MODE for Mobile
– An Efficient Overlay Multicast Protocol for Heterogeneous Users –

Thilmee M. Baduge, Hirozumi Yamaguchi and Teruo Higashino
Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, JAPAN
{thilmee, h-yamagu, higashino}@ist.osaka-u.ac.jp

ABSTRACT

In this paper, we present a protocol called MODE-for-mobile (MODE-m in short). MODE-m constructs a *Degree-Bounded Minimum Diameter Tree (DBMDT)* as an overlay network in a distributed manner. MODE-m is extended from our previous protocol called MODE, to incorporate instable and less powerful hosts such as mobile nodes. We have shown from the experimental results that this feature is very effective to keep the diameter as small as possible under the existence of mobile nodes, compared with MODE.

Keywords: Overlay Multicast, Degree-bounded Minimum Diameter Tree, Decentralized Algorithm and Mobile Hosts

1 Introduction

The deployment of mobile devices has brought us a new innovation of our communication styles. It enables us to communicate with other people from/to anywhere easily. Accordingly, people will change their communication styles so that, using mobile devices, they can use advanced communication facilities which they formerly used for desktop PCs in their office and home. In particular, many-to-many interactive communication is one of such facilities. A typical example is multi-user video-chatting/conference systems. Let us assume that some people use their hand-held devices and others use their desktop PCs, and they all together hold a conference where each participant's voice/video is interactively broadcast to all the other members.

Overlay multicast[1]–[5] is a reasonable solution where a spanning tree or a mesh-like network involving all active participants (referred to as *nodes* hereafter) of the session is constructed by connecting those nodes via unicast channels (referred to as *overlay links* hereafter) as shown in Fig.1. Each node relays incoming data packets from an overlay link to the other overlay links attached to the node. Considering the characteristics of the many-to-many interactive applications including mobile nodes, the overlay multicast is required to satisfy the following requirements. (i) The maximum delay of the tree, called *diameter*, should be minimized. This is because the longest delay affects the delay of the session. In Fig.1, the diameter path is $a - b - c - d$ and the diameter is 8 units of time. This means that each node must wait for 8 units of time to guarantee all the other nodes have received the transmitted data. (ii) Bandwidth constraints around nodes should be taken into account. Since overlay links through a node actually use the same network interface of the node, the

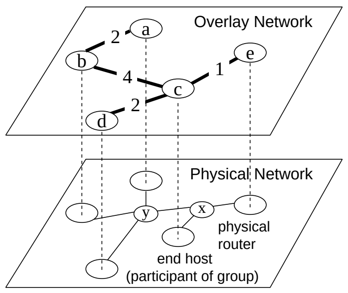
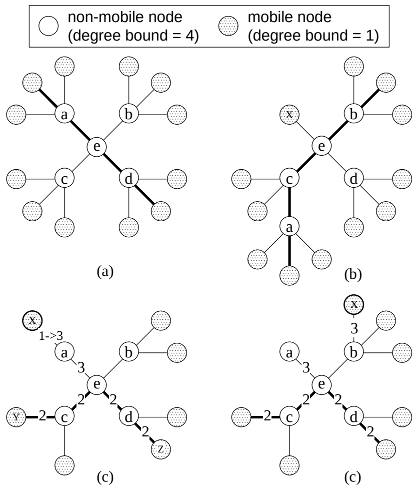


Figure 1: Concept of overlay multicast

traffic amount of the node depends on its *degree*, the number of overlay links of the node. For example, since node c has three overlay links, the degree of c is three. So the degree should not exceed the capability limitation called *degree bound*, of the node. (iii) Some nodes may be mobile terminals. Therefore, those hosts may not be stable due to limitation of batteries, computing and communication capabilities and so on, and thus not stable to relay data packets. Moreover, they may use wireless links and thus delay of the overlay links connected to those nodes may not be stable also. In such an environment, overlay multicast should be constructed carefully and dynamically to prevent those hosts from staying at critical positions that affect the diameter and stability of the overlay multicast.

The protocols MODE in our previous work [4] and OMNI in Ref.[5] have presented distributed solutions for the requirements (i) and (ii). However, as far as we know, none of the existing work, including MODE and OMNI, has not dealt with the third issue, which is a very important issue to realize seamless communication between non-mobile and mobile nodes in group communication.

In this paper, we present a protocol called MODE-for-mobile (MODE-m in short). MODE-m is extended from MODE protocol to incorporate instable and less powerful hosts such as mobile nodes. The original MODE aims at constructing a *Degree-Bounded Minimum Diameter Tree (DBMDT)* in a distributed manner. MODE-m has the same goal as MODE, however supporting mobile nodes is a new feature to handle group communication with diverse hosts. In MODE-m, if mobile nodes occupy intermediate positions of the current tree and the diameter becomes large, such nodes are set to leave and re-join the tree so that the diameter of the tree can be held short. Our experimental results have shown this feature is very effective to keep the diameter as small as possible under the existence of mobile nodes.



non-mobile node (degree bound = 4)
mobile node (degree bound = 1)
(a)
(b)
(c)
(c)
1->3

Figure 2: Concept of MODE-m

2 Related Work and Motivation

Most researches of overlay multicast pursue the stability (*i.e.* fault-tolerance) and efficiency of overlay multicast trees under the constraints of bandwidth and delay. For example, ALMI [2] proposes a centralized algorithm for constructing minimum spanning trees. Yoid [3] is similar to ALMI, however, Yoid together uses shared tree connection and mesh-like connection for robustness. Narada [1] considers mesh-like overlay networks where a shortest path tree per source is constructed. NICE [6] considers hierarchical topology where leaders organize their logical sub-domains.

The original MODE protocol is different from the above approaches, since *it presents distributed heuristics to optimize diameters which is adaptive to multiple nodes' failures*. The detailed comparison of MODE with the existing work is described in Ref.[4]. Inheriting the advantage, MODE-m in this paper can adaptively determine the positions of mobile nodes to *prevent these incapable nodes from affecting the diameters*.

We illustrate how MODE-m works in comparison with MODE in Fig.2. For simplicity, in the following figures, we only illustrate overlay networks, and physical networks are omitted. Also an integer on an overlay link represents its delay, and description of delay "1" is omitted. Finally, diameter paths are denoted by thick lines. The essential difference between MODE and MODE-m is illustrated in Fig.2(a) and Fig.2(b). In Fig.2(a), non-mobile nodes (white circles) with degree four are connected near the center (node e in this case) of the tree, and mobile nodes (meshed circles) with degree one (*i.e.* they never relay packets) are located in the leaf positions. On the other hand, in Fig.2(b), a mobile node X is connected to node e, thus node a is connected to node c and consequently they form a longer diameter path. The case of Fig.2(b) will happen in MODE, since nodes incrementally join the current tree and tree optimization is done only when a node in the tree leaves. Therefore, once a mobile node occupies a position close to the center, it continues staying their without providing degrees. This results in pushing the following joining nodes like node a away from the center and making the diameter longer. In MODE-m, if a non-mobile node with higher degree bound joins the current tree, our protocol makes it preempt the position of a mobile node near the center by letting the mobile node leave and re-join the tree. Thus we can expect that a tree like Fig.2(a), which is well organized and has a shorter diameter, will be formed eventually. Another feature is that mobile nodes adaptively move if they know that they become to form diameter paths due to variation of overlay link delay. For example, in Fig.2(c), the diameter path is the path between node Y and Z. Then let us assume that the delay of overlay link X-a changes to 3 and the path X-Z becomes the diameter path. In the original MODE, each node knows the current diameter and the "height" of the tree rooted at the neighbor of the node by periodical collection of diameter information[1]. Using this information, node X can know whether the variation of delay makes the path be the diameter path or not. In MODE-m, if it is true, node X spontaneously leaves and re-joins the tree to find a better position (Fig.2(d)). If this variation is caused by the "last one hop" wireless link, the overlay link delay between X and the new neighbor may be the same as before. However, the diameter, and thus the session delay, will be improved as in Fig.2(d).

3 Overview of MODE-m

3.1 DBMDT Problem

The definition of Degree-Bounded Minimum Diameter Tree (DBMDT) is given below. Let $G = (V, E)$ denote a given undirected complete graph where V denotes a set of nodes and E denotes a set of potential overlay links which are unicast connections between nodes. Also let $d_{max}(v)$ denote the degree bound (the maximum number of overlay links) on each node $v \in V$, and let $c(i, j)$ denote the cost (delay in this paper) of each overlay link $(i, j) \in E$. DBMDT is a spanning tree T of G where the diameter of T (the maximum overlay delay between two arbitrary nodes on T) is minimum and the degree of each node $v \in V$ (denoted as $d(v)$) does not exceed $d_{max}(v)$. Hereafter, d_{max} is used to represent the maximum degree bound of all the nodes (*i.e.* $\max_{v \in V}\{d_{max}(v)\}$).

In this paper, we assume that nodes are classified into two types, mobile nodes and non-mobile nodes. Also we assume that the degree bounds of the mobile nodes are all one and the delay of overlay links connected to mobile nodes may change from time to time.

3.2 Outline of MODE for Mobile

The main effort of our research is to present a new DBMDT constructing protocol to support *mobile nodes*. The basic idea is inspired from the protocol MODE[4], which is one of our prior works. Mobile nodes we discuss here are nodes with low-bandwidth and less capability of computing, which are connected via wireless links to wired networks. Therefore,

[1]The information is aggregated at each node and the amount of information is very small in each collection packet. This is one of the key advantages of MODE.

we set mobile nodes' degree bounds to only one (*i.e.* they never relay packets) and treat them in a special way where we always keep them as leaf nodes of the tree.

In MODE-m, we follow MODE's concept of repeating two phases, the *collection phase* and the *normal phase*, alternately. The protocol carries out a certain information gathering in the collection phase and this information is used to construct, refine and repair the spanning tree in the normal phase which starts right after that collection phase. The normal phase stands for the time-period between two collection phases. MODE-m has three procedures, *join procedure*, *refresh procedure* and *leave procedure*, in addition to the information collection. These are responsible for adding newly joining nodes, stabilizing the diameter fluctuation caused by mobile nodes and repairing isolated trees caused by nodes' disappearance, respectively. The following sections describe these procedures.

Nodes may disappear from a tree at any timing. In order to discuss the consistency of MODE-m, we give the following assumptions concerned with the disappearances of nodes.

G1. Any node's disappearance does not affect the physical (underlying) network, and each node can immediately detect its neighboring node's disappearance. And each mobile node can detect any change of the delay between itself and it's peer node immediately.

G2. At least one node never disappears and each new node which wants to join the tree knows the network address (*e.g.* IP address) of this node. We call this node *root node*. This does not mean that the root node plays a role of a centralized node. It only works as a well-known node required for new nodes' participations.

G3. A node's disappearance doesn't result in disappearance of any control message.

3.3 Information Collection

In this section, we provide a brief explanation about the collection phase protocol.

The task of the collection phase protocol is to let each node know the *sub-tree information*. Let us assume that a spanning tree has been constructed among nodes. For a pair of two adjacent nodes x and y, let $T_{x,y}$ denote the sub-tree rooted at node y, which does not contain x. The *sub-tree information* of $T_{u,v'}$ consists of the IDs and addresses of, (i) node v' and its neighbors except u and (ii) the center node of $T_{u,v'}$ (denoted as $centerNode(T_{u,v'})$) (Fig.3). Node v has sub-tree information of $T_{u,v'}$ for every pair of a neighboring node u and u's neighboring node v' (v' may also be v). For example, node v in Fig.3 knows the sub-tree information of $T_{u,v}$, $T_{u,v'}$ and $T_{u,v''}$.

Then we briefly describe how messages are exchanged to collect those information. As we mentioned above, we assume that the root node never disappears (see assumption G2). The root node starts information collection, (*i.e.* the collection phase) at every regular interval by sending *synchronization messages* to the other nodes along the current tree. A

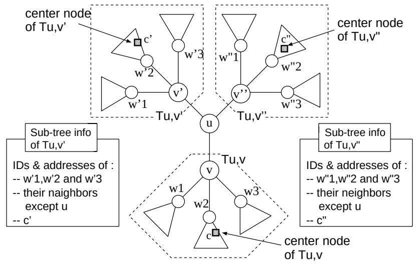


Figure 3: Sub-tree information

node enters the collection phase if it has received a synchronization message from its parent (the node in the direction to the root node) and sent synchronization messages to all its children. A leaf node does not send synchronization messages. Instead, when it receives a synchronization message, it enters a collection phase and sends a *collection message* to its parent. Each non-leaf node in the collection phase acts as follows. Whenever it receives collection messages from all the neighboring nodes except one neighboring node (say x), it sends a collection message to node x. Each node u leaves the collection phase if, u has received collection messages from each neighbor node v and has sent a collection message to v. This means that in a collection phase, $2(n-1)$ collection messages are exchanged on the tree (n is the number of nodes on the current tree). Sub-tree information of $T_{x,y}$ is computed using sub-tree information of $T_{y,z}$ where z stands for each neighboring node of y. This recursive definition of sub-tree information is incorporated in the above message exchange. For more details readers may refer to Ref. [4].

3.4 Join Procedure

In MODE-m, the basic policy to accept a new node is to connect the new node to a existing tree's node with more than one residual degree, which places the new node closest to the center node. This basic approach provides quite reasonable diameters in an incremental way. However, as we stated in Section 2, this approach fails to make trees with reasonable diameters when mobile nodes' participation takes place. So we move the mobile nodes so that they become leaf nodes in the following manner. Every non-mobile node replaces one of its mobile node neighbors (say m) with the newly joining non-mobile node by forcibly disconnecting m, only if that makes the new node closest to the center node. Here the node m is selected randomly, and m is subjected to *rejoin* the tree in the same way as a newly joining mobile node. Note that newly joining mobile nodes cannot replace them with the existing mobile nodes. The detailed join procedure is as follows.

A new node which wants to join the current tree first sends a query message to the root node [2] to ask the address of the center node. In MODE-m, any node can calculate the center node of the tree using the sub-tree information described in the previous sub-section. So the root node can send the center node's address to the new node. In case of when the

[2]This is because we assume that a new node only knows the root node as the well-known node (see assumption G2). To avoid access concentration, several well-known nodes may be assumed rather than a single node.

root node does not know the center node (*i.e.* before the first collection phase has been completed) or the center node has already disappeared, the root node sends the address of itself instead.

Once the joining node receives the reply from the root node, it sends a connection request message to the center node (Fig.4 (a)). The center node (say c), which receives the connection request message, sends a connection permission message to the joining node if at least one of the following two conditions, (i) c has more than one residual degree or (ii) the joining node is a non-mobile node and c has at least one mobile node as a neighbor, is satisfied. At the same time, it broadcasts the connection request message to its neighbors[3]. The neighbors also treat the connection request message in the same way. Note that these nodes send no message to the joining node if they satisfy neither of the conditions (i) and (ii) stated earlier. Here the delay from the center node is also added to the connection request message before it is broadcast to the neighboring nodes. So every node which sends a connection permission message to the joining node also includes the delay from the center node to the node itself. Then the joining node uses these values and the delay to each node which has sent a connection permission message (this can be measured using *ping* for instance) to select the node (say p) which can place this node closest to the center node. Then it establishes an overlay link with the node p. If this overlay link violates the degree constraint at p (Fig.4(b)), that is p has accepted the new node according to the condition (ii), then the node p forces one of its neighboring mobile nodes to rejoin the tree (Fig.4(b)). As a result of these rejoin processes, the mobile nodes shift themselves from positions close to the center node to leaves of the tree.

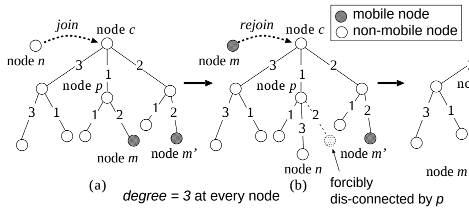


Figure 4: Mobile node rejoin in join procedure

3.5 Refresh Procedure

Refresh procedure is applied only to mobile nodes, to prevent the diameter from being affected by mobile nodes' delay fluctuation. Mobile nodes become leaf nodes according to the above join procedure and that helps to construct a spanning tree with a shorter diameter. However, the diameter path's both ends may be occupied with mobile nodes in many cases. Therefore, the diameter becomes sensitive to the mobile nodes' delay fluctuation. To avoid this phenomenon we make mobile nodes rejoin the tree if its wireless characteristics result in a longer diameter. We name this process *refresh*.

[3]Here we can set a suitable forwarding count limit to prevent the joining node from receiving a huge number of connection permission messages.

Note that mobile nodes can detect the change of the delay to its neighbor according to the assumption G1.

A mobile node m performs a refresh procedure if the following statement is true. Here n denotes the neighboring node of m.

$$c(m,n) + depth(T_{m,n}) > current\ tree\ diameter + \alpha$$

Here, $c(m,n)$ denotes the delay of the link from m to n and $depth(T_{m,n})$ denotes the longest depth of the subtree $T_{m,n}$. Here α regulates the refresh frequency. In other words the refresh procedure does not take place if the new diameter exceeds the previous one by not more than α. Without this the mobile nodes may oscillate in the tree. Fig.5 illustrates this refresh strategy.

Fig.5(a) shows a case that a mobile node(m) finds that the above condition is true after detecting the change of the delay to its peer ($c(m,n)$). Then m rejoins the tree as shown in the Fig.5(b) to prevent the diameter from increasing. Note that m keeps checking the above condition whenever it detects a new delay, despite its previous refresh tasks. To avoid infinite refresh tasks, we have set the mobile nodes not to perform a second refresh procedure, if it got connected to the same peer after rejoining the tree.

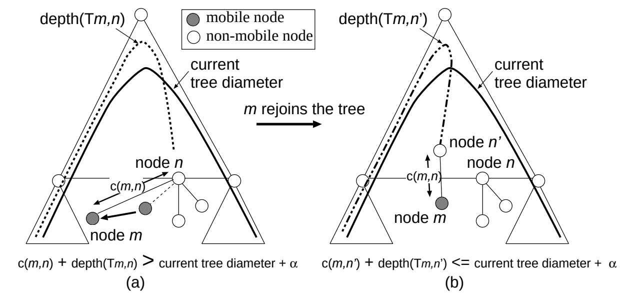


Figure 5: Mobile node refresh strategy

3.6 Repair Procedure

For simplicity of discussion, we say that node u is said to be the parent of node v if u is a neighbor of v and the root node resides in the direction of u, otherwise u is said to be a child of v. Let v denote a node which has disappeared, u denote v's parent and w_j denote v's j-th child ($1 \leq j \leq d(v) - 1$) where $d(v)$ is the current degree of node v. Also, let $R(v)$ denote the repair procedure for node v's disappearance. For simplicity we explain $R(v)$ without considering other disappearances.

We illustrate the behavior of the repair procedure for a single (non-leaf) disappearance in Fig.6. For node v's disappearance, its parent (node u, this is called the *repair initiator*) activates the repair procedure and sends the information of T_{v,w_j} ($1 \leq j \leq d(v) - 1$) to the center node of $T_{v,u}$ (this is called *repair master*). This step is called phase 1 (see Fig.6(a)). The repair master then sends its address to each w_j ($1 \leq j \leq d(v) - 1$) (called *repair sub-initiators*) (phase 2). Then each repair sub-initiator w_j sends the address of repair master to the center node of sub-tree T_{v,w_j} (called *repair sub-master*) (phase 3). Now each repair sub-master of T_{v,w_j} ($1 \leq j \leq d(v) - 1$) knows the address of repair master, so it

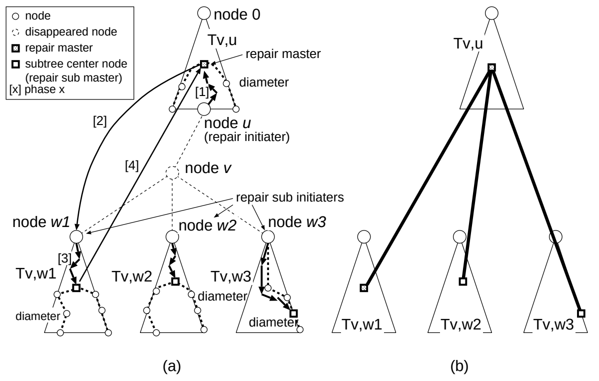


Figure 6: Repair procedure (single disappearance)

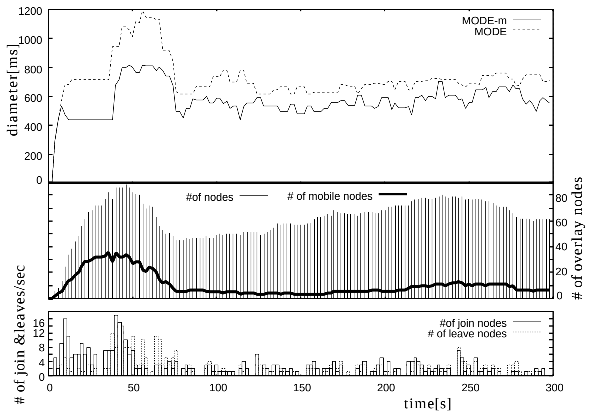


Figure 7: Dynamics of diameters

connects itself to the repair master (phase 4). Note that if the repair sub-master has no residual degree, its closest node with at least one residual degree is selected for this task. Once this repair process is done, the repair master and the repair sub-master exchange their neighboring nodes over the new overlay link. By this procedure, nodes near the "center" of the sub-trees are connected, and the diameter of the repaired tree is expected to be equal or smaller than before.

Also, MODE-m can repair the tree in a simple, fast and decentralized way when multiple nodes' disappearances occur. The repair procedure for multiple disappearances is described in Ref. [4].

4 Performance Evaluation

4.1 Simulation Setup

We have implemented our MODE-m protocol on ns-2 to evaluate the enhancement against MODE. In our experiments, networks with about 400 physical nodes have been generated and used as underlying networks. We have selected 200 nodes, including both mobile and non-mobile nodes, as overlay participant nodes. Here we define the mobile node occupancy (as a percentage of the entire nodes) as an evaluation parameter m. In the simulation, we have changed the end-to-end delay (overlay link delay) for non-mobile nodes from 50ms to 300ms during the simulation. For the overlay links between non-mobile nodes, the initial delay has been set to vary from 10ms to 200ms, and these initial values were set not to change during the simulation.

Considering practical situations, we have prepared the following scenario that simulates a real-time session in collaborative applications such as video-based meetings or groupwares. Note that we have set the interval between collection phases to 30 seconds. The degree bound on each non-mobile node was set to a randomly selected constant value, which ranges from 3 to 7, while that of each mobile node was set to 1. The scenario is as follows. (i) The session period is 300 seconds. (ii) Each of 200 nodes joins the session only once and eventually leaves the session. (iii) Within the first 30 seconds, about 60 nodes join the session. (iv) From 30 seconds to 250 seconds, additional joins are processed. Also some existing nodes leave the session. The collection phases take place at 30, 60, 90, 120, 150, 180, 210, 240, 270 and 300 seconds successively. (v) Mobile nodes are set to refresh their delay in each 5 seconds. And the value α (see Section 3.5) is set to 50[ms]. (vi) After 250 seconds, no join takes place and about 40 nodes leave the session.

We have evaluated MODE-m following the above scenario setting m (ratio of mobile nodes to all the nodes) to 5%, 10%, 20%, 30% and 40%. Here, we have limited the upper margin of m to 40% considering the fact that construction of DB-MDT might be impossible with a larger percentage of mobile nodes where $d_{max} = 1$ [4]. In Fig.7 and Fig.8, the number of nodes on the tree at every second together with the numbers of join/leave operations are shown, to make it facilitate to see the dynamics of the metrics according to the scenario.

4.2 Experimental Results

[Diameter Against the # of Nodes] We have measured the diameters of the trees at every one second for MODE-m and MODE for various m's. The result for $m = 30\%$ is shown in Fig.7.

Obviously MODE-m could archive a shorter diameter. Especially in the period with higher mobile node occupancy (within 0[ms] to 75[ms] in Fig.7), MODE's performance has become poorer. Here the diameter difference of MODE and MODE-m in the time period of 0[ms] to 30[ms] in Fig.7, shows the validity of the improvement done to the join procedure, while the rest part of the session shows the performance gained by the mobile node specific refresh procedure. Considering both results we can say that MODE-m well supports mobile nodes.

[Average Diameter and Rejoin-cost] We have measured the average diameters (in milli seconds) for MODE-m and MODE after performing simulations for ten different sessions, where each follows the above scenario. And also we calculated the *rejoin-cost* for the MODE-m in each session and calculated the average value for the ten sessions. *Rejoin-cost* is the sum of the number of overlay links which are established and disconnected due to the rejoin processes during the join and refresh procedures over the session. Note that the rejoin-

[4]Readers may note that constructing of DBMDT will become possible with the existence of powerful mobile nodes (nodes where $d_{max} > 1$).

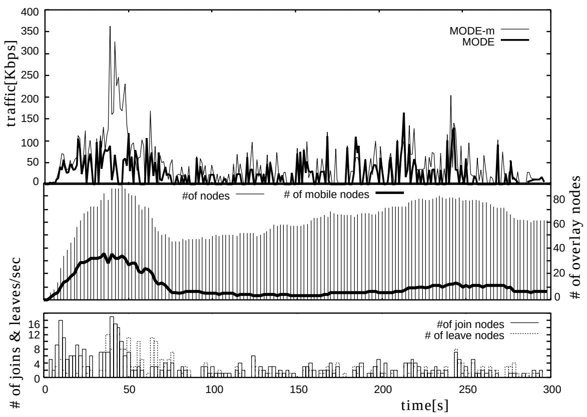


Figure 8: Control traffic

cost becomes two times of the number of rejoin processes applied. The results for different values of m's are shown in Table. 1.

m		5%	10%	20%	30%	40%
MODE -m	diameter	467	513	565	585	646
	rejoin-cost	32	64	162	197	222
MODE	diameter	494	569	633	736	801

Table 1: Comparison of diameter [ms] and rejoin-cost [number of attached/detached links] of mobile nodes

The result shows that the performance of MODE-m increases with the growth of the mobile node occupancy. At the same time we can note that the rejoin-cost also keeps increasing with the mobile node occupancy.

The rejoin-cost discussed here reflects the disconnecting frequency of mobile nodes. Considering that the mobile nodes always resides as leaf nodes, we can understand the fact that non-mobile nodes remain unharmed in this regard. This implies that the session remains consistent for the rest of nodes, even if the rejoin-cost is considerably large. However, we can say the rejoin-cost is reasonable enough considering the diameter reduction gained compared to MODE, by these rejoin processes. Furthermore, the rejoin-cost can be reduced by increasing the value of α (see Section 3.5), though this might increase the diameter, which is in a trade-off with the rejoin-cost.

[Traffic] We have measured the control traffic flows on the entire tree for the MODE-m and MODE. The result is shown in Fig.8. The increased traffic amount in MODE-m is mainly the traffic due to rejoin processes. And the highest traffic amount has been generated around 30[ms] as the number of mobile nodes has reached to top. Taking this peak value, 350[Kbps], together with the number of nodes in the session at that time (90 nodes approximately), the average traffic amount on a single node can be calculated as 4[Kbps]. We can say this value is small enough for real world applications.

[Time Required for Procedure Execution] Finally, we have measured *join-time* and *refresh-time*, the time required to execute join and refresh procedures successively. Note that here refresh procedure does not count for MODE, and the comparison of the time required to repair is omitted here, since repair procedure is common to MODE and MODE-m. Here, the *join-time* of a node n includes n's join time plus the time needed for the mobile node, which have been forcibly disconnected due to the join procedure of n, to rejoin the tree. The expected average values for join-time and refresh-time when $m = 30$ are shown in Table. 2. According to those results both the join-time and the refresh-time hold values less than 1 second, which can be considered small enough for real applications. Note that the values in Table. 2 remain almost same for different values of m as well, according to the nature of the protocol.

	join-time[s]	refresh-time[s]
MODE-m	0.93	0.79
MODE	0.71	-

Table 2: Comparison of the time required for join and refresh

5 Concluding Remarks

In this paper, we have proposed a new overlay multicast protocol called MODE-m that constructs a degree-bounded minimum diameter tree, supporting mobile hosts. MODE-m is considerably enhanced from MODE so that it can achieve reasonable diameter under the existence of heterogeneous hosts. The experimental results have shown the advantage of MODE-m compared with MODE.

REFERENCES

[1] Y. H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proc. of ACM SIGMETRICS*, 2000. Tools are provided: `http://www-2.cs.cmu.edu/~streaming/index.html`.

[2] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proc. of 3rd Usenix Symp. on Internet Technologies and Systems*, 2001.

[3] P. Francis. Yoid: Extending the internet multicast architecture. `http://www.isi.edu/div7/yoid/`, 2002.

[4] H. Yamaguchi, A. Hiromori, T. Higashino, and K. Taniguchi. An autonomous and decentralized protocol for delay sensitive overlay multicast tree. In *Proc. of IEEE ICDCS*, pages 662–669, 2004.

[5] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proc. of IEEE Infocom*, 2003.

[6] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. of ACM SIGCOMM 2002*, pages 205–217, 2002.