A Distributed Approach for Dynamic Multicasting

DEBASISH CHAKRABORTY †
† Research Institute of Electrical Communication
Tohoku University, Sendai
deba@shiratori.riec.tohoku.ac.jp
SALAHUDDIN MUHAMMAD SALIM ZABIR †
szabir@shiratori.riec.tohoku.ac.jp

GOUTAM CHAKRABORTY
Dept. of Software & Information Science
Iwate Prefectural University
goutam@iwate-pu.ac.jp
NORIO SHIRATORI †
norio@shiratori.riec.tohoku.ac.jp

Abstract

For many applications, the set of destinations will be changing dynamically, with destinations joining and leaving the multicast group during the communication. Under such condition the computation of an optimal spanning tree is not the best way to proceed. Furthermore, finding optimal multicast route (also known as Steiner tree) has been proved to be NP-complete. In this paper, we propose a heuristic destination driven distributed dynamic multicast routing algorithm, and tried to minimize the total cost of the Steiner tree over the whole session period, where information about the resource reservation (i.e. joining and leaving times of participants) are available at the time of joining of the node. The efficiency of our algorithm and comparison with other existing algorithms are shown by various simulation results.

keywords: dynamic multicast, distributed routing, resource reservation, optimization

1. Introduction

Multicast can be defined loosely as the ability to logically connect a subset of hosts in a network. A packet switched network is said to be able to provide a multicast service, if it can deliver copies of a packet to a set of destinations simultaneously. The optimization criterion for a multicast routing algorithm can be classified into two general categories. One is the shortest path tree (SPT), which minimizes the cost from source to each destination. Another optimization goal is to minimize the overall cost of the multicast tree. The second category is known as a Steiner tree problem, which is proved to be NP-complete problem [9] in general case. A survey of both exact and heuristic solutions can be found in [5].

A multicast group can be either (*i*) *static* - once set up, the multicast tree remains unmodified until all the nodes are discarded or torn down. (*ii*) *dynamic* - destinations join and leave the session at any time independently. This dynamic multicast can again be classified into two. One, where advance information about the whole session and participation time for each individual is available and the other, where no such information is available.

There exists several dynamic multicast routing algorithms. Most of them are centralized and optimum solutions are often too complicated [7] for practical application, specially in dynamic situation. Though there are simpler solutions [3, 10], they are proved to be costly in the sense of total tree cost. KMB [8] can produce near optimal tree, but it requires total re-routing for every membership changes, which is unsuitable for maintaining continuous packet transmission. Even partial re-routing procedure [6] will cause re-ordering and thus synchronization problem for continuous stream data. It is complex as cell-ordering at ATM switches has to be preserved. Therefore, the algorithm should be able to maintain the near optimal dynamic multicast tree without restructuring the existing tree.

In this paper, without reconstructing the existing multicast tree, we minimize the overall tree cost for the whole session duration, where the duration of each participation is known only at the time of joining. This is practical especially when, for reserving the network resource, one has to pay for it. Longer the duration of the connection, higher will be the cost. Naturally a reservation request is expected to be made carefully by the user, mentioning nearly exact duration of using the network. The time span, mentioned by the user while joining the session, is used as a parameter to re-calculate the link cost of the tree. The routing takes a greedy strategy using Dijkstra's shortest path [1]. Extensive simulation results show the efficiency of the proposed algorithm over the existing ones.

The proposed algorithm has the following advantages:

1. *Distributed in nature:* Each node operates based on

its local routing information and coordination with its neighboring nodes via network message passing.

2. *Suboptimal Routing Trees for the whole duration:* We have successfully reduced the total cost of the tree for the whole duration of the session.

3. *Dynamic Membership Changes of Multicast Groups:* A new destination can join or an existing member can leave without restructuring the routing tree.

The rest of the paper is organized as follows. In Section 2, we describe the advance resource reservation strategy. In Section 3, we define the dynamic multicast routing problem, then describe the related works. In Section 4, we present our algorithm, a Distributed and Modified Greedy algorithm (DMG). In Section 5, simulation results are shown comparing with Greedy and Naive multicast. Concluding remarks and scope of further works are in Section 6.

2. Advance Resource Reservation Environment

In the advance resource reservation environment, the requests made by the application specify, not only the parameters that define the traffic flow specifications and its QoS requirements, but also the following two quantities: (a) the starting time, and (b) the duration time. There may be two cases when the starting time is known: (i) it is known at the time when the request for resource reservation arrives. The resource will be used immediately after they have been successfully reserved. (ii) it is known a priori, i.e., there is a request for the resource reservation earlier to actually using of the resources, which will be in the future. Similarly, the duration can be: (i) known, or (ii) unknown. Combining these two quantities, we can classify the advance resource reservation model into 4 schemes as follows:

Scheme 1: $\tau_{cur} = \tau_{arr} = \tau_{st}$, and τ_{dep} is known.

Scheme 2: $\tau_{arr} < \tau_{st}$, and τ_{dep} is known.

Scheme 3: $\tau_{cur} = \tau_{arr} = \tau_{st}$, and τ_{dep} is unknown.

Scheme 4: $\tau_{arr} < \tau_{st}$, and τ_{dep} is unknown.

where τ_{cur} is the current time, τ_{arr} is the time when the application requests for a resource reservation, starting time τ_{st} is the time when the application want to get connected and use network resources, and τ_{dep} is the time when the application will disconnect. The duration time δ is then defined by $\tau_{dep} - \tau_{st}$. And $\tau_a < \tau_b$ means τ_a is earlier than τ_b.

Employing these information wisely is quite complex. In this paper, we consider situation as in Scheme 1 which is very practical too. The discussion of using scheme 2 is postponed to a future publication.

3. Dynamic Multicasting Problem Specification

3.1. Problem Formulation

The network is modeled as a directed graph $G = (V, E)$ where the nodes in the graph represent network routers and the edges correspond to communication links between nodes. Each link $l \in E$ is associated with two parameters $c(l)$ and $b(l)$. $c(l)$ denotes the communication cost of l. It can be considered as a *cost function*, which maps l into the set R of non-negative real numbers, $c(l) \rightarrow R$. $b(l)$ is a measure of bandwidth available on link l. Similarly, $b(l)$ is a *bandwidth function*, which maps l into the set R of non-negative real numbers, $b(l) \rightarrow R$.

Given a source node $s \in V$, a set of destination nodes $D \subset V$, with $s \notin D$, a routing tree for a multicast connection is subtree of the graph $G(V, E)$ rooted from s, that contains all of the nodes of D and an arbitrary subset of $(V - D)$. When multicasting a message to the nodes of D, source node s (the root of a routing tree) sends a copy of the message to each of its children along the tree. These children in turn transmit the message to their children until all nodes in the tree (thus, all nodes in D) have received the message. If $|D| = 1$, it becomes a unicast, and if $|D| = |V| - 1$, it becomes a broadcast. According to the nature of trees, a multicast message flows through each branch of the routing tree once and only once to reach all the destinations. Therefore the network cost of multicasting a message to a group of destinations is proportional to the sum of the cost of all links in the multicast tree.

$R_t = \{r_t(1), r_t(2), ..., r_t(m)\}$ is the set of join requests asked by participants at time t. $r_t(i)$, the i^{th} request at time t, is expressed as the tuple :

$$\langle z_i, \tau_{dep}(z_i), \Delta_b(z_i) \rangle$$

where $z_i \in D$ is the destination node that want to join at time t, $\tau_{dep}(z_i)$ is the departure (leaving) time of node z_i, and $\Delta_b(z_i)$ is the bandwidth requirement of destination node z_i. m is the number of nodes that want to join at time t. T is the total session period.

The problem is to find a sequence of multicast tree routes (without re-routing) with minimal total cost of the tree over the whole session period T. The bandwidth requirement of each destination has to be satisfied i.e. for destination z_i, each link l along the path from s to z_i must have available bandwidth greater or equal than $\Delta_b(z_i)$. At each time instant t, a multicast tree is defined as a subtree of the graph $G(V, E)$, rooted from s, that contains the destination nodes

which join before (or at) time t, and have not departed from the session yet.

There are two basic requirements to be satisfied in our problem. First, to guarantee cell ordering at the destination (without any additional hardware), re-routing will not be allowed. A route chosen for a connection request, will be unaltered throughout their life-time. Second, the bandwidth constraint for each destination must be satisfied.

Most of the multicast routing algorithms try to minimize the cost of the connection tree for individual instant of time without taking the whole session time T into consideration, as shown in Eq.(1). Here MC is the multicast connection tree and l is the individual links that make up the multicast tree i.e.

$$minimize \left\{ \sum_{l \in MC} c(l) \right\} \quad (1)$$

In this paper, the goal is to minimize the total cost of the tree over the session time T. It is the time integral of the cost of multicast tree over the whole session period, as shown in Eq.(2), where $Cost_{tree}(t)$ is the cost of multicast tree at time t i.e.

$$minimize \left\{ \int_0^T Cost_{tree}(t)dt \right\} \quad (2)$$

4. Algorithm Description

Our algorithm is a modification of Greedy algorithm with virtual link-costs [2]. Virtual link-costs are calculated at the time of arrival of a new node, and it depends on the staying time of the newly arrived node and the remaining time of other tree nodes. The proposed algorithm is fully distributed in nature, and named as *Distributed and Modified Greedy*, in short DMG.

4.1. Virtual Link-Cost (VLC) calculation

In the proposed algorithm, with every link there is an associated *time-tag*. This *time-tag* is the time until which the link has to remain connected in the multicast-tree. Naturally those links outside the multicast tree assumed to have *time-tag* zero.

In virtual link-cost calculation for DMG, where a link e with actual cost $c(e)$ connects node u and v and have $time-tag = \tau_{tag}(l)$, and $\tau_{dep}(new)$ is the new node's departure time, the virtual link-cost (VLC):

1. between two nodes u and v, forming a link, which is outside the running multicast tree is $c(e) \times (\tau_{dep}(new) - \tau_{cur})$.

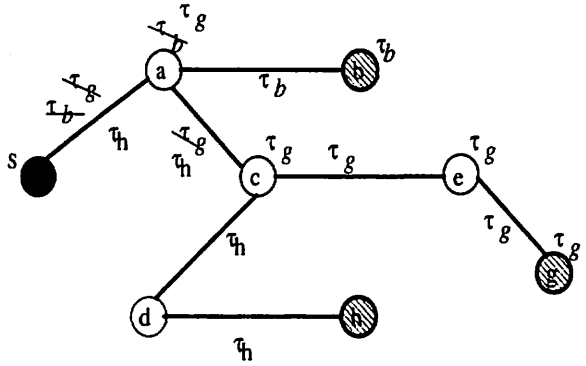

Figure 1. Example of tag updating

2. between two nodes, forming a link of running multicast tree with $\tau_{dep}(new) > \tau_{tag}(l)$ is $c(e) \times (\tau_{dep}(new) - \tau_{tag}(l))$.

3. between two nodes, forming a link of running multicast tree, but $\tau_{dep}(new) < \tau_{tag}(l)$ is considered as zero.

Link-updating: Whenever a new node join the multicast tree the node and link tag from that node to source will be updated if the tags are less than the new node. For example, in Fig. 1, g has joined before h through e, c, a to S. But when h join with $\tau_h > \tau_g$, and h join through d, c, a to S, the tag of c and a will be updated from τ_g to τ_h.

4.2. Outline of the Algorithm

In our algorithm, while constructing the multicast tree we recalculate the virtual link-costs. The basic idea is that, when a link is in the tree and will remain connected for time longer than the time duration of connection of the new node, its virtual cost with respect to connecting the new node is zero. But if it is necessary to be kept in the existing tree (for already connected nodes) for a duration of time less than the time of connection of the new node, its virtual cost will be positive. Final route searching is by a distributed shortest path algorithm, where message-passing will be done along the existing tree nodes. We assume that each node has the information about the shortest path, in terms of actual link-cost (which is fixed) to every other node in the network. It also has the time-tag information of its next-hop nodes only and the corresponding links. This information is stored in the node's local routing table. This information is modified, if any change occur. We calculate the virtual cost of the shortest paths from source to new destination node, directly and through each existing multicast group members. We call this as SPVC (shortest path's virtual cost). The shortest among the possible paths will be selected as the best route

to join the new node to the existing tree. These shortest paths' virtual costs from source to destination will vary as VLCs between links vary according to newly arriving destination nodes' staying duration. By minimizing the virtual tree cost, we minimize the cost of multicast tree over the whole duration of session, even with dynamic joining and leaving of participants.

All nodes and links in the networks will have *time-tag* initialized to zero. The first arrived node (or selected as the first node from a group), will join the source with simple shortest path route. All the nodes and the links in this path will be tagged as the first destination's departure time. The later destination nodes, will be added with virtually shortest route to the source and the nodes' and links' *tags* will be updated, if the previous *tag-values* are less than the new one along the path from source to the new destination. The destination node can select the suitable node from the multicast member to join the session independently.

Whenever a new node will join the tree and if it is not included in the tree already, it will first send its departure time τ_{dep} and its $node_{id}$ to the source by the shortest path. Starting from source, every member of the multicast node will do the following calculation. It will first calculate the virtual-cost of the shortest path (SPVC) from itself to this new node and add the VLC from source to itself, to find the virtual shortest path from source to new node, through that node. If it finds that its calculated cost is lesser than the present minimum, it will replace the value in two places of the table. First, the $node_{id}$ of the multicast node, from which the previous virtual shortest path was found by its own and Second, the route information.

Source is the first node that will initiate the process. After calculating the SPVC from itself to new node, it will send to the next node a message table that consists of **(i) new node's id, (ii) it's own id** (because it is the virtual nearest node from the new node so far) **(iii) departure time of the new node, (iv) SPVC from itself to new node and (v) virtual link cost (VLC) from source to itself** (which is zero in case of source node). If there is a branch, the same message-table will be send to all branches. Now, the next node will also do its own calculation of SPVC from itself to new node, and add the VLC from source to itself, so that it can get the total virtual cost from source to new node, through that node.

This process will be repeated until it reaches a leaf-node. Thus the information will reach all the tree nodes along the existing tree-routes. Every tree node can execute the necessary calculation to search virtual path cost through that node to new destination from source and update the table if it can find a virtual shorter path than the one found so far by its previous nodes from which it received the message-table.

So, when the message table will reach a leaf node of any particular branch, it will have the most updated information about the shortest virtual cost path from source to new node in that branch. Every leaf-node will send this message to the destination node. The destination node will check the messages-table coming from different leaf-nodes and select the shortest virtual path from source to itself, and connect itself to a node accordingly. Once the virtual shortest path is determined, the departure time of the newly joining node will be the *tags* for all the nodes and the links from source to the newly joined node, if it is more than the existing *tag-value*.

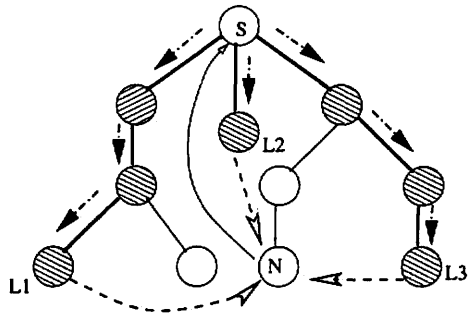

Figure 2. The process of message passing in the tree

Fig. 2 shows how the process starts from the new node and with the new node's $node_{id}$ and its τ_{dep}. The searching for least-cost virtual path initiates from the source (S) and it gradually reaches the leaf-nodes. The shaded nodes are the multicast nodes and only these already participating multicast tree nodes join the searching process. The leaf-nodes (L1, L2, L3) ultimately sends the most updated information (routing table) to the new node (N) through the shortest path and the new node will join to that node which is virtual shortest from it.

Only the initial information of shortest path which is actually available in most implementation of current routing table, is required at every node. The algorithm requires simple message passing along the existing tree. The recalculation of link-cost to find the virtual-cost of a link is also simple and reasonable, as we are aiming for a low-cost tree for the whole duration of a session. We have used Dijkstra's shortest path algorithm to find the shortest route between two nodes. This algorithm has been already in use in Internet routing protocol, such as OSPF. With our algorithm, we may not always produce a perfect optimum tree, but it certainly produces less costly tree than Greedy or Naive on average.

4.3. Deletion of Nodes

When the remaining time of a node in the tree become zero, that node is supposed to leave the group. The deletion process will be initiated by *deletion request* from the receiving node. If it is a leaf-node, it can leave immediately and the corresponding link will also be removed. This deletion process will be executed recursively until it reach a node that has higher *time-tag*. The deleted nodes' and links' *time-tag* will be initialized as *zero*.

5. Experimental Set up and Simulation Results

Random graphs were constructed with their connectivity characteristics approximated to those observed in real networks. The total number of nodes in our model is 50. The network model to be considered are as in [10], where graphs are constructed by distributing n nodes across a Cartesian coordinate grid. Edges were added to the graph by considering all possible pairs (u, v) of nodes and using the following probability function

$$P_e(u, v) = \beta \times exp\left(\frac{-d(u, v)}{\alpha L}\right)$$

to create an edge. Here $d(u, v)$ is the Euclidean distance between the nodes' locations, L is the maximum possible distance between two nodes and α and β are parameters in the range $0 \leq \alpha, \beta \leq 1$. A large value of α increases the number of connections to nodes further away from it, while a large value of β increases the number of edges from each node. The cost of an edge was defined simplistically as $w(u, v)$, the distance between its nodes. For every graph is was ensured that every node is connected i.e. a spanning tree exists covering all the nodes in G.

The number of total multicast nodes present at any particular time, in our simulation is limited to 40% of the total nodes in the network. To imitate actual situation involved we considered that *Every new node is coming one at a time with their individual staying time.* Repeated joining was considered, i.e. a node may rejoin after disconnection.

5.1. Random Arrival and Departure Time

A node can arrive at any time of the session, provided it is not already in the multicast tree. The arrival times were simulated using uniform random function. The frequency of joining is made high at the beginning and decreases at the end. The departure time of any node, for obvious reason should not exceed the end-time of the session. The duration of connection is restricted to be at least one-tenth of the whole session time. The average staying time is nearly 60% of the total session time. The nature of arrival and departure has been made, emulating the practical situation observed in multicast sessions. We ran our simulation by using different set of random arrival and departure times and different network topologies. We took the average value by running it one thousand times. So we can assume that the performance of the algorithm is unbiased.

As is obvious, the source node is connected from the beginning to the end of the session for all occasions. The number of destinations is a proper subset of the total number of nodes.

5.2. Results

Simulation programs were written in C and run on a SUN SPARC workstation under SOLARIS operating system. Simulations results are shown in Fig. 3 to Fig. 4.

We have calculated both instant and cumulative tree cost for the whole session and the comparison graph is shown consisting of Naive, Greedy and DMG.

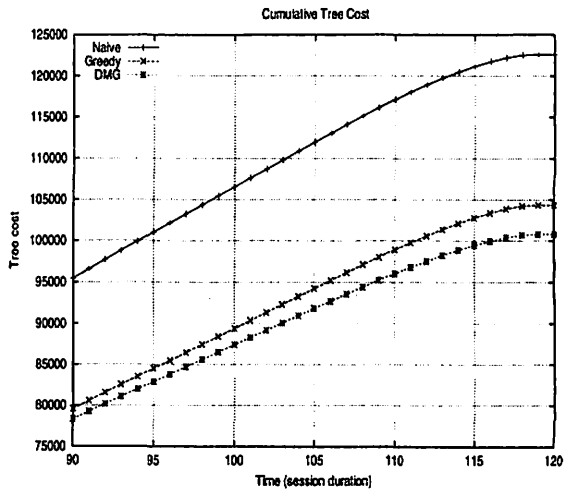

Figure 3. Cumulative cost

The results are shown in Fig. 3 and Fig. 4. The Naive algorithm performs much worse than other two algorithms. From the cumulative cost graph (Fig. 3) the improvement of DMG becomes clear.

The mean and maximum tree cost with different ratio of the number of destinations to total nodes, are shown in Fig. 4(a) and Fig. 4(b). Mean tree cost is the average tree cost of an individual session and maximum tree cost is the highest tree cost at any instant of a session. On both the occasion, the average of several instances has been taken. Our algorithm proved more efficient than other two algorithms in each cases.

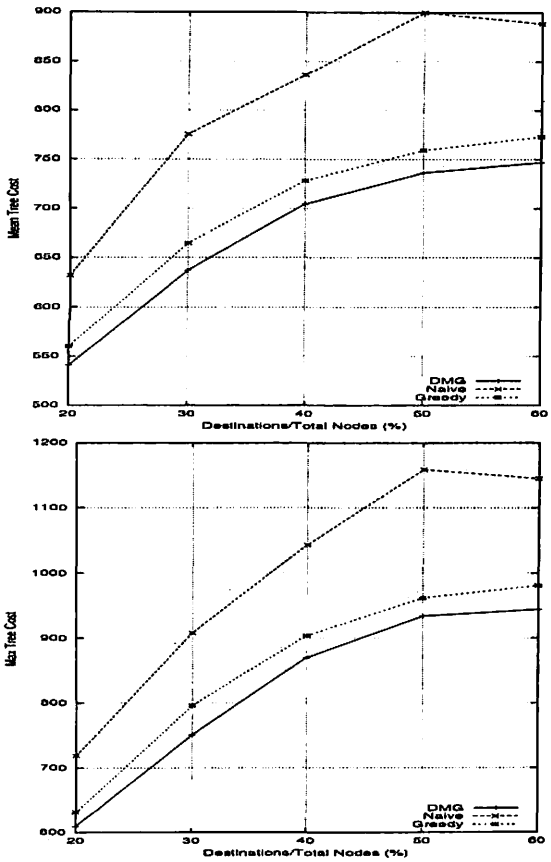

Figure 4. (a) Mean Tree Cost for the whole session, (b)Average of maximum tree cost of the session

To emphasize the effect of the respective algorithms, we have shown the end part of the cumulative cost (Fig. 3). As the time passes and nodes start leaving, Greedy based algorithms start performing worse than our algorithms because the number of actual node deletion becomes less compare to our algorithms. They remain connected in the multicast tree as Steiner nodes even after crossing their actual departure times.

6. Conclusion

In this paper we presented a distributed algorithm for dynamic multicast routing that biases toward existing routes with longer departure times. The actual thought behind our algorithm is to increase the probability of creating leaf nodes in the multicast tree, such that, when a node's departure time will arrive, it may leave instead of staying in the multicast tree as Steiner node. Our aim here is to minimize the overall cost of the tree and the primary concern is to keep costs down over the entire transmission session and also without any rerouting when member changes in a dynamic situation.

Experimenting with different batch of arrival and departure time as well as network with varying topologies, it has been seen that DMG algorithm could produce, in fully distributed manner, less costlier tree, considering the whole duration of session time.

Additional goals for future work include experimenting the performance of our algorithm while applying in actual multicast application. We are also considering the coexistence of more than one multicast session and to find a suitable algorithm to handle that situation.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures And Algorithms*. Addision-Wesley, April 1987.

[2] D. Chakraborty, G.Chakraborty, C. Pornavalai, and N. Shiratori. Optimal routing for dynamic multipoint connection. *European Transaction on Telecommunications*, 10(2):183–189, March-April 1999.

[3] Matthew Doar and Ian Leslie. How bad is naive multicast routing? In *Proceedings of IEEE INFOCOM*, San Francisco, CA, pages 82–89, April 1993.

[4] A. Gupta, D. Ferrari, and G. Ventre. Distributed advance reservation of real-time connections. April 1995.

[5] F.K. Hwang and D.S. Richards. Steiner tree problems. *Networks*, 22(1):55–89, January 1992.

[6] James Kadirire. Minimizing packet copies in multicast routing by exploiting geographic spread. In *ACM SIGCOMM CCR*, July 1994.

[7] V.P. Kompella, J.C. Pasquale, and G.C. Plyzos. Multicast routing for multimedia communication. *In IEEE/ACM Trans. Networking*, 1(3):286–292, June 1993.

[8] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for steiner trees. *Acta Informatica*, 15(11):141–145, 1981.

[9] M.R.Garey and D.S. Johnson. *Computers and Intractablitity: A Guide to the Theory of NP-Compleness*. W.H. Freeman Company, San Francisco, November 1990.

[10] Berman M. Waxman. Routing of multiple connections. *IEEE Journal on selected areas in Communications*, 6(9):1617–1622, December 1988.