K-MCore for Multicasting on Mobile Ad Hoc Networks

Yamin Li and 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

A k-cluster of a tree includes a single path and $k-1$ subpaths growing from that path. A k-MCore is a k-cluster that minimizes the sum of the distances of all vertices to the cluster plus the size of the cluster. This structure is motivated by the applications on overlay multicasting. The overlay multicast protocol constructs a virtual mesh spanning all member nodes of a multicast group. It employs standard unicast routing and forwarding to fulfill multicast functionality. In this paper, we propose effective distributed algorithms for constructing k-MCore on a tree network. The k-MCore is more stable and easier to maintain than the spanning tree in virtual mesh. The simulation results show that our approach handles the flexibility and mobility issues in an overlay multicast protocol effectively, especially when the group size is large.

1. Introduction

Mobile ad hoc network (MANET) refers to a form of infrastructureless network connecting mobile devices with wireless communication capacity. Each node in MANET 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 MANET, the multicast support is critical since the close cooperation among team members is required for many MANET applications.

Multicasting in MANET faces many challenges due to the continuous changes in network topology (mobility) and limited channel bandwidth. Many multicast routing protocols have been proposed for MANET [1, 4, 5, 7, 8, 9, 11, 12, 13]. 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 MANET, 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 [13] 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 an effective structure called k-MCore, for the overlay multicast on the virtual mesh. The selection of the value k largely depends on size of the spanning tree at hand. A small k is enough practically for most of the networks and the communication groups. The k-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 MANET. Section 3 presents k-MCore, a new structure for the multi-

cast on virtual mesh. Section 4 gives distributed algorithms for finding k-MCore. Section 5 provides simulation results on the performance of multicasting using the k-MCore and compares these results to those of 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 [13], 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 MANET environment. 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 the previous section, 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 and update of the multicasting tree will become costly and inefficient. 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.

In this paper, we propose a *k-MCore* for handling the multicast communication in mobile ad hoc networks. We say a subgraph of the tree is a k-cluster if it includes a single path C and $k-1$ subpaths; each subpath is a path with one-end at a node in C. A k-MCore of a tree is a k-cluster in the tree that minimizes the total cost of the multicast communication (to be defined later). The k-MCore is a simple infrastructure to support the overlay multicast in mobile ad hoc networks at application level. A similar structure called k-tree core in tree networks had been proposed and studied by Peng et al [10]. However, the k-tree core is a subtree with k leaves and the object function to be minimized for the k-tree core is also different with the k-MCore defined in this paper.

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 with edge e. Let G' be a connected subgraph of G, we define $\alpha(G') = \sum_{e \in G'} w(e)$. For any node $u \in V(G)$, we define $d(u, G') = \min\{d(u, v) | v \in V(G')\}$ and $\beta(G') = \sum_{u \in V(G)} d(u, G')$. Then, we define $\gamma(G') = \alpha(G') + \beta(G')$. In this paper, we consider $\gamma(G')$ as the object function that we want to minimize.

In the wireless ad hoc networks, we can consider $\alpha(G')$ as an *inner cost* and $\beta(G')$ as an *outer cost* for the overlay multicast using G' as infrastructure. For example, if Steiner tree T of the virtual mesh is used then $\beta(T) = 0$, and in case of stateless networks (no infrastructure), we have $G' = \{u\}$ and $\alpha(G') = 0$. The motivation of this paper is to propose a new structure, called k-MCore, which has a simpler struc-

ture than the spanning tree and the total cost $\gamma(G')$ for the multicast is minimized.

Let T be an edge-weighted tree with vertex set $V(T)$ and edge set $E(T)$. Each $e \in E(T)$ has a weight $w(e) > 0$. Let C_k be a k-cluster of T. Let F be the set of all k-clusters of T. Then a k-MCore is a k-cluster that minimizes the total cost $\gamma(C_k) = \alpha(C_k) + \beta(C_k)$ among all C_k in F, where $\alpha(C_k) = \sum_{e \in E(C_k)} w(e)$, $\beta(C_k) = \sum_{u \in V(T)} d(u, C_k)$, and $d(u, C_k) = \min\{d(u, v) | v \in V(C_k)\}$. In this paper, we propose efficient distributed algorithms for finding a k-MCore of the spanning tree of a virtual mesh and perform simulations for the overlay multicast using k-MCore.

To keep the structure simple, we want k to be a small integer. If the network is not large, say the number of nodes is 100 or less, then an 1-MCore is sufficient for efficient overlay multicast. However, when the size of the virtual mesh grows, the k-MCore with $k > 1$ should be considered. We expect that $k \leq 5$ is practically sufficient for handling the overlay multicast efficiently in large networks.

In order that the proposed k-MCore can be used as a multicast protocol at application level, efficient distributed algorithms for constructing a k-MCore from a weighted tree that uses local information only must be developed. We propose efficient distributed algorithms for finding a k-MCore in the next section.

4. The Algorithms for Finding a *K*-MCore

4.1. Algorithm for 1-MCore

We use MCore to denote 1-MCore. 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. The local MCore within the branch is constructed and saved in the candidate (we will describe how to construct the MCore late in details). Then, the leaves 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 construct 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 constructing 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.

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 v sends messages to each node in $B(u)$. Then the unicast cost

$$C_U(u, v) = \sum_{t \in B(u)} d(t, v). \quad (1)$$

Let $L(u)$ be a set of current leaf nodes in $B(u)$. If there are $n-1$ nodes in $L(u)$, 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

$$C_U(u, v) = \sum_{i=1}^{n-1} C_U(l_i, u) + |B(u)| \times w(u, v). \quad (2)$$

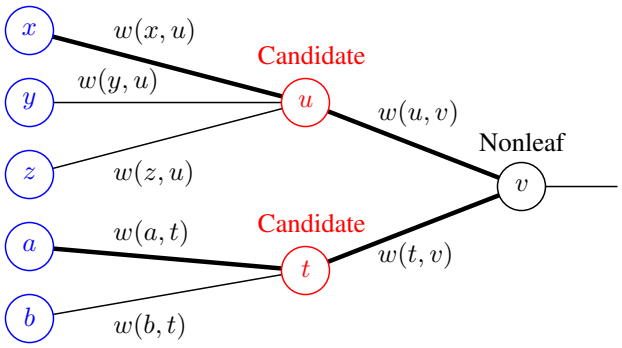

Figure 1. Core path selection

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. We define *saved cost* $C_S(u, v) = C_U(u, v) - C_C(u, v)$, the amount of cost 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. Suppose x is also an MCore node, then

$$C_S(u, v) = C_S(x, u) + (|B(u)| - 1) \times w(u, v). \quad (3)$$

The saved cost is the key measure used for determining the MCore node candidate. The node with larger saved cost should be selected as an MCore node candidate. Referring to Fig. 1, After the two candidates u and t finish the saved cost calculation, node v becomes a candidate, and we simply compare $C_S(u, v)$ and $C_S(t, v)$ to select node u or node t as an MCore node candidate.

In our algorithm, we store the following information in 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) $x.path = \{x\}$. 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 $u.path$.

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,$\emptyset$,$\emptyset$) 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++;
      if (TABLE[i].state = NONLEAF)
        n_nonleaves++;
        v = TABLE[i].node_id;
      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, $1 \le i \le n$;
        broadcast $u.path = u_j.path \cup u \cup u_k.path$;
        exit();
      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;
      if (TABLE[i].state = REQ)
        if (calculated = FALSE)
          send Message(u,ACK,0,0,$\emptyset$,$\emptyset$) 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();
      if (TABLE[i].state = ACK)
        send Message(u,LEAF,$u.nnodes$, $u.cs$, $u.list$, $u.path$) to v;
        exit();
end

The multicast using MCore is described below. If the source node broadcasting a message is an MCore node, it sends the message to its MCore neighbors as well as all the local non-MCore nodes. Once receiving the message, each MCore node sends the message to all the local non-MCore nodes 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 two cases at which the branch-cut procedure terminates: Case 1 — a candidate has no nonleaf neighbor. This node is marked as an MCore node. Case 2 — two candidates act as the nonleaf neighbors each other. In this case, only one of these two nodes is allowed to proceed by setting priority based on their *node_id*s. The handshaking or synchronization between the two nodes is required.

The proposed distributed algorithm for finding an MCore in a weighted tree is formally presented in Algorithm 1. The algorithm uses only local information and works asynchronously. 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 the nonleaf node is u, by comparing their *node_id*s, one of the two nodes u and v will do 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.

4.2. Algorithm for *K*-MCore

The algorithm for constructing k-MCore with $k > 1$ adopts an incremental approach. That is, the construction of a k-MCore is based on the $(k-1)$-MCore. It first finds an MCore and then use a greedy strategy to extend the MCore into a 2-MCore and then 3-MCore and so on.

Algorithm 2: Find_k-MCore
Input: A tree T, an integer k, an MCore L , local core P_v
  and $C_S(P_u)$ for each $v \in N(u)$ and each $u \in L$
Output: A k-MCore of T for $k > 1$
begin
  for $i = 1$ to $k - 1$ **do**
    $u = L$.first; */* initial setting */*
    $B = \emptyset$;
    $C_S(B) = 0$;
    s = null;
    while $u \neq$ null */* search for the ith branch */*
      find $v \in N(u)$ s.t. $C_S(P_u) \le C_S(P_w)$ for $w \in N(u)$;
      if $C_S(P_v) > C_S(B)$
        $B = P_v$;
        $s = v$;
      $u = L$.next;
    remove s; */* Node s is excluded for next round */*
    output B; */* The ith branch is found */*
end

After the algorithm for finding an MCore is done, we have a linked list L that is the MCore. Each core node u in L keeps a list $N(u)$ of all its neighbors that are not core nodes together with their local core P_v, $v \in N$and the values of the saved cost $C_S(P_v)$. The algorithm for finding a k-MCore for $k > 1$ is formally presented in Algorithm 2.

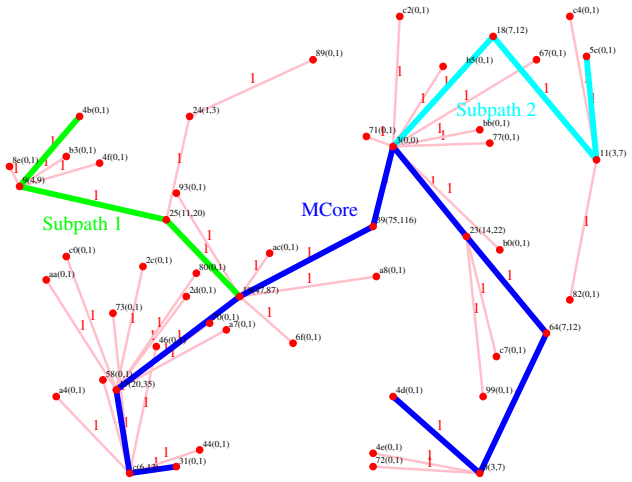

Figure 2. An example of 3-MCore

Fig. 2 shows an example of mobile ad hoc networks. The multicast group size is 50. There are 200 mobile nodes randomly locating within a 2000m $\times$ 1500m area, but only the group members are shown in the figure. Each member node is given with $id(cs, cu)$ where id is the node identification, cs is the saved cost, and cu is the unicast cost. Node 3 is the last candidate when the MCore is constructed. The MCore is marked with dark-blue lines; and the two subpaths in a 3-MCore are marked with green and cyan lines, respectively.

The k-MCore should be periodically updated as that in AMRoute. The simulation results presented in the next section show that multicasting based on the k-MCore structure is more stable and can tolerate the higher mobility than that in AMRoute.

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 300m, 400m, and 500m. The group size is chosen to be 10 to 100, stepped by 10. Each configuration runs 100 trials.

Fig. 3 shows the MCore size — the number of nodes of MCore. The MCore size is relatively small compared to the multicast group size. Also, increasing the radio transmission range reduces the MCore size.

Fig. 4 shows the message delivery cost. The message delivery cost here is simply defined as the sum of physical

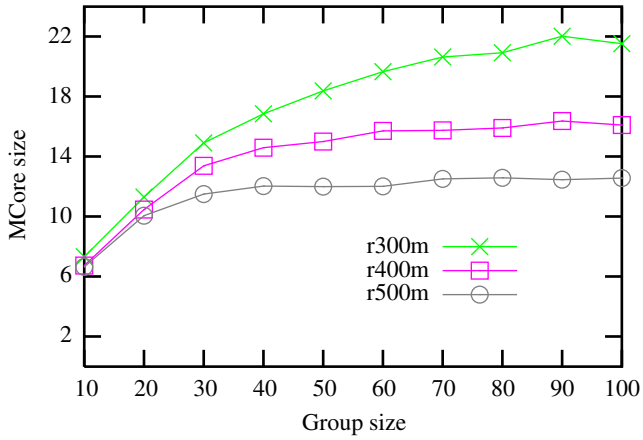

Figure 3. Average MCore size

hop length of virtual links of the MCore when a message is multicasted to all the group members. Increasing the radio transmission range will decrease the cost but the effect is not obvious.

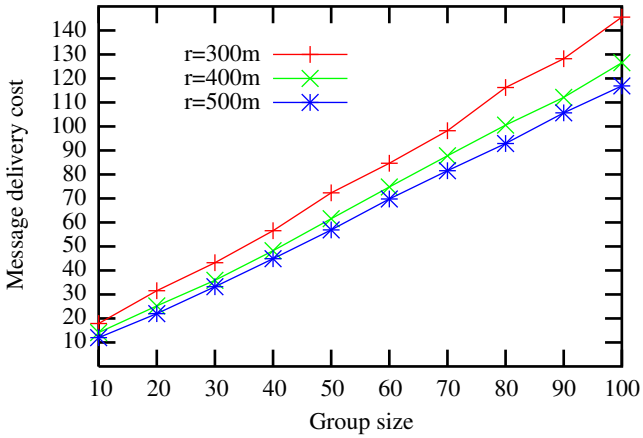

Figure 4. Average cost

Tab. 1 lists the message delivery costs of k-MCore ($1 \leq k \leq 4$) and AMRoute with the radio transmission range of 300m. The table also lists the cost for stateless transformation in which the message is sent to every member individually by unicast routing.

Fig. 5 depicts the costs listed in the table. By simply adding subpaths, the message delivery cost is closed to that of AMRoute, but k-MCore structure maintains fewer nodes than AMRoute. From our simulation, we conclude that the k-MCore with two or three subpaths provides better maintenance-cost/performance.

The virtual k-MCore remains static even though the underlying physical topology is changing. We also inves-

Table 1. Average cost

#Mem	Stateless	MCore	AMRoute	2-MC	3-MC	4-MC
10	35.85	17.88	17.33	17.28	17.11	16.94
20	76.22	31.53	27.46	28.30	27.57	27.49
30	115.7	43.17	35.58	38.02	36.41	35.90
40	154.0	56.60	43.85	48.98	46.12	44.72
50	191.8	72.28	52.01	61.33	56.32	53.94
60	234.5	84.68	61.03	71.67	66.51	63.95
70	273.0	98.19	70.06	84.07	77.95	74.60
80	311.9	116.2	79.78	98.22	89.32	85.18
90	346.1	128.2	89.32	108.0	100.3	96.15
100	394.5	145.5	99.25	123.7	113.0	107.8

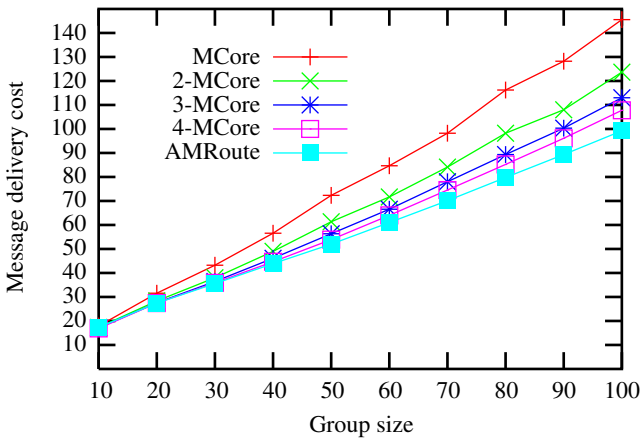

Figure 5. Average cost

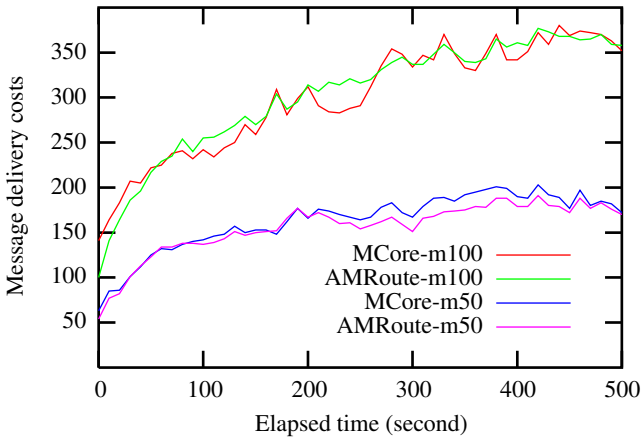

Figure 6. Cost with mobility

tigated the mobility effect on the message delivery cost. The movement of each node follows the random waypoint model [6]: 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.

Fig. 6 shows the time-line of the costs of the AMRoute and the MCore for multicast group sizes 50 and 100 at the radio transmission range of 300m. As the member node moves, the message delivery costs of both the AMRoute and MCore increase. In practice, the MCore structure must be re-constructed periodically, like the AMRoute does.

6. Concluding Remarks

We proposed k-MCore for overlay multicasting on mobile ad hoc network, and evaluated the performance through simulations. The k-MCore structure may be applied to some other areas. Some theoretical study on the k-MCore is interesting for further research.

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] D. B. Johnson and D. 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] R. Novak, J. Ruge, and G. Kandus. Steiner tree based distributed multicast routing in networks. *Steiner Trees in Industries*, 8(5):1–25, 2000.

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

[11] 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.

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

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