A Distributed Algorithm for Finding a Tree Trunk and Its Application for Multicast in Mobile Ad Hoc Networks

Yamin Li, Shietung Peng
Department of Computer Science
Hosei University
Tokyo 184-8584 Japan
{yamin;speng}@k.hosei.ac.jp

Wanming Chu
Department of Computer Hardware
University of Aizu
Aizu-Wakamatsu 965-8580 Japan
w-chu@u-aizu.ac.jp

Abstract

Overlay multicast protocol constructs a virtual mesh spanning all member nodes of a multicast group and employs standard unicast routing to fulfill multicast functionality on application layer. The advantages of this approach are simplicity and flexibility. However, efficiency and stability are the issues that must be addressed as the size of the multicast group grows in the mobile ad hoc networks (MANETs). In this paper, we propose tree trunk for overlay multicast to solve these problems in MANETs. A tree trunk is a path that minimizes the sum of the distances of all vertices to the path plus the length of the path. We give an efficient distributed algorithm for finding a tree trunk in a tree network. We also perform some empirical analysis based on the tree trunk and compare the results with those using spanning tree.

Key words: Mobile ad hoc networks, multicast, overlay mesh, efficiency, stability, distributed algorithm.

1 Introduction

Mobile ad hoc networks (MANETs) refer to a form of infrastructureless networks connecting mobile devices with wireless communication capacity. Each node in MANETs behaves as a router as well as an end host, so that the connection between any two nodes is a multi-hop path supported by other nodes. In MANETs, the multicast support is critical since the close cooperation among team members is required for many MANET applications.

Multicasting in MANETs faces many challenges due to the continuous changes in network topology (mobility) and limited channel bandwidth. Many multicast routing protocols have been proposed for MANETs [1, 9, 3, 5, 11, 15, 16, 17, 4]. A review paper was given by Cordeiro et al. [2]. For multicast protocols, robustness and overhead are key issues since the protocols maintain state information at all nodes involved — both member nodes and non-member nodes that act as routers for supporting the multicast session.

Most multicast research for ad hoc networks has focused on IP layer multicast protocols. Such protocols require the cooperation of all the nodes of the network. Application layer multicasting (overlay multicasting) is an alternative approach to IP layer multicasting. The overlay multicast has the following advantages: First, it does not require changes at the network layer; second, routing complications are hidden; and third, intermediate nodes do not have to maintain per group state for each multicast group. However, the use of application layer multicast can result in the transmission of multiple copies of multicast messages over each physical link. This effect is especially visible when there are a large number of multicast group members.

In the overlay multicast approach for MANETs, a virtual infrastructure is built to form an overlay network on top of the physical network. Each link in the virtual topology is a unicast path in the physical network. The overlay network implements multicast functionalities such as dynamic membership maintenance, packet duplication and multicast routing. AMRoute [4] is an ad hoc multicast protocol that uses the overlay multicast approach. The protocol does not need to track the network mobility since it is handled by the underlying unicast protocols. Thus, it can operate seamlessly on multiple domains that use different unicast routing protocols [11].

To handle the efficiency issue in overlay multicast approach, minimum cost spanning tree on the virtual mesh is built. The minimum cost spanning tree problem is to find a subgraph that connects all vertices such that the sum of the costs of the edges in the subgraph is minimum. The cost of constructing and maintaining the tree depends very much on the size of the tree. For this reason, the overlay multicast approach works well for small groups but the performance degrades rapidly when the group size grows. Instead of using spanning tree of the virtual mesh, we propose a specific path, called tree trunk, for the overlay multicast on the virtual mesh. A tree trunk is a path that minimizes the sum of the distances of all vertices to the path plus the length of the path. The tree trunk significantly reduces the cost for the maintenance and provides higher stability under the mobile environment.

The rest of the paper is organized as follows. Section 2 reviews the previous work on overlay multicast in MANETs. Section 3 presents the theoretical background for tree trunk. An efficient distributed algorithm for finding a tree trunk is given in Section 4. Section 5 gives simulation results on the performance of multicasting using tree trunk and compares these results to those of the AMRoute. Section 6 concludes this paper.

2 Preliminaries and Previous Work

We consider an ad hoc network as a graph $G = (V, E)$, where V is a set of nodes and E is a set of bidirectional links. In an overlay multicast approach, a virtual mesh connecting all group members is built first. Each member node starts a neighbor discovery process using the expanded ring search technique. The maximum degree of the virtual topology is controlled. Each member node keeps track of other members in its vicinity. This is done by a query to its route table maintained by unicast protocol, or by a periodic neighbor discovery operation. Each member node also maintains the topology map of the virtual mesh. This is done by the link state exchange technique. At each node, the topology map is represented as a link state table. Through the link table, each node has a local view of the whole virtual topology. After the virtual mesh is built, multicast tree is set up on the virtual mesh for efficient multicasting.

There are two kinds of approaches for tree-based multicast: shared tree and source-based tree. The source-based tree approach is more efficient for data delivery. However, since each node constructs its own tree the cost is higher. We review three overlay multicast protocols that are presented in the literature, namely, AMRoute [4], PAST-DM [9], and ALMA [8].

AMRoute is an ad hoc multicast protocol that uses the overlay mesh and a shared user-multicast tree for robust IP multicast in mobile ad hoc networks. Bidirectional unicast tunnels are used to connect the multicast group members into a virtual mesh. After the mesh creation phase, a shared tree for data delivery is created and maintained within the mesh. One member node is designated as the logical core which is responsible for initiating the tree creation process periodically. The core node is not a preset node and changes dynamically according to the core-resolution algorithm. The tree constructed in AMRoute is not necessary to be the minimum cost tree.

In PAST-DM protocol, each source constructs its own data delivery tree based on its local link state table. A novel source-based Steiner tree algorithm is used to minimize the total cost of multicast tree. The tree is then periodically refreshed. During the construction process, the source makes all its logical neighbors its first-level children in the multicast tree and divides the remaining members into subgroups. Each of these sub-groups forms a subtree rooted at one of the first-level children. Each of the source's first-level children then repeats the source-based Steiner tree algorithm to establish their own subtrees and forwards the message to the subtree.

In ALMA protocol, an overlay multicast tree of logical links between the group members is constructed to tackle the efficiency problem in MANETs. The virtual topology gradually adapts to the changes in underlying network topology. A source-based Steiner tree algorithm was proposed for constructing the multicast tree. The multicast tree is progressively adjusted according to the latest local topology information. Its advantages are: receiver-driven, flexible, and adaptive. From their simulations, ALMA performs significantly better for small group sizes.

3 Tree Trunk for Overlay Multicast

As mentioned in Section 1, overlay multicasting protocol is an application layer protocol that constructs an overlay multicast tree of logical links among the group members. For small group this approach works well. However, as the size of the group grows, the maintenance cost of the multicasting tree will become higher and the stability of the tree will become worse due to the node mobility. Instead of using a spanning tree, our new approach uses a very simple linear structure, called tree trunk, for multicasting on the virtual mesh. This approach is beneficial when the multicast group is not small.

A core of a tree is a path that minimizes the sum of the distances of all vertices to the path. A linear sequential algorithm for finding a core of a tree was given by Morgan and Slater [14]. Peng et al. developed a parallel algorithm for finding a core of a tree [6, 7]. A core of a tree does not count the distances between the nodes on the path. However, for multicast on mobile ad hoc networks, the length of the path should be included in the optimization criteria. Li, Peng, and Chu proposed distributed algorithms [12, 13] for multicasting in mobile ad hoc networks, but the algorithms do not guarantee to generate a tree trunk.

3.1 Definition of tree trunk

A tree trunk is a path that minimizes the sum of distance of all vertices to the path *plus* the length of the path. Let G be an edge-weighted graph with vertex set $V(G)$. Each edge $e = (u, v)$ has a weight $w(e)$, or $w(u, v)$, where nodes

u and v are neighbors connected by edge e. Let G' be a connected subgraph of G, we define the *inner cost* $w(G')$ and *outer cost* $\delta(G')$ as

$$w(G') = \sum_{e \in G'} w(e) \quad (1)$$

$$\delta(G') = \sum_{u \in V(G)} d(u, G') \quad (2)$$

where $d(u, G') = \min\{d(u, v) | v \in V(G')\}$ and $d(u, v)$ is the distance between nodes u and v. Our goal is to minimize $w(G') + \delta(G')$. Notice that if G' is a spanning tree of G, $\delta(G') = 0$ and if $G' = \{u\}$, stateless broadcast for instance, $w(G') = 0$.

Let T be an edge-weighted tree with vertex set $V(T)$ and $P(s, t)$ be a path in T with two end nodes s and t. A path $P(s, t)$ is a tree trunk if $w(P(s, t)) + \delta(P(s, t))$ is minimum for any $s, t \in V(T)$. The *cost* of a path $P(s, t)$ is defined as $w(P(s, t)) + \delta(P(s, t))$.

Because the number of nodes in a tree trunk is much less than the number of nodes in the corresponding tree, especially when the tree size is large, the maintenance of a tree trunk is easer than that of the tree. Sending message to a node that is not on the trunk is done with unicast, therefore, there is no affect on the topology when a non-trunk node quits from the group membership. This means that the tree trunk is more *stable* that the tree.

3.2 Theoretical background for finding a tree trunk

For efficiently constructing a trunk, we orient tree T into a rooted tree T_r with root r. A root r is arbitrarily selected. After the tree trunk is constructed, r becomes an ordinary member node. Selecting a different root r does not affect the final tree trunk. Any distributed selection algorithm can be used for selecting the root r and the root r has much less job than the logical core in AMRoute.

For any vertex $v \in T_r$, we denote the parent of v as $p(v)$, the subtree rooted at v as T_v, and the number of vertices in T_v as $|T_v|$. Let a rooted trunk $P(r, l_0)$ be a path from root r to leaf l_0 which minimizes $\delta(P(r, l)) + w(P(r, l))$ among all paths from r to leaf l in T_r. We show that the problem of constructing a trunk in T can be reduced to the problem of constructing a rooted trunk in a rooted tree T_r. The following lemmas form the theoretical background for the reduction.

Lemma 1 Let rooted tree T_r be an orientation of T and $P(r, l_0)$ a rooted trunk in T_r. Then $P(r, l_0) \cap P(l_1, l_2) \neq \emptyset$ for any trunk $P(l_1, l_2)$ in T.

Proof: Assume that $P(r, l_0) \cap P(l_1, l_2) = \emptyset$ for a trunk $P(l_1, l_2)$. Let i be the closest vertex in $P(r, l_0)$ to $P(l_1, l_2)$ and j the closest vertex in $P(l_1, l_2)$ to $P(r, l_0)$ (see Figure 1). Let path $C = P(l_0, i) \cup P(i, j) \cup P(j, l_2)$. Since

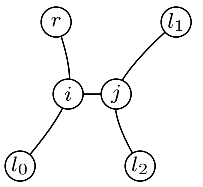

Figure 1. $P(r, l_0)$ is a rooted trunk in T_r and $P(l_1, l_2)$ is a trunk

$P(r, l_0)$ is a rooted trunk, $\delta(P(l_0, i)) + w(P(l_0, i))) \leq \delta(P(l_1, i)) + w(P(l_1, i))$. Since i is not a leaf, we have $\delta(P(l_1, i)) + w(P(l_1, i)) < \delta((P(l_1, j)) + w(P(l_1, j))$. Similar, we have $\delta(P(l_0, j)) + w(P(l_0, j)) < \delta((P(l_0, i)) + w(P(l_0, i))$. From these equations, we get $\delta(P(l_0, j)) + w(P(l_0, j)) < \delta((P(l_1, j)) + w(P(l_1, j))$. This implies $\delta(C) + w(C) < \delta(P(l_1, l_2)) + w(P(l_1, l_2))$, a contradiction to the fact that $P(l_1, l_2))$ is a trunk. Therefore, the lemma must be true. ❑

Theorem 1 Let rooted tree T_r be an orientation of T and $P(r, l_0)$ a rooted trunk in T_r. Then a rooted trunk in rooted tree T_{l_0}, a new orientation of T, is a trunk in T.

Proof: Let $P(l_0, l_0')$ be a rooted trunk in T_{l_0}. Assume that $P(l_1, l_2)$ is a trunk in T. From Lemma 1, $P(l_0, l_0') \cap P(l_1, l_2) \neq \emptyset$. Let $P(i, j) = P(l_0, l_0') \cap P(l_1, l_2)$, where i is the vertex in $P(i, j)$ closest to vertices l_0 and l_1 (see figure 2). Since $P(r, l_0)$ is a rooted trunk, we have $\delta(P(l_0, i)) + w(P(l_0, i)) \leq \delta(P(l_1, i)) + w(P(l_1, i))$. Similarly, Since $P(l_0, l_0')$ is a rooted trunk, we have $\delta(P(l_0', j)) + w(P(l_0', j)) \leq \delta(P(l_2, j)) + w(P(l_2, j))$. Therefore, we get $\delta(P(l_0, l_0')) + w(P(l_0, l_0')) \leq \delta(P(l_1, l_2)) + w(P(l_1, l_2))$. We conclude that $P(l_0, l_0')$ is a trunk in T. ❑

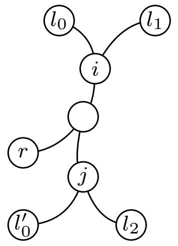

Figure 2. $P(i, j) = P(l_0, l_0') \cap P(l_1, l_2)$

From Theorem 1, the problem of constructing a trunk in T can be solved as follows:

1. Orient tree T into a rooted tree T_r with an arbitrary vertex r;
2. Construct a rooted trunk $P(r, l_0)$ in T_r;
3. Re-orient T into T_{l_0};
4. Construct a rooted trunk in T_{l_0}.

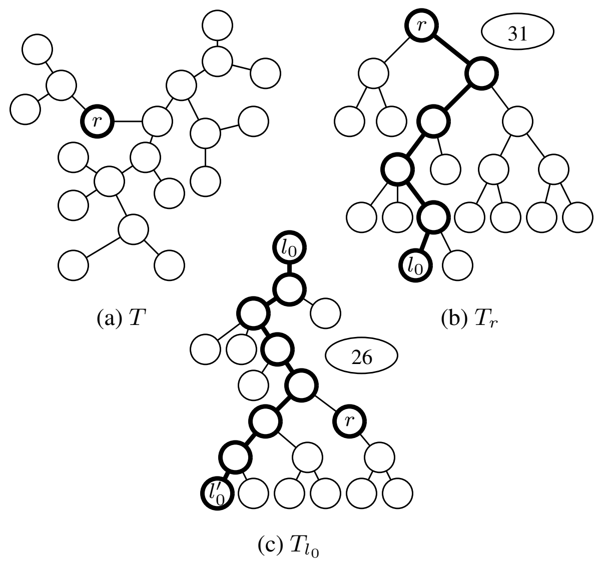

Figure 3. (a) An example tree; (b) a rooted trunk in T_r; (c) a rooted trunk in T_{l_0} which is a trunk in T

In Figure 3, we first show an example tree T with an arbitrarily selected vertex r in Figure 3(a). Then, in Figure 3(b), we show the rooted tree T_r and a rooted trunk $P(r, l_0)$ in T_r. We have $\delta(P(r, l_0)) + w(P(r, l_0)) = 26 + 5 = 31$. Finally in Figure 3(c), we show the rooted tree T_{l_0} and a rooted trunk $P(l_0', l_0)$ in T_{l_0}. The path $P(l_0, l_0')$ is a trunk in T, and we have $\delta(P(l_0, l_0')) + w(P(l_0, l_0')) = 19 + 7 = 26$.

4 A Distributed Algorithm for Tree Orientation and Finding a Rooted Trunk

We propose a distributed algorithm for tree orientation and finding a rooted trunk of a tree T with a given root node r in this section. The algorithm is based on *branch-cut* operation. The branch-cut operation works inward from leaves ($\neq r$). The branch-cut operation first identifies *candidates*. A node $u \neq r$ is a candidate if the following conditions are satisfied: (1) u is a nonleaf node; (2) if $r \notin N(u)$ (the set of neighbor nodes of u), node u has exactly one nonleaf neighbor, otherwise, all nodes in $N(u) - \{r\}$ are leaves. The root r is a candidate if all its neighbors are leaves. If u becomes a candidate then branch-cut is performed on u; the neighbors of u that are leaves are cut-off from the tree and u becomes a leaf (for tree orientation, we set all edges connecting u and its leaf neighbors the direction toward u). The *branch* $B(u)$ is a subtree in T that includes all edges oriented toward u or its descendants through branch-cut. Figure 4 depicts a tree T_r that contains a candidate u. Notice that w is not a candidate although it has only one nonleaf neighbor v.

Through branch-cut operation, the rooted trunk of the branch $B(u)$ with root u (called local trunk) is calculated

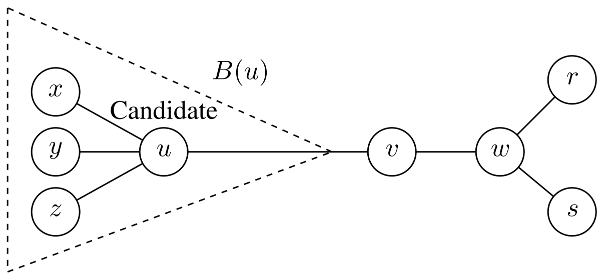

Figure 4. Candidates in tree T_r

and saved in u. Since all candidates that are not root r calculate the disjoint local trunks for different branches at the same time, the algorithm inherits natural parallelism. In a distributed environment, global clock and global information are not available, so branch-cut operation should be done asynchronously, and based on the local information only.

To find the rooted trunk based on branch-cut, if we use the formula $w(P) + \delta(P)$ directly, $\delta(l)$, for all leaves l of tree T, should be calculated first. However, calculating the value of $\delta(l)$ requires global information. To overcome this problem, we define *cost saving* that needs local information only. The cost saving of a path from a leaf l to node v, denoted as $C_s(P(l, v))$, is defined as follows:

$$C_s(P(l, v)) = \delta(v) - \delta(P(l, v)) - w(P(l, v)) \tag{3}$$

Now, from the definition of rooted trunk, to find a rooted trunk in branch $B(u)$ equals to find a path $P(l, u)$ in $B(u)$ such that $C_s(P(l, u))$ is maximized. It is the key in the design of distributed algorithm for finding rooted trunk based on branch-cut that $C_s(P(l, u))$ in any branch $B(u)$ can be computed using local information only. The formula for computing cost saving while extending path from v to u is

$$C_s(P(l, u)) = C_s(P(l, v)) + (|B(u)| - 1) \times w(u, v) \tag{4}$$

where v is a child of u and $v \in P(l, u)$).

The cost saving of a local rooted trunk in $B(u)$, denoted as $C_s(u)$ is defined as

$$C_s(u) = \max_{l \in B(u)} C_s(P(l, u)) \tag{5}$$

We give an example to demonstrate how the cost saving are calculated through branch-cut. As shown in Figure 5, given tree T and root r, four nodes a, b, c, and d are identified as candidates. We perform branch-cut and these four nodes become leaves with $C_s(a) = C_s(b) = C_s(c) = C_s(d) = 0$. Next, nodes e, g, and f are identified as candidates. We perform branch-cut and calculate the cost savings $C_s(e) = 0 + (4 - 1) \times 1 = 3$, and $C_s(f) = C_s(g) = 2$ in parallel by using Equation 4. Finally, since all three neighbors of node r are leaves, we perform branch-cut at r and

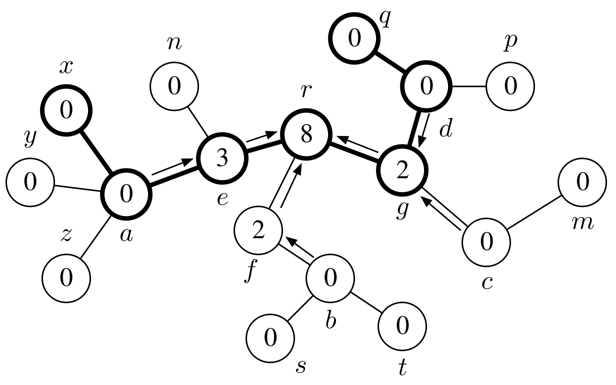

Figure 5. Calculation of cost saving during branch-cut

```
Algorithm 1: Find_Rooted_Trunk
Input: A weighted tree $T$ and a root $r$
Output: A rooted trunk $P_{r,l}$ of $T$ with root $r$
begin
  $u$ = my_node_id;
  $u.size = 1$;
  $u.saving = 0$;
  $u.path = \{u\}$;
  $n = degree(u)$;
  $L = N(u)$;          /* $N(u)$ is the set of neighbor nodes of u */
  if $(n = 1)$ and $(u \neq r)$                      /* a leaf */
    send Message($u.size, u.saving, u.path$) to $v \in L$;
    set direction of $(u, v)$ as $u \rightarrow v$;
    exit();
  else
    while (true)
      receive Message ($v.size, v.saving, v.path$) from $v \in L$;
      $n = n - 1$;
      $L = L - \{v\}$;
      $u.size = u.size + v.size$;
      if $(u.saving < v.saving + (v.size - 1) \times w((u, v)))$
        $u.saving = v.saving + (v.size - 1) \times w((u, v))$;
        $u.path = v.path \cup (u, v)$;
      endif
      if $(n = 1)$ and $(u \neq r)$                  /* branch-cut */
        send Message($u.size$, $u.saving$, $u.path$) to $v \in L$;
        exit();
      endif                                      /* my_node_id finish */
      if $(n = 0)$                                   /* u is root */
        return $u.path$;
        exit();                                  /* Rooted trunk found */
      endif
    endwhile
  endif
end
```

calculate three cost saving from three branches. They are $C_s(P(x, r)) = 8$, $C_s(P(s, r)) = 5$, and $C_s(P(q, r)) = 8$, respectively. Therefore $C_s(r) = 8$ and either $P(x, r)$ or $P(q, r)$ can be selected as rooted trunk.

In our algorithm, we use the local information to compute the following three variables in each node u:

1. The number of nodes in $B(u)$, denoted as $u.size$.
2. The cost saving $C_s(u)$, denoted as $u.saving$.
3. The local rooted trunk in $B(u)$, denoted as $u.path$ ($C_s(u.path) = C_s(u)$).

The proposed distributed algorithm for finding a rooted trunk in a weighted tree is formally presented in Algorithm 1. The algorithm uses only local information and works asynchronously. We show in theorem 2 that algorithm 1 finds the tree orientation and a rooted trunk correctly.

Theorem 2 Algorithm 1 returns an orientation of T with root r and a rooted trunk in T_r.

Proof: From the initial values assigned to leaves and the iteratively extending formula 4, it can be verified easily that the $u.saving = Cs(u)$. Next, When node $u \neq r$ becomes a leaf ($|L| = n = 1$), it sends a message to the only node v left in L that is either a nonleaf node or the root and exits. Therefore, the message from u to v is sent only once and no message will be sent from v to u (or the direction of edge (u, v) is set from u to v one and only once). That is, there is no conflict during asynchronous communication. Since the message is sent from u (u is cut-off) only if $u \neq r$, root r will receive message from its neighbor only and the edges are oriented from its neighbors toward r. Therefore, root r must be the last node remained during the branch-cut process. When the neighbor list L of node r becomes empty ($n = 0$), the orientation of T is done and the rooted trunk of T_r is also found. ❑

Next, we shows in Theorem 3 that finding a tree trunk can be done efficiently in a distributed environment using local information only.

Theorem 3 Given a weighted tree T, there exists a distributed algorithm that finds a tree trunk in T in $O(d)$ time, where d is the diameter of T. Assume that the degree of node $v \in T$, $deg(v) = O(1)$.

Proof: To find a tree trunk of T, we perform algorithm 1 twice: first with a randomly selected root r, and second with root l, where $P_{r,l}$ is the rooted trunk of T with root r. From the theorems 1 and 2, we conclude that the rooted trunk $P_{l,l'}$ of T with root l is a tree trunk of T. ❑

Figure 6 shows an example of mobile ad hoc networks. There are 160 mobile nodes randomly locating within a 2000m $\times$ 1500m area. The radio transmission range is 250 meters. The multicast group size is 80. The solid cycles represent the group members. The maximum d-hop distance is 2. The tree is marked with thicker lines and the tree trunk is marked with thickest lines.

The proposed algorithm is designed to operate in a distributed fashion in a mobile environment. The network topology may change due to host mobility or adds/drops of group membership. With the spanning tree topology, every group member's mobility or quit from the group may

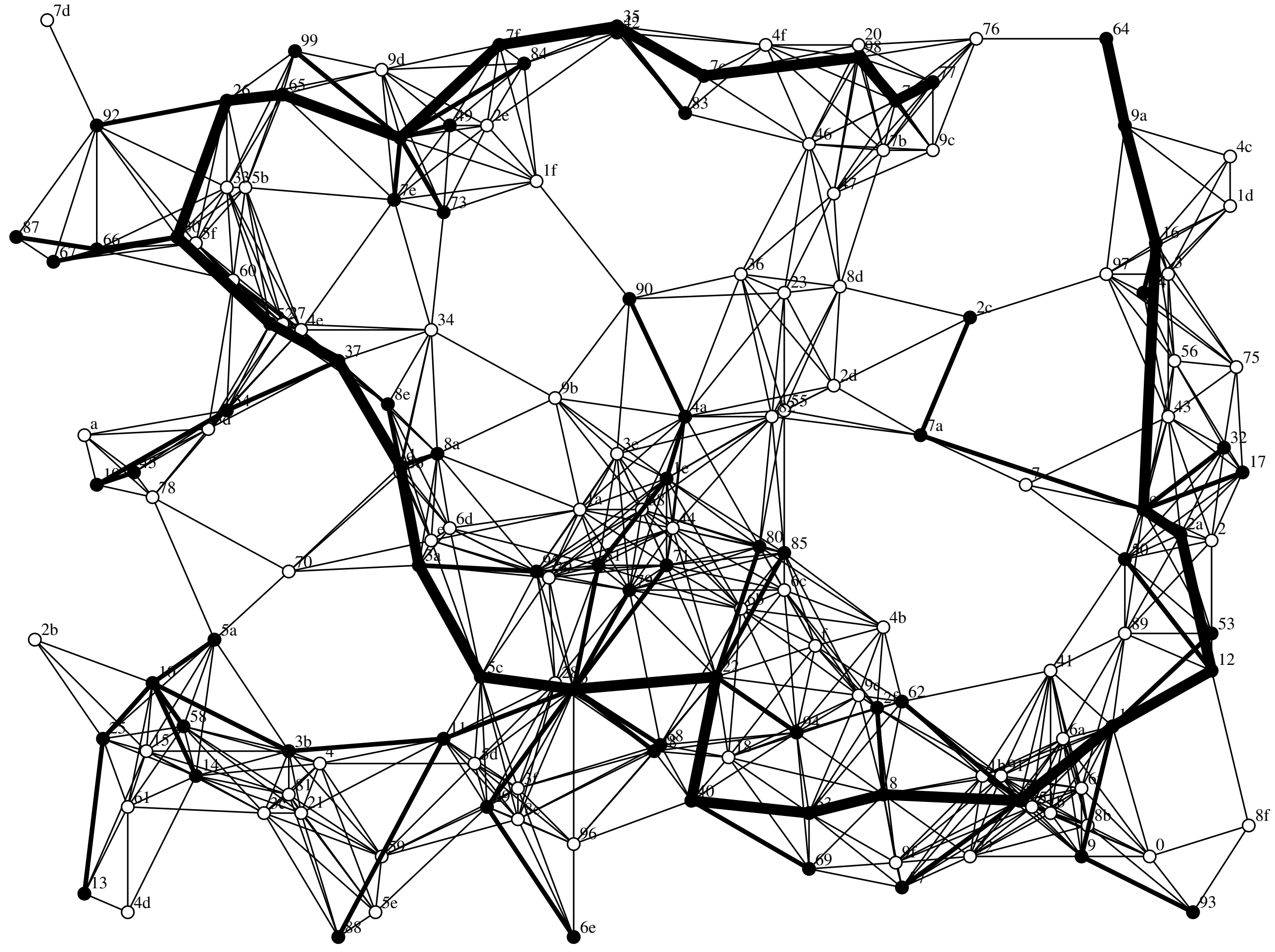

Figure 6. An example of tree trunk

affect the connectedness of the network, but with the tree trunk topology, the mobility or quit of a member that is not a node of the tree trunk does not affect the connectedness of the network. We will demonstrate the benefit of using a tree trunk for multicast in the mobile ad hoc networks in the next section.

5 Performance Analysis and Simulations

The network for the performance simulation is configured as below. There are 200 nodes randomly roaming within a 2000m $\times$ 1500m area. The radio transmission range of each node is set to be 250m, 350m, and 450m. The group size is chosen to be 10 to 100, stepped by 10. Each configuration runs 100 trials.

Figure 7 shows the average hop distances for constructing the overlay mesh. Increasing the radio transmission range will decrease the hop distance at the cost of increased power consumption.

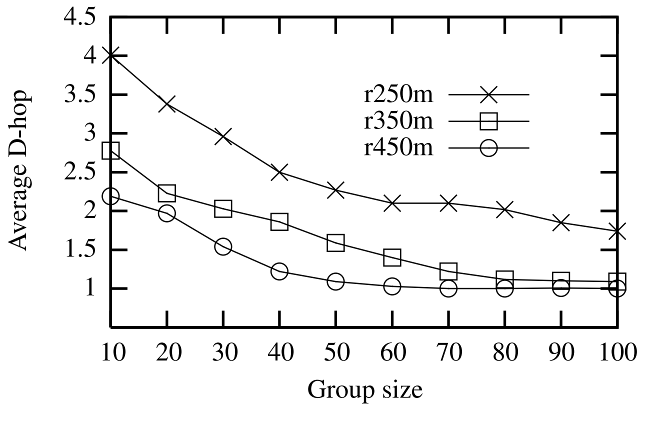

Figure 7. Average D-Hop

Figure 8 shows the size of the trunk, i.e., the number of member nodes that form the trunk. This size is relatively

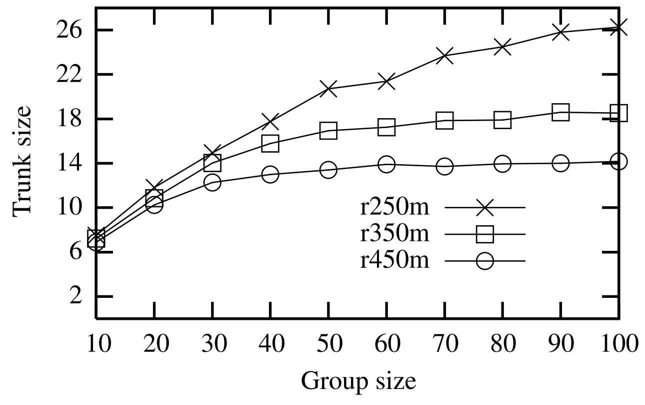

Figure 8. Core size

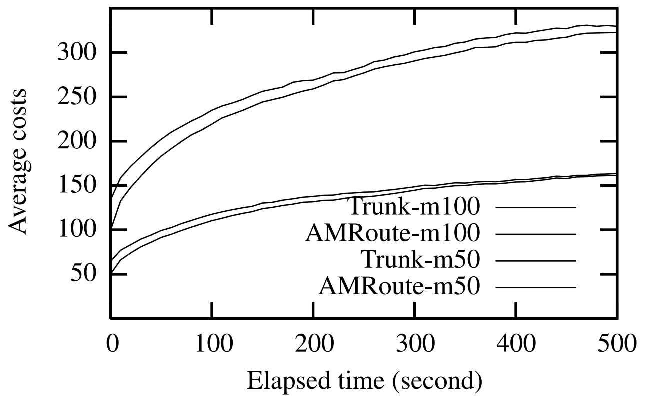

Figure 10. Average cost

small compared to the group size. Also, when the group size is large, say 50, adding new members to the group will not affect the size of trunk obviously.

The trunk maintains fewer nodes that re-send the received message than AMRoute for multicast. Of course, the message delivery cost of the trunk is higher than that of AMRoute. But, from Figure 9, we can see that the increased cost is quite small. The message delivery cost here is simply defined as the sum of physical hop length of virtual links of AMRoute or the trunk when a message is multicasted to all the group members. The figure also shows the cost for stateless transformation in which the message is sent to every member individually by unicast routing.

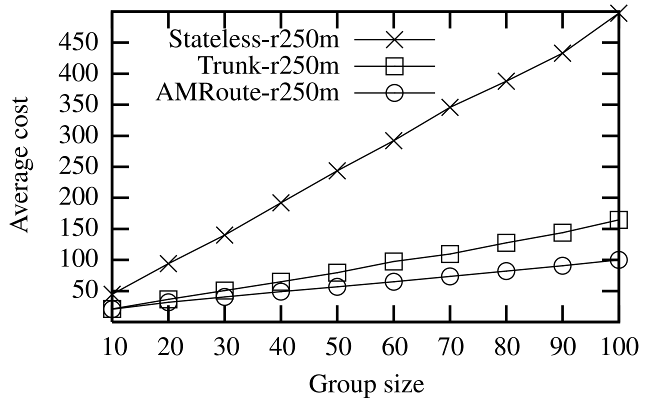

Figure 9. Average cost

The virtual trunk remains static even though the underlying physical topology is changing. We investigated the mobility effect on the cost. The movement of each node follows the random waypoint model [10]: Each node selects a destination location randomly and moves straight toward the destination with a constant speed which is uniformly distributed over [0,20] meters/second. After arrival, the node pauses for 10 second and then moves to another location, and so on.

Figure 10 shows the time-line of the average costs of AMRoute and the trunk for multicast group sizes 50 and 100. As shown in the figure, the costs increase as the mobile nodes move; the costs for AMRoute and the trunk are almost the same as the time elapses. Figure 11 shows the relative increased costs which are obtained by dividing *cost*(t), the cost at the time t, by *cost*(0). As the time elapses, the relative increased cost of the trunk is less than that of AMRoute.

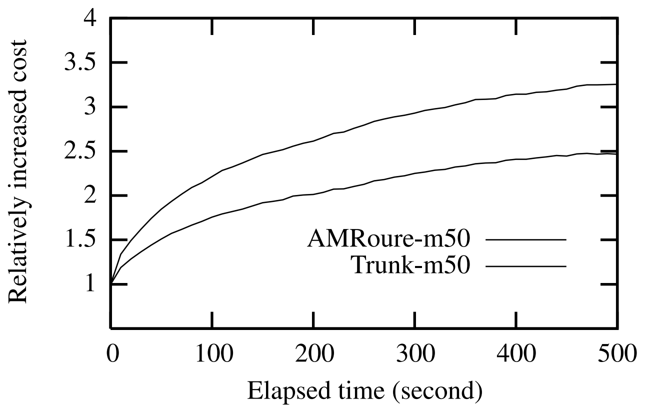

Figure 11. Relatively increased cost

For tree topology, if a leaf node quits from the group, there is no effect on the connectedness of the tree; if it is a nonleaf node, the original tree is partitioned into two or more subtrees. Similarly, if the quited node is a trunk node, the trunk topology may lose the virtual connection. Figure 12 shows the delivery ratio when mobile nodes quit from the multicast group. The delivery ratio is obtained as below.

We simulated 100 trials for a fixed number of quit nodes.

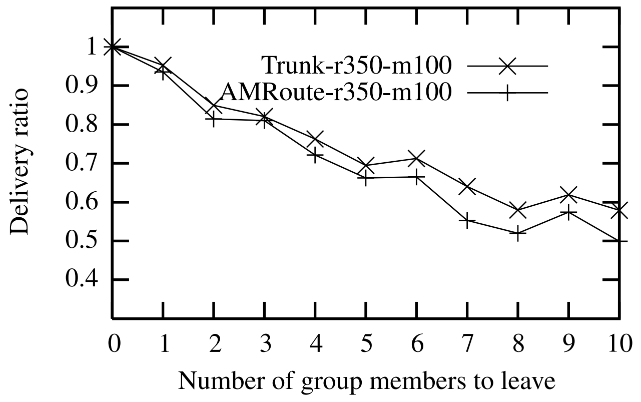

Figure 12. Delivery ratio

The quit nodes are randomly chosen among the multicast group members. For each trial, a randomly chosen node broadcasts a message. Because the trunk and the virtual tree in AMRoute may be partitioned due to the node quits, the message may not reach to the all other nodes. The relative delivery ratio is calculated by dividing the number of nodes that received the message by the group size. As shown in the figure, the delivery ratio of the trunk is better than that of AMRoute.

This also means that as the node quits, the probability of the connectedness of the trunk is higher than that of AMRoute. In a practical implementation, when disconnected, AMRoute and the trunk must be reconstructed. From Figure 12, we conclude that trunks are more stable than AMRoute.

6 Concluding Remarks

A new infrastructure based on trunk for overlay multicasting on mobile ad hoc network was proposed, an efficient distributed algorithm for finding a trunk was developed, and the performance was evaluated through simulations. Our future work includes theoretical study and finding alternative infrastructures for multicasting on mobile ad hoc networks.

The following comes from the reviewer comments: "As far as a mobile ad hoc network-based application is concerned, the withstanding or resilience to network merging and partitioning is generally required. A network partitioning splits the communication environment into two or more disjoint parts, a common occurrence in wireless networks containing mobile nodes. On the contrary, network merging will cause issues on changes in coalition membership or on reconciling the effects of partitioned multicast communication. Such related issues require further investigation in the future."

References

[1] K. Chen and K. Nahrstedt. Effective location-guided tree construction algorithm for small group multicast in manet. In *Proc. of IEEE INFOCOM'02*, June 2002.

[2] C. Cordeiro et al. Multicast over wireless mobile ad hoc networks: present and future directions. *IEEE Network*, 17(1), Jan. 2003.

[3] D. Janotti et al. Overcast: reliable multicasting with an overlay network. In *Proc. of the 4th Symposium on Operating System Design and Implementation*, Oct. 2000.

[4] J. Xie et al. Amroute: ad hoc multicast routing protocol. *ACM Mobile Networks and Applications*, 7(6), Dec. 2002.

[5] S. J. Lee et al. On-demand multicast routing protocol in multihop wireless mobile networks. *ACM Mobile Networks and Applications*, 7(6), Dec. 2002.

[6] S. Peng et al. Algorithms for a core and k-tree core of a tree. *Journal of Algorithms*, 15:143–159, 1993.

[7] S. Peng et al. A simple optimal parallel algorithm for a core of a tree. *Journal of Parallel amd Distributed Computing*, 20:388–392, 1994.

[8] M. Ge, S. V. Krishnamurthy, and M. Faloutsos. Overlay multicasting for ad hoc networks. In *Proc. of the Third Annual Mediterranean Ad Hoc Networking Workshop (MedHocNet 2004)*, pages 131–143, June 2004.

[9] C. Gui and P. Mahapatra. Efficient overlay multicast for mobile ad hoc networks. In *Proc. of IEEE WCNC2003*, March 2003.

[10] David B. Johnson and David A. Maltz. *Dynamic source routing in ad hoc wireless networks*. Kluwer Academic Publishers, 1996.

[11] S. J. Lee and W. Su. Performance comparison study of ad hoc wireless multicast protocols. In *Proc. of IEEE INFOCOM'00*, Mar. 2000.

[12] Yamin Li, Shietung Peng, and Wanming Chu. Mcore: A simple structure for effective overlay multicast on mobile ad hoc networks. In *Proceedings of The IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 341–346, Nov. 2006.

[13] Yamin Li, Shietung Peng, and Wanming Chu. K-mcore for multicasting on mobile ad hoc networks. In *Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 109–114. IEEE Computer Society Press, Dec. 2006.

[14] C. A. Morgan and P. J. Slater. A linear algorithm for a core of a tree. *Journal of Algorithms*, 1:247–258, 1980.

[15] Roman Novak, Joze Rugelj, and Gorazd Kandus. *Steiner tree based distributed multicast routing in networks, Steiner trees in industries (Combinatorial optimization, Vol. 11) Xiuzhen Cheng Ed.* Kluwer Academic Publishers, 2001.

[16] E. Royer and C. E. Perkins. Multicast operations of the adhoc on-demand distance vector routing protocol. In *Proc. of ACM MOBICOM'99*, Aug. 1999.

[17] C. W. Wu and Y. C. Tay. Amris: a multicast protocol for ad hoc wireless networks. In *Proc. of IEEE NILCOM'99*, Nov. 1999.