Overlay Multicast for MANETs Using Dynamic Virtual Mesh *

Chao Gui and Prasant Mohapatra †
Computer Science Department
University of California, Davis
Davis, CA 95616
Email: {guic,prasant}@cs.ucdavis.edu

October 29, 2004

Abstract

Overlay multicast protocol builds a virtual mesh spanning all member nodes of a multicast group. It employs standard unicast routing and forwarding to fulfill multicast functionality. The advantages of this approach are robustness and low overhead. However, efficiency is an issue since the generated multicast trees are normally not optimized in terms of total link cost and data delivery delay. In this paper, we propose an efficient overlay multicast protocol to tackle this problem in MANET environment. The virtual topology gradually adapts to the changes in underlying network topology in a fully distributed manner. To save control overhead, the participating nodes only keep a fisheye view of the dynamic mesh. The multicast tree is progressively adjusted according to the latest local topology information. Simulations are conducted to evaluate the tree quality. The results show that our approach solves the efficiency problem effectively.

Keywords: MANET, Overlay Multicast, Stateless Multicast, Virtual Topology, *Source-Based Steiner* Tree Algorithm.

*This research was supported in part by the National Science Foundation under the grants CCR-0296070 and ANI-0296034.

†The preliminary results of this work is presented in "Efficient Overlay Multicast in Mobile Ad Hoc Networks," Proc. IEEE WCNC 2003.

1 Introduction and Motivation

Mobile Ad Hoc Network (MANET) [1] refers to a form of infrastructureless network connecting mobile devices with wireless communication capability. The connection between any two nodes is either a single hop or a multi-hop path supported by other nodes. Most of the applications of MANETs require group communication support in the network. Various group communication models are surveyed in [2]. Among them, multicasting is the most general and a frequently used model. Multicasting in MANET faces many challenges due to the continuous changes in network topology and limited channel bandwidth. Thus, conventional multicast schemes designed for wire-line networks cannot be directly applied. Many multicast routing protocols have been proposed for MANET, which can be classified into tree-based protocols [3, 4, 5], mesh-based protocols [6, 7, 8, 9], and combined methods [10, 11].

1.1 Overlay Multicast in MANETs

Overlay multicast [12, 13, 14] has been proposed as an alternative approach for providing multicast services in the Internet. A virtual topology can be built to form an overlay network on top of the physical Internet. Each link in the virtual topology is a unicast tunnel in the physical network. The IP layer provides a best-effort unicast datagram service, while the overlay network implements all the multicast functionalities such as dynamic membership maintenance, packet duplication and multicast routing. When the overlay multicasting technique is applied to the MANETs, the manner in which the overlay layer interacts with the physical network is quite different from that of overlay multicasting in the Internet. In MANETs, each node acts as a router as well as an end host. In most cases, we can assume the bandwidth homogeneity among the nodes in a MANET topology. Whereas in the Internet topology, there is a significant difference in available bandwidth at the end hosts and the routers. Forwarding and duplicating packets at the bandwidth limited end-hosts are inherently less efficient than at the routers. Thus, there is a major efficiency problem in overlay multicasting in the Internet, compared to the network layer multicast. However, this problem does not exist for overlay multicasting in MANET.

AMRoute [11] is the first overlay multicast protocol proposed for MANETs. The protocol has two components: the mesh construction procedure and the multicast tree maintenance procedure. It uses a broadcast-based procedure to build the unicast tunnels for the virtual mesh. During the mesh construction process, a logical core node is elected for the entire mesh. Then, the core node periodically disseminates control packets within the virtual mesh to build a shared multicast tree. The data packets flow along the shared tree for delivery. Though the physical network poses dynamic topology, the virtual mesh remains unchanged once built. Figure 1 illustrates the concept of virtual mesh and the multicast tree built by the AMRoute protocol within the mesh. Using this overlay method, AMRoute does not need any support from the non-member nodes, i.e., all multicast functionality and state information are maintained within the group member nodes.

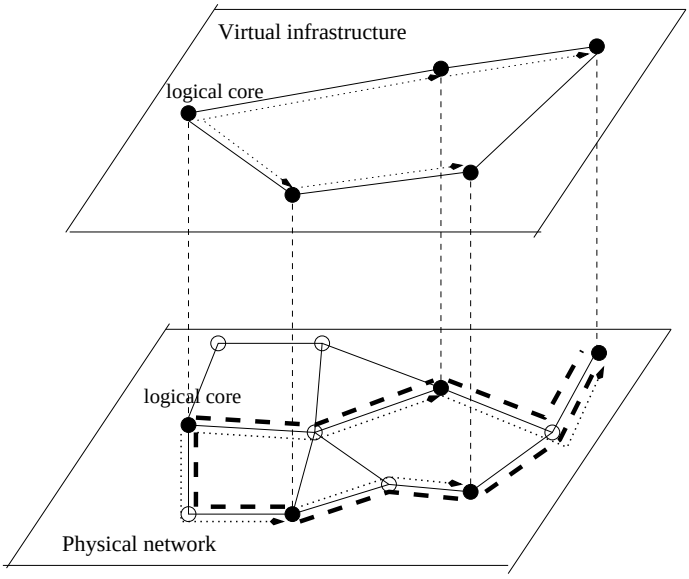


Figure 1: Concept of virtual topology for overlay multicast.

Other advantages are simplicity and flexibility. The protocol does not need to track the network mobility since it is totally handled by the underlying unicast protocols. Unlike many other multicast schemes, AMRoute protocol has no special requirements imposed on the unicast protocol. Thus, it can operate seamlessly on multiple domains that use different unicast routing protocols.

Several other overlay multicast methods are proposed for MANETs, aiming at improving the efficiency and reduce the latency of overlay multicast. They include the Location-Guided Tree Method (LGT) [15] and Prioritized Overlay Multicast (POM) [16]. Under the LGT scheme, the overlay topology is not built in the explicit manner as other protocols do. With the knowledge of the full member list and their geometric locations, the source node computes the topology of the overlay multicast tree. It uses the geometric distance between member nodes as the heuristic of link costs. Each data packet is unicasted to the children nodes on the computed virtual tree. And the tree is also encoded in the packet header so that the children nodes know how to duplicate and forward the packet. LGT complies to the "stateless" architectural principle of network protocols, i.e. there is no multicast routing tables to be maintained. However, the costs are the heavy packet headers and the cost of a universal node location service. POM builds multiple overlay trees with different priority levels. Nodes within the higher priority tree carry more important traffic flows in overlay networks and low-priority trees are rearranged whenever some nodes temporarily move to a high-priority tree. Similar to LGT, the tree formation procedure would use the location of group member nodes as heuristics of optimization. The overlay topology is dynamic in accordance to change in network traffic properties as well as node mobility.

1.2 Motivations and Objectives

For an overlay multicast protocol, the key property is the controlled spread of its state information, i.e. the multicast routing tables are kept within the group member nodes. On the other hand, the traditional MANET multicast protocols maintain state information at all network nodes, i.e., both member nodes and non-member nodes need to run the protocol in order to support the multicast session. This widespread maintenance of state information lowers the protocol robustness against the node mobility. If the routing topology involves some fast moving nodes, even though they are not member nodes, the multicast session is hampered. Further, when members join or leave the group, the state information in all the involved nodes should be updated. This is another burden to both member nodes and the participating non-member nodes. Thus, we expect overlay multicast protocols to be more robust, due to the constrained spread of state information.

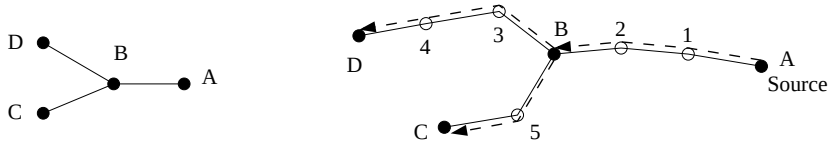
However, the advantages of overlay multicast come at the cost of low efficiency of packet delivery and long delay. When constructing the virtual infrastructure, it is very hard to prevent different unicast tunnels from sharing physical links, which results in redundant traffic on the physical links. Furthermore, if the virtual mesh remains static in the face of node mobility, it can cause performance problems such as the formation of loops and sub-optimal multicast tree. As pointed out in [17], AMRoute achieves near perfect packet delivery ratio for no mobility, and its performance drops significantly even at low mobility. To tackle this problem, we propose an overlay multicast scheme that constantly optimizes the quality of generated multicast trees.

Thus, we propose an efficient overlay multicast protocol based on dynamic overlay mesh. Our protocol is composed of two parts: the Dynamic Mesh (DM) scheme and the Progressively Adapted Sub-Tree forwarding (PAST) scheme. The DM scheme connects the group members using low-cost virtual links, and continuously eliminates high-cost virtual links from the mesh. While using the DM scheme, each member maintains a partial view of the virtual mesh. Based on the sender's local view, the multicast delivery tree is calculated, instead of forming the tree by passing control packets. Thus, the control overhead is minimized. Finally, when a packet reaches the root of a sub-tree, the sub-tree is re-calculated based on the most accurate local view of the DM. The tree construction protocol, called PAST, is used to achieve a balance between low control overhead and high multicast tree quality. Simulation studies have shown 40% reduction in average tree cost in terms of the number of physical links in groups of 40 members.

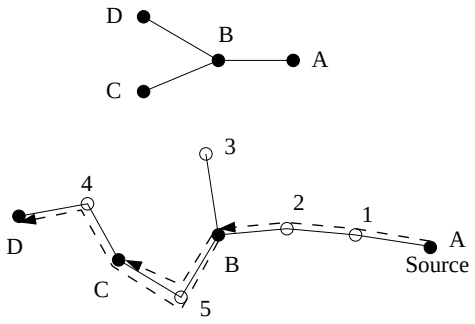
The rest of the paper is organized as follows. In Sections 2 and 3, we present our multicast protocol. The basic concepts of the protocol are introduced in Section 2, whereas the detailed procedures are given in Section 3. In Section 4, we present the results of simulation studies. The related work are discussed in Section 5. Finally, the concluding remarks are presented in Section 6.

2 Basic Concepts of PAST-DM Protocol

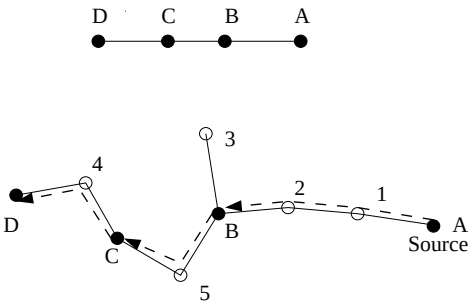
Figure 2 illustrates an example case where the static virtual topology could cause excessive redundancy. Figure 2(a) depicts an initial setup of both virtual topology and physical topology. Node A is the source node. The virtual topology can serve as a multicast routing tree for data delivery. The dashed lines in the physical topology are unicast tunnels corresponding to the virtual topology. As the nodes move, unicast routing protocol will form different tunnels for the same virtual topology, which is shown in Figure 2(b). As shown in the figure, 9 physical links are used for the virtual topology. Physical links B-5 and 5-C are redundantly included. Figure 2(c) shows the adapted virtual topology and its corresponding tunnels, which only requires 7 physical links. Initially, nodes C and D have higher hop distance and the virtual link C-D is not included in the virtual topology. However, with the movement of network nodes, the hop distance between C and D reduces significantly. With static virtual topology, the virtual link C-D can never be utilized, resulting in high redundancy. AMRoute[11] periodically rebuilds the shared routing tree in order to account for the node mobility. However, it still encounters the situations as shown in Figure 2(b) because the shared tree is always built using the static virtual mesh.



a) Initial virtual topology and physical topology



b) Changed physical topology and initial virtual topology



c) Changed physical topology and adapted virtual topology

Figure 2: Efficiency of overlay multicast.

2.1 Dynamic Mesh Maintenance by Link State Exchange

The goal of the Dynamic Mesh (DM) method is to keep the virtual mesh optimized by always using the shortest logical links (unicast tunnels) to connect the group members. The mesh is dynamic in the sense that it is updated according to the current physical topology, which prevents it from deterioration due to the node mobility. Each member node in a DM keeps track of the member nodes in its vicinity. Each node records its virtual neighbors (VN) as its *virtual link state*. The

maximum degree of the virtual topology is controlled. Each member node of a DM maintains the topology map of the virtual mesh. This is done by the link state exchange technique, which is used in the Fish-eye State Routing protocol [19] for ad hoc network. At each member node, the topology map is represented as a Link State Table (LST). The entries of the LST are the link state information of all group nodes obtained from virtual neighbors (VNs). Every node periodically exchanges this link state table with its neighbor nodes only (no flooding). Each entry in a LST carries a sequence number, and the entry with a higher sequence number will always replace the one with a lower sequence number. The link state of a node will eventually be carried to the far away nodes after several exchanges. Through the LSTs, each node has a local view of the whole virtual topology. To avoid a storm of link state exchanges, each node can make the interval between two consecutive exchanges as the period plus a small random offset. This will give a more stable overall network performance. Figure 3 shows an example network with a virtual mesh on five nodes. The four sub-figures shows the LSTs at different mesh nodes after each round of LST exchanges. The top-left sub-figure shows the initial LSTs. The top-right sub-figure illustrates the packets involved for one round of LST exchanges. The bottom-left sub-figure shows the LSTs after fist round of LST exchanges. And the bottom-right sub-figure shows the LSTs after the second round of LST exchanges. And after two rounds, each mesh node has a complete view of the virtual mesh.

The link state exchange procedure requires each group member to deliver its LST to all its VNs at the same time. This can be done by a series of unicasts. However, if all intended receivers are located within a local range of less than certain number of hops away, a TTL bounded local flooding will deliver the packet to all receivers at the same time, and much more efficiently. This is justified by the statistical results obtained from the simulations described in Section 4. For a group of 20 nodes randomly chosen from a network of 100 nodes, the average hop length of the logical links is 3.8 within the virtual mesh.

2.2 Detection of Remote Island

The TTL bounded flooding is efficient only when the TTL bound is small enough, normally no greater than 4. Thus, this method alone cannot be enough for all cases of link state exchanges. When one or a small number of clustered member nodes moves far from the cloud of the majority, long tunnels are needed to keep the virtual mesh connected. Both the ends of a long unicast tunnel should exchange LST. We term the cluster of a small number of remote members a "remote island". The cloud of the majority of group members is called the "mainland". It is notable that whether there is a remote island situation is depending on the range of TTL bounded flooding. At certain TTL value, one or a small number of member nodes can not be reached by the packets from the other member nodes. Thus, they form a remote island. However, if the TTL value had been greater, even greater by one hop, the packets from the "mainland" could have reached them. Then, there is no remote island situation.

It is necessary to make each node able to detect the case of inefficient communication due to

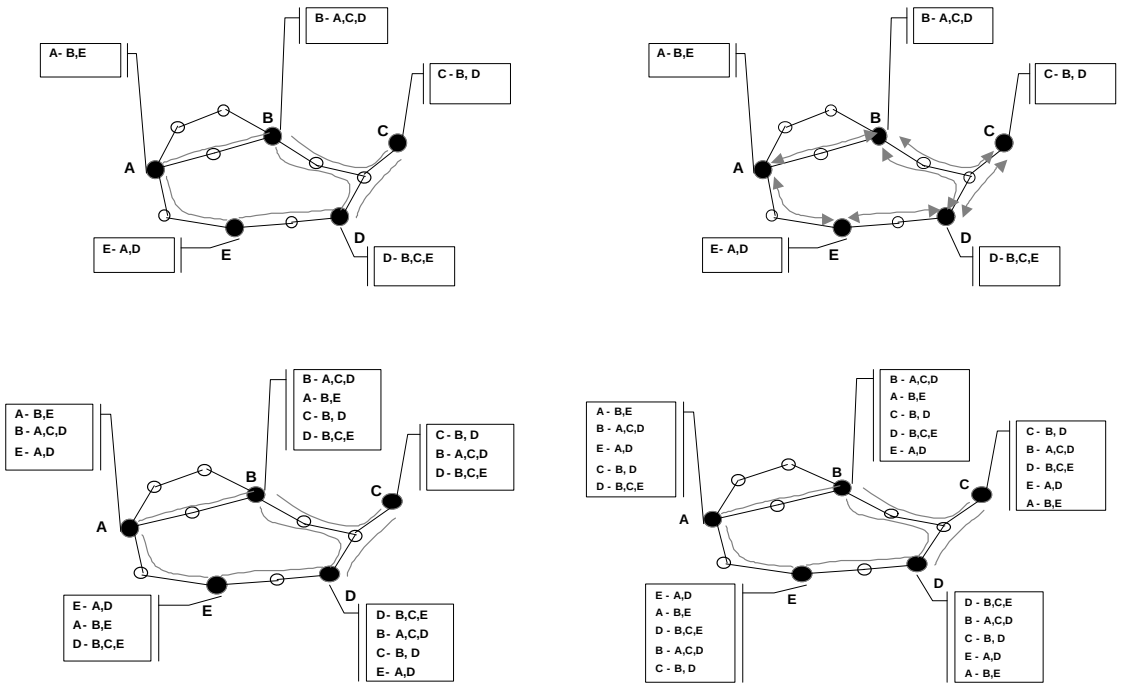


Figure 3: Example of LST Exchange

the remote island problem. The solution of this approach uses the "Leaky Bucket" algorithm. The initial level of the "Leaky Bucket" (LB) at each node is determined by its link state. Each LB leaks at constant rate. Each time a member node, I, receives a fresh LST packet from another member node, it adds certain amount of "liquid" to its own LB. When node I stays in the mainland, it constantly receives LST packets from other virtual neighbors, which maintains its LB level. Only when node I goes away far off or to an isolated island, its LST packet arrival rate drops gradually toward zero. Its LB level eventually becomes empty, which makes node I aware of its situation.

2.3 Connecting the Remote Island

For a remote island, it is important that there are bidirectional tunnels between the mainland of group members and the remote island. To ensure this, when the LB of a member node in remote island runs empty, it broadcasts a Group_REQ packet. The packet will be rebroadcasted until it reaches other group members. A group member receiving this Group_REQ packet will not rebroadcast it, but will intend to maintain a tunnel to the sender of Group_REQ. By following the distributed "Reconnection Procedure" described in Section 3.1.5, a limited number of bidirectional tunnels are set up between the remote island and the mainland.

2.4 Group_REQ Flooding

A Group_REQ packet is only rebroadcasted by non-member nodes, not by member nodes. This constraint reduces control overhead. Furthermore, this type of propagation introduces a good connectivity for the unicast tunnels. For a member node inside the mainland, there are some

member nodes certain hops away in all directions. If it broadcasts a Group_REQ packet, the packet will be consumed by the surrounding members. Thus, the higher the population density of a multicast group in the network, the higher the possibility that the Group_REQ flooding from one node is constrained within the sender's VNs. This constraint makes the bootstrap procedure feasible. Even though all group members are broadcasting Group_REQs during this stage, the overlap of the flooding ranges are insignificant. Thus, the overall overhead is much smaller than the case when all group members are broadcasting with network-wide flooding without constraining. Furthermore, the virtual mesh built from the bootstrap procedure are optimized. All logical links that makes up the virtual mesh are the shortest possible ones based on the current physical topology.

When a node in a remote island tries to connect to the mainland by broadcasting Group_REQ packets, the packet can only reach the member nodes at the border of the mainland. Since it is consumed by the member nodes, the packet have a very small chance to get inside the mainland. This property ensures that only the border nodes are maintaining the tunnels for remote islands.

2.5 Progressively Adapted Source-Based Tree

Compared to the shared tree method, the source-based tree approach is more efficient for data delivery. Each source constructs its own data delivery tree based on its local link state table. No extra overhead of control message is needed. This is the key difference between our method and other source-based tree protocols. To minimize the total cost of multicast tree, the source needs to construct a *Steiner* tree for the virtual mesh. As the local view of the virtual mesh at the source node is based on its link state table, the topology information close to the source is more up-to-date and accurate. It is progressively less accurate as the hop distance increases. Thus, if there is a tie between two virtual links with the same cost during the tree construction, the one that is closer to the source node is favored. Specifically, the virtual links adjacent to the source should always be included in the tree since they are from the most up-to-date link state information. To address this property, we propose a *Source-Based Steiner tree* algorithm as discussed in the next section.

By applying the *source-based Steiner tree* algorithm, the source makes all its neighbors its children in the multicast tree and divides the remaining nodes into subgroups. Each subgroup forms a subtree rooted at one of the first-level children. The source node does not need to compute the whole multicast tree. It puts each subgroup into a packet header, combines the header with a copy of the data packet, and unicasts the packet to the corresponding child. Each child is then responsible for delivering the data packet to all nodes in its subgroup. It does so by repeating the *Source-Based Steiner tree* algorithm. Eventually, the subgroup will become empty, and the process stops. If the subgroup has only one node, the data packet is directly unicast to the final receiver.

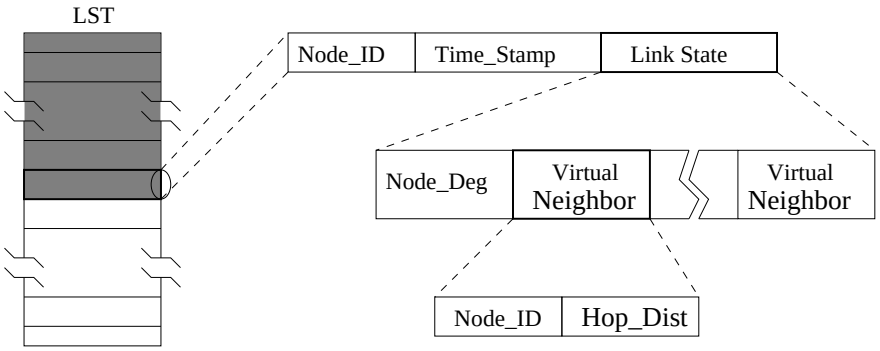


Figure 4: Link state table for virtual mesh.

3 Detailed Operations of PAST-DM Protocol

Our proposed protocol for overlay multicast is called PAST-DM. The virtual mesh topology gradually adapts to the changes of the underlying network topology in a fully distributed manner with minimum control cost. The multicast tree for packet delivery is also progressively adjusted according to the current topology. Exploiting the advantages of overlay multicast approach, the join and leave operations can be simple and robust. The descriptions of the protocol are detailed in the following subsections.

3.1 Dynamic Virtual Mesh by Link State Exchange

3.1.1 Data Structure

The data structure that is maintained at each member node is the link state table (LST) for virtual mesh, which is shown in Figure 4. Each LST entry has three fields: Node_ID, TimeStamp, and LinkState. The LinkState field is the virtual link state of the given member node, say node I, indicated by the Node_ID field. The TimeStamp field indicates the time when this link state entry is originally released from node I. Periodically, each member node releases a packet to its Virtual Neighbors (VNs), carrying a subset of its LST. This subset, named as Propagate Link State Table (PLST), contains the entries of LST which are fresher than the "Propagate_Threshold". Entries that are older than the "Propagate_Threshold" but fresher than the "Stale_Threshold" are valid but not propagate entries. Entries that are older than the "Stale_Threshold" should be expunged from the LST table. The LinkState field of each LST entry has the following structure. The Node_Deg field indicates the node degree of the given member node, I, in the virtual mesh. It is followed by a variable number of VN descriptors, each of which describes a VN of node I by its Node_ID and its physical hop distance from node I.

3.1.2 Control Packets and Formats

PAST-DM uses four control packets for the maintenance of dynamic virtual mesh. They are Group_REQ, Group_REP, Near_PLST and Far_PLST. Among these four control packets, Group_REP

and Far_PLST are unicast packets. Group_REP and Near_PLST are flooding packets, however, they are flooded with two different bounding techniques. Near_PLST packet flooding is bounded by the TTL field, i.e. the packet is locally flooded within a hop distance specified by its TTL field. The bounding method of Group_REQ packet flooding is different. The packet is only rebroadcasted by the non group member node, and it gets consumed by the group member nodes. Network-wide flooding is intrinsically costly and error-prone. These bounding methods for flooding will reduce control overhead. The purpose of these control packets and their format are summarized as follows:

- Group_REQ. This packet is flooded by each group member during the bootstrap process. A new member joins a multicast group by flooding the Group_REQ packet. Finally, when a member node detects that it moved far away from all other members and becomes a single remote member, it will flood Group_REQ packet in order to maintain connection with other members. This detection mechanism is described in the upcoming subsection. Though this packet is purely for signaling, it contains the following fields: SenderID, TimeStamp, and HopCnt. The TimeStamp field records the time when the packet is released. The HopCnt field records how many hops the packet has been forwarded. Every time the packet is rebroadcasted, the forwarding node will increment the HopCnt field by one.

- Near_PLST. This packet is used by each member node to locally flood within a limited region. This packet carries the sender's PLST information, and reaches all its VNs in the flooding range. It has the following fields: SenderID, TimeStamp, TTL, HopCnt and PLST. The PLST field is used to carry the sender's PLST. The sender assign the TTL field an initial value, and it is decremented every time the Near_PLST packet is rebroadcasted. The packet will not be further forwarded if the TTL field becomes zero.

- Group_REP. This packet is used by a receiver of the Group_REQ packet to unicast a reply back to the originator. Thus, both the receiver and the originator will regard each other as neighbors in the virtual mesh. Later in this section, it is shown that this packet is used to set up long unicast tunnels for the virtual mesh. It contains the following fields: SenderID, TimeStamp, and PLST. The node sending this packet should copy its PLST to the PLST filed of this Group_REP packet.

- Far_PLST. This packet is used to maintain long unicast tunnels in a virtual mesh. Both end nodes of a long unicast tunnel will unicast to each other the Far_PLST messages to exchange their PLSTs. It has three fields as SenderID, TimeStamp and PLST. This packet has the same format and content as the Group_REP packet.

3.1.3 Bootstrap Procedure

At the bootstrap stage of a multicast session, all member nodes first broadcast a Group_REQ packet. Each member then waits for a period termed as REQCollect_Period to collect Group_REQ

packets from other group members. When a node receives a Group_REQ packet, the receiver can know the current hop distance between the receiver and the originator from the packet's HopCnt field. The Group_REQ packets with HopCnt field less than a threshold value termed as REQHop_Threshold are regarded as normal REQs. Other Group_REQ packets are regarded as long REQs. All member nodes follow the same bootstrap protocol. We use one member node, I, as an example to describe the process. Assume node I has collected n normal REQs and l long REQs during the waiting period. Node I selects its VNs from the senders of these $n + l$ REQs, and further determine the bounding TTL value of its Near_PLST packets. Each collected normal REQ is represented by a two-tuple <HopCnt, TimeStamp>. Node I first sorts the normal REQs by their tuples in an increasing order of HopCnt field and decreasing order of TimeStamp field. Let the sorted normal REQs be of the following order: $Rn_1^I, Rn_2^I, \cdots, Rn_n^I$. The TTL bound value for node I will be:

$$PLST\_TTL(I) = \min\{HopCnt(Rn_{VDB}^I),\ PLST\_TTLMAX\},$$

where VDB is the bound on node degrees in the virtual mesh, and PLST_TTLMAX is the bound on PLST_TTL values for all nodes.

The receivers of the long REQs should be responsible for making sure that the senders of the long REQs are connected to the main mesh. To achieve this objective, node I randomly selects k REQs, $\{Rl_1, Rl_2, \cdots, Rl_l\}$, from the l collected long REQs, and regards the senders of $\{Rl_1, Rl_2, \cdots, Rl_l\}$ as its far VNs. To ensure mesh connectivity, especially when the multicast group population is sparse in the network, one should choose large k values, or, even enforce k be equal to l. On the other hand, larger k value results in more longer links in the mesh, and more overhead in bootstrap and maintenance procedure. In our simulation setups, we can achieve good result with k being set as one. Up to now, node I can determine its VN set as the following:

$$\begin{aligned} VNset(I) &= VNset\_Near(I) \cup VNset\_Far(I);\ where \\ VNset\_Near(I) &= \{SenderID(Rn_j^I) \mid HopCnt(Rn_j^I) \le PLST\_TTL(I), j \in [1,n]\}\ and \\ VNset\_Far &= \{SenderID(R_l)\} \end{aligned}$$

SenderID(R) indicates the sender of the REQ packet R. This VN set is actually node I's initial link state in the virtual mesh. It is the first entry that node I can fill into its Link State Table (LST). Node I then need to unicast a Group_REP packet back to $SenderID(R_l)$, carrying its current PLST. The detailed steps are represented in Algorithm 1.

3.1.4 Maintenance Procedure

When the member nodes have determined their initial link states, they begin their maintenance procedure. Therefore, a multicast session enters the maintenance stage. The main operation of the maintenance stage is to keep up the periodical link state exchange among the group members. To illustrate the maintenance procedure, let us consider an example with node I as a group member. After a period termed as PLST_EXPeriod, node I repetitively generates a Near_PLST packet.

Algorithm 1: Bootstrap Procedure

```
Data: Group_REQ packet
Result: Initialize the VNset, the lists of neighbors on the virtual mesh
begin
    Initialize the NormalREQ_List and LongREQ_List;
    Broadcast Group_REQ packet;
    Wait for Group_REQ packets from other member nodes;
    foreach received Group_REQ packet, denoted as R do
        if HopDistance(ThisNode, SenderID(R)) < NormalREQ_MAX then
            put R into NormalREQ_List;
        else
            put R into LongREQ_List;
        end
    end
    Select member nodes for VNset from the NormalREQ_List and LongREQ_List;
    Set TTL for PLST packets;
end
```

The TTL field of the packet is assigned the value of PLST_TTL(I), and the PLST field is filled by node I's current PLST. The packet is then broadcasted by node I and further flooded within its local range. Both member and non-member nodes will rebroadcast the packet, and its HopCnt field get incremented every time by the forwarding node. If its far VN set is not empty, node I should immediately unicast Far_PLST packets to all its far VNs. This combined unicast/broadcast method is the most efficient way of propagating node I's current PLST to all its VNs.

After propagating its PLST, node I uses the rest of the period collecting fresh Near_PLST packets from other members. A PLST packet, P, is regarded as fresh according to the following definition. If SenderID(P) is in node I's VN set, the TimeStamp(P) should be later than the last received PLST packet from SenderID(P). Otherwise, the TimeStamp(P) should be later than any PLST packet collected in this period. At the end of the period, node I adjust its PLST_TTL parameter based on the collected Near_PSLT packets and their HopCnt fields. Using the same process as the initial PLST_TTL determined during the bootstrap stage, node I determines its PLST_TTL value for the next period. Node I's link state is also updated at the same time, as node I updates its VN set. The VNset_Near(I), which represents node I's near VNs are updated as the senders of the newly collected PLST_Near packets during current period. In spite of updating the near VN set, node I keeps its far VN set the same.

Another function of the maintenance procedure is to detect if one or a small number of members have moved far away from all other members. This is done using a "leaky bucket" algorithm. At the beginning, node I decides the initial level of its "leaky bucket", LB(I), as the following:

$$LB\_Level(I) = \begin{cases} \|VNset\_Near(I)\| \times LBUnit\_Normal & : \ if\ VNset\_Near(I) \neq \emptyset \\ \|VNset\_Far(I)\| \times LBUnit\_Far & : \ otherwise \end{cases}$$

During the procedure, LB(I) is leaking at constant rate, termed as the LBLeak_Rate. Certain amount of "liquid" is added each time node I receives a fresh PLST packet. Depending on the

PLST packet being Near_PLST or Far_PLST, the added "liquid" amount is $LBAddUnit\_Normal$ or $LBAddUnit\_Far$, respectively. Given the fact that a node generally has less far VNs than near VNs, and it takes longer delay for Far_PLST packets to reach its destination, the parameter $LBUnit\_Far$ is larger than $LBUnit\_Normal$, and similarly is $LBAddUnit\_Far$ is larger than $LBAddUnit\_Near$.

To achieve higher accuracy, we propose to add variable amount of "liquid" upon receiving a fresh PLST packet. The added liquid is proportional to the number of hops that the message has traveled. Thus, depending on the PLST packet being Near_PLST or Far_PLST, the amount of added liquid is $LBAddUnit\_Near \times Hops(PLSTpacket)$ or $LBAddUnit\_Far \times Hops(PLSTpacket)$, respectively. The detailed steps are represented in Algorithm 2.

Algorithm 2: Mesh Maintenance Procedure

```
Data: PLST packet
Result: Maintain the VNset, adjust the LB level
begin
    Set initial LB level;
    At periods in length as PLST_EXPeriod, do begin
        Generate and broadcast Near_PLST packet;
        if VNset_Far(ThisNode) is not empty then
            Unicast Far_PLST packet;
        if LB level == 0 then enter Reconnection Procedure;
        Wait for PLST packets;
        foreach received PLST packet do
            Update VNset;
            if PLST packet is Near_PLST then
                amount of added liquid to LB level is
                LBAddUnit_Near × Hops(PLSTpacket);
            if PLST packet is Far_PLST then
                amount of added liquid to LB level is
                LBAddUnit_Far × Hops(PLSTpacket);
    end
end
```

3.1.5 Reconnection Procedure

The purpose of this procedure is to setup and maintain the bi-directional tunnels between the mainland of group members and a remote island. When the LB level at a member node, I, becomes empty, it should start the reconnect procedure. Node I first broadcasts a Group_REQ packet, and the packet is rebroadcasted until it reaches the other group members. All other member nodes, which are in the maintenance stage should respond when receiving a Group_REQ packet. Suppose node J has received a Group_REQ packet, R^I from node I. If HopCnt(R^I) is not greater than PLST_TTL(J), node J will ignore the packet as an abnormal event. Otherwise, node J should include I into its far VN set. J then unicasts back to node I the Group_REP packet, carrying its

current PLST. Node I waits for a period REPCollect_Period, to collect any responding Group_REP packets. Suppose node I collect p REPs during the period, it only needs to maintain certain number of the collected REPs, and make the senders of those REPs as its VNs. It then unicasts each of its VNs the Far_PLST packets with its current PLST. At the chosen VNs, upon receiving the Far_PLST, the link state entry in LST that includes this unicast tunnel will be refreshed. At the senders of other Group_REPs, which are not chosen by the destination node, the LST entry will soon become outdated, and they cease to keep the unicast tunnel. In this way, a controlled number of tunnels is setup between the mainland and a remote island. The detailed steps are represented in Algorithm 3.

Algorithm 3: Reconnection Procedure

```
Data: Group_REP packet
Result: Re-establish the list of neighbors on the virtual mesh
begin
    Broadcast Group_REQ packet;
    Wait for Group_REP packets from other member nodes;
    foreach received Group_REP packet, denoted as P do
        put P into REP_List;
    end
    Set VNset(ThisNode) according to REP_List;
    Unicast PLST packets to nodes in VNset;
end
```

3.2 Progressively Adapted Source-Based Tree (PAST) Algorithm

As discussed in the previous section, the link state exchange algorithm for DM maintenance gives a special property to the LSTs in the member nodes. The link state entries are progressively less accurate with increasing hop distance. To address this issue, we developed a novel *Source-Based Steiner* tree algorithm for the tree computation.

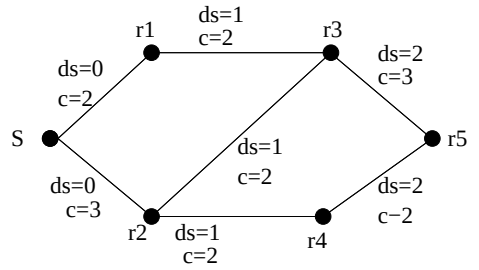
Let $ds(n)$ denote the hop distance from source node s to node n (regardless of the costs on the links). For a virtual link (n1,n2), its hop distance to source node is defined as follows.

$$ds(n_1, n_2) = min[ds(n_1), ds(n_2)].$$

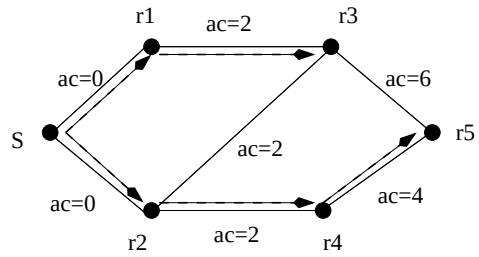
Let $c(n_1, n_2)$ be the cost of the virtual link (n_1, n_2). We define the *"adapted cost"* of the link as its cost multiplied by its distance to the source.

$$ac(n_1, n_2) = ds(n_1, n_2) \cdot c(n_1, n_2).$$

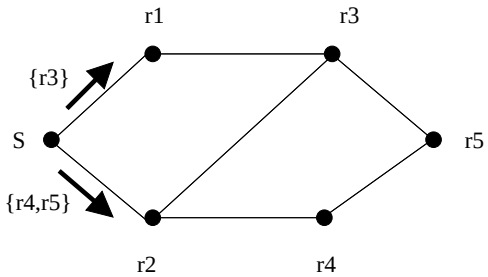
Thus, for any link that is adjacent to the source node, its distance value and the *adapted cost* should both be 0. With each link using its *adapted cost* value, we apply the following heuristic [20] for computing the *Steiner* tree. With the zero-cost links, the initial tree can have the source node as root and all neighbors as its first-level children. The partially constructed tree expands toward the *Steiner* tree by including the nearest receiver to it, together with the shortest path connecting them.



(a) Distance and cost of each link



(b)Adapted cost of each link and Source−Based Steiner tree



(c) Receiver lists in the header of the packets sent from S to its children

Figure 5: Example of Tree Construction.

Figure 5 shows an example. The source node is S, and its receiver list is $\{r1 \cdots r5\}$. Figure 5(a) shows its local view of virtual topology. The *source-based Steiner* tree using adapted costs is shown in Figure 5(b). S generates two smaller lists: {r3} and {r4, r5}. They are included into the header of the packets sent to r1 and r2 as shown in Figure 5(c). The detailed steps are represented in Algorithm 4.

To reduce the computational complexity, the source node can cache the recent tree construction result for the incoming data packets. This cached result can be kept valid until the local link state table is updated during the next incoming link state exchange.

3.3 Join and Leave

Contrary to the complicated join and leave process with conventional "stateful" multicast, overlay multicast supports dynamic membership in a simple and robust manner. When a node intends to join the multicast group, it starts with a normal neighbor discovery process described in Section 3.1. As multiple groups may co-exist in the network, it needs a group address in its Group_REQ packets. As the member nodes of the intended group respond with Group_REP packets, it can collect its

Algorithm 4: PAST Algorithm (for group member node "I")

Data: Virtual link state table local at node I; and header of the data packet;
Result: Partition of node I's sub-tree into smaller trees, one for each children;
begin
  foreach *pair of group member nodes* n_1, n_2 **do**
    if n_1 *and* n_2 *are neighbors in link state table* **then**
      Edge (n_1,n_2) is included in graph G[V,E];
      Assign weight to (n_1,n_2) as: $ac(n_1, n_2) = ds(n_1, n_2) \cdot c(n_1, n_2)$;
  end
  Use SubTreeList(I) denote the node list in the packet header to node I;
  Apply *Steiner* Tree algorithm to G on its node subset SubTreeList(I), the result is SubTree(I);
  foreach *node J as the direct children of I in SubTree(I)* **do**
    SubTreeList(J) is the set of all J's descendants in SubTree(I);
    Make a copy of the data packet;
    Output SubTreeList(J) into the header of the new packet;
    Forward the data packet to J;
  end
end

own virtual neighbors and set up its own link state. As the responding group nodes also include the newcomer as their neighbor, they will start to exchange link state tables with the new member. In this way, the new member will gradually build its own view of the virtual mesh. At the same time, the new member's neighbors on the virtual mesh will exchange their new link state to their neighbors, carrying the new member's identity further away into the mesh.

The new member's identity will eventually reach any data source node in the mesh, then, it can be included in the header of later data packets. However, the new member can start receiving data packets right after it is recognized by its virtual neighbors. This is achieved by an additional data forwarding function on each child node on the multicast tree. When a child node has forwarded the data packet to all nodes in its subgroup, it checks if it has any virtual neighbor as a new member in the mesh. A new member can indicate its newcomer status in its first Group_REQ packets. An additional unicast is needed for the missing newly joined neighboring receiver. As a new member may have multiple virtual neighbors, it will receive multiple copies of the same data packet. It should discard the duplicate ones. However, once the source recognizes the new member and puts it into the delivery list, it will no longer receive any more duplicate packets.

To leave the group, a member node needs to unicast a Group_LV message to its current virtual neighbors. The link state exchange between it and its neighbors will stop. The neighbors will not deliver the data packet to it even though it may still appear in the subgroup node list for a while.

4 Performance Evaluation

4.1 Simulation Configuration

Our simulation network has 100 mobile nodes randomly roaming within a 2000m × 750m free space. The radio transmission range of each node is 250 meters. In this setup, the probability of partition in network topology is relatively small. The average hop distance is 4.4. The longest hop distance is 11. Each simulation run lasts for 500 simulation seconds. The movement of each node follows the random waypoint model [21]. Each node selects a destination location randomly within the roaming area and moves straight toward the destination with a constant speed. The moving speed is uniformly distributed over $[\frac{Max}{2}, Max]$ m/s, where MAX is the maximum moving speed normally set as 20 m/s. For the multicast sessions, we choose the group size to be 5, 10, 20, 30 and 40.

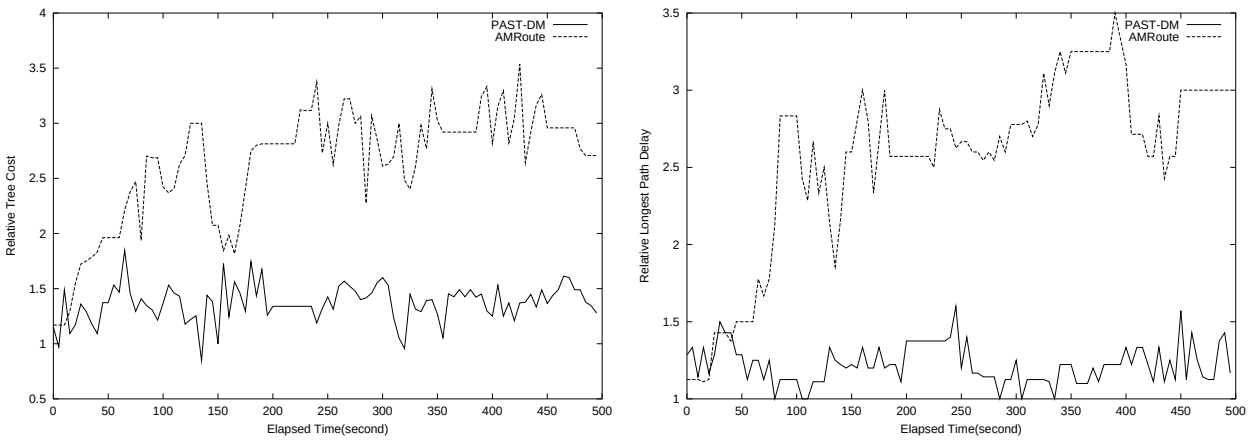
This performance study contains two parts. The first part, as shown in Section 4.2, is for evaluation of multicast tree quality. We use a specially developed simulator to study the topology of multicast tree constructed by the PAST-DM protocol, compared with that of AMRoute and the optimal tree. The simulator implements the random waypoint mobility model, and generates the instant network topology at each point of simulation time. The second part, as shown in Section 4.3, is for evaluation of protocol performance, in terms of control overhead and delivery ratio. We use the wireless network simulator, GloMoSim, for this part of study.

4.2 Study of Multicast Tree Quality

To study the performance of multicast tree quality, we measure the following metrics: (a) (relative) tree cost; and (b) (relative) longest path delay. The cost of a data delivery tree is the sum of physical hop lengths of all virtual links of the tree. Longest path delay is the number of physical hops along the longest path from the source to any of the receivers on the tree. In order to compare the efficiency of overlay multicast method, we compute the optimal tree cost and optimal longest path delay at each simulation step. The optimal tree cost is computed as the overall hops of a *Steiner* tree built on the physical topology for the same group of nodes. The optimal longest path delay is the longest path delay of the *Shortest Path Tree* built on the physical topology with the same source node. By definition, the optimal values are the best we can have based on existing physical network topology. The relative value of the metrics are the ratio of measured value over the corresponding optimal value, which represent quality and efficiency of the generated multicast trees.

4.2.1 Time-line of tree quality

Figures 6(a) and (b) show the time-line of multicast tree quality for a multicast group of size 40 under AMRoute and PAST-DM. The tree quality indicators considered here are relative tree cost and relative longest path delay, which are shown in the two figures respectively. The period of AMRoute tree creation process is 40 simulation seconds. The period of virtual link state exchange



(a) Relative Tree Cost.

(b) Relative longest path delay.

Figure 6: Time-line of multicast tree quality (group size is 40).

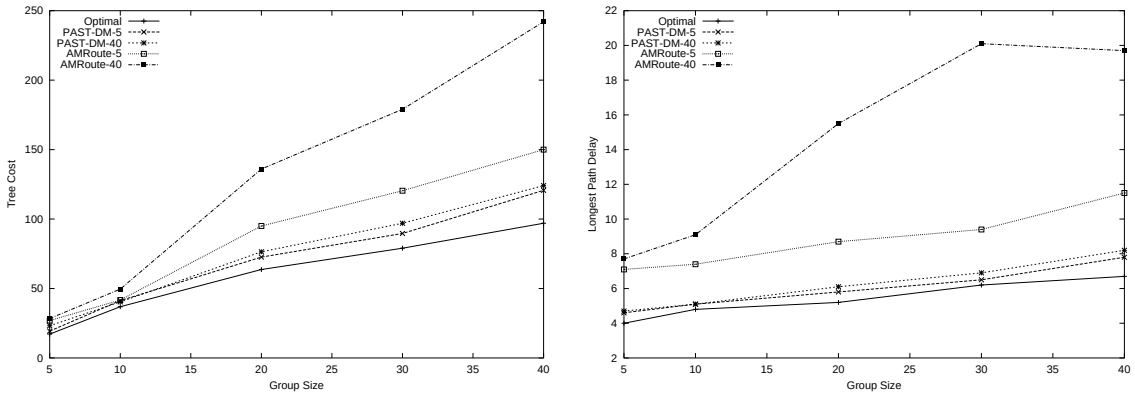
for PAST-DM is also set to 40 simulation seconds.

In both figures, the time-lines of the two protocols are intermingled together during the first 20 seconds. This is because the virtual mesh is being just built up by AMRoute. As the mobile nodes have not moved far from their initial places, the virtual mesh still reflects the underlying physical topology. As time elapses, both the relative tree cost and the relative longest path delay of AMRoute increase rapidly. PAST-DM, however, presents a stable tree quality, which is the effect of the dynamic virtual mesh protocol implemented by the virtual link state exchange process. A noticeable gap between the time-lines in both sub-figures indicates that PAST-DM consistently builds more efficient multicast trees than AMRoute.

The oscillation of the time-lines is also noticeable. High oscillation in relative tree cost will result in unstable network traffic, while oscillation in relative longest path delay can cause out-of-order delivery problem. In both sub-figures, the time-lines for PAST-DM have much less oscillation compared to those of AMRoute. This will yield a much more stable network performance for the PAST-DM protocol.

4.2.2 Tree quality versus group size

Figures 7(a) and 7(b) show the average tree cost and longest path delay of the multicast trees versus the group size. At each simulation step of PAST-DM protocol, a data delivery tree is constructed with the source randomly chosen from the group members. The average tree cost and longest path delay is recorded for all step-wise multicast trees. For PAST-DM, we set the parameter of update period to be 5 seconds and 40 seconds, which is labeled as "PAST-DM-5" and "PAST-DM-40" in the figures. Similarly, the legend labels "AMRoute-5" and "AMRoute-40" mean the AMRoute protocol with 5 and 40 second of tree re-build period. PAST-DM yields close to optimal trees for all



(a) Tree Cost.

(b) longest path delay.

Figure 7: Comparison of multicast tree quality with different group size.

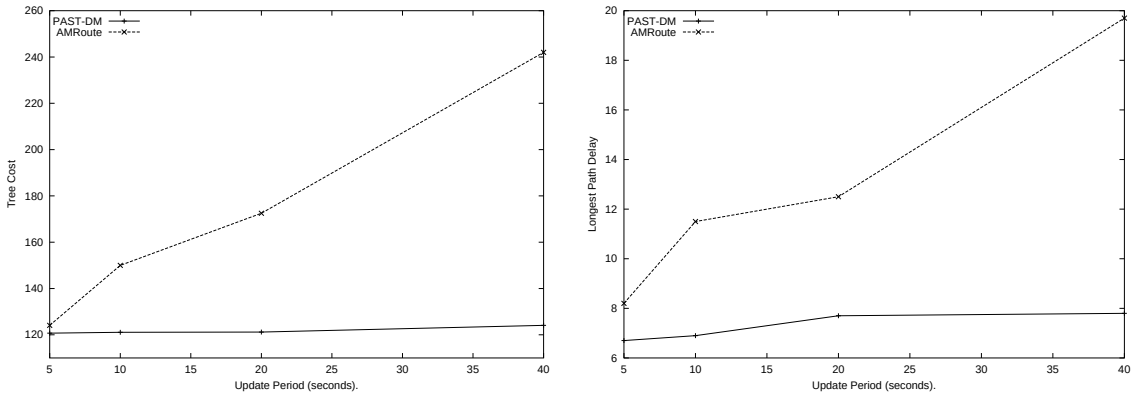
group sizes. The cost of the trees built by AMRoute increases faster than the optimal value as the group size increases. The same is true for the longest path delay. Thus the multicast trees become less efficient as the multicast group grows. We can also calculate the average hop length of virtual links in the multicast tree. For example, for any delivery tree spanning a group of 20 nodes needs just 19 virtual links. The total costs of trees by AMRoute-5 and PAST-DM-5 are 95.9 and 72.6 respectively. So the average length of virtual links in both trees are 5.0 and 3.8.

4.2.3 Tree quality versus update period

Figures 8(a) and 8(b) show the change of multicast tree quality with different update periods. The update period for AMRoute and PAST-DM refers to the tree re-creation period and the virtual link state exchange period, respectively. As higher update period results in less protocol overhead, we are looking for a high update period that does not result in much worse tree quality. As shown in the figure, tree cost and longest path delay of AMRoute increases rapidly with update period. This means it needs frequent re-creation of multicast tree to keep up with the change of physical hop lengths of the virtual links. PAST-DM yields a stable tree quality as update period increases. It is less sensitive to the frequency of link state exchange. Thus, with PAST-DM, we can make the link state exchange among group members to be less frequent in order to reduce the overhead of the protocol.

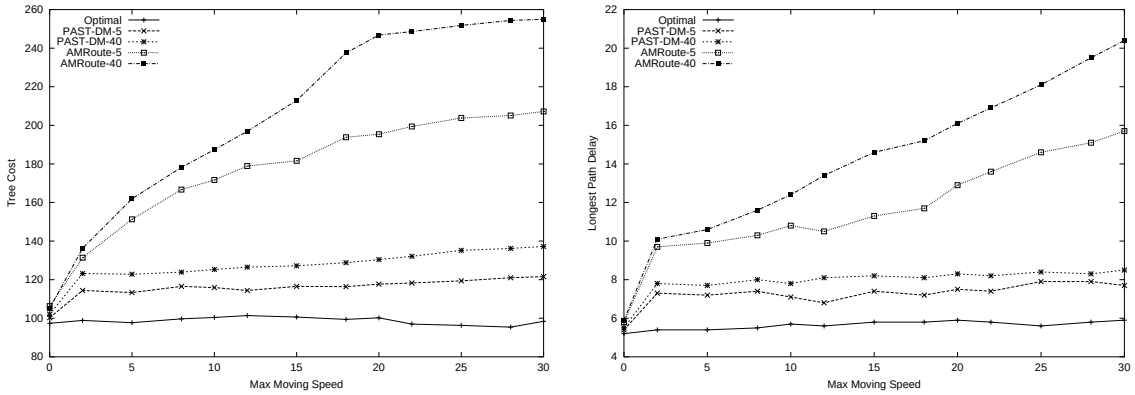
4.2.4 Tree quality versus mobility

Figure 9(a) and 9(b) show the change of tree quality with respect to different mobility speeds. We use the *random waypoint* mobility model, and the actual speed is randomly chosen from the range of $[\frac{Max}{2}, Max]$. The maximum speed ranges from 0 to 30 m/s. The non-zero minimum speed prevents the harmful situations in *random waypoint* model discussed in [22].



(a) Tree Cost. (b) longest path delay.

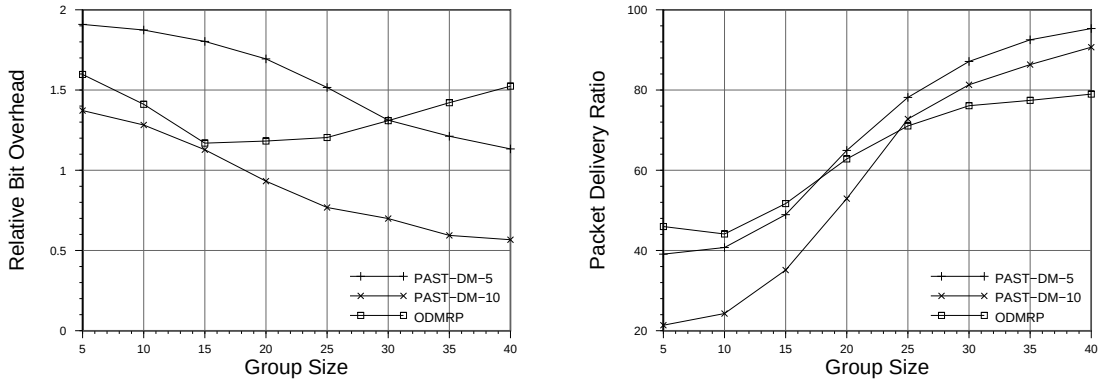
Figure 8: Comparison of multicast tree quality with different update period (group size is 40).



(a) Tree Cost. (b) Longest path delay.

Figure 9: Comparison of multicast tree quality with node mobility (group size is 40).

For both tree quality metrics, the protocols present similar performance value which are close to the optimal, when the moving speed is zero. As mobility increases, the average tree cost for both AMRoute-5 and AMRoute-40 increases dramatically as shown in Figure 9(a). Beyond 20 m/s moving speed, the increase of tree cost slows down because the formation of sub-optimal trees is close to a saturated situation. However, the tree cost value of DM-PAST is less than half of the value of AMRoute. It remains stable in the face of increasing node mobility. The longest path delay has similar trend, as shown in Figure 9(b). It is slightly different in the sense that the increase of longest path delay for AMRoute does not slow down. This indicates that though the cost of a sub-optimal tree does not increase, the the longest delivery path can still be even longer.



(a) Relative bit overhead.

(b) Packet delivery ratio.

Figure 10: Comparison of protocol performance.

4.3 Study of Protocol Performance

We study the performance of PAST-DM protocol by protocol overhead and the packet delivery ratio. Protocol overhead is measured by *relative control bit overhead*, defined as the amount of control overhead in bits per delivered bit. The transmitted control bits includes the control packets and the bytes in each packet header. Moreover, the involved unicast control bit overhead is also included. The packet delivery ratio of multicast is defined as the sum of the received data packets at all member nodes divided by product of the number of sent packets from the source node multiplied by the number of member nodes. We use the GloMoSim[25], a wireless network simulator developed by UCLA. The parameters are described in Section 4.1. At the physical layer, GloMoSim uses a comprehensive radio model that includes the Two-Ray radio propagation model and also accounts for noise power. At the MAC layer, IEEE 802.11 is used.

We focus on the performance with regard to the different group sizes. Thus, for all simulations, the mobility parameters remains the same. We use random waypoint mobility model, with 40 seconds in pause time and 10 m/s in maximum moving speed. We compare the the PAST-DM with the ODMRP [8], which is a mesh-based multicast protocol.

4.3.1 Relative Control Overhead

Figure 10(a) shows the result for protocol control overhead, with regard to the group size. In both figures, the legend labels of "PAST-DM-5" and "PAST-DM-10" mean the update period being set respectively. It is noticeable that the curve for ODMRP first decreases with growing group size, then the curve goes up when group size is 15. From 5 members to 15 members, though the amount of control packets increases, the number of delivered packets increases faster with more receivers. As the group population becomes more dense, more control packets sent by all members nodes need

to gather at the sender node. Thus, the packet implosion problem becomes causing higher collision and retransmission of the control packets.

PAST-DM, on the other hand, shows steady decrease of relative bit overhead with growing group size. The control packets sent by each node only need to reach its local area, as opposed to the ODMRP, where the periodic JOIN_REPLY packets from each receiver need to reach the sender node. As the group population becomes denser, the reachable region of each control packets should shrink accordingly, which further reduces bit overhead. Thus, we see that, even though PAST-DM has extra bit overhead in packet headers, the overall overhead is less, compared to ODMRP.

4.3.2 Packet Delivery Ratio

Figure 10(b) shows the result for packet delivery ratio, also with regard to the group size. As shown by the curves, when the group is small PAST-DM does not perform as well as ODMRP. This is because the group population is sparse, and the mesh connectivity is under worse challenges with the distance between adjacent members nodes being larger. When the update period time is 20 seconds, the PAST-DM performance is very poor. However, as the group grows larger, PAST-DM-5 out-performs ODMRP when group size is up to 20. Similarly, PAST-DM-20 gets better performance when group size is larger than 25. We can also observe that, the performance difference between PAST-DM-5 and PAST-DM-20 becomes less when the group size is larger. Thus, it is possible that, when the group population becomes dense, the update period can be longer, so that the protocol overhead can be significantly reduced while the packet delivery ratio is only slightly lower.

5 Related Work

Besides overlay multicast, our work is also closely related to stateless multicast. This approach is similar to overlay multicast in the sense that it also employs standard unicast routing and forwarding. The difference is that there is no explicit concept of overlay virtual topology. SGM[23] was first proposed for small group multicasting on the Internet. The source node first performs a route table look up to determine the next hop for each receivers in its group. It then partitions the group based on the next hops. Each subgroup is encoded into a SGM packet header and the packet is unicast to the corresponding next hop. The receiving next hop will further partition its subgroup and forward the data to it next hops. SGM is considered as a scalable solution to support large number of small multicast groups on the Internet. DDM[24] is a stateless multicast protocol proposed for MANET. In DDM, receivers can be listed in the DDM packet header in a differential manner, which means it only includes the difference with respect to the receiver list in the last packet. Each node in the forwarding paths remembers the subset it has been forwarded to last time, together with the corresponding next hop information. By caching routing decisions in the intermediate nodes, the source does not need to list the group members in future packets. In making routing decisions, intermediate nodes need to query the unicast route table.

With GPS devices, each mobile node is aware of its location within the network area. Location Guided Tree (LGT) construction scheme[15] builds overlay multicast tree using geometric distance between member nodes as the heuristic of link costs. Two tree construction algorithms are proposed: greedy k-ary tree construction (LGK) and *Steiner* tree construction (LGS). With LGK, the source nodes selects k nearest neighbors as its children, and partitions the remaining nodes according their distance to the children nodes. LGS constructs the *Steiner* tree using link costs as their geometric lengths. Each children node is then responsible for packet delivery to its own subgroup using the same algorithm. Our work is inspired by the stateless multicast approaches. The difference is that our approach uses virtual mesh so that no GPS or location information are needed. This makes the protocol applicable to more situations.

6 Conclusions

In this paper, we present an overlay multicast protocol aiming to solve the efficiency problem of overlay multicast approach. To achieve this goal, we propose dynamic virtual mesh that adapts itself to the mobility of network nodes. Virtual links with long physical hop lengths are replaced by short links. A novel tree construction algorithm is proposed that fully utilizes the latest local topology information. Simulation studies have shown that the yielded multicast tree is close to optimal in terms of total hop cost with a stable quality. Thus stable and efficient multicast performance is observed. The control overhead can be reduced to a low level since the tree quality is only moderately hampered when the periodic update behaviors are conducted less frequently.

References

[1] M. Gerla, and J.T. Tsai, "Multicluster, mobile, multimedia radio network," ACM-Blatzer Wireless Networks 1(1995) 255-65.

[2] P. Mohapatra, C. Gui and J. Li, "Group Communications in Mobile Ad Hoc Networks," IEEE Computer, vol. 37, no. 2, Feb. 2004

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

[4] L. Ji, and M. S. Corson, "A Lightweight Adaptive Multicast Algorithm," In Proceedings of IEEE GLOBECOM'98, November 1998.

[5] C.W. Wu, Y.C. Tay, and C.-K. Toh, "Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) Functional Specification, Internet Draft.

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

[7] C.-C. Chiang, M. Gerla, and L. Zhang, "Forwarding Group Multicasting Protocol for Multihop, Mobile Wireless Networks," ACM-Baltzer Journal of Cluster Computing: Special Issue on Mobile Computing, vol. 1, no. 2, pp. 187-96, 1998.

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

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

[10] P. Sinha, R. Sivakumar and V. Bharghavan, "MCEDAR: Multicast Core-Extraction Distributed Ad Hoc Routing", IEEE WCNC 1999.

[11] J. Xie, R. Talpade, T. McAuley, and M. Liu, "AMRoute: Ad Hoc Multicast Routing Protocol", ACM Mobile Networks and Applications (MONET) Journal , 7(6): 429-439, Dec 2002

[12] H. Eriksson, "MBone: The Multicast Backbone," Communications of the ACM, Aug. 1994, Vol. 37, pp.54-60.

[13] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," In Proceedings of ACM/Sigmetrics, June 2000.

[14] M. Kwon and S. Fahmy, "Topology-Aware Overlay Networks for Group Communication," In Proceedings of ACM NOSSDAV, May, 2002.

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

[16] L. Xiao, A. Patil, Y. Liu, L. M. Ni, and A.-H. Esfahanian, "Prioritized Overlay Multicast in Mobile Ad Hoc Environments," IEEE Computer, vol. 37, no. 2, Feb. 2004

[17] S.-J. Lee, W. Su, M. Gerla, and R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols," IEEE INFOCOM'00, Tel-Aviv, Israel, March, 2000.

[18] C.E. Perkins, E.M. Royer, and S.R. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing," Internet Draft, draft-ietf-manet-aodv-10.txt, March 2002 (Work in Progress).

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

[20] H. Takahashi, and A. Matsuyama, "An approximate solution for the Steiner problem in graphs," Mathmatica Japonica, 1980, pp.573-77.

[21] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pp. 153-81, Kluwer Academic Publishers, 1996.

[22] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," IEEE INFOCOM'03, San Francisco, Mar, 2003.

[23] R. Boivie, N. Feldman and C. Metz, "Small Group Multicast: A New Solution for Multicsting on the Internet," IEEE Internet Computing, May.June 2000.

[24] L. Ji and M. S. Corson, "Differential Destination Multicast - A MANET Multicast Routing Protocol for Small Groups," In Proceedings of IEEE/Infocom'01, April, 2001.

[25] GloMoSim, http://pcl.cs.ucla.edu/projects/glomosim