MCORE: A SIMPLE STRUCTURE FOR EFFECTIVE OVERLAY MULTICAST ON 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 an effective structure for overlay multicast to solve these problems in MANETs. Instead of using a spanning tree on the virtual mesh, we adopt a simple structure called MCore for multicast. An MCore is a path that minimizes the sum of the distances of all vertices to the path plus the length of the path. The MCore is more stable and easier to maintain than the spanning tree in MANETs. The simulation results show that our approach handles the flexibility and mobility issues in overlay multicast protocols effectively for large multicast group size.

KEY WORDS
Mobile ad hoc networks (MANETs), 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, 4, 5, 7, 8, 10, 13, 14, 15]. 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 [15] 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 [8].

To handle the efficiency issue in overlay multicast approach, minimum cost spanning tree on the virtual mesh is built. 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. We propose a new structure, called MCore, for the overlay multicast on the virtual mesh. An MCore is a path that minimizes the sum of the distances of all vertices to the path plus the length of the path. The MCore 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 MCore structure and distributed algorithm for the multicast in MANETs. Section 4 gives simulation results on the performance of multicasting using the MCore and compares these results to those of the AMRoute. Section 5 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 [15], PAST-DM [4], and ALMA [3].

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 A New Structure 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 for multicasting on the virtual mesh. This approach is beneficial when the multicast group is not small.

3.1 The Definition of MCore

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 [9]. Peng et al. developed a parallel algorithm for finding a core of a tree [11, 12]. A core of a tree is a path that minimes the total sum of the distances from every node in the tree to that path. It does not count the distances between the nodes on the path. However, for overlay multicast on mobile ad hoc networks, the cost of unicast among nodes on the path should be counted.

In this paper, we define an MCore for dealing with the multicast communication in mobile ad hoc networks. An MCore 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* $\alpha(G')$ and *outer cost* $\beta(G')$ as

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

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

where $d(u, G') = \min\{d(u, v) | v \in V(G')\}$ and $d(u, v)$ is the distance between nodes u and v.

We define the total cost $C_T(G') = \alpha(G') + \beta(G')$. Then, if G' is a spanning tree of G, $\beta(G') = 0$ and $C_T = \alpha(G')$. If $G' = \{u\}$ (stateless broadcast for instance), $\alpha(G') = 0$ and $C_T(G') = \beta(G')$.

For constructing an MCore from a spanning tree, 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. Then the inner and outer costs of $P(s, t)$ are

$$\alpha(P(s,t)) = d(s,t) = \sum_{e \in P(s,t)} w(e) \quad (3)$$

$$\beta(P(s,t)) = \sum_{u \in V(T)} d(u, P(s,t)) \quad (4)$$

where $d(u, P(s,t)) = \min\{d(u,v) | v \in V(P(s,t))\}$.

A path $P(s,t)$ is an MCore if the total cost $C_T(P(s,t)) = \alpha(P(s,t)) + \beta(P(s,t))$ is minimum for any $s, t \in V(T)$.

3.2 Basic Algorithmic Concepts

The MCore algorithm performs *branch-cut* operation inward from leaves. The branch-cut operation first identifies *candidates*. A candidate u is a nonleaf node and has at most one nonleaf neighbor v. The *branch* $B(u)$ is a subtree in T rooted at u. Figure 1 depicts a candidate u that has three leaf neighbors x, y, and z and a nonleaf neighbor v.

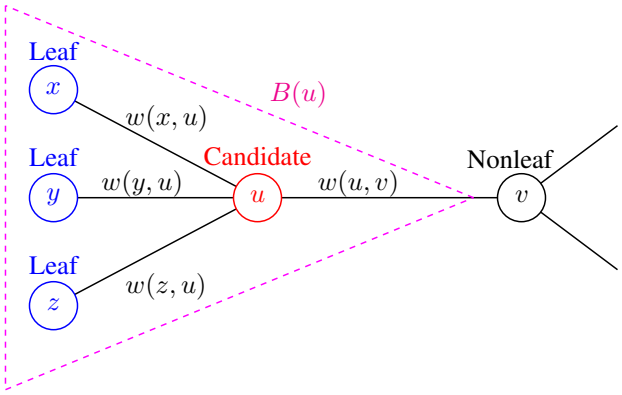

Figure 1. A candidate in a tree

The local MCore within the branch is calculated and saved in the candidate (we will describe how to calculate the MCore late in detail). Then, the leaf neighbors of the candidate are cut off and hence the candidate becomes a leaf node. The branch-cut operation continues and terminates when there is no more candidates.

Since all candidates can calculate the disjoint local MCores 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.

Now we introduce the algorithm for calculating the MCore. Because the tree connects the group members only, the multicast becomes broadcast to all the nodes on the tree. To let the nodes that are not on the MCore receive the broadcast message, the MCore nodes have the responsibility to send messages to those non-MCore nodes. This is done with unicast. That is, an MCore node unicasts the message to each non-MCore node in its branches individually. Referring to Figure 1, if node v is an MCore node and the others are not, node v must unicast the message to nodes x, y, z, and u.

A branch $B(u)$ has a *root* node u connecting to a nonleaf neighbor v. Let $C_U(u,v)$ be the unicast cost at which node v sends messages to each node in $B(u)$. Then the unicast cost

$$C_U(u,v) = \sum_{t \in B(u)} d(t,v) \quad (5)$$

For example, in Figure 1, $C_U(u,v) = d(x,v) + d(y,v) + d(z,v) + d(u,v) = (w(x,u) + w(u,v)) + (w(y,u) + w(u,v)) + (w(z,u) + w(u,v)) + (w(u,v)) = w(x,u) + w(y,u) + w(z,u) + |B(u)| \times w(u,v)$, where $|B(u)|$ is the number of nodes in $B(u)$.

Let $L(u)$ be a set of current leaf nodes in $B(u)$. If node u has $n-1$ leaf neighbors, we use l_i to denote each leaf node, where $i = 1, 2, \ldots, n-1$. In general, a leaf node l_i in $L(u)$ is a root of $B(l_i)$ and has $C_U(l_i, u)$, generated in the previous iteration. Then the cost of unicasting to $B(u)$ from v becomes

$$C_U(u,v) = \sum_{i=1}^{n-1} C_U(l_i, u) + |B(u)| \times w(u,v) \quad (6)$$

Let $C_C(u,v)$ be the cost of broadcasting from v to $B(u)$ if node u is selected as an MCore node candidate. Then,

$$\begin{aligned} C_U(u,v) &= C_U(x,u) + C_U(y,u) + C_U(z,u) \\ &+ |B(u)| \times w(u,v) \\ C_C(u,v) &= C_C(x,u) + C_U(y,u) + C_U(z,u) \\ &+ w(u,v) \end{aligned}$$

$C_U(u,v)$ is the same as Equation 6. In the $C_C(u,v)$ equation, the first term, $C_C(x,u)$, is the cost of broadcasting by u to $B(x)$ since node x was supposed to be an MCore node candidate. Nodes y and z are not on the MCore; the message must be unicasted by node u to the nodes in $B(y)$ and $B(z)$, so the second and third terms are $C_U(y,u)$ and $C_U(z,u)$, respectively. The last term, $w(u,v)$, is the cost for sending message from v to u. We have

$$\begin{aligned} C_U(u,v) - C_C(u,v) &= C_U(x,u) - C_C(x,u) \\ &+ (|B(u)| - 1) \times w(u,v) \end{aligned}$$

We define *saved cost*

$$C_S(u,v) = C_U(u,v) - C_C(u,v)$$

It is the cost that will be saved when node v broadcasts a message to $B(u)$ if node u changes its status from a non-MCore node to an MCore node. Then

$$C_S(u,v) = C_S(x,u) + (|B(u)| - 1) \times w(u,v) \quad (7)$$

where node x in $B(u)$ is also an MCore node.

The saved cost can be used for determining the MCore node candidate. In the next iteration, node v becomes a candidate and selects node u or node t as an MCore

node candidate. If node u is selected, then node v uses the MCore path $(l, \ldots, x, u, v)$ to multicast messages to all the nodes in $B(u)$ at cost of $C_C(u, v)$; and unicasts message to all the nodes in $B(t)$ at cost of $C_U(t, v)$.

The total cost $C_T(u, v)$ in this case is

$$C_T(u, v) = C_C(u, v) + C_U(t, v)$$

If node t is selected, the total cost $C_T(t, v)$ is

$$C_T(t, v) = C_C(t, v) + C_U(u, v)$$

Obviously, a node with the lower total cost should be selected as an MCore node candidate. That is, if $C_T(t, v) - C_T(u, v) > 0$, node u is selected. Note that

$$\begin{aligned} & C_T(t, v) - C_T(u, v) \\ = \; & (C_C(t, v) + C_U(u, v)) - (C_C(u, v) + C_U(t, v)) \\ = \; & (C_U(u, v) - C_C(u, v)) - (C_U(t, v) - C_C(t, v)) \\ = \; & C_S(u, v) - C_S(t, v) \end{aligned}$$

If $C_S(u, v) = C_S(t, v)$, we can select either node u or t as the MCore candidate.

We conclude that if $C_S(u, v) \geq C_S(t, v)$, node u is selected as an MCore node candidate. In case node v has more than two leaf neighbors, a neighbor node with the highest saved cost is selected. Any one can be selected if there are two or more neighbors with the equal highest saved cost.

3.3 Distributed Algorithm

In this subsection, we show a distributed algorithm that implements the algorithmic concepts described in the previous subsection. The algorithm uses only local information and works asynchronously.

In our algorithm, we store the following information in each node u:

1. The number of nodes in $B(u)$, denoted as $u.nnodes$ in Algorithm 1 which we will give later.
2. The saved cost $C_S(u, v)$, denoted as $u.scost$.
3. The node list, $u.list$, containing all the nodes in $B(u)$, used for unicast.
4. The MCore path candidate, $u.path = l_k.path \cup u$, where $l_k.path$ is the MCore path candidate stored in node l_k and $C_S(l_k, u)$ is the highest one among $C_S(l_i, u)$ for $i = 1, 2, \ldots, n-1$ and $l_i \in L(u)$.

The information stored in an original leaf node x consists of 1) $x.nnodes = 1$; 2) $x.scost = 0$; 3) $x.list = \{x\}$; and 4) $x.path = \{x\}$.

Figure 2 shows an example of the tree network. Note that the links in the tree are paths that may contain non-member nodes. For clarity, the non-group member nodes are not shown in the figure.

As shown in Figure 2, nodes a, b, c, and d calculate their saved costs $C_S(a, e)$, $C_S(b, f)$, $C_S(c, g)$, and

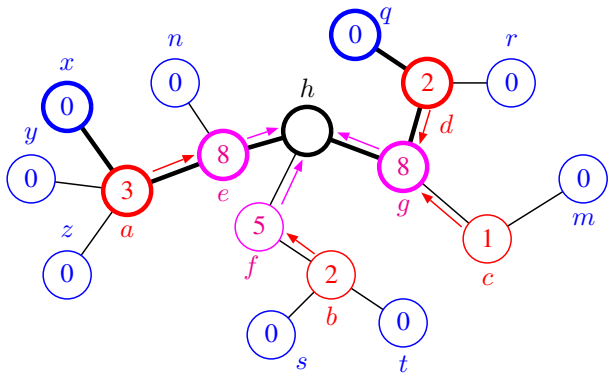

Figure 2. Saved cost calculation during branch-cut

$C_S(d, g)$ in parallel by using Equation 7. For simplicity, we set all the edge-weights to 1. $B(a)$ has 4 nodes: 3 leaf nodes and a root node a. The saved cost $C_S(a, e)$ is $0 + (4 - 1) \times 1$, or 3. The calculated saved costs are shown inside the cycles. All the leaves are cut-off and nodes a, b, c, and d are marked as leaves. Then, nodes e, f, and g calculate their saved cost $C_S(e, h)$, $C_S(f, h)$, and $C_S(g, h)$. After marking nodes e, f, and g as leaves, node h finds that it has leaves only (no nonleaf neighbor), this means that the branch-cut finishes.

To generate $u.path$ for u, we select the MCore path candidate with the highest saved cost generated in the previous iteration and add u to form $u.path$. For example, in Figure 2, node g has two branches: $B(c)$ and $B(d)$. We select $d.path$ and let $g.path = d.path \cup g$.

Let node h in Figure 2 be an MCore node. From the branches of node h, we select two neighbors with the highest saved cost and the second highest saved cost (nodes e and g in Figure 2) as the MCore nodes. Then, we get the final MCore path: $h.path = e.path \cup h \cup g.path = \{x, a, e, h, g, d, q\}$. The resulting MCore in Figure 2 is drawn with thick cycles (nodes) and lines (links). Note that the two end nodes of the MCore, nodes x and q, have no contributions to minimize the cost of multicasting and can be deleted from the MCore: just treat them as the same as the non-MCore nodes.

When the MCore is used for multicasting, each node in the MCore except for the two end nodes should unicast message to nodes in its *cluster*, which is a set of all nodes in the branches whose roots are not MCore nodes. If the source node broadcasting a message is an MCore node, it sends the message to its MCore neighbors as well as the nodes in its cluster. Once receiving the message, each MCore node sends the message to all the nodes in its cluster by unicast and to the next MCore node, and so on. If the source node is a non-MCore node, it sends the message to its MCore node.

There are two cases at which the branch-cut procedure will terminate: A candidate has no nonleaf neighbor (Case 1), or there are two candidates, each acts as the nonleaf neighbor of the other (Case 2). In Case 2, only one of these

```
Algorithm 1: Find_MCore
Input: A weighted tree T of size > 2
Output: An MCore of T
begin
  u = my_node_id;
  n = degree(u);
  if (n = 1)
    send Message(u,LEAF,1,0,{u},{u}) to the neighbor;
  else send Message(u,NONLEAF,0,0,∅,∅) to all neighbors;
    n_leaves = n_nonleaves = 0;
    calculated = FALSE;
    while (true)
      receive Message from u_i and store to TABLE[i];
      if (TABLE[i].state = LEAF)
        n_leaves++;
      endif
      if (TABLE[i].state = NONLEAF)
        n_nonleaves++;
        v = TABLE[i].node_id;
      endif
      if (n_leaves = n)
        find u_j and u_k such that u_j.scost and u_k.scost are
          the highest two among all u_i, 1 ≤ i ≤ n;
        broadcast u.path = u_j.path ∪ u ∪ u_k.path;
        exit();
      endif
      if (n_leaves = n − 1) and (n_nonleaves = 1)
        calculate u.nnodes, u.scost, u.list, and u.path;
        send Message(u,REQ,u.nnodes, u.cs, u.list, u.path) to v;
        calculated = TRUE;
      endif
      if (TABLE[i].state = REQ)
        if (calculated = FALSE)
          send Message(u,ACK,0,0,∅,∅) to u_i;
        else
          if (u > TABLE[i].node_id)
            send Message(u,LEAF,u.nnodes, u.cs, u.list, u.path) to v;
            exit();
          endif
        endif
      endif
      if (TABLE[i].state = ACK)
        send Message(u,LEAF,u.nnodes, u.cs, u.list, u.path) to v;
        exit();
      endif
    endwhile
  endif
end
```

two nodes can proceed; the other should wait until the first one is done. In our algorithm, we set a priority according to their *node_id*s.

The proposed distributed algorithm for finding an MCore in a weighted tree is formally presented in Algorithm 1. Each nonleaf node u maintains a table of local status of neighboring nodes, TABLE, of size $n = degree(u)$. Let the neighboring nodes of u be $u_1, \ldots, u_n$. Each nonleaf node receives messages from its neighbors and stores into the corresponding entry of TABLE. Each entry of TABLE has 6 fields: TABLE$[i]\langle node_id,\ type,\ nnodes,\ scost,\ list,\ path\rangle$. Field $node\_id$ indicates who sent the message; $type$ indicates the types of the message. There are 4 types: `LEAF` means that $node\_id$ is or becomes a leaf node; `NONLEAF` indicates that $node\_id$ is a nonleaf node; `REQ` is a request used for synchronization between a candidate and its nonleaf node when the candidate attempts to change its status from nonleaf to leaf; and `ACK` is an acknowledgment to the request; $nnodes$ is the number of nodes in branch $B(node\_id)$; $scost$ is the saved cost of $node\_id$; $list$ is the node list in $B(node\_id)$, and $path$ is an MCore subpath candidate with an end node $node\_id$.

If a candidate u wants to change its status to leaf, it must get a permission from its nonleaf node v. This is done by sending a request to v. If v is not a candidate, v sends an acknowledgment to u. Once u gets the acknowledgment, it changes status to leaf and reports to v. In case v is also a candidate and its nonleaf node is u, by comparing their *node_id*s, one of the two nodes u and v will be done first and the other is responsible for generating the final MCore. Note that in Algorithm 1, after receiving a message from neighbor, the executions of `if` clauses can be out of order.

The simulation results presented in the next section show that multicasting based on the MCore structure is more stable and can tolerate higher mobility than AM-Route.

4 Performance Analysis and Simulations

The network for the performance simulation is configured as below. There are 200 nodes randomly roaming within a 2000m × 1500m area. The radio transmission range of each node is set to be 300m, 400m, and 500m. The group size is chosen to be 10 to 100, in steps of 10. Each configuration runs 100 trials.

Figure 3 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.

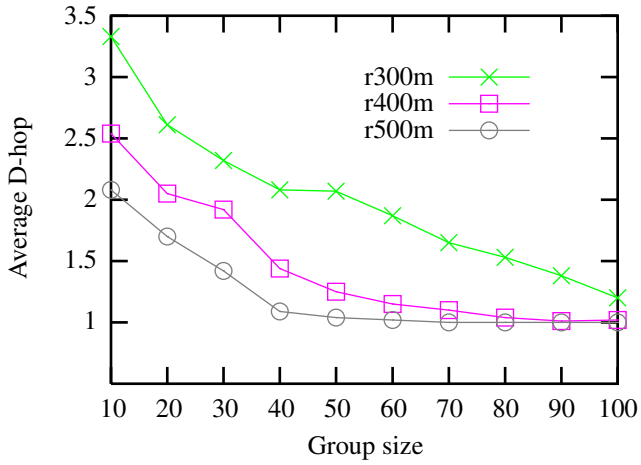

Figure 3. Average D-Hop

Figure 4 shows the size of the MCore, i.e., the number of member nodes that form the MCore. This size is relatively small compared to the group size. Also, when

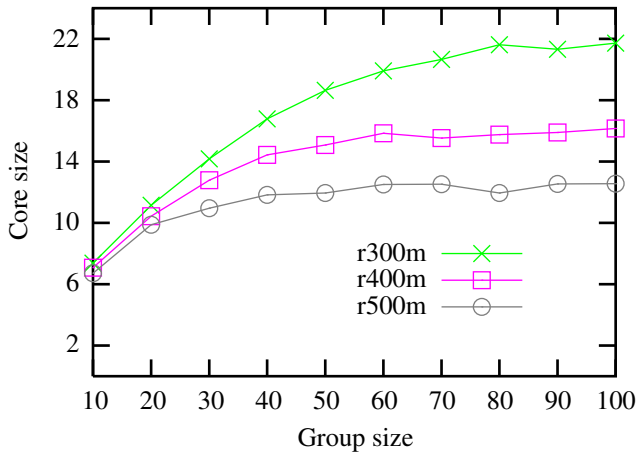

Figure 4. Core size

the group size is large, say 50, adding new members to the group will not affect the size of MCore obviously.

The MCore maintains fewer nodes that re-send the received message than AMRoute for multicast. Of course, the message delivery cost of the MCore is higher than that of AMRoute. But, from Figure 5, we can see that the increased cost is quite small. The message delivery cost here is just the sum of physical hop length a message travals when the 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.

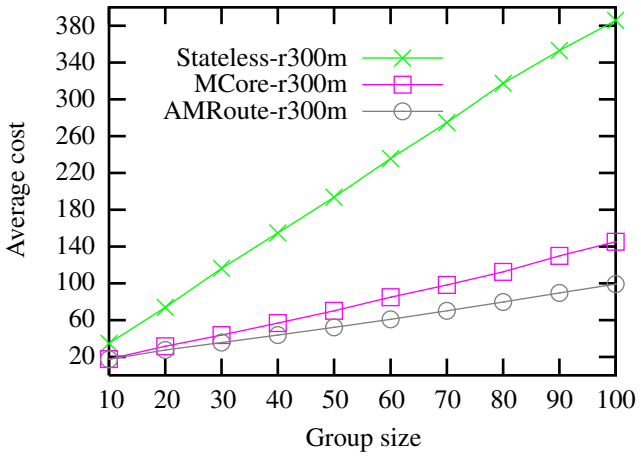

Figure 5. Average cost: resp. group size

5 Concluding Remarks

A new infrastructure based on MCore for overlay multicasting on mobile ad hoc network was proposed, an efficient distributed algorithm for finding an MCore 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.

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] 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.

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

[5] 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.

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

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

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

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

[10] 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. pp. 327-351.

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

[12] 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.

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

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

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