A Distributed Multicast Routing Algorithm for Real-Time Applications in Wide Area Networks

Tzu-Lun Huang
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan 106

E-mail: d88002@csie.ntu.edu.tw

D. T. Lee[1]
Institute of Information Science, Academia Sinica, NanKang, Taipei, Taiwan
E-mail: dtlee@iis.sinica.edu.tw

Abstract

In this paper we propose a delay-constrained distributed multicast routing algorithm based on token passing. This algorithm is fully distributed and generates a multicast routing tree, which not only meets the real-time requirement, but also has a sub-optimal network cost. Simulations have been done and the results have shown that the multicast routing tree generated by our algorithm has better performance than previously known results.

Keywords: *multicast routing, distributed algorithm, real-time routing, constrained Steiner Tree.*

1. Introduction

Due to prevalence of the Internet, the demand for obtaining multimedia information from the network is increasing. How to design algorithms that distribute the information with guaranteed Quality of Service (QOS) in real-time systems has been a cardinal issue addressed by researchers. Multicast routing, which refers to transmitting data from a source to multiple destinations within a time constraint, thus is an important problem to consider. It is a generalization of the concepts of one-to-one unicast and one-to-all broadcast in a communication network. It usually uses a tree topology, called a *multicast routing tree*, for data transmission over a wide area network. Only one copy of the data gets transmitted over a link between two nodes in the routing tree. Real-time multicast routing is a kind of QOS routing in which all recipients must receive data from the sender within a specified delay bound.

Real-time multicast routing arises in many multimedia applications; e.g., videoconference, distance learning and so on. In these applications we require that data must be sent to all destinations within certain time delay, otherwise the service quality will suffer greatly. Multimedia applications often need sufficient network resources (i.e. bandwidth) for video/audio data transmission. A logical or physical connection from the source to destination(s) needs to be established before any data transmission can occur. During connection setup, sufficient network resources (i.e. bandwidth or buffer) are reserved between two nodes in the connection so that user required Quality of Service (end-to-end delay, bandwidth, error rate, delay jitter, etc.) can be guaranteed at data transmission time.

Resource reservation consists of admission control, packet scheduling, and resource management and assignment. Routing is an important part of resource reservation. A route satisfying the QOS requested by the application is first determined, and then resources are reserved hop by hop along the route by the resource reservation protocol.

Conventional non-real time point-to-point transmission control protocols (i.e. TCP/IP) do not provide multicasting and guaranteed QOS. They are not adequate for multimedia data transmission. In recent years there have been many studies on best effort multicasting. In these works, the minimum cost (or hop-count) route is

[1] Also with Department of Computer Science and Information Engineering, National Taiwan University and with Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois; Research supported in part by the National Science Council under the Grants NSC-89-2219-E-001-002, NSC-89-2219-E-001-003 and by the National Science Foundation under the Grant CCR-9731638.

computed by examining the network topology. Even though these schemes exhibit simplicity and excellent scalability, they are not adequate for real-time multicast route computation.

Optimization techniques for multicasting can be categorized into two objectives, one is delay optimization and the other is cost optimization. Different optimization objectives can be used in multicast routing algorithms to define what constitutes a good multicast routing tree. One such objective is to provide the minimum delay from source to destinations along the tree, which is important for delay-sensitive multimedia applications, such as real-time teleconferencing. Another is to construct a minimum cost spanning tree, which is important in managing network resources efficiently. The tree cost is defined as the total cost of the links of the tree. This objective is referred to as utilization-driven in [21], because it optimizes the total utilization of links.

For real-time multicast routing, there are two important factors that we have to consider. The most important one is the end-to-end delay latency and the second is the network cost of the multicast routing tree. End-to-End delay latency is the latency from the source to each destination and includes physical propagation delay, transmission delay and queuing delay at each node. Propagation delay reflects the physical distance from source node to each destination node in the network, transmission delay reflects the bandwidth available for data transmission, and queuing delay reflects how congested each route is between the source and each destination. Network cost is the sum of the costs on all links in the multicast routing tree. The link cost may be hop counts or bandwidth required on the link and reflects the network load between two adjacent nodes in the routing tree. In this paper, we regard each link cost as the total bandwidth reserved by all sessions on that link. If the resource available on the link is less, then the link cost will be higher. Finding a minimal cost real time multicast routing tree in a network is also called the Constrained Steiner Tree problem, which is NP-Hard [3]. Many heuristic algorithms have been proposed to find approximate solutions to the Constrained Steiner Tree problem and they have different performance.

In this paper, we propose a token-based distributed real-time multicast routing algorithm that finds the real-time multicast routing tree in less time, using fewer messages and the tree cost is better than that reported in [5].

2. Problem definition and related work

The network is modeled as a directed graph $G(V, E)$, where V is the set of nodes and E is the set of links such that:

a. The weighting functions for delay and cost are defined respectively on each link:
$$d_e : E \rightarrow Z^+ \;;\; c_e : E \rightarrow Z^+$$
b. The cost and delay for $e(u, v)$ and $e(v, u)$ may or may not be the same depending on the network architecture (logical link or physical link), and they are not related.

The problem of finding a real-time multicast routing tree can be formulated as follows:

a. Given is a network $G(V, E)$ with delay and cost functions on the set of links, d_e, c_e, the source node s, a set M of destination nodes, and a real-time constraint Δ_i for each destination m_i.
b. For each link e belonging to $E(G)$, if $P(s, m_i)$ is the path from s to m_i in the multicast routing tree T then
$$\sum_{e \in p(s, m_i)} d_e(e) \leq \Delta_i$$
c. Our goal is to find a real-time multicast routing tree T, such that
$$\sum_{e \in T} c_e(e) \quad \text{is} \quad \text{minimized}$$

Centralized [2,6,8,9,11,14,15,20,21] and distributed [5,10,12,13,16,19] heuristic algorithms for the real time multicast routing problem or Constrained Steiner Tree problem have bee proposed. In the centralized scheme, every node in the network has a full knowledge of the entire network topology and its status. The source node (also called the root node) is responsible for computing the entire routing tree. The computation is quite straightforward in most centralized schemes. However, the overhead to maintain the status of the entire network can be very large. In the distributed scheme, on the contrary, each node in the network only has the information about its neighboring nodes, i.e., a partial knowledge of the network topology. It exchanges messages with adjacent nodes in the network. The distributed scheme is slow and complex, but it need not maintain the whole network in each node.

Kompella et al. [9] proposed a centralized algorithm for solving the Constrained Steiner Tree problem. This heuristic algorithm is based on Prim's Minimum Spanning Tree algorithm. A routing tree grows from source s and selects node v, which has the smallest weight and adds it to the tree. Jia et al. [7] proposed a centralized algorithm, which improves the performance of Kompella et al.'s heuristic. When selecting a node v to be added in the tree, if the delay at node v, $Delay[v] > \Delta$, it backtracks the tree formed so far and tries to find a tree structure which can

link v to the tree and satisfy $Delay[v] \leqq \Delta$. It then reconstructs a new tree to include node v. Zhu et al. [21] proposed another centralized algorithm called Bounded Shortest Multicast Algorithm. BSMA starts with a Shortest Path Tree to all destinations. It then iteratively replaces super-edges in the tree with lower cost paths until the total cost of the tree cannot be reduced any further. Although BSMA has a near optimal tree cost, it is not suitable for real networks due to its high computation cost.

Kompella et al. [10] also proposed a Minimum Spanning Tree based distributed algorithm for solving the Constrained Steiner Tree problem. In the algorithm, a *FIND* tree-broadcast message is first sent from *s* down to the tree, node by node, in order to determine the best out-going edge from the multicast tree. Upon receiving the *FIND* message, each node in the tree will decide the best out-going edge and send a *Reply* message back to *s*. Upon receiving all replies from all nodes in the tree, *s* will choose a best out-going edge and send a *Construct* message to add the edge into the tree. The algorithm continues to run until all destinations are included in the multicast routing tree.

3. Algorithm Description

3.1 Routing Information

In this section, we discuss the routing information needed for the proposed algorithm to compute the real-time multicast tree. Each node in the network should know the delays and costs of all outgoing links and must maintain the following information: a least delay vector and a least cost vector. They are collectively referred to as LD table and LC table respectively. Entries in these tables will be modified periodically as network states change.

In the least delay vector, each node $v_i \in V$ consists of $|V|$ - 1 entries, one entry for every other node. The entry for node $v_j \in V$ ($v_j \neq v_i$) contains the following items:

a. the destination node ID: $id[v_j]$;
b. the end-to-end delay of the least delay path $P_{ld}(v_i, v_j)$ from v_i to v_j: $Delay[P_{ld}(v_i, v_j)]$;
c. the cost of the least delay path $P_{ld}(v_i, v_j)$ from v_i to v_j: $Cost[P_{ld}(v_i, v_j)]$;
d. the next hop node on the least delay path $P_{ld}(v_i, v_j)$ from v_i to v_j : $id[P_{ld}(v_i, v_j)]$.

The least cost vector at node $v_i \in V$ also consists of $|V|$ - 1 entries, one entry for every other node. The entry for node $v_j \in V$ ($v_j \neq v_i$) contains the following items:

a. the destination node ID: $id[v_j]$;
b. the end-to-end cost of the least cost path $P_{lc}(v_i, v_j)$ from v_i to v_j: $Cost[P_{lc}(v_i, v_j)]$;
c. the delay of the least cost path $P_{lc}(v_i, v_j)$ from v_i to v_j: $Delay[P_{lc}(v_i, v_j)]$;
d. the next hop node on the least cost path $P_{lc}(v_i, v_j)$ from v_i to v_j : $id[P_{lc}(v_i, v_j)]$.

The above least delay vectors and least cost vectors are obtained by using a distributed Bellman-Ford algorithm-based Distance Vector Routing Protocol [1]. Furthermore, we assume that the least cost vectors and the least delay vectors at all nodes have been obtained, and that their content including the link costs, and link delays do not change during the execution of the routing algorithm.

To record the cost from the source to each relaying node, a routing-token is introduced. The token consists of four possible components, containing token state which shows the state of the algorithm (*Fork* State or *Setup* State or *Complete* State), the delay D_u which shows the delay from source to node u, d, the destination node whose connection is to be set up, and a table T2D (tree to destinations). The entries in the T2D table may change, as destination nodes are included in the tree. Each entry in the T2D table has five fields: destination node ID d_i, the cost from the tree to node d_i, the relaying node on the tree that is closest to the d_i, the delay constraint Δ_i and the type of path, least cost path (LCP) or least delay path (LDP), from the relaying node to d_i.

3.2 The Algorithm

Every node in the system executes the same algorithm. It is initially in an idle state waiting for connection setup. When a node receives a request for opening a multicast connection with parameters M and Δ_i, it becomes the source s of the multicast connection and the routing-token is initiated. According to the results of table lookup in LC or LD table, the source node s may fill the cost for each destination $d_i \in M$ in the T2D table. The routing tree RT consists of the source s initially. The destination dn closest to a node v$\in$ RT, whose delay satisfies $D_{dn} \leqq \Delta_{dn}$, is selected and marked. Then the routing-token is created and the token state is set to be the *Setup* State. Then the token is sent to the neighboring node *v' of v*, referred to as the relaying node, on either the least delay path or the least cost path from v to dn.

When the setup routing-token arrives at a relaying node, say node u. D_u is extracted from the token and recorded at this node. For each destination d_i not already in the tree, the following condition is checked:

$(D_u + \text{LC\_Table}[u, d_i].d \leqq \Delta_i)$ AND (LC_Table[u, d_i].c < T2D[d_i].c). (1)

where LC_Table[u, d_i].d and LC_Table[u, d_i].c denote the delay and the cost of LCP from node u to d_i respectively.

If inquality (1) is satisfied, entry T2D[d_i] is updated as follows:

T2D[d_i].c: = LC_Table[u, d_i].c; T2D[d_i].node := u

If inequality (1) is not satisfied, the following is checked:

(D_u + LD_Table[u, d_i].d $\leqq \Delta_i$) AND (LD_Table[u, d_i].c < T2D[d_i].c). (2)

If inequality (2) is satisfied; entry T2D[d_i] is updated as follows:

T2D[d_i].c: = LD_Table[u, d_i].c; T2D[d_i].node := u

Otherwise, T2D table is unchanged.

When the connection S*etup* token reaches the designated destination node, the next destination node dx in T2D table whose T2D[dx].c is minimum is selected as the next one to be included in the routing tree. The token state is changed from *Setup* state to *Fork* state, and the token is sent to the corresponding node *w* represented in T2D[dx].node. Upon the receipt of the *Fork* token, node *w* changes the *Fork state* to *Setup* state and the connection setup from node *w* to destination dx repeats.

The above operation repeats until all destinations are included in the tree. When the last destination is added to the tree, a token of *Complete* state is sent to the source *s* by changing token state to *Complete* state.

However, loops may occur in our algorithm when constructing a path to a new destination. A loop can be easily detected by checking whether the next hop-node has an existing routing table for the current multicast session. If a tree node *v* is detected to be in a loop, then node *v* will send a PRUNE message to its parent node v_p. Upon receiving a PRUNE message, node v_p removes its link to node *v* and checks if it is either a destination node or a relaying node. If node v_p is neither a destination node nor a relaying node, then node v_p continues to send PRUNE message to its parent node. Otherwise the process stops.

Our algorithm also supports dynamic change of multicast participants. In many multimedia applications, such as distance learning and teleconferencing, multicast participants are free to leave or join a multicast tree dynamically. Our algorithm ensures that any change of multicast participants will not affect the traffic of the current connection and the tree cost remains sub-optimal after the change.

When *m* nodes want to join an existing multicast group, each of them sends a request message to the source of the group. The source *s* may then send a token *of tree-broadcast* state to all nodes in the tree. When a node in the tree receives the *tree-broadcast* token, it computes the best destination node by looking up its LC and LD tables. When the request arrives at a leaf node, the leaf node sends a reply token including the cost and tree-node information back to its parent node. This parent will compare the cost from all of its child nodes' replies with its own cost, choose the best solution and send a reply token to its parent. Once the source node has decided which destination node to include first, it will broadcast the information along with the tree node responsible for the connection to all the nodes. The tree node will receive a *Setup* token and the algorithm enters into the state of establishing a connection setup from the tree node to the new destination. This process terminates until all the requests have been fulfilled.

When a destination node requests to leave a multicast group, it is "disconnected" from the connection. If the destination is a leaf node of the routing tree, it just sends a *leaf* token (a token of *leaf* state) to its parent node that will terminate its connection to the leaf node. This parent node, which may now become a leaf node, will repeat this process. If the node is not a leaf node, it simply changes its status from destination node to relaying node.

3.3 Complexities and Correctness of our Algorithm

We summarize the results below with the proofs omitted.
Theorem.1: The message complexity and time complexity of our proposed algorithm are $O(|V|^2)$ where $|V|$ is the number of nodes in the network.

Theorem 2: Our proposed algorithm always finds a tree that satisfies the delay (or real-time) constraint, if it exists.

4. Simulation model and results

4.1 Simulation model

We use the same simulation environment (model) as described in [14] for the performance evaluation of our proposed algorithm.

4.2 Simulation results

The overall network performance is evaluated with two parameters: multicast group size and real-time constraint. In our simulation, we generate a random graph with 100 nodes, with multicast group sizes ranging from 5 nodes to 40 nodes and real-time constraints from 75 ms to 250 ms. At each simulation point (group-size vs. real-time constraint), we ran the simulation 300 times using 95% conference interval. Each time the source node *s* and the destination nodes are randomly selected from the graph. The simulation result is the mean value of the results produced by these 300 runs. Table-1 shows our simulation results. In the table the first row indicates real-time

constraints ranging from 75 ms to 250 ms and the first column denotes the group sizes ranging from 5 nodes to 40 nodes. The numbers in Table-1 are the percentages of improvement over the results found in [5]. Form this table; we see that our proposed algorithm has a significant improvement in sparse networks, approximately 20% superior. Figure.1 and Figure.2 show the same result but in a different way. The percentage of our improvement in Figure.1 is computed as follows:

$$\text{improvement } \% = \frac{\text{Cost (Compared Algorithm)} - \text{Cost (Our Algorithm)}}{\text{Cost (Comapared Algorithm)}}$$

Figure.2 is the percentage of our improvement versus real-time delay constraint when the multicast group size is fixed and ranging from 5 nodes to 40 nodes. From this figure, we can see that our algorithm has a better performance than the compared paper [5], especially in sparse mode.

	75	100	125	150	175	200	225	250
5	19.013	19.013	19.013	19.013	19.013	19.013	19.013	19.013
10	14.302	14.374	14.374	14.374	14.374	14.374	14.374	14.374
15	9.628	9.701	9.701	9.701	9.701	9.701	9.701	9.701
20	9.229	9.629	9.629	9.629	9.629	9.629	9.629	9.629
25	8.328	8.319	8.319	8.319	8.319	8.319	8.319	8.319
30	5.962	6.197	6.203	6.203	6.203	6.203	6.203	6.203
35	5.670	5.884	5.863	5.863	5.863	5.863	5.863	5.863
40	4.821	5.541	5.563	5.563	5.563	5.563	5.563	5.563

Table-1. Tree cost improvement (%) relative to those in [5]

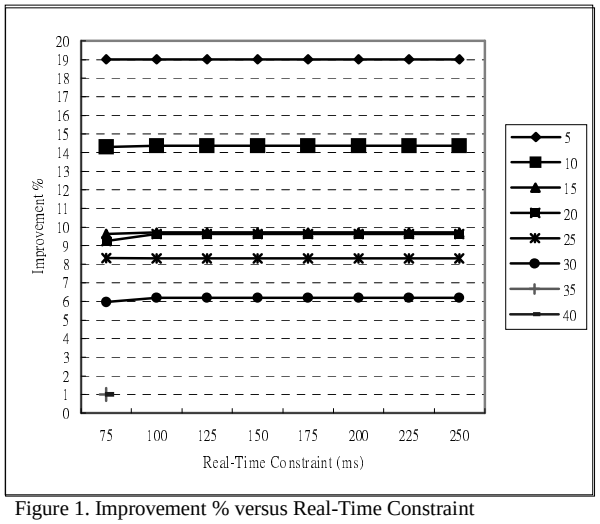

Figure 1. Improvement % versus Real-Time Constraint

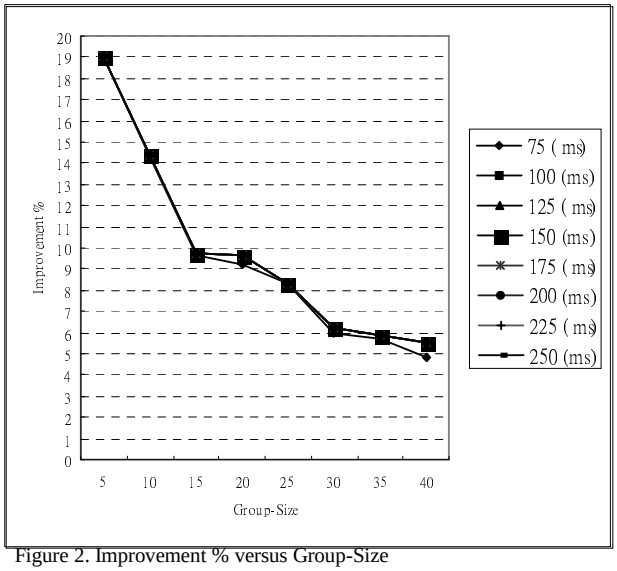

Figure 2. Improvement % versus Group-Size

5. Conclusions

In this paper, we have proposed a token-passing based distributed real-time multicast routing algorithm for WAN. Simulation results have shown that our algorithm is superior to that proposed in [5]. The algorithm is shown to run in time quadratic in the number of nodes and its message complexity is also quadratic. Furthermore the algorithm allows for the multicast participants to join or leave the multicast group dynamically without affecting the connection quality of the existing multicast tree. Whether or not the time and message complexities can be further reduced remains to be seen.

6. References

[1] R. Bellman, "Dynamic Programming," Princeton University Press, 1957.

[2] K. Bharath-Kumar and J.M. Jaffe, "Routing to multiple destinations in computer networks," IEEE Transactions on communication, 31(March), pp. 343-351, 1983.

[3] T.H. Cormen, C.E. Leiserson and R.L. Rivest, "Introduction to algorithms," The MIT Press, 1990.

[4] M.R. Garey and D.S. Johnson, "Computer and Intractability: A guide to the theory of NP-Completeness," W.H. Freeman, 1979.

[5] Y. Im, Y. Lee, S. Wi and Y. Choi, "Delay constrained distributed multicast routing algorithm," Computer Communications 20, pp. 60-66, 1997.
[6] X. Jia, C.H. Lee, K. Makki and N. Pissinou, "Efficient multicast tree algorithm in ATM networks," Computer Communications 19, pp. 637-644, 1996.
[7] X. Jia, N. Pissinou and K. Makki, "A real-time multicast routing algorithm for multimedia applications," Computer Communications 20, pp. 1098-1106, 1997.
[8] D.D. Kandlur, K.G. Shin and D. Ferrari, "Real-time communication in Multi-hop networks," Proc. IEEE 11th Intl. Conf. on Distributed Computing Systems, pp. 300-307, 1991.
[9] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Multicast Routing for Multimedia Communication," IEEE. Trans. on Networking. Vol. 1, No. 3, June, pp. 286-292, 1993.
[10] V.P. Kompella, J.C. Pasquale and G.C. Polyzos, "Two distributed algorithms for the constrained Steiner tree problem," Proc. 2th Intl. Conf. on Computer Communication, pp. 343-349, 1993.
[11] K. Makki, "A new approximation algorithm for the steiner tree problem," Congress Numerantium 84, pp. 135-140, 1991.
[12] J. Rugelj and S. Klavzar, "Distributed Multicast Routing in Point-To-Point Networks," Computer. Ops. Res. Vol. 24, No. 6, pp. 512-527, 1997.
[13] D.S. Reeves and H.F. Salama, "A Distributed Algorithm for Delay-Constrained Unicast Routing," IEEE/ACM trans. on. Networkin, Vol. 8, No. 2, April, pp. 239-250, 2000.
[14] H.F. Salama, D.S. Reeves and Y. Viniotis, "Evaluation of multicast routing algorithms for real-time communication on high speed networks," High Performance Networking," IEEE Journal of Selected Area in Communications, Vol. 15, No.3, April, pp. 332-345, 1997.
[15] H. Salama, "Multicast Routing for Real-time Communication on High-Speed Networks," PHD thesis, North Carolina State University, Department of Electrical and Computer Engineering , 1996.
[16] Q. Sun and H. Langendorfer, " A new Distributed Routing Algorithm for Supporting Delay-Sensitive Unicast Routing," Computer Communications 21, pp. 572-578, 1998.
[17] B.M. Waxman, "Routing of multipoint connection," IEEE Journal on Selected Areas in Communications, Vol. 6, no. 9, pp. 1617-1622, Dec, 1988.
[18] B.M. Waxman, "Performance evaluation of multipoint routing algorithm," IEEE INFOCOM, pp. 980-986, 1988.
[19] S. Wi and Y. Choi, "A delay constrained distributed multicast routing algorithm," 12th Int. Conf. on Computer Communication, ICCC'95, pp. 833-838, 1995.
[20] Z. Wang and J.Crowcroft, "Quality-of-service routing for supporting multimedia applications," IEEE Journal on Selected Area in Communications 14(7), pp. 1228-1234, 1996.
[21] Q. Zhu, M. Parsa and J.J. Garcia-Luna-Aceves, "A source-based algorithm for Near-optimum delay-constrained multicasting," IEEE INFOCOM, pp. 377-385, 1995.