loss due to collisions. This, we believe, is an important reason for the relatively low multicast delivery performance of these protocols.

As expected, the multicast delivery ratio of LTMP (and most of the other protocols) decreases as node mobility increases. This is because with higher node speeds, there is a strong possibility that the paths set up between

the multicast source and the receivers become useless more often, with the result that the delivery ratio suffers.

4.2 Multicast Efficiency:

In a wireless medium, a multicast protocol can take advantage of the MAC layer broadcast facility to reduce channel overhead. For instance, in FGMP and LTMP a multicast packet broadcast by an upstream is received and processed by each of its immediate downstream nodes. An efficient MANET multicast protocol thus makes good use of the MAC layer broadcast facility. We use a parameter called multicast efficiency to measure how effectively the broadcast facility is used to reduce channel overhead:

$$\text{Multicast efficiency ME} = \frac{\text{Total number of multicast receptions (along the path as well as at the receiver)}}{\text{Total number of multicast transmissions (at each node)}}$$

To measure the total number of multicast receptions, we accumulate the hopcount of every multicast packet at each multicast receiver. Lower values of ME imply a lot of redundant multicast transmissions. This leads to unnecessary channel overhead. The other source of channel overhead is the use of control messages. This has been studied using parameter 4.4, the number of control bytes sent per data byte received.

Figure 15 shows how the multicast efficiency varies with the number of multicast groups in the network. The network configuration is the same one used for measuring multicast delivery. As before, several trials were conducted and the average value of efficiency was plotted.

LTMP clearly has the best multicast efficiency among the protocols under analysis. This is a reflection on the average forwarding group size of LTMP, which is the lowest among the protocols under analysis. Both DDM and FGMP operate in a similar manner in that they use AODV to determine and maintain the forwarding group for a multicast sender. As a result, these two protocols have similar values for multicast efficiency. The number of groups in the network has little effect on the multicast efficiency, as can be

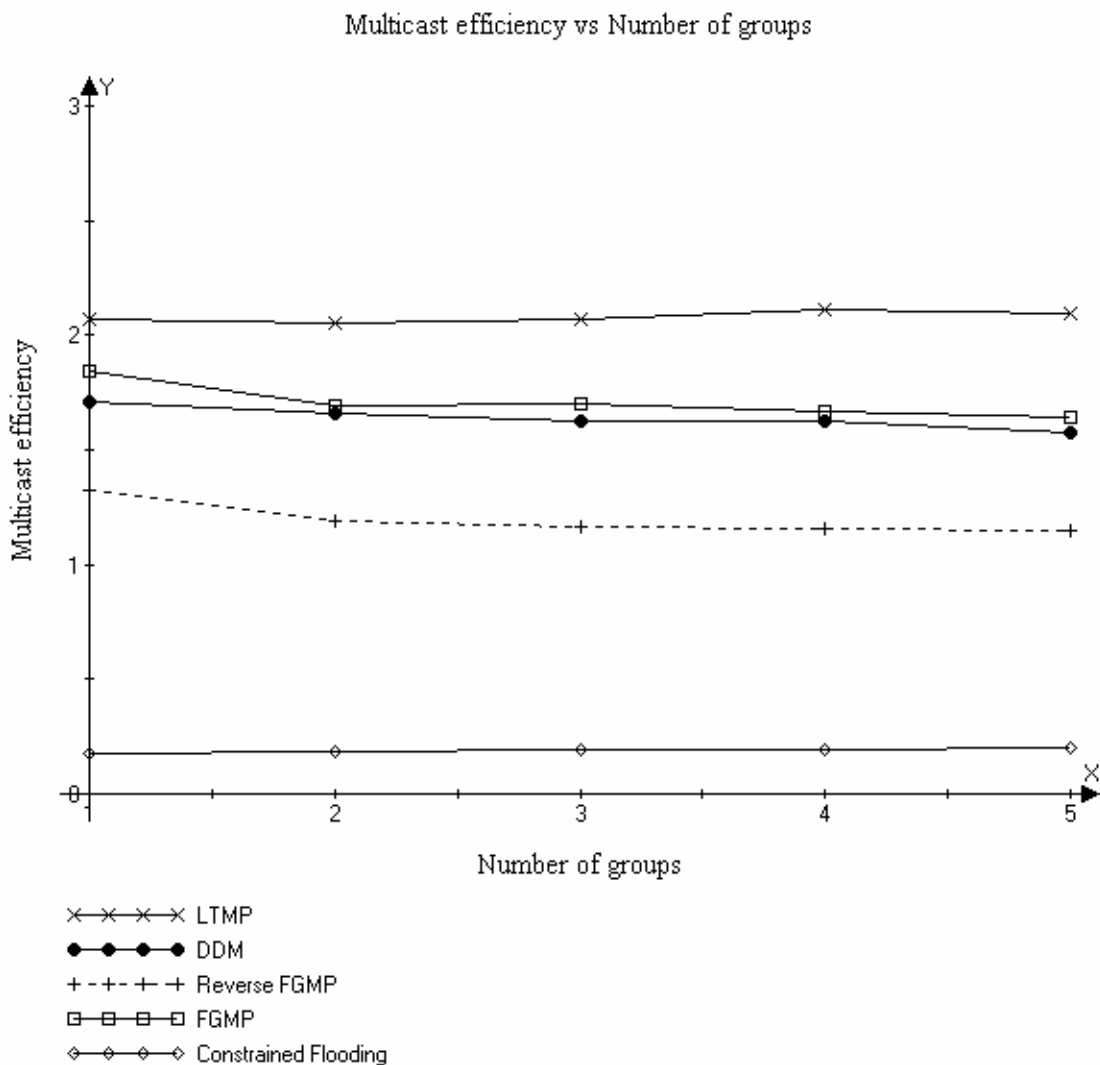


Figure 15 – Multicast Efficiency vs. Number of groups

seen from the graph.

Also, as expected, flooding is the least efficient multicast protocol (with $ME < 1$, meaning there are a lot of redundant multicast transmissions). This is because every node in the network is a multicast forwarder for all groups. As a result, nodes receive a lot of redundant multicast packets and the broadcast nature of the medium is not utilized efficiently.

Figure 16 shows how multicast efficiency varies with node mobility. It greatly resembles the previous graph. Again, LTMP has the best multicast efficiency among the protocols under analysis. As in the previous graph, flooding has the lowest multicast efficiency. Node mobility has a moderate degrading effect on the multicast efficiency of LTMP. This is again a reflection on the average FG size of the LTMP protocol, which increases slightly with increasing mobility.

Multicast efficiency vs Node mobility

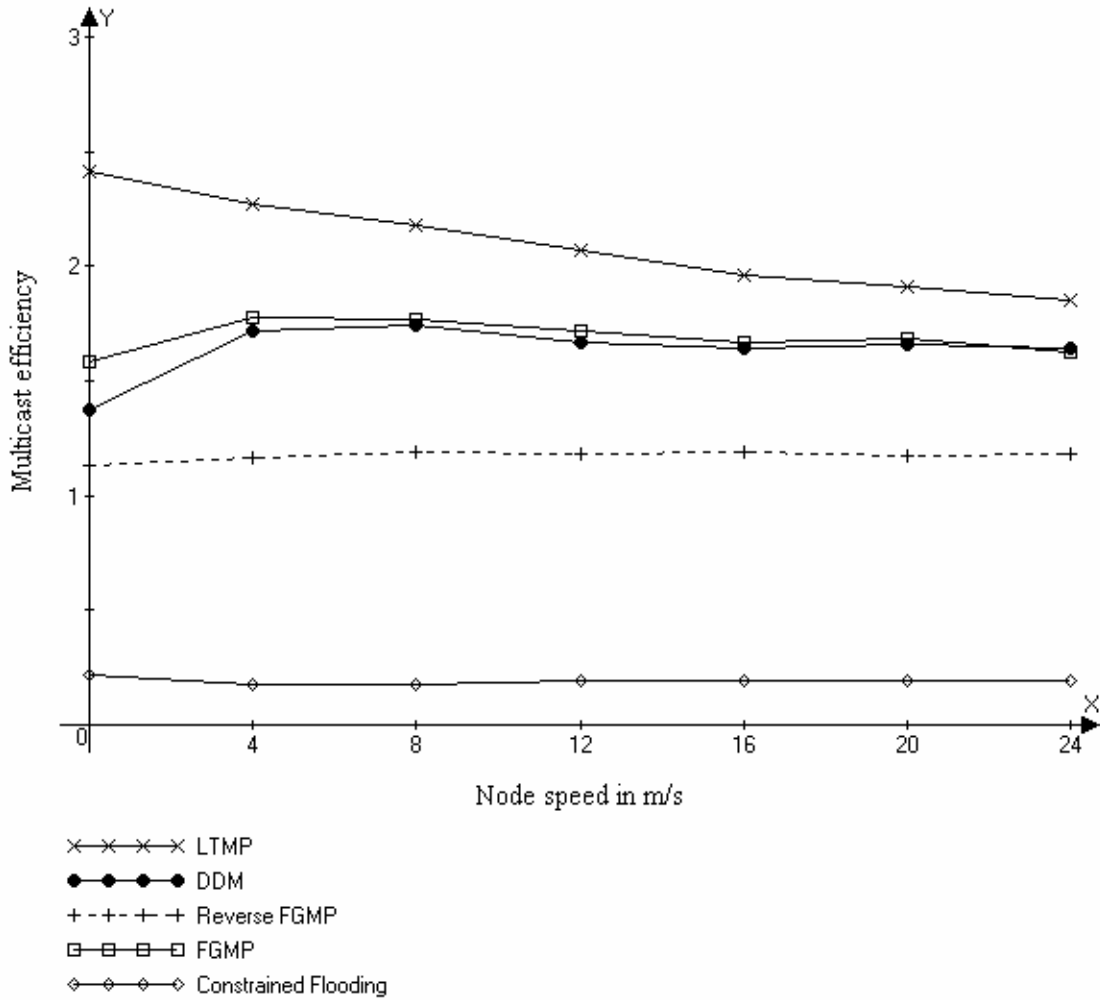


Figure 16 - Multicast Efficiency vs. Node mobility

4.3 Forwarding Group size:

The size of the forwarding group FG should be kept as small as possible to avoid unnecessary channel overhead. Also, larger forwarding groups affect multicast delivery because of the increased likelihood of packet loss due to collisions. Figure 17 shows the average size of the forwarding group for the various protocols under analysis.

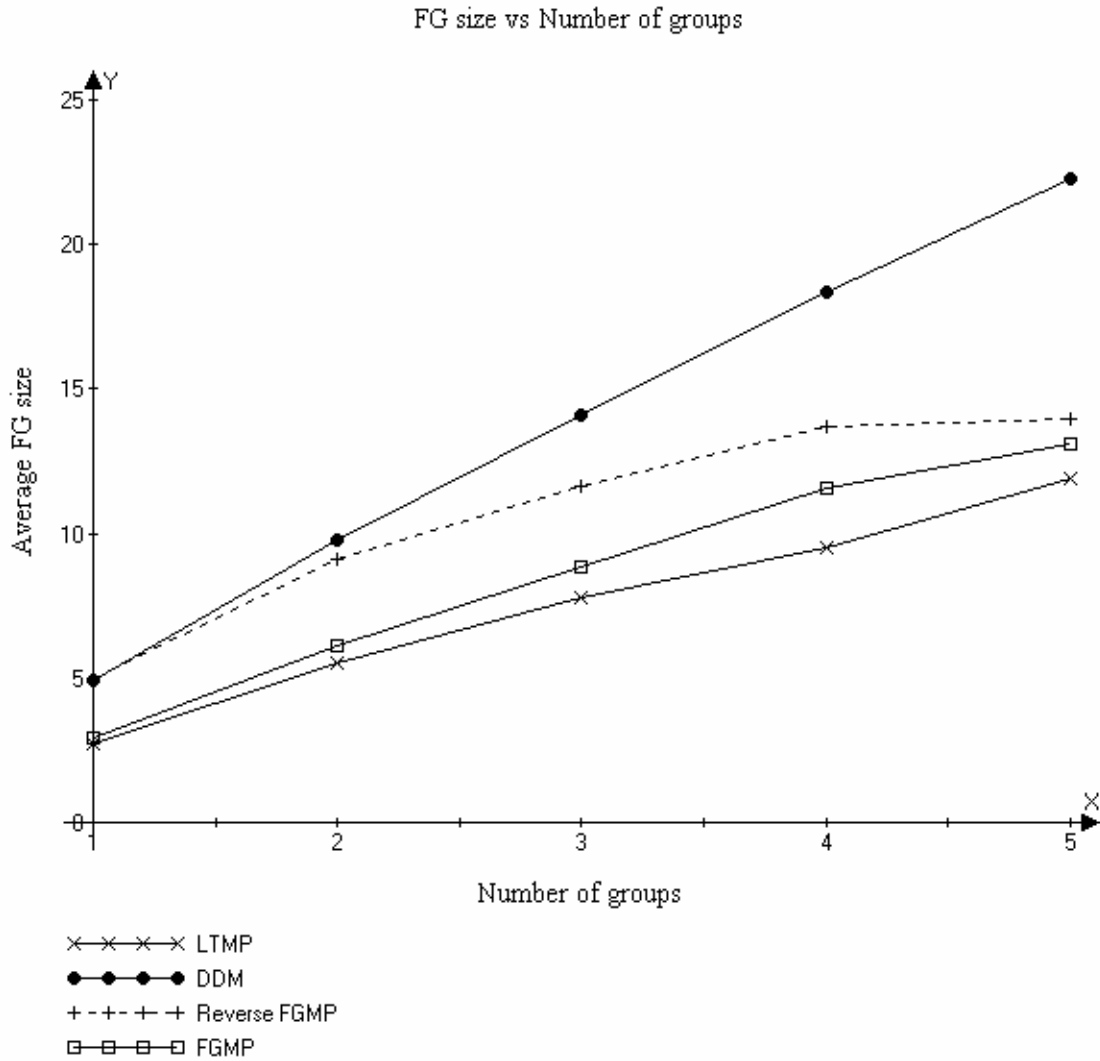


Figure 17 – Average FG size vs. Number of Groups

The graph shown above (Figure 17) shows how the average FG size (FG is assumed to include every multicast forwarder in the network and FG size is averaged over the experiment) varies with the number of groups, N . As expected, the FG size grows as more multicast groups are added to the network. It must be noted that the FG size for flooding has not been plotted in this graph. This is because the multicast flooding protocol we have

implemented uses a constant FG size of 50 (which is the number of nodes in the network- every node is a forwarder).

LTMP has the lowest average FG size among the protocols under analysis. One reason for this could be that LTMP, unlike other protocols, adds new forwarders to the FG only when absolutely necessary. Localized tree changes allow for stable multicast delivery with minimal additions to the forwarding group. New forwarders are added to the FG only when the *receiver* is unable to obtain a valid upstream to the multicast source. Protocols such as FGMP and DDM, on the other hand, frequently employ the route discovery mechanism of the AODV protocol if a valid route to the receiver does not exist. In highly mobile scenarios, involving frequent link failures, this might mean frequent additions to the FG as new routes to the receiver are discovered as part of the AODV route discovery process.

The graph (Figure 18) illustrates how the average forwarding group size (averaged over the experiment) varies with the speed of nodes in the network. As in the previous graph, LTMP has the lowest average forwarding group size. There is a slight increase in FG size as node mobility increases. This is expected, because at higher node speeds, the number of link failures due to nodes moving out of range is also higher. As a result, the number of such failures that actually get propagated to the receiver is also likely to be higher. This leads to an increase in the frequency of receiver floods, thus resulting in relatively frequent additions to the FG (when compared with low mobility scenarios).

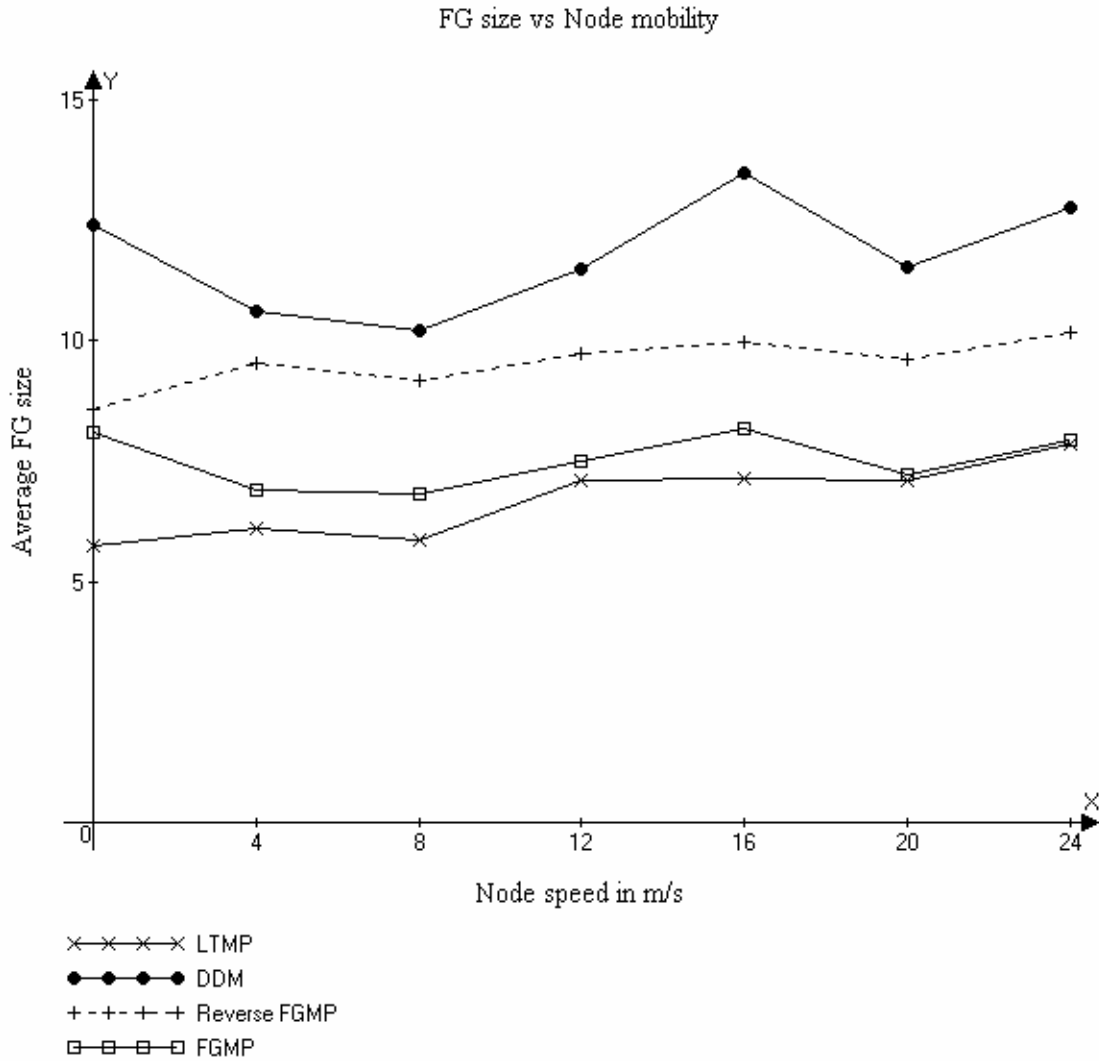


Figure 18 – Average FG size vs. Node mobility

4.4 Control Overhead:

To measure the control overhead, we use a parameter that represents the number of control bytes transmitted per data byte received. Unnecessary control messages increase channel overhead and result in increased contention for the channel. Control overhead has an adverse impact on multicast delivery, because a protocol that uses a large number of control

messages is also likely to suffer data packet loss due to collisions with control packets.

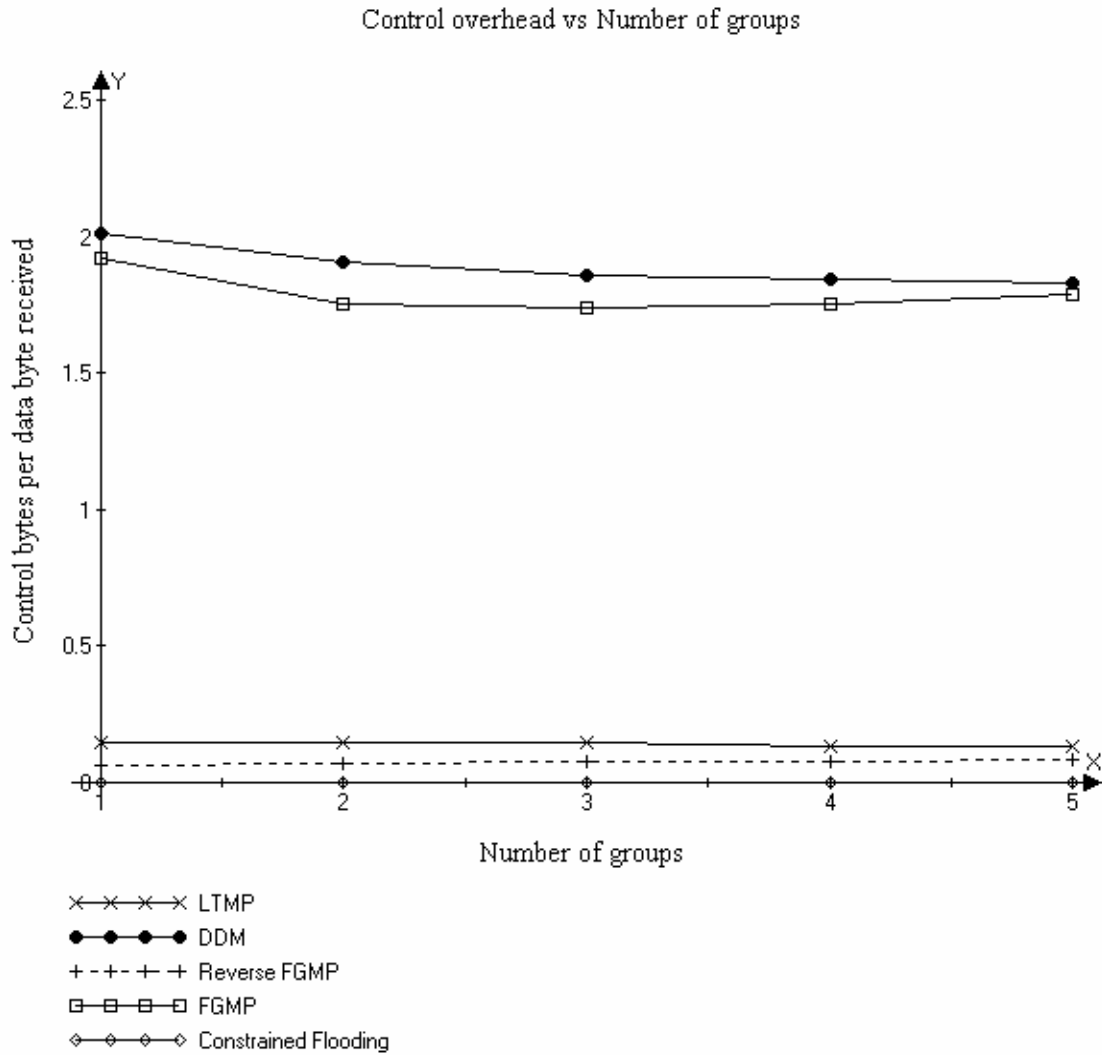


Figure 19 – Control overhead vs. Number of groups

Figure 19 shows how the control overhead varies with the number of groups for the various protocols. The Reverse FGMP protocol has the lowest control overhead among the protocols under analysis. And as expected, its control overhead is fairly constant (both in Figure 19 and in Figure 20,

which shows how the control overhead varies with node mobility). The Reverse FGMP protocol uses membership request messages that are periodically flooded by the receivers in order to refresh membership status with the multicast source and establish paths between the multicast source and its receivers. The protocol implementation does not involve any other control messages to handle link failures or other abnormal network conditions. This low control overhead of the Reverse FGMP protocol should be weighed against the fact that its delivery characteristics are poor, as seen in Figures 13 and 14.

LTMP clearly has very low control overhead, when compared to protocols like DDM and FGMP. As explained before, these two protocols use excessive control messages. These protocols, in many cases, use *global flooding* to transmit the control messages, which only adds to the control overhead. The control overhead for the flooding protocol is set to zero. This is because the flooding protocol does not use any control packets. All multicast data packets are *globally flooded* using a constrained flooding protocol.

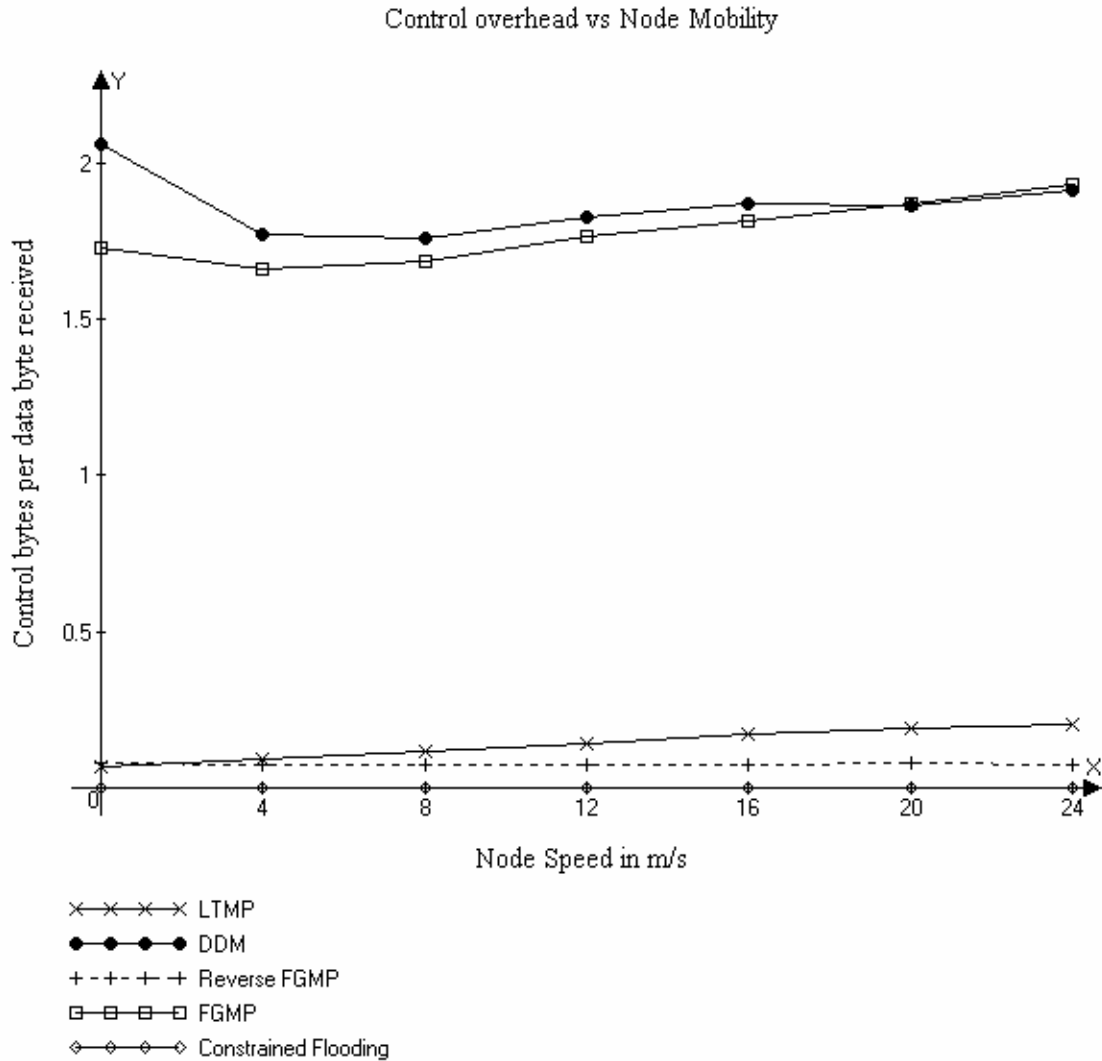


Figure 20 – Control overhead vs. Node mobility

Figure 20 shows how the control overhead varies with node mobility. In many ways, it resembles the previous graph. One noticeable feature is a gradual increase in the control overhead of the LTMP protocol as node mobility increases. As nodes become more mobile, there is a greater chance that multicast routes set up between the source and the multicast receiver(s) might become invalid. In some cases, this might lead to the receiver flooding (using a special flooding protocol aimed at reducing channel overhead) a

Join request to the source. This could be one reason for the moderate increase in control overhead with node mobility.

CHAPTER 5

CONCLUSION

In this project, we have designed and evaluated a novel source based multicast protocol for MANETs – the LTMP protocol. The key distinguishing features of our protocol are its low control overhead and its ability for localized tree change. For evaluation purposes, we have formulated and implemented our protocol in the MANET simulator, Glomosim. We have also implemented four other existing protocols for comparison purposes, namely FGMP, Reverse FGMP, DDM and Constrained Multicast flooding. From the simulation results, it is clear that LTMP outperforms most of the protocols under analysis with regard to a number of performance parameters, including multicast delivery, multicast efficiency, average forwarding group size and control overhead.

An interesting outcome of our project is an understanding of the relationship between the control traffic employed by a protocol and the delivery characteristics of that protocol in a MANET environment. From the performance evaluation, it is clear that LTMP, with its relatively lower levels of control traffic, has the best delivery characteristics (after flooding, which is hardly a feasible option in a MANET environment). Lesser control traffic translates to lesser amounts of packet loss due to collisions at the MAC layer (multicast transmissions at the MAC layer are unacknowledged). Also, a protocol with higher control traffic is likely to spend more time and bandwidth on control operations, rather than on actual data delivery. Additionally, the provision for localized tree changes in our protocol negates the need for frequent RREQ flooding by intermediate forwarder nodes

(unlike in protocols like DDM and FGMP); In LTMP, only receivers flood join requests (using a procedure designed to minimize control overhead) aimed at multicast sources.

The LTMP protocol was thus designed with reducing control overhead as top priority and as indicated by the evaluation results, does an excellent job. Though the performance of LTMP may not be comparable to mesh based protocols as network traffic increases (just like any other tree based protocol), LTMP is one of the finer tree based protocols around.

REFERENCES

1. A.Ballardie, P.Francis, and J.Cowcroft (1993), “Core Based Trees” ,Proceedings of SIGCOMM 1993
2. Ching-Chuan Chiang, Mario Gerla and Lixia Zhang (1998), “Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks”, CLUSTER COMPUTING 1998
3. S. E. Deering and D. R. Cheriton (1990), “Multicast Routing in Datagram Internetworks and Extended LANs”, ACM Transactions on Computer Systems 1990.
4. Lusheng Ji and M. Scott Corson (2001), “Differential Destination Multicast-A MANET Multicast Routing Protocol for Small Groups”, INFOCOMM 2001.
5. David B. Johnson and David A. Maltz (1996), “Dynamic Source Routing in Ad Hoc Wireless Networks”, Mobile Computing 1996.
6. Charles E. Perkins and Pravin Bhagwat (1994), “Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers”, Proceedings of SIGCOMM 1994.
7. Charles E. Perkins and Elizabeth M. Belding-Royer(1999) ,“Ad hoc On-Demand Distance Vector (AODV) Routing”, Proceedings of the 2nd IEEE Workshop on Mobile Computing systems and applications 1999.
8. Elizabeth M. Royer and Chai-keong Toh (1999), “A review of current routing protocols for ad-hoc networks”, IEEE Personal Communications 1999.