New Distributed Multicast Routing and Its Performance Evaluation

Takuya Asaka[1][2][3], Takumi Miyoshi[2][3], and Yoshiaki Tanaka[2][3]

[1] NTT Service Integration Laboratories
9-11, Midori-Cho 3-Chome, Musashino-Shi, Tokyo, 180-8585 Japan.
[2] Global Information and Telecommunication Institute, Waseda University
3-10, Nishi-Waseda 1-Chome, Shinjuku-Ku, Tokyo, 169-0051 Japan.
[3] Okinawa Research Center
Telecommunications Advancement Organization of Japan
21-1, Nishi-Waseda 1-Chome, Shinjuku-Ku, Tokyo, 169-0051 Japan.

Abstract. With conventional dynamic routing algorithms, many query messages are required in a distributed environment for efficient multicast routing of any traffic volume. We have developed a dynamic routing algorithm that uses a predetermined path search in which an appropriate multicast path is dynamically constructed by searching only a few nodes. This algorithm can construct an efficient multicast tree for any traffic volume. Simulation has shown that the proposed algorithm is advantageous compared with conventional dynamic routing algorithms when nodes are added to or removed from the multicast group during steady-state simulation.

1 Introduction

Future computer-network applications such as teleconferencing or remote collaboration will rely on the ability of networks to provide multicast services. Multicasting is expected to become widely used [1,2,3], and is well suited to these services because it uses network resources efficiently. In multicasting, a point-to-multipoint (multicast) connection is used to copy packets only at branching nodes, which ensures network efficiency. Naturally, the smallest possible amount of network resources should be used to set up the multicast connection.

Multicast routing problems are either *static* or *dynamic*. In *static* routing problems, the members of the multicast group remain unchanged during the lifetime of the multicast connection. In *dynamic* routing problems, members can join or leave the group during the lifetime of the connection. Dynamic multicast routing is important for actual multicast applications and is supported by protocols in ATM network [3] and Internet protocols [4,5,6].

Here, we focus on a dynamic multicast routing algorithm that can satisfy the following requirements.

(1) *Minimized average multicast tree cost.* Every link has a cost (or a metric), and the tree cost is the sum of the costs for all links included in the multicast tree. Minimizing the tree cost ensures efficient use of bandwidth. This is called the dynamic Steiner problem.

(2) *Scalability in a distributed environment.* A central server should not be used to control joining nodes because a large-scale network could overload the server, thus degrading performance. Moreover, minimizing the overhead of nodes is important from the viewpoint of algorithm scalability even if there is no central server does.

(3) *Robustness against the number of joining nodes.* The number of joining nodes strongly affects the performance of conventional algorithms. The performance should be independent of the number of joining nodes.

(4) *Minimized worst-case cost of the multicast tree.* The worst-case cost produced by the algorithm should be theoretically bounded as small as possible.

Many dynamic multicast routing algorithms have been proposed [4,5,6,7,8,9] [10,11,12,13,14]. However, none of there algorithms can satisfy all of the above requirements. The greedy algorithm [8,9] selects the shortest path to an existing multicast tree when a node is added. It can construct a near-optimal multicast tree, but requires many query/reply messages between nodes when implemented in a distributed environment [13,14]. Thus , the algorithm is not scalable in a distributed environment. The pruned shortest-path tree algorithm [4,5,6,7] finds the shortest path from the source node (or the center node) to the nodes in the multicast group when a node is added to the multicast tree. This algorithm cannot construct an appropriate multicast tree, though, from the viewpoint of tree cost. The virtual trunk dynamic multicast (VTDM) routing algorithm [10] constructs multicast trees based on the virtual trunk, which is the tree of the underlying graph. However, the "trunk number" for constructing the virtual trunk must be determined according to the estimated number of nodes that will join and it is not flexible.

We have developed a dynamic routing algorithm that can satisfy all four of the about requirements. It uses a predetermined path search where only a few nodes are searched to determine to which existing node a newly added node should be connected. In this algorithm, the searched nodes are on predetermined paths: the paths from new added nodes to the source node on the minimum-spanning tree and the shortest-path tree. The node and network overheads are small because only a few nodes are searched. The performance is similar to the greedy algorithm but does not depend on the number of joining nodes. Our simulation has shown that our algorithm is advantageous when nodes are added to or removed from the multicast group in the steady state. We discuss the competitive ratio of our algorithm, and show its advantage over the greedy algorithm.

2 Problem Definition

To define the dynamic multicast routing problem formally from the viewpoint of minimizing multicast tree cost, we use the terminology of graph theory for the models. In the model used for the conventional dynamic multicast routing problem, the network is modeled as a graph whose edges have costs. If nodes can be added or removed during the lifetime of the multicast connection, this problem becomes the dynamic multicast routing problem, i.e., the dynamic Steiner tree problem. Let $R = \{r_1, r_2, ..., r_K\}$ be a sequence of requests, where r_i is either adding or removing a destination node to or from the multicast group. Let S_i be the set of nodes in the multicast group after request r_i has been made. In response to request r_i, multicast tree T_i is constructed using a dynamic multicast routing algorithm. The dynamic multicast routing problem can thus be formally defined as follows.

> Given graph $G = (V, E)$, a nonnegative weight for each $e \in E$, and $Z \subseteq V$, and a sequence R of requests, find a sequence of multicast trees $\{T_1, T_2, ..., T_K\}$ in which T_i spans Z_i and has minimum cost.

The dynamic multicast routing problem considered in this paper does not allow re-routing of existing connections when additional requests are received. One node is the source for a multicast communication, and this node cannot be removed from the multicast group during the life of the multicast connection.

In this paper, vertices that are included in a multicast tree are called existing nodes, and the source node is an existing node. Vertices that do not participate in a multicast group but that are included in the multicast tree are called intermediate nodes. Moreover, vertices that are neither an existing nor an intermediate node are called non-existing nodes.

3 Conventional Algorithms

Several dynamic multicast routing algorithms have been reported [8,9,10,11,12]. As mentioned earlier, the greedy algorithm [8,9] selects the shortest path to an existing multicast tree when adding a node. The shortest path between a pair of nodes can be calculated using Dijkstra's algorithm before routing. In the greedy algorithm, selecting a new path is the best way to add a node. However, when greedy algorithm is implemented in a distributed environment, a newly addednode must flood the entire network with query messages [13]. That is, a newly added node sends a query message to all its neighbors on the shortest-path tree rooted at the node, and this continues until the query message reaches either an existing node or a leaf node of the shortest-path tree. Consequently, there is a large processing overhead for the query message in each node. The network overhead becomes particularly large when there are many non-existing nodes. As another approach [15], a manager router is introduced to control the addition and removal of nodes. However, the processing overhead of the manager router is high when there are many nodes in the network, and a failure of the manager router would cause a fatal error in the construction of the multicast tree.

As mentioned above, the pruned shortest-path tree algorithm (pruned SPT) [4,5,6,7] finds the shortest path from a source node [7] (a "core" in CBT [4] or a “rendezvous point” in PIM-SM [5]) to the nodes in the multicast group when a node is added to the multicast tree. The pruned SPT algorithm has been implemented as an actual routing protocol [4,5,6] because it is easier to implement than the greedy approach. The shortes t-path tree is spanned from a source node (or a core or rendezvous point) to every node in the multicast group. Moreover, the multicast tree is obtained by deleting nonessential edges. The cost of a multicast tree made using this approach is higher [16] than for one made using the greedy approach.

The virtual trunk dynamic multicast (VTDM) routing algorithm [10] constructs multicast trees based on the virtual trunk, which is the tree of the underlying graph. The virtual trunk and the multicast tree that uses it are constructed as follows:

Step 1: Find the shortest paths for all pairs of nodes. Count the number of shortest paths passing through each node.
Step 2: Define a set F as vertices having the top $\lceil \theta N \rceil$ largest numbers; these vertices are called virtual-trunk nodes.
Step 3: Construct a complete graph for F by replacing the shortest paths between pairs of vertices in G with distances between pairs of vertices. Find the minimum-spanning tree for the complete graph.
Step 4: Convert edges in the minimum-spanning tree back to the corresponding shortest paths in the graph G. Run the minimum-spanning tree algorithm and remove any unnecessary nodes or links. The obtained tree is called the virtual trunk.
Step 5: Connect nodes not in the virtual trunk to the nearest node in the virtual trunk.

N is the number of nodes in graph G, and parameter θ ($0 \leq \theta \leq 1$) determines the number of trunk nodes. When $\theta = 0$, the VTDM algorithm is the same as the pruned SPT because only the source node is selected as a trunk node. Similarly, when $\theta = 1$, the VTDM algorithm is the same as the pruned minimum-spanning tree (pruned MST), where the MST is used instead of the SPT as in the pruned SPT. In this algorithm, θ must be determined according to the estimated number of nodes that will be join. However, actual traffic estimation is difficult, and setting θ is a complicated task for network management.

The other conventional dynamic multicast routing algorithms take different approaches. For example, some allow re-routing of multicast connections [11,12,17].

4 Dynamic Multicast Routing Algorithm Using Predetermined Path Search

Our algorithm is based on the greedy algorithm and is intended to have the same performance. It uses *query* and *reply* messages to obtain membership information, as does the greedy algorithm. Thus, this approach does not require a manager server for multicast routing. Furthermore, our dynamic multicast algorithm searches only a few nodes to determine which existing node should be connected. A newly added node connects to this existing node using the shortest path. The node and network overheads are small because the number of searched nodes is restricted. An appropriate path is selected based on the current multicast tree. Thus, our algorithm is suitable for a distributed environment.

In this algorithm, two kinds of spanning trees are prepared for the predetermined paths: a minimum-spanning tree and a shortest-path tree having a source node as a root. Figure 1 shows examples of the MST and SPT. The minimum-spanning tree can be calculated using either Kruskal's or Prim's algorithm with computational complexity $O(m \log n)$, and the shortest-path tree can be calculated using Dijkstra's algorithm with computational complexity $O(m + n \log n)$. Our algorithm works as follows.

Step 1: A newly added node sends query messages to nodes on two predetermined paths toward the source.
Step 2: If a queried node is an existing node, it sends a reply message to the new added node and the query message is not transmitted to the next node. Otherwise, the query message is transmitted to the next node on the predetermined path.
Step 3: After the added node receives reply messages from existing nodes on each predetermined path, it connects to the closest node among the nodes that sent reply messages.

An example of how this procedure works is shown in Fig. 2. The newly added node #6 sends query messages to nodes on two predetermined paths toward the source. First, we describe the case of using the MST as a predetermined path. In Fig. 2, only node #1 is an existing node. The predetermined path sfor a newly added node is $\langle \#6 \rightarrow \#4 \rightarrow \#2 \rightarrow \#1 \rightarrow \text{source} \rangle$. When a new node requests to join, a query message is transmitted to node #6. Since node #6 is not an existing node, the query message is retransmitted to the next node. This retransmission continues until node #1 receives the query message. Node #1 is an existing node, so it sends a reply message to the newly added node. Next, we describe

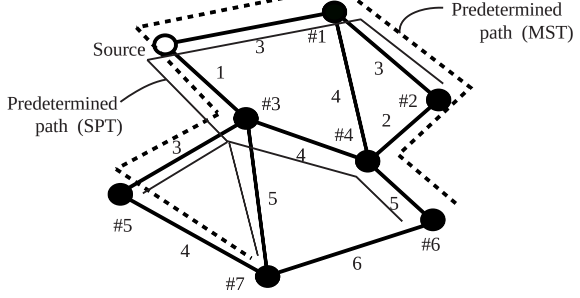

Fig. 1. Example of MST and SPT.

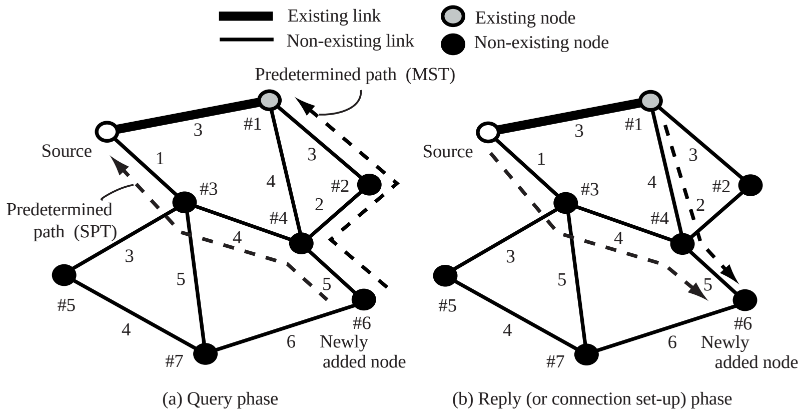

Fig. 2. Dynamic multicast routing algorithm using predetermined path search.

the case of using the SPT as a predetermined path. The predetermined path for a newly added node is ⟨#6 → #4 → #3 → source⟩. The algorithm works similar to the case of MST and the newly added node receives a reply message from the source. Thus, the newly added node #6 receives two reply messages from node #1 and from the source node. Node #1 is closer to the newly added node than the source, so the newly added node is connected to node #1 using the shortest path between them.

In this algorithm, the added node should be connected to the source node using the path that is nearly the shortest path when there are few existing nodes. On the other hand, when there are many existing nodes, the added node should be connected to the existing node using nearly the minimum spanning.

Moreover, the node and network overheads are lower than with the greedy algorithm. This is because only nodes on predetermined paths are searched and only these nodes receive query messages. In the example in Fig. 2, the proposed algorithm needs six query messages. These messages are ⟨#6 → #4⟩, ⟨#4 → #2⟩, ⟨#2 → #1⟩, ⟨#6 → #4⟩, ⟨#4 → #3⟩ and ⟨#3 → source⟩. The greedy algorithm, on the other hands, needs seven. These are ⟨#6 → #4⟩, ⟨#4 → #2⟩, ⟨#2 → #1⟩, ⟨#4 → #3⟩, ⟨#3 → source⟩, ⟨#6 → #7⟩, ⟨#7 → #5⟩. Similary, if node #4 instead of node #6 is newly added to the multicast group, the proposed algorithm needs four and the greedy algorithm needs seven. As the network size increases, the advantage of the proposed algorithm increases because the greedy algorithm broadcasts query messages.

Another version of this algorithm uses only the MST as the predetermined path, rather than both the MST and SPT. This does not require reply messages from the existing nodes because a connection set-up message can be used instead. Since the newly added node need not determine which existing node is nearer. However, the tree cost may be higher than in the previous case. There is a trade-off relationship between processing overhead and cost performance.

Table 1. Parameters used for simulations.

Parameter	Value	Parameter	Value
N	50	$\bar{e}$	3
α	0.25	k	25
β	0.20	μ	0.9
γ	0.20		

5 Simulation Model

We evaluated the performance of our algorithm through simulation, using the same model and parameters that were used by Lin and Lai [10]. A network is modeled as a random graph possessing some of the characteristics of an actual network [7]. The vertices representing nodes are randomly distributed on a rectangular coordinate grid, and each vertex has integer coordinates. For a pair of vertices, say u $(0 \leq u \leq 1)$ and v $(0 \leq v \leq 1)$, an edge is added according to the following probability:

$$P_e(u,v) = \frac{k\bar{e}}{N}\beta \exp\frac{-d(u,v)}{L\alpha}, \tag{1}$$

where N is the number of vertices in the graph, $\bar{e}$ is the mean number of degrees of a vertex, k is a scale factor related to the mean distance between two vertices, $d(u,v)$ is the Euclidean distance between vertices u and v, L is the maximum distance between any two vertices in the graph, and α and β are parameters (real numbers between 0 and 1). The edge density is increased by increasing the value of β. The edge density of shorter edges relative to that of longer one s is decreased by decreasing the value of α. Setting the values of α and β to 0.25 and 0.20, respectively, generates a graph that roughly resembles a geographical map of the major nodes in the Internet [7]. Once the vertices and edges have been generated, we can be sure that the graph is composed of only one component.

In the simulations, requests to add or remove a node to/from the multicast group are periodically generated. We used a probability model to generate the sequence of requests, i.e., to determine whether each request was to add or remove. The probability for adding a node was determined by

$$P_c(q) = \frac{\gamma(N-q)}{\gamma(N-q)+(1-\gamma)q}, \tag{2}$$

where q is the current number of nodes in the multicast group and γ $(0 \leq \gamma \leq 1)$ is a parameter (a real number). That determines the size of the multicast group in equilibrium. Each node had a different rate for joining the multicast, i.e., a different probability that the ith node would be selected to join. We defined A as a set of non-joining nodes and the joining rate of the ith node was given by

$$P_{join}(i) = \begin{cases} (1-\mu)\mu^{i-1}/F_0, & \text{for} \quad i \in A \\ 0, & \text{for} \quad i \notin A \end{cases} \tag{3}$$

where $F_0 = \sum_{i \notin A}(1-\mu)\mu^{i-1}$ and a parameter μ $(0 \leq \mu \leq 1)$ determined the bias of the joining rate. In simulations, if a joining event is generated with $P_c(q)$, a node that will join is determined with $P_{join}(i)$. If a removal event is generated with $P_c(q)$, a node that will be removed is randomly determined.

We compared our algorithm with four conventional algorithms: the greedy algorithm, the pruned SPT, the VTDM algorithm, and the pruned MST. For the VTDM algorithm, parameter θ was set to 0, 0.2, 0.4, or 1. The VTDM algorithm with $\theta = 0$ corresponded to the pruned SPT, and with $\theta = 1$ it corresponded to the pruned MST.

In the simulations, the average multicast tree cost was used as a measure of performance. We generated ten different networks and calculated the average tree cost. Each multicast connection consisted of a sequence of 20,000 requests to add or remove nodes. The costs for the first 2,000 requests were not used in calculating the average costs to eliminate the effect of the initial conditions in the simulation. Table 1 shows the default values of the simulation parameters.

6 Simulation Results

Basic cases

The relationship between the tree cost and the average multicast group size is shown in Fig. 3 for 20 and 50 nodes, where the multicast tree costs are normalized by the costs with the greedy algorithm. In both cases, the cost with the proposed algorithm was close to that with the greedy algorithm. The reason is as follows. In the proposed algorithm, the selected node on the tree and the newly added node are connected using the shortest path. Thus, the proposed algorithm works like the greedy algorithm. Moreover, the proposed algorithm can find an appropriate existing node on predetermined paths. When the multicast group is large, an existing node on the MST is likely to be selected as the node to be connected for the newly added node. On the other hand, when the multicast group is small, an existing node on the SPT is likely to be selected as the node to be connected for the newly added node. Thus, the number of wasteful searches is small.

Furthermore, the proposed algorithm is slightly superior to the greedy algorithm when the group size is small. In these cases, the greedy algorithm may cause the construction of roundabout routes since many nodes are removed. The SPT as the predetermined path can prevent the construction of these routes because nodes nearby the source are selected as nodes to be connected.

The other algorithms, however, could not match the performance achieved with the greedy algorithm for any multicast group size. These algorithms were especially poor for extremely small group sizes, and require that the algorithm or parameters be selected according to the group size. In comparison, our algorithm worked well for any multicast group size without requiring a parameter for control, and was not greatly influenced by the number of nodes.

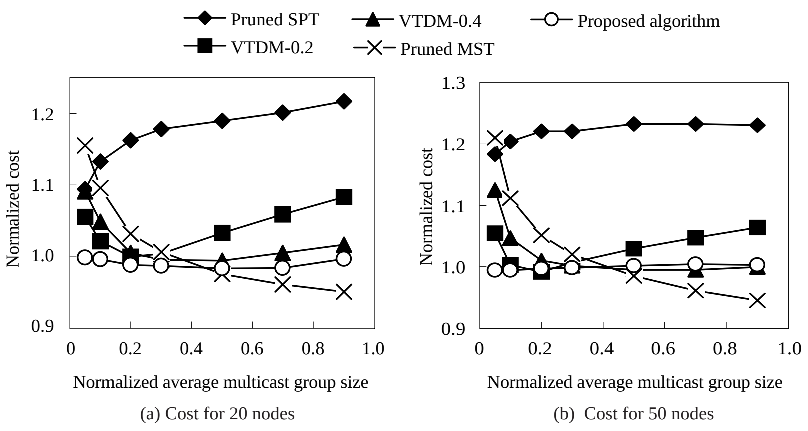

Fig. 3. Cost of average multicast tree with the conventional and proposed algorithms.

Traffic bias

The relationship between the tree cost and the average multicast group size is shown in Fig. 4 when the joining rate of each node was varied. The traffic bias was small when parameter μ for the joining rate was large, and each node had the same rate when $\mu = 1$.

The tree cost of all the algorithms became worse than that with the greedy algorithm, but the proposed algorithm was slightly more robust for traffic bias than the other algorithms. This is because the proposed algorithm - just like the greedy algorithm - can construct multicast trees depending on existing nodes.

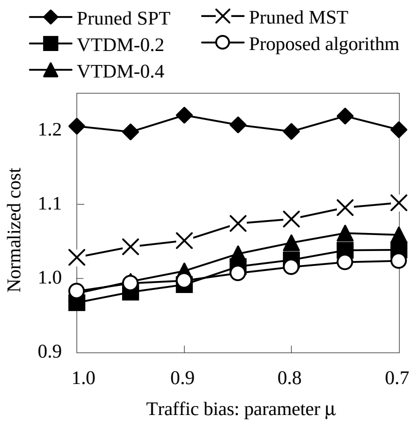

Fig. 4. Comparison of average multicast tree cost at different traffic biases.

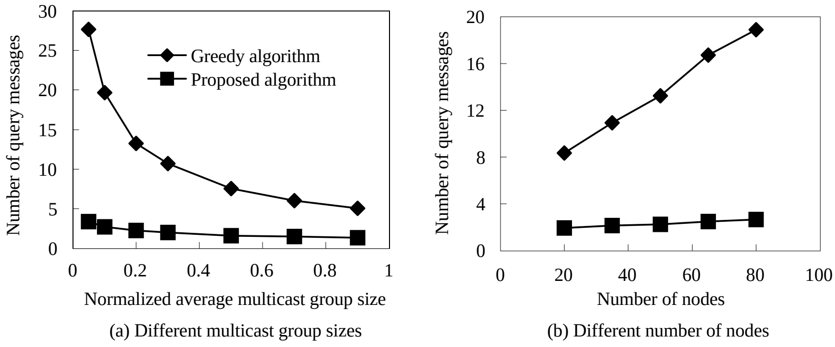

Fig. 5. Number of query messages with the proposed and greedy algorithms.

Number of query messages

The number of query messages with the proposed and greedy algorithms [13] is compared in Fig. 5. In this figure, the number of query messages is the total number of query messages received by all nodes when a node is newly added to the multicast group.

When the multicast group size is small, more messages were needed with the greedy algorithm (Fig. 5(a)). This is because the greedy algorithm requires query message transmission until the query message reaches either an existing node or a leaf node of the shortest-path tree. When the number of nodes was increased, the number of query messages increased with both algorithms (Fig. 5(b)). However, the increase was smaller with the proposed algorithm than with the greedy algorithm. Thus, the loads offered to nodes and the network should always be smaller with the proposed algorithm than with the greedy algorithm.

7 Competitiveness Analysis

Here, we discuss the *competitive ratio* of the proposed algorithm, defined as the maximum ratio of the cost of the algorithm over the optimal one. The competitive ratios of conventional algorithms and the proposed algorithm are shown in Table 2. In the static case, the members of the multicast group remain unchanged during the lifetime of the multicast connection. In the join-only case, the members of the multicast group do not quit a session. In the join-remove case, the members of the multicast group join and quit a session during the lifetime of the multicast connection. We denote the number of participant nodes as M in the static case, and the maximum number of participant nodes in the lifetime of a multicast connection in the join-only and join-remove cases.

The competitive ratios of conventional algorithms are described in [18,19]. The competitive ratio of the greedy algorithm had not been derived in the join-remove case, but the lower bound of the competitive ratio had been derived. For the proposed algorithm, the competitive ratio is $\max(N-M, M)$ since the worst case is always bounded by either the pruned MST or the pruned SPT.

Table 2. Competitive ratios.

	Static	Join-only	Join-remove
Pruned MST	$N-M$	$N-M$	$N-M$
Pruned SPT	M	M	M
Greedy algorithm	$2-2/M$	log$M$	2^M $(*)$
Proposed algorithm	$\max(N-M, M)$	$\max(N-M, M)$	$\max(N-M, M)$

$(*)$lower bound

Table 2 shows that the greedy algorithm has an advantage over the other algorithms in the static and join-only cases. On the other hand, the greedy algorithm fails dramatically in the join-remove problem, while the other algorithms do not become any worse than in the static and join-only cases. Although our algorithm does not have an advantage over the pruned MST and the pruned SPT in all cases, it has a large advantage over the greedy algorithm.

8 Conclusion

We proposed a dynamic routing algorithm that uses a predetermined path search, in which an appropriate multicast path is dynamically constructed by searching only a few nodes. Simulation showed that the performance of this is close to that of the greedy algorithm and that it is superior when nodes are added to or removed from the multicast group during steady-state simulation. The node overhead is lower than with the greedy algorithm since it requires only a few query messages, so our algorithm is suitable for a distributed environment. Moreover, it can construct an efficient multicast tree that is independent of the multicast group size. We also showed that the competitive ratio of our algorithm is superior to that of the greedy algorithm.

Some research problems remain concerning multicasting using this method. One is the performance of multicasting under diverse practical traffic patterns and network topologies. The development of a multicast routing protocol scheme also deserves attention.

References

1. The IP Multicast Initiative, The IP Multicast Initiative Home page, http://www.ipmulticast.com/, Dec. 1998.
2. The MBone Information Web, Home Page, http://www.mbone.com/, Dec. 1998.
3. C. Diot, W. Dabbous and J. Crowcroft, "Multipoint Communication: a Survey of Protocols, Functions, and Mechanisms," IEEE JSAC., Vol. 15, No. 3, pp. 277–290, April 1997.
4. A. Ballardie, "Core Based Trees (CBT Version 2) Multicast Routing – Protocol Specification –," RFC2189, Sept. 1997.

5. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma and L. Wei, "Protocol Independent Multicast-sparse Mode (PIM-SM): Protocol Specification," RFC2362, June 1998.
6. J. Moy, "Multicast Extensions to OSPF," RFC 1584, March 1994.
7. M. Doar and I. Leslie, "How Bad is Naïve Multicast Routing?" IEEE INFOCOM'93, pp. 82–89, March 1993.
8. B. M. Waxman, "Routing of Multipoint Connections," IEEE JSAC., Vol. 6, No. 9, pp. 1617–1622, Dec. 1988.
9. B. M. Waxman, "Performance Evaluation of Multipoint Routing Algorithms," IEEE INFOCOM'93, pp. 980–986, March 1993.
10. H. Lin and S. Lai, "VTDM - a Dynamic Multicast Routing Algorithm," IEEE INFOCOM'98, pp. 1426-1432, March–April 1998.
11. J. Kadirire, "Minimizing Packet Copies in Multicast Routing by Exploiting Geographic Spread," ACM SIGCOMM Computer Communication Review, Vol. 24, pp. 47–63, 1994.
12. J. Kadirire, "Comparison of Dynamic Multicast Routing Algorithms for Wide-area Packet Switched (Asynchronous Transfer Mode) Networks," IEEE INFOCOM'95, pp. 212–219, March 1995.
13. R. Venkateswaran, C. S. Raghavendra, X. Chen and V. P. Kumar, "DMRP: a Distributed Multicast Routing Protocol for ATM Networks," ATM Workshop'97, May 1997.
14. K. Carlberg and J. Crowcroft, "Building Shared Trees using a One-to-many Joining Mechanism," ACM Computer Communication Review, Vol. 27, No. 1, pp. 5–11, Jan. 1997.
15. M. Faloutsos, A. Banerjea, and R. Pankaj, "QoSMIC: Quality of Service Sensitive Multicast Internet Protocol," ACM SIGCOMM '98, Sep. 1998.
16. L. Wei and D. Estrin, "The Trade-offs of Multicast Trees and Algorithms," ICCN'94, Sept. 1994.
17. F. Bauer and A. Varma, "ARIES: a Rearrangeable Inexpensive Edge-based On-line Steiner Algorithm," IEEE INFOCOM'96, pp. 361–368, March 1996.
18. M. Faloutsos, R. Pankaj and K. C. Sevcik, "Bounds for the On-line Multicast Problem in Directed Graphs," 4th International Colloquium on Structural Information and Communication Complexity, July 1997.
19. P. Winter, "Steiner Problem in Networks: a Survey," Networks, Vol. 17, pp. 129–167, 1987.