DBpia

논문 99-24-1A-9 한국통신학회논문지 '99-1 Vol.24 No.1A

정적 및 동적그룹을 지원하는 지연시간 보장형 분산 멀티캐스팅

정회원 임 용 준*, 최 양 희*

A Distributed Delay-Constrained Multicasting for Static and Dynamic Multicast Groups

Yong jun Lm*, Yang hee Choi* *Regular Members*

ABSTRACT

Multicasting refers to concurrently transmitting the same data from a source to multiple destinations in a packet network. To handle large number of multicast sessions, a network must minimize the sessions's resource consumption, while meeting their QoS requirements. In this paper, we propose a distributed multicast algorithm for constructing minimum-cost multicast trees with delay constraints. The proposed algorithm requires limited network state information at each network node and the routing tree is computed through a single round of message exchanges between network nodes. We prove the algorithm's correctness by showing that it is always capable of constructing a delay-constrained multicast tree, if one exists, and it has the worst case message complexity of $O(|V|^2)$, where $|V|$ is the number of network nodes. The proposed algorithm is verified by simulation, and it is shown that the proposed algorithm exhibits much better performance than the existing ones for static and dynamic multicast groups in all circumstances, which include various network sizes, multicast group sizes and end-to-end delay bounds.

I. Introduction

One of the pressing needs for enhanced communication protocols comes from realtime multipoint (or group) applications These involves more than two users and requires strictly controlled QoS(Quality of Service) at the network layer. Such applications cover a very wide spectrum, including software distribution, replicated database update, command and control systems, audio/video conferencing, distributed games, and distributed interactive simulation(DIS).

Multicasting refers to concurrently transmitting the same data from a source to multiple destinations in a packet network. Multicasting can be provided at any layer in the protocol stack, but the protocol efficiency increases at the lower layer. Datagram networks can not provide QoS guarantees to realtime applications. Circuit-switching networks are capable of providing guaranteed service to the applications, but they do not manage the network bandwidth efficiently. Recently, B-ISDN(Broadband Integrated Services Digital Network) is evolving at a fast speed. B-ISDN uses the ATM(Asynchronous Transfer Mode) which is based on high speed packet-switching technology. ATM is a connection-oriented technology in the sense that an application first has to establish virtual connections between the sources and the destinations prior to the actual transmission of packets (called as cells in ATM). While setting up the virtual connections, the application

* 서울대학교 컴퓨터공학과(yjin@nnlab.sun.ac.kr)
논문번호 : 98339-0803, 접수일자 : 1998년 8월 3일

www.dbpia.co.kr

specifies the QoS requirements. The paths for the connections are selected and resources are allocated, based on the requested QoS and the available network resources. Thus ATM is capable of providing guaranteed services to the applications. ATM is designed to allow different applications with different QoS requirements to coexist and share the same underlying network. In addition, multicasting, which manages network resources(bandwidth etc.) efficiently in comparison with the repeated unicasting, is as part of ATM service.

However, the ATM layer itself only provides the means for fast end-to-end transmission of packets. Efficient network layer mechanisms are needed in order to benefit from the services provided by ATM. Currently, much work is underway to develop such network layer mechanisms which include routing, resource reservation, admission control and flow control.

In the past, the routing problem in communication networks was simple. The applications utilized a modest percentage of the available bandwidth and none of them had QoS requirements. In addition, very few applications involved more than two users. That is why simple routing techniques[1, 3, 4, 5] were sufficient in the past. The situation is different, however, for the emerging realtime applications discussed above. These applications are usually bandwidth-intensive, have QoS requirements, involve more than two users, and they are already available over current networks. For example, the Internet MBONE service and a popular conferencing tool, which are based on DVMRP (Distance Vector Multicast Routing Protocol)[5, 15] already use the multicasting support recently added to the Internet. Even though these schemes exhibit simplicity, they do not provide QoS guarantees for realtime multimedia applications. Thus the routing problem for realtime multimedia applications is more complex than that of the past. Future applications will also rely on the network capability to perform multicast communications. Thus, multicasting will be an essential part of future networks. This paper focuses primarily on routing mechanisms for multicast communication over high speed, connection-oriented, wide-area networks carrying realtime multimedia traffic with QoS requirements.

To handle large number of multicast sessions, a network must minimize the sessions's resource consumption, while meeting their QoS requirements. Algorithms are needed in the network to compute multicast routing trees; we call such algorithms multicast algorithms. Different optimization goals can be used in multicast algorithms to define what constitutes a good tree. One such goal is providing the minimum delay from source to destinations along the tree, which is important for delay-sensitive multimedia applications, such as videoconferencing. Another optimization goal is constructing the minimum cost tree, which is important in managing network resources efficiently. Tree cost normally refers to the amount of network resources needed to transport packets over the tree. Therefore, minimizing the tree cost is equivalent to the efficient use of network resources. Efficient multicast routing algorithms are essential for services spanning wide areas and involving large number of nodes in order to optimize the utilization of the network resources. However, finding the minimum cost route is different from finding QoS constrained route. In the past, several QoS-constrained multicast algorithms have been proposed[7, 8, 9]

Route computation can be done in two different ways: centralized and distributed. In the centralized one, also called source routing, one node which is aware of the status of the whole network computes the route. The computation is easy and fast in most centralized schemes. However, the overhead to maintain the whole network status in the route computing node can be very large. In the distributed scheme, on the contrary, each network node participates in the route computation. The route is generated by exchanging messages between nodes that have only partial knowledge of the network status. The distributed scheme is slow and complex, but it need not maintain the whole network status in each node. Most of the existing multicast algorithms[7, 8, 9] are centralized ones and some of them can not be applied to large scale networks due to its high complexity.

Here, we are interested in a minimum cost distributed route computation algorithm that satisfies the end-to-end delay requirement, which is specified by the application. Multicast algorithms that perform cost

www.dbpia.co.kr

optimization have been based on computing the minimum Steiner tree in a graph. The problem of finding the minimum Steiner tree is known to be NP-complete[10] and a number of heuristics have been developed to solve this problem in polynomial time and producing near-optimum results. The minimum cost tree with delay bound contraint is a special case of the minimum cost tree, thus it is also an NP-complete problem. The proposed algorithm is therefore a heuristic one.

In section Ⅱ the previous works on this issue are presented. Section Ⅲ describes the network model and problem definition, and section Ⅳ presents the proposed distributed multicast algorithm. In section Ⅴ simulation model and performance results obtained by simulation are presented. We conclude the paper in section Ⅵ.

Ⅱ. Related Works

Most of the previous delay-constrained multicast algorithms are centralized ones[7, 8, 9] Few distributed delay-constrained multicast algorithms have been presented in the literature[12] One of the simplest distributed algorithms is the shortest path tree(SPT). The multicast tree for the SPT consists of the least delay path from a source to all destinations in the multicast group. The shortest paths are established using the existing unicast routing algorithm, such as Dijstra's shortest path algorithm. When a new destination D joins a multicast group, the source S determines the least delay path from S to D. If part of this path from S to a node K is already in the multicast tree, the multicast tree needs only be extended by the least delay path from K to D. While SPT minimizes the delay, it does not try to minimize the total cost of the multicast tree.

In[12], distributed multicast algorithm, called WAVE algorithm, was presented for finding a delay-constrained multicast tree for static and dynamic multicast groups. In this algorithm, when a node D wants to join a multicast group, it sends a request Req to the source S. Starting from S, this request is propagated throughout the multicast tree and answered (Rsp) by nodes that receive that request. Each node that receives such a message can compare the requested QoS(cost, delay) with the QoS characteristics of the path taken by this message. The message will only be forwarded if the path taken so far meets the QoS requirement. Each response received by D has the form Rsp = (n_{id}, *cost, delay*), where n_{id} denotes the node that generated the response, the *cost* represents the cost of the path from n_{ia} to D, and the *delay* denotes the path delay from S to D via n_{ia}. For each response Rsp, the destination computes a weighted cost $WC(n_{id})=w_c \times (cost/max\_cost)+w_d \times (delay/max\_delay)$, where $w_c \in [0, 1]$, $w_d \in [0, 1]$ and *max_cost* and *max_delay* are maximum cost and delay values over all received responses. The destination calculates the weighted costs for all responses and selects the node with the minimal weighted cost. The selection of appropriate w_c and w_d value allows the receiver to trade-off cost versus delay. Howerver, WAVE algorithm does not support a delay constraint explicitly.

In[8], near-optimal centralized multicastalgorithm called BSMA(bounded shortest multicast algorithm) was presented. The algorithm starts by computing the least delay tree for a given source and multicast group. Then it iteratively replaces superedges[1] in tree with lower cost superedges not in the tree, without violating the delay bound, until the total cost of the tree can not be reduced any further. Although BSMA is a centralized algorithm with near-optimal performance regarding the resulting tree cost, it can not be applied to real networks due to its high time complexity. In our simulation experiments, we will show the performances of the distributed algorithms relative to BSMA.

Ⅲ. Network Model and Problem Definition

A network is modeled as follows:

- A network is represented by a directed graph $G=(V,E)$, where V is a set of nodes and E is a set of links. The existence of a link $e = (v, w)$

1 A superedge is a path in the tree between two branching nodes or two multicast group members or between a branching node and a multicast group member.

www.dbpia.co.kr

from node v to node w implies the existence of a link $e'=(w, v)$ for any $v, w \in V$.

- Weighting functions for cost and delay respectively exist on each link and have non-negative real values.

$$C(e): E \rightarrow R^{+} \quad (1)$$

$$D(e): E \rightarrow R^{+} \quad (2)$$

A link's cost is a measure of that link's resource utilization. Thus cost should be a function of the amount of traffic traversing the link and expected buffer space needed for that traffic. A link's delay is the delay data packet experiences on that link (sum of switching, queueing, transmission and propagation delays). Links are asymmetrical, namely the cost and delay for the link $e = (v, w)$ and the link $e' = (w, v)$ may not the same.

A multicast group $G=\{g_1, \cdots, g_n\} \subseteq V$ is a set of nodes participating in the same session and is identified by a unique group address. A node $s \in V$ is a multicast source for the multicast group. A multicast source s may or may not be itself a member of the group G. A multicast tree $T(s,G) \subseteq E$ is a tree rooted at the source s and spanning all members of the group G. The total cost of a tree $T(s,G)$ is simply $\sum_{e \in T(s,G)} C(e)$. A path $P(s,g) \subseteq T(s,G)$ is a set of links connecting s to $g \in G$. The cost of the path $P(s,g)$ is $\sum_{e \in P(s,g)} C(e)$ and the end-to-end delay along that path is $\sum_{e \in P(s,g)} D(e)$.

The problem of finding the multicast tree satisfying the end-to-end delay bound can be formulated as follows.

Given $G=(V,E)$, $C(e)$, $D(e)$, source node s, multicast group $G \subseteq V$ and a delay bound Δ, minimize the tree cost $\sum_{e \in P(s,g)} C(e)$, where $T(s,G)$ is a multicast tree rooted at s and spanning all of the nodes in G such that for each node g in G, $\sum_{e \in P(s,g)} D(e) < \Delta$, and $P(s,g)$ is the unique path in $T(s,G)$ from s to g,

When Δ is ∞, the delay-constrained multicast routing problem reduces to the Steiner tree problem[10] and therefore NP-complete. When the multicast group size $|G|=1$, it reduces to the delay-constrained unicast routing problem and was shown also to be NP-complete. A heuritstic solution to the delay- constrained multicast routing problem should always find a constrained multicast tree, if one exists.

Ⅳ. DDCMT Algorithm

1. Routing Information

In the distributed route computation, the overhead of routing table management in a node is small because only the partial knowledge of the network status is sufficient to carry out the algorithm at each node. On the contrary, the computation time takes longer than that of the centralized algorithm since many nodes in the network participate in the route computation by exchanging messages repeatedly over the links. The proposed algorithm is also distributed one. We have enhanced the $DCMT_{md}$[11] algorithm developed by authors and call the proposed algorithm DDCMT (Distributed Delay-Constrained Multicast Tree).

In this section, we first discuss the routing information which needs to be present for the proposed algorithm to compute the delay-constrained multicast tree. The routing information of the proposed algorithm is similar to that of [13]. Every node $v \in V$ should know the delays of all outgoing links and must maintain the following information during the route computation: a delay vector and a cost vector.

The delay vector at node $v \in V$ consists of $|V|-1$ entries, one entry for each other node w in the network. The entry for node $w \in V (v \neq w)$ holds the following information:

- the destination node ID: w
- the end-to-end delay of the least delay path $P_{ld}(v,w)$ from v to w: $D[P_{ld}(v,w)]$
- the cost of the least delay path $P_{ld}(v,w)$
- from v to w: $C[P_{ld}(v,w)]$
- the ID of the next hop node on the least delay path $P_{ld}(v,w)$ from v to w: $N[P_{ld}(v,w)]$

The cost vector at node $v \in V$ also consists of $|V|-1$ entries, one entry for each other node w in the network. The entry for node $w \in V (v \neq w)$ contains the following information:

- the destination node ID: w
- the end-to-end delay of the least cost path $P_{lc}(v,w)$

www.dbpia.co.kr

from v to w: $D[P_{lc}(v,w)]$
- the cost of the least cost path $P_{lc}(v,w)$ from v to w: $C[P_{lc}(v,w)]$
- the next hop node on the least cost path $P_{lc}(v,w)$ from v to w: $N[P_{lc}(v,w)]$

The cost vectors and delay vectors are similar to the distance vectors of existing routing protocols[2]. Distance-vector based protocols treat carefully about how to update the distance vectors in response to topology changes, and how to prevent instability. These procedures are simple and require the contents of the distance vector at each node to be periodically transmitted to neighboring nodes. The same procedure can be used for maintaining the cost vectors[2] and delay vectors. Thus existing distance-vector based protocols[2] can be modified to maintain the routing information of the proposed algorithm. We assume that the cost vectors and delay vectors at all nodes are up-to-date. We also assume that the link costs, the link delays, the contents of the cost and delay vectors do not change during the route computation.

2. The Proposed Algorithm

DDCMT algorithm operates in two different modes: static and dynamic modes. The static mode is used when group membership is fixed or changes infrequently, and dynamic mode is executed when nodes join or leave a multicast group dynamically. The operations of dynamic mode are subset of those of static mode. We first describe the static mode operations and later add some comments for the dynamic mode.

In static mode, the source node s sends TEST_METRIC message to nodes already included in the tree. At first, the message is sent to the source node itself. Upon receiving TEST_METRIC, each node v verifies its cost vector and delay vector to see if there exist delay-constrained paths to the destination nodes not included in the tree (node v is aware of the delay from the source to itself from the previous steps and calculates the total delay to destination w by adding $D[P_{ld}(v,w)]$ or $D[P_{lc}(v,w)]$ value for the destination to the source-to-itself delay).

At first, node v checks Eq. (3) for each destination node w not included in the tree. Node v is aware of the delay from the source s to itself, $\sum_{e\in P(s,g)} D(e)$, from the previous steps. We define $COST_{min}(v,w)$ as the cost of the least delay path or least cost path from node v to new destination w. If Eq. (3) is satisfied, $COST_{min}(v,w)$ is set to $C[P_{lc}(v,w)]$. Otherwise it checks Eq. (3) and if it is satisfied, $COST_{min}(v,w)$ is set to $C[P_{ld}(v,w)]$.

$$\sum_{e\in P(s,v)} D(e) + D[P_{lc}(v,w)] < \Delta \quad (3)$$

$$\sum_{e\in P(s,v)} D(e) + D[P_{ld}(v,w)] < \Delta \quad (4)$$

If both Eq. (3) and (4) are not satisfied, no path exists that can satisfy the given delay constraint from source s to w, therefore, $COST_{min}(v,w)$ is set to ∞. After node v finds $COST_{min}(v,w)$ for all destinations not included in the tree, it selects the one with minimum $COST_{min}(v,w)$ among the paths satisfying source-to-destination delay bound. And the related information is carried back to the source node s in TEST_METRIC_ACK message. At this time, if $COST_{min}(v,w)$ is equal to ∞ it is not necessary to send the TEST_METRIC_ACK message to the source node.

Before sending the TEST_METRIC_ACK message to the source node, each tree node collects all TEST_METRIC_ACK messages from its children nodes and selects one with the least $COST_{min}(v,w)$ among them including its own one.

Then, the selected TEST_METRIC_ACK message is propagated to the next tree node on the path to the source node. This message merging is repeated at every tree node on the path to the source node. Therefore, the number of TEST_METRIC_ACK messages received by the source node is equal to the number of multicast branches it has.

In the source node, it collects TEST_METRIC_ACK messages from its children nodes, and selects one with the least $COST_{min}(v,w)$. And then the node that sent the selected information is notified by PATH_FIND message. The selected message may have been received from the source node itself.

2 Note that the procedures for maintaining the cost vectors should give the end-to-end delay (instead of the cost) of the least cost path)

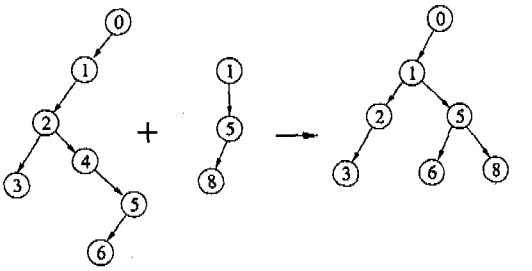
www.dbpia.co.kr

```
/* When a node v receives a PATH_FIND message */
   Existing tree   New path from 1 to 8   Resulting tree
                s=0, G={3, 6, 8}, v-5, V_D=4, w=8
if (delay + D[P_lc(v,w)]) < Δ then
   next_node ← N[P_lc(v,w)];
   path_dir ← LC;
   delay ← delay + D(v, N[P_lc(v,w)]);
else
   next_node ← N[P_ld(v,w)];
   path_dir ← LD;
   delay ← delay + D(v, N[P_ld(v,w)]);
endif
PATH_SETUP ← (w, path_dir, Δ, delay);
send the PATH_SETUP message to next_node;

/* When a node v receives a PATH_SETUP message */
if v ≠ w then
   if path_dir in PATH_SETUP = LC or
   delay in PATH_SETUP + D[P_lc(v,w)] < Δ then
      next_node ← N[P_lc(v,w)];
      path_dir ← LC;
      delay ← delay + D(v, N[P_lc(v,w)]);
   else
      next_node ← N[P_ld(v,w)];
      path_dir ← LD;
      delay ← delay + D(v, N[P_ld(v,w)]);
   endif
   PATH_SETUP ← (w, path_dir, Δ, delay);
   send the PATH_SETUP message to next_node;
else
   send the PATH_SETUP_ACK message to the source
endif
```

Fig. 1 The procedures to find a path from node v to destination node w

Upon receiving PATH_FIND message, node v constructs a path one node at a time in the similar manner described in [13] from the node v to the new destination node w. If Eq. (3) is satisfied, node v reads the next hop node on the least cost path towards w, namely $N[P_{lc}(v,w)]$, from its cost vector and sends a PATH_SETUP message with the information, as depicted in Fig. 1, to the next hop node. Otherwise, node v reads the next hop node on the least delay path towards w, namely $N[P_{ld}(v,w)]$, from its delay vector and sends a PATH_SETUP message with appropriate information, shown in Fig. 1, to the next hop node. When a node $v \neq w$ receives a PATH_SETUP message, it executes the procedure shown in Fig. 1. The nodes on the path to the added destination w are also included in the multicast tree, and participate in the subsequent tree computation steps. And each intermediate node to a new destination stores the accumulative delay, which is the *delay* field in Fig.1, from the source node to itself. The destination node w replies back by PATH_SETUP_ACK message when it receives a PATH_SETUP message. The source node repeats the whole procedure until there are no destinations left.



Existing tree New path from 1 to 8 Resulting tree
s=0, G={3, 6, 8}, v-5, V_D=4, w=8

Fig. 2 An example of loop scenario

Note that loops may appear in constructing a path to a new destination. Loops can be simply detected by checking if the next hop node, $N[P_{lc}(v,w)]$ or $N[P_{ld}(v,w)]$, has a routing table entry for the current source and multicast group during the route construction. Upon receiving PATH_SETUP message, node v checks if it has a routing table entry for the current source and multicast group. Then node v sends a PRUNE message to its parent node v_p. Upon receiving PRUNE message, node v_p checks if it is neither a multicast group member nor an intermediate node, which leads to any multicast group member, after removing link (v_p,v). Then node v_p removes the routing table entry for the current source and multicast group, and sends a PRUNE message to its parent node. The same procedure is repeated on the path towards the

www.dbpia.co.kr

source node. Fig. 2 shows a loop scenario. Because the new path (1,5,8) should be included in the resulting tree, some of the links in existing tree must be removed when loop appears. In this case link (4,5) must be removed. Note that link (2,4) must also removed, because node 4 is neither a multicast group member nor an intermediate node that leads to any other multicast group member after removing link (4,5).

In the dynamic mode of DDCMT algorithm, when a node w wants to join a multicast group, it sends a JOIN message to the source node s When the source node s receives a JOIN message, it sends a TEST_METRIC message with *joining node identifier* (w) to the nodes already included in the tree. If the TEST_METRIC message is propagated throughout the multicast tree, the message will only be forwarded if the path delay taken by the message meets the end-to-end delay requirement. A tree node v, upon receiving a TEST_METRIC message, finds the delay-constrained minimum cost path to the node w and replies by TEST_METRIC_ACK message to the source node. The next steps are the same as those of static mode. When a node w want to leave a multicast group, if it is a leaf node in the multicast tree, node w sends a PRUNE message to its parent node.

We now present the formal algorithm. The notations used in the algorithm are:

- s: source node
- G: set of destination nodes, where $G \subseteq V$
- T: set of nodes in the multicast tree
- $COST_{min}(v,w)$: cost of the least delay path or least cost path from node v to node w
- $DELAY_T(v)$: delay from node s to node v in the multicast tree T
- Δ: end-to-end delay bound

Step 1: s sends TEST_METRIC to $v \in T$, where T is the set of the nodes already included in the tree and it initially includes source node s only.

Step 2: Node $v \in T$, upon receiving TEST_METRIC, determines $COST_{min}(v,w)$ for each $w \in G-T$. If $DELAY_T(v)+D[P_{lc}(v,w)] < \Delta$, $COST_{min}(v,w) = C[P_{lc}(v,w)]$. Otherwise, if $DELAY_T(v)+D[P_{ld}(v,w)] < \Delta$, $COST_{min}(v,w) = C[P_{ld}(v,w)]$. Otherwise $COST_{min}(v,w) = \infty$. After selecting w with the minimum $COST_{min}(v,w)$ for $w \in G-T$, node v constructs TEST_METRIC_ ACK.

Step 3: Before TEST_METRIC_ACK is sent to s, $v \in T$ collects all TEST_METRIC_ACK's from its children nodes, and selects one with the least $COST_{min}(x,y)$ among them including v's own one, where $x \in T$ and y is a destination node which is not included in the multicast tree. The selected message is sent to s and other messages are discarded by v.

Step 4: s selects the node whose $COST_{min}(v,w)$ in the TEST_METRIC_ACK is minimum, then sends PATH_FIND} to v. The newly added destination is also notified to each $x \in T-\{s\}$.

Step 5: Node $v \in T$, upon receiving PATH_FIND}, checks if $DELAY_T(v)+D[P_{lc}(v,w)] < \Delta$. If it is satisfied, v sends PATH_SETUP} to $N[P_{lc}(v,w)]$. Otherwise it sends PATH_SETUP to $N[P_{ld}(v,w)]$.

Step 6: Node x, upon receiving PATH_SETUP, forwards PATH_SETUP to the next node y, $N[P_{lc}(x,w)]$ or $N[P_{ld}(x,w)]$, determined by the procedure shown in Fig.1. If $x \in T$, x sends PRUNE to the next hop node along the path to s. x is included in T.

Step 7: Node $v \in T$, upon receiving PRUNE, checks if v is a leaf node, $v \notin G$ and $v \neq s$. Then, v is excluded from T and sends PRUNE to the next hop node on the path to s.

Step 8: Destination node w, upon receiving PATH_SETUP, sends PATH_SETUP_ACK to s indicating that w is included in the tree.

Step 9: The algorithm is finished when $G \subseteq T$. Otherwise go to Step 1.

It should be noted, however, that a TEST_METRIC message need not be sent to all tree nodes at every cycle. For tree nodes which didn't have the newly added destination node in their previous TEST_METRIC_ACK messages, TEST_METRIC need not be repeated since we would get the same answers. The above policy and message merging scheme, which was described in Step 3 of the formal algorithm, contribute largely to make the algorithm a scalable one in large network with large multicast group.

Property: *DDCMT always finds a multicast tree when*

www.dbpia.co.kr

there exists one that satisfies the delay bound \end{theorem}

Proof: Assume that DDCMT cannot find a delay-constrained path to a node $w \in G$, although there exists one. Then $D[P_{id}(s,w)] > \Delta$ since $D[P_{id}(v,w)] \leq D[P_{ic}(v,w)]$ and $DELAY_T(v)+D[P_{id}(v,w)]>\Delta$ for $\forall v \in T$. This is in contradiction to the assumption that there exists a delay-constrained path from s to w. Therefore DDCMT always finds a delay-constrained tree when there exists one.

Property 2 *The worst case message complexity of DDCMT is $O(|V|^2)$, where $|V|$ is the number of network nodes.*

Proof: The computational complexity of the proposed algorithm at any node is $O(1)$, because each time a node receives messages, it performs a fixed amount of computation, irrespetive of the size of the network. The number of messages needed to construct a path for a given source and a new destination is proportional to the number of links in the tree and the number of links in the path from a selected tree node to a new destination, because a node running DDCMT exchanges at most three messages. For a network size of $|V|$ nodes, the maximun number of tree links is $|V|$-1 and the longest possible path from a selected tree node to a new destination node consists of $|V|$-1 links. Therefore the number of messages needed in the worst case is $O(|V|)$. Because the above procedure must be repeated for each group member and the maximum multicast group size is $|V|$-1, the number of messages needed to construct a tree is $O(|V|^2)$ in the worst case.

The previous works on distributed multicast algorithm\cite[11, 12] have the same message complexities as DDCMT. BSMA[8], which is a centralized algorithm, has computational complexity of $O(k|V|^3 log|V|)$, where k may be very large in case of large, densely connected networks, and it may be difficult to achieve acceptable running times.

V. Performance Evaluation

We used simulation for our experimental investigations to avoid the limiting assumptions of analytical modeling. The multicast routing simulation environment described in[14] was modified to evaluate and compare performances of DDCMT, modified WAVE algorithm[12], shortest path tree algorithm (called SPT) and near-optimal centralized algorithm (called BSMA)[8]. In this simulation, full duplex ATM networks with homogeneous link capacities of 155 Mbps(OC3) were used. The positions of the nodes were fixed in a rectangle of of size 3000×2400 Km^2, roughly the area of the United States. A random graph generator based on Waxman's generator\cite[6], which is known to generate graphs similar to real networks, was used to create networks. The generator first creates a list of nodes at random locations and then creates links between these nodes. The probability of a link to exist between two nodes v and w is given by

$$P_e(v,w) = \beta \exp \frac{-d(v,w)}{(\alpha L)} \qquad (5)$$

where $d(v,w)$ is the distance between node v and w, L is the maximum possible distance between two nodes on the plane, and α and β are parameters in the range $0 < \alpha, \beta \leq 1$. We can control the degree and connectivity pattern of the generated graphs by setting appropriate values to α and β. The probability of link existence between remote nodes increases as α gets larger, and the degree of generated graph increases as β gets larger. Link delay is set to be linearly proportional to the distance between two end nodes of the link. The simulated network is always conntected and has an average degree of 4 through adjusting the parameters.

Each node in the network represents a non-blocking ATM switch. Each link is assumed to have a small output buffer. The propagation speed through the links is assumed to be two thirds of the speed of light. The propagation delay was dominant under these conditions, and the queueing component was neglected when calculating the link delay. For the multicast sources we used variable bit rate(VBR) video sources. Any session traversing a link reserved a fraction of the link bandwidth equal to the equivalent bandwidth of the traffic it generated. The link cost was taken equal to the reserved bandwidth on that link, because it is a suitable measure of the utilization of both the link's bandwidth and its buffer space. Therefore, the cost of a heavily utilized link was larger than that of a lightly

www.dbpia.co.kr

utilized link. A link can be used in a new multicast session and reserve bandwidth for it, only if the sum of the total currently reserved bandwith on the link and the newly requested equivalent bandwidth is less than 85% of the link's capacity. This simple admission control policy is sufficient for the purpose of our study of the multicast routing algorithms.

Realtime applications have tight delay requirements. We used a value of 30 *ms* for delay constraint which represents only an upper bound on the end-to-end propagation delay across the networks. This relatively small value was chosen in order to allow the higher level end-to-end protocols enough time to process the transmitted information without affecting the quality of the application.

When we consider the case that a multicast group evolves dynamically by nodes joining or leaving we use the function $PD(k)$ introduced by Waxman[6].

$$P_D(k) = \frac{\gamma(n-k)}{r(n-k)+(1-\gamma)k} \qquad (6)$$

where n is the number of nodes in the network, k is the current number of group members in the multicast tree, and γ is a parameter between (0,1). γ represents the ratio of the number of group members to the number of network nodes at equilibrium. For example, when $\gamma = k/n$, $P_D(k)=1/2$. To determine whether the next group modification will be a join or leave, we compute a random number r, $0 \le r < 1$, to compare with $P_D(k)$. If $r > P_D(k)$, the modification is leave and a randomly chosen group member will leave the multicast group. For $r \le P_D(k)$, the modification is join and a node is randomly selected as a new group member.

1. Simulation Results

For each run of the experiments, we generated a random set of links to interconnect the fixed nodes and generated random background traffic for each link. And then we selected a random source node and a multicast group of randomly chosen destination nodes. The equivalent bandwidth of each link's background traffic is a random variable between B_{min} and B_{max}. As the range of the link loads, i.e., the difference between B_{min} and B_{max}, increases, the asymmetry of the link loads also increases. The routing algorithm was applied to create a constrained multicast tree for a random source generating video traffic with an equivalent bandwidth of 0.5 Mbps and a group of randomly chosen destinations. The experiment was run repeatedly until confidence intervals of less than 5%, using 95% confidence level, were achieved for the measured quantities. BSMA[8] is a centralized delay-constrained multicast algorithm which has a near optimal performance regarding resulting tree cost. Therefore, we will show the percentage increase in the total cost of the distributed algorithms relative to that of BSMA.

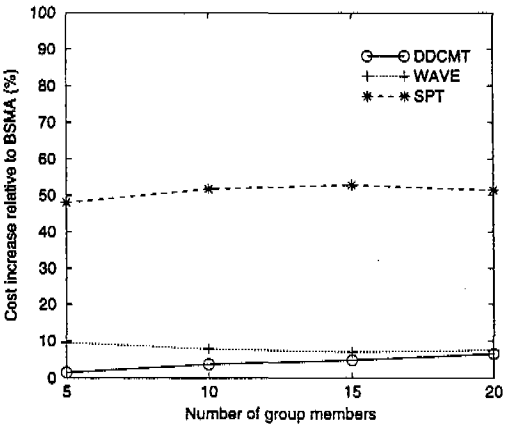


Fig. 3 Tree cost relative to BSMA, 20 nodes, average degree=4, Bmin=10Mb/s, Bmax=120Mb/s, Δ =30ms

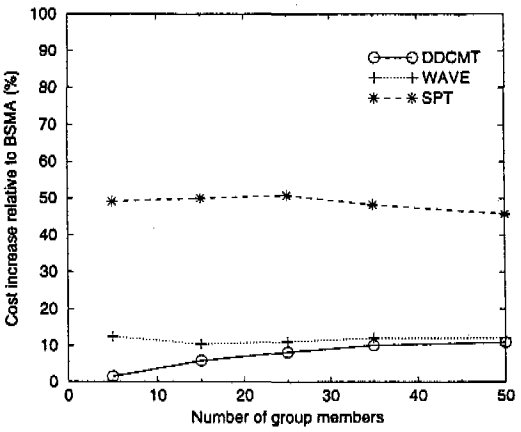


Fig. 4 Tree cost relative to BSMA, 50node, average degree=4, Bmin$=10Mb/s, Bmax=120Mb/s, Δ =30ms

Although WAVE algorithm does not support delay constraint explicitly, we modified it by adding

www.dbpia.co.kr

end-to-end delay bound concept and set w_c=1, w_d=0 in the weighted cost function of WAVE algorithm for fair comparison. The modified WAVE algorithm has performances equivalent to [11].

We first consider scenarios where a multicast group is static, i.e., a multicast tree is constructed for a fixed multicast group of which membership does not change. Fig. 3 shows the percentage increase in the total cost of DDCMT, WAVE and SPT relative to BSMA, for the multicast group size of 20-node networks.

The costs of DDCMT and WAVE are approximately 7% and 10% worse than BSMA respectively, when the mulcast group size is equal to the network size, which reduces to a spanning tree. The cost performance of SPT is significantly worse (over 50%) than BSMA. For a practical multicast group size, which is less than 10% of the number of nodes, DDCMT has a much better performance compared to two other algorithms. Fig. 4 and 5 shows the cost performance of the algorithms when applied to 50-node and 100-node networks respectively. Comparison with Fig. 3 indicates that the cost performance remains approximately unchanged as the network size increases. Fig. 6 shows the cost performance of the algorithms versus the network size for a fixed multicast group of five members. The costs of DDCMT and WAVE are approximately 3% and 12% worse than BSMA respectively. The cost performance of SPT is significantly worse (over 50%) than BSMA. The performance of DDCMT approaches that of BSMA, which is a centralized algorithm with near-optimal performance. And DDCMT has a superior performance compared to other two algorithms for all network sizes. Fig. 7 shows the cost performance of the algorithms versus the delay bound for a fixed multicast group of 10 members, when applied to 100-node networks. For the very stringent delay bound of 20 *ms* (a value close to the largest propagation delay between the nodes in the network), we observes that the costs of DDCMT, WAVE and SPT are approximately 4%, 9% and 30% worse than BSMA respectively. Under this very small delay bound condition the probability that the delay of the least cost path satisfies the delay bound is small, therefore DDCMT's cost performance is slightly worse than BSMA. As the delay bound increases, the cost of SPT increases abruptly and that of WAVE changes slightly, wherease that of DDCMT approaches the cost of BSMA. Therefore, DDCMT exhibits much better performance than WAVE and SPT in all circumstances, which include various network sizes, multicast group sizes and delay bounds.

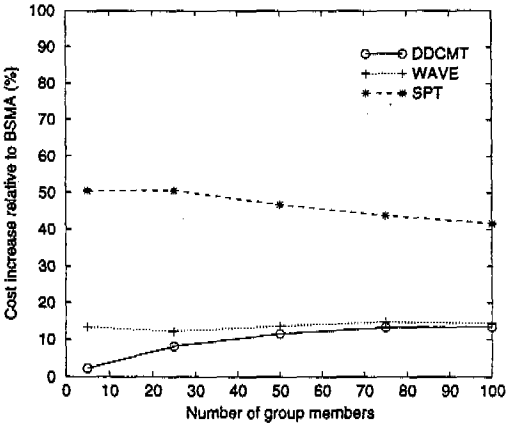


Fig. 5 Tree cost relative to BSMA, 100 nodes, average degree=4, Bmin=10Mbs, Bmax=120Mb/s, Δ =30ms

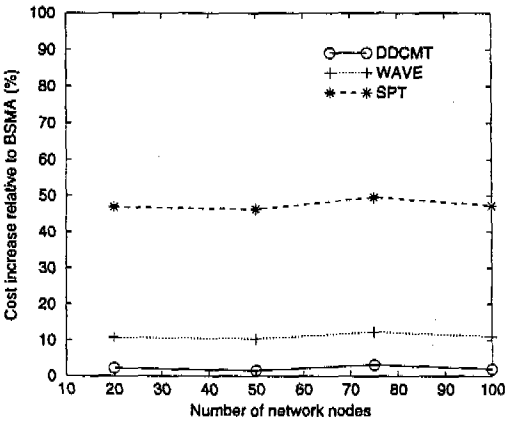


Fig. 6 Tree cost relative to BSMA, variable network size, number of group members=5, average degree=4, Bmin=10 Mb/s, Bmax=120 Mb/s, Δ =30ms

In many cases, a multicast group changes because new destinations join or leave the multicast group during the multicast session. A multicast algorithm should be able to allow for changes in the multicast group without disrupting the communications between the source and existing destinations of the multicast group. Most of the centralized multicast algorithms

www.dbpia.co.kr

does not meet this requirement. Any changes in the multicast group membership will require to recompute the complete multicast tree.

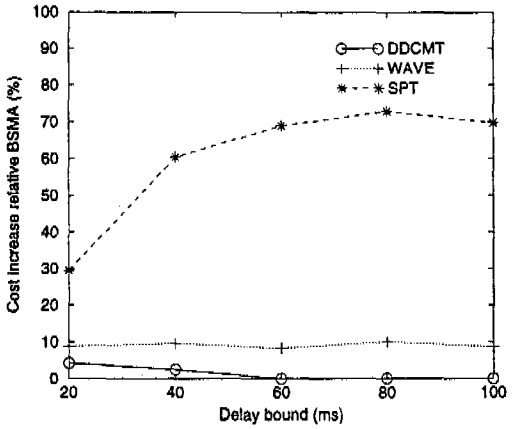


Fig. 7 Tree cost relative to BSMA, 100 nodes, number of group members=10, average degree=4, Bmin=10Mb/s, Bmax=120 Mb/s, variable delay bound (△)

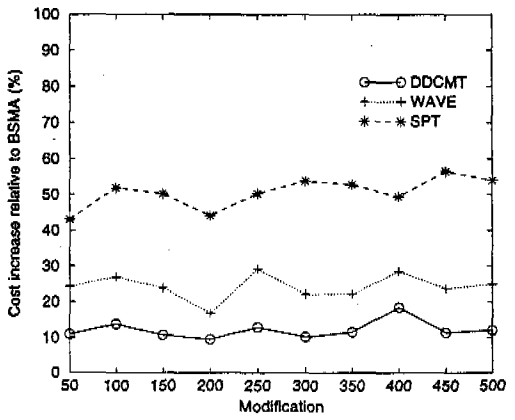


Fig. 8 Tree cost relative to BSMA, 100 nodes, number of group members at equilibrium=5, average degree=4, Bmin=10 Mb/s, Bmax=120Mb/s, △=30 ms

Changes in the multicast tree therefore affect the existing members. To compare distributed multicast algorithms with respect to the cost increase relative to BSMA, we recompute the multicast tree using BSMA after 50 multicast group modifications. The cost of the multicast tree obtained for BSMA was then compared with the cost of multicast tree for DDCMT, WAVE and SPT that was dynamically evolving with each modification. When interpreting the results, we therefore keep in mind that this comparison is in some respect unfair towards DDCMT, WAVE and SPT because BSMA is not able to smoothly grow the multicast tree each time a change in the group occurs. Fig. 8 and 9 shows the cost increase of DDCMT, WAVE and SPT relative to BSMA, for the multicast group of 5 and 20 members repectively at equilibrium, γ=0.05 and 0.2 in Eq. (6), when applied to 100-node networks. DDCMT exhibits much better performace than WAVE and SPT during all modifications.

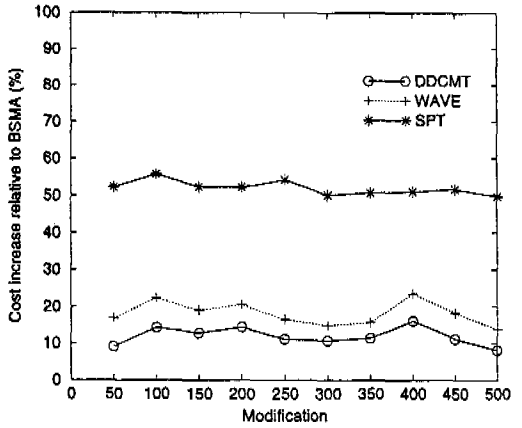


Fig. 9 Tree cost relative to BSMA, 100 nodes, number of group members at equilibrium=20, average degree=4, B_{min}=10 Mb/s, B_{max}=120 Mb/s, △ =30ms

VI. Conclusions

Multicast routing algorithms satisfying stringent delay bound are becoming increasingly important. We proposed a new distributed multicast route computation algorithm that finds a least cost tree under an end-to-end delay bound. The proposed algorithm requires limited network state information at each node: cost and delay vector. We proved that it can always construct a delay-constrained multicast tree, if one exists, and it has a relatively small message complexity, $O(|V|^2)$. We verified the algorithm's performance by simulation. Compared with the previous works on the distributed multicast algorithms, it is shown that the proposed algorithm exhibits much better performance in all circumstances. There are, however, a number of remaining research topics in the area of QoS-constrained multicast routing. Algorithm consi-

www.dbpia.co.kr

dering bandwidth requirement, delay variation of the multicast connection as well as the delay constraint is required. And for multimedia applications, it will be necessary to have a multicast routing algorithm that finds multiple parallel multicast connections with related QoS parameters (example: multipoint multimedia conferencing) in a dynamically changing network topology environment.

References

[1] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Trans. on Computer Systems*, vol. 8, no. 2, pp. 85-110, May. 1990.

[2] C. Hedrick, "Routing Information Protocol," *Internet RFC 1058*, Jun. 1988.

[3] J. Moy, "Multicast extension to OSPF," *Internet RFC 1585*, Sep. 1992.

[4] A. Ballardie, P. Francis and J. Crowcroft, "Core Based Tree(CBT) - An architecutre for scalable inter-domain multicast routing," *Proc. ACM SIGCOMM '93*, pp. 85-95, Sep. 1993.

[5] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu and L. Wei, "An architecture for wide-area multicast routing," *Proc. ACM SIGCOMM '94*, pp. 126-135, Aug. 1994.

[6] B. M. Waxman, "Routing of multipoint connections," *IEEE JSAC*, vol. 6, no. 9, pp. 1617-1622, Dec. 1988.

[7] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast routing for multimedia communications," *IEEE/ACM Trans. on Networking*, vol. 1, no. 3, pp. 286-292, Jun. 1993.

[8] Q. Zhu, M. Parsa, and J. J. Garcia-Luna- Aceves, "An iterative algorithm for delay-constrained minimum-cost multicasting," *IEEE/ACM Trans. on Networking*, vol. 6, no. 3, pp. 461-474, Aug. 1998.

[9] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," International Computer Science Institute, University of California at Berkeley, *Tech. Rep. ICSI TR-94-024*, Jun. 1994.

[10] P. Winter, "Steiner problem in networks : a survey," *Networks*, vol. 17, pp. 129-167, 1987.

[11] Yongjun Im, Youngseok Lee, Sunjoo Wi and Yanghee Choi,"Delay constrained distributed multicast routing algorithm,"*Computer Communications*, vol. 20, no. 1, pp. 60-66, Jan. 1997.

[12] E. Biersack and J. Nonnenmacher, "WAVE: A new multicast routing algorithm for static and dynamic multicast groups," *Proc. NOSSDAV '95*, 1995.

[13] Q. Sun and H. Langendorfer, "A new routing algorithm for supporting delay-sensitive applications," *Computer Communications*, vol. 21, no. 6, pp. 572-578, May. 1998.

[14] H. F. Salama, D. S. Reeves and Y. Viniotis, "Evaluation of multicast routing algorithms for real-time communication on high-speed networks," *IEEE JSAC*, vol. 15, no. 3, Apr. 1997.

[15] D. Waitzman, C. Partridge and S. Deering, "Distance vector multicast routing protocol," *Internet RFC 1075*, Nov. 1988.

임 용 준(Young Jun Im) 정회원

1968년 5월 2일생
1987년 3월~1991년 2월 : 서울대학교 컴퓨터공학과 (공학사)
1991년 3월~1993년 2월 : 서울대학교 대학원 컴퓨터공학과 (공학석사)
1993년 3월~현재 : 서울대학교 대학원 컴퓨터공학과 박사과정
<관심분야> 멀티캐스트 라우팅, ATM망 트래픽 제어

최 양 희(Yang hee Choi) 정회원

1955년 7월 27일생
1971년~1975년 : 서울대학교 전자공학과 (공학사)
1975년~1977년 : 한국과학원 전기공학과 (공학석사)
1977년~1979년 : 한국통신기술연구소 전임연구원
1980년~1984년 : 프랑스 ENST 대학교(공학박사)

www.dbpia.co.kr

1981년~1984년 : 프랑스 CNET 연구소 방문연구원
1984년~1991년 : 한국전자통신연구소 책임연구원
1981년~1995년 : 서울대학교 컴퓨터공학과 조교수
1995년~현재 : 서울대학교 컴퓨터공학과 부교수
<관심분야> 멀티미디어 시스템, 고속 통신망

www.dbpia.co.kr