An Algorithm for Multicast with Multiple QoS Constraints and Dynamic Membership

Aiguo Fei and Mario Gerla

Network Research Lab, Department of Computer Science
University of California, Los Angeles, CA 90095, USA
{afei,gerla}@cs.ucla.edu

Abstract. In this paper we present an algorithm to construct low-cost source trees for multicast with multiple QoS constraints and dynamic membership. Assuming the availability of link-state information, a join path is computed for a new joining multicast receiver. An algebraic formulation is introduced to show how to determine if the QoS requirements for a new receiver can be satisfied at an intermediate node along the join path and how to adjust the tree without breaking QoS requirements for existing members if they are not. Our scheme builds multicast tree incrementally and thus supports fully dynamic membership. It also supports heterogeneous receivers seamlessly. Moreover, our algorithm can support any number of arbitrary QoS metrics without assuming any dependencies among them, if they satisfy some normal mathematical property. If implemented in a distributed fashion, our approach doesn't require any node to have explicit knowledge of the multicast tree topology, thus it scales well for multicast of large group. Simulation studies have been carried out to study the behavior of our algorithm and compare its performance with other schemes.

1 Introduction

In multicast[3], data packets are distributed through a tree structure. A central task in multicasting is to build such a distribution tree (the routing problem). Over the years, many multicast routing algorithms and protocols had been proposed and developed. Several routing protocols had been standardized or are in the process of being standardized by IETF[11, 4] for IP networks, and some of them had been deployed and used on the experimental MBone and some Internet Service Providers'(ISP) networks[11].

In recent years, great effort has been undertaken to introduce and incorporate quality of service(QoS) into data communication networks such as ATM and IP networks[6]. Many multicast applications, such as video conferencing, multimedia broadcasting, and distance-learing, are QoS-sensitive in nature, thus they will all benefit from the QoS support of the underlying networks. The new challenge is how to build a multicast tree (or multiple trees) to deliver the multicast data from source (or sources) to all receivers so that QoS requirements satisfied and the cost of the multicast tree(s) and/or network resources demanded are

G. Pujolle et al. (Eds.): NETWORKING 2000, LNCS 1815, pp. 132-143, 2000

minimized. This problem is NP-complete in general[8] as a constrained Steiner tree problem.

A number of multicast routing algorithms that are QoS-aware have been proposed. Most of these algorithms are "static" algorithms in the sense that they need explicit knowledge of all group members[9, 14]. For many such algorithms it is difficult and expensive for members to dynamically join the group since this usually requires a re-computation of the whole tree. Morever, most of them are not designed to handle heterogeneous receivers (different receivers have different QoS requirements), or multiple QoS constraints. On the other hand, a "dynamic" routing algorithm will build the multicast tree through joining of group members one by one. When a new member joins, the routing algorithm doesn't reconstruct the whole tree, instead it would try to connect the new receiver to an existing tree member without affecting existing group members. It might be harder for such type of algorithms to produce a globally-optimized multicast tree (in terms of cost), but they are necessary for some applications that have frequent membership dynamics. We anticipate both types of multicast algorithms are needed to support different types of applications.

In this paper, we propose an incremental algorithm to support multicast with dynamic membership and multiple QoS constraints. Assuming the availability of information about the global network (as in an OSPF environment in IP networks), a join route is computed when a new member joins. Our algorithm specifies how this new member would be connected to the existing multicast tree without breaking QoS requirements for existing members and how the existing tree would be adjusted if necessary. The algorithm tries to minimize the total cost of the tree by computing a low-cost join path for new members utilizing a QoS unicast routing algorithm. It supports arbitrary number of QoS metrics, dynamic group membership and heterogeneous receivers. It also has good scalability if implemented in a distributed fashion.

The rest of this paper is organized as follows. Section 2 presents network model and some related work. Sections 3 introduces classification of QoS metrics and definitions necessary for the presentation of the algorithm. Section 4 describes the algorithm, followed by simulation results in section 5. We conclude our paper with a short summary as section 6.

2 Network Model and Related Work

Most existing QoS multicast routing algorithms have real-time applications in mind and have end-to-end delay as the QoS constraint needed to be satisfied. The network is modeled as a connected graph $G(V, E)$, where V is a set of vertices(nodes) and E is a set of edges(links). For any edge $e \in E$, $d(e)$ is the delay and $c(e)$ is the associated cost. One of the node is a traffic source. End-to-end delay on a path from the source to a destination is the summation of $d(e)$ for all edge e on the path. The goal is to bound the end-to-end delay on paths from the source to all destinations and minimize the total cost of the tree (as a summation of $c(e)$ for all $e \in T$ where T is the multicast tree). In IP networks,

a source(receiver) node would be a router which has end-user computer in its subnet as a traffic source(receiver) of the multicast group. A survey of those algorithms can be found in [2], and some performance evaluation can be found in [12].

Many centralized multiple-metric(e.g. cost and delay) heuristic algorithms are variations of the well-known single-metric algorithms: Shortest Path Tree(SPT) that optimizes delay from source to all destinations, and Minimum Cost Spanning Tree(MCST) that minimizes the total cost of the tree. The one proposed by V. Kompella et al. in [9] is based on Prim's minimum spanning tree algorithm. In this algorithm, a tree grows from the source, when selecting a new node to join the tree, instead of picking the node connecting to the existing tree via a minimum-cost link, it selects a link that minimizes a given selection function. Another centralized heuristic is Bounded Shortest Multicast Algorithms(BSMA[14]), which starts with an SPT and then replaces some paths in the tree with non-tree paths without breaking the delay constraint but of lower cost. Distributed version of Kompella's algorithm can be found in [10].

As mentioned earlier, most of these algorithms have three main drawbacks: they are not designed to support dynamic membership, they don't consider heterogeneous receivers, and they are difficult to be extended to handle multiple constraints (i.e., jitter and loss in addition to delay). These are the problems our algorithm intends to address.

3 QoS Metrics Classification and Definitions

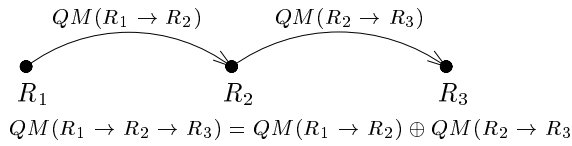

Fig. 1. Concatenation of paths.

Let R_i denote a network node. Consider the concatenation of a path from R_1 to R_2 and a path from R_2 to R_3 to make up a path from R_1 to R_3. Consider a QoS metric, say, QM, which is a function of the path. We have $QM(R_1 \to R_2)$, $QM(R_2 \to R_3)$ and $QM(R_1 \to R_3)$, where $R_i \to R_j$ denotes a specific path from R_i to R_j, and $R_1 \to R_3$ refers to the path concatenated by paths $R_1 \to R_2$ and $R_2 \to R_3$. In general, $QM(R_1 \to R_3)$ is a function of the complete path from R_1 to R_3. However, for some metric QM, $QM(R_1 \to R_3)$ only depends on $QM(R_1 \to R_2)$ and $QM(R_2 \to R_3)$. For example, delay $D(R_i \to R_j)$ is such a metric,

$$D(R_1 \to R_2 \to R3) = D(R_1 \to R_2) + D(R_2 \to R_3). \quad (1)$$

To be general, we can define an operator $\oplus$ on QM such that,

$$QM(R_1 \to R_3) = QM(R_1 \to R_2) \oplus QM(R_2 \to R_3). \tag{2}$$

Delay is a metric that obeys the regular *addition*(+) operation, we call such a metric as an *additive QoS metric*. Number of hops is another example of additive metrics. Generally delay jitter is also considered to be additive. We call a metric a *transitive QoS metric* if it has the *operator* $\oplus$ defined as:

$$t_1 \oplus t_2 = min[t_1, t_2]. \tag{3}$$

or,

$$t_1 \oplus t_2 = max[t_1, t_2]. \tag{4}$$

To simplify our discussion, from now on, we consider only transitive metric defined by the *min* operator, which is often called *concave* metric in literature. Available bandwidth of a path is a transitive metric, e.g., $BW(R_1 \to R_3) = min[BW(R_1 \to R_3), BW(R_2 \to R_3)]$.

We call a metric a *multiplicative QoS metric* if it has the *operaotr* $\oplus$ defined as:

$$m_1 \oplus m_2 = m_1 \times m_2. \tag{5}$$

If one defines reliability r as $r = 1 - \ loss\ rate$, then r is a multiplicative metric. On the other hand, if loss rate L on all link is considered small enough such that $L(R_1 \to R_3) = L(R_1 \to R_2) + L(R_2 \to R_3)$, then *loss rate* L can be considered as an additive metric.

Consider a path from S to D that has a *QoS descriptor* denoted as $QD =< a_1, .., a_{n_a},\ t_1, .., t_{n_t},\ m_1, .., m_{n_m} >$, where a_i is an additive QoS metric for the path, t_j is a transitive metric and m_k is a multiplicative metric; n_a, n_t, n_m are number of additive, transitive and multiplicative metrics respectively. To make a meaningful problem, we assume $a_i \geq 0$ for any additive metric a_i and $0 < m_i \leq 1$ for any multiplicative metric m_i. Consider a *QoS requirement* QR, which can be denoted in the same way as a *QoS descriptor*, $QR =< a_1^{'}, .., a_{n_a}^{'},\ t_1^{'}, .., t_{n_t}^{'},\ m_1^{'}, .., m_{n_m}^{'} >$. We define the $\leq$ operator between QR and QD (and between two QD's) as

$$QR \leq QD = \begin{cases} true & if\ a_i^{'} \geq a_i, t_j^{'} \leq t_j\ and\ m_k^{'} \leq m_k\ for\ all\ i, j, k \\ false & otherwise. \end{cases} \tag{6}$$

We say *a QoS requirement QR is satisfied by the path $S \to D$ with QoS descriptor QD if $QR \leq QD$.*

We can also define the $\oplus$ operator between two QD's as

$$QD \oplus QD^{'} =< .., x_i \oplus x_i^{'}, .. >, i\ runs\ over\ all\ valid\ members, \tag{7}$$

where QoS descriptor $QD =< .., x_i, .. >$ and $QD^{'} =< .., x_i^{'}, .. >$. For a path $R_1 \to R_2 \to R_3$, $QD(R_1 \to R_2 \to R_3) = QD(R_1 \to R_2) \oplus QD(R_1 \to R_3)$.

Consider network nodes S and D and an intermediate node I, the path $S \to I$ has QoS descriptor $QD(S \to I)$ and the path $I \to D$ has QoS descriptor $QD(I \to D)$, we have:

Lemma 1. A QoS requirement QR is satisfied by the path $S \to I \to D$ if $QR \leq QD(S \to I) \oplus QD(I \to D)$.

To simplify future discussion, if $QR \leq QD$, we define a $\ominus$ operator between them as

$$QR \ominus QD =< ..., a'_i - a_i, ..., t'_j, ..., m'_k/m_k, ... >, \tag{8}$$

where $QR =< ..., a'_i, ..., t'_j, ..., m'_k, ... >$, and $QD =< ..., a_i, ..., t_j, ...,m_k, ... >$. The $\ominus$ operator is only defined between a QR and a QD, and the result is a new QoS requirement. It is not defined if $!(QR \leq QD)$.

These operators defined above obey similar rules as regular arithmetic operators, e.g., if $QR \ominus QD_2 \leq QD_1$ then $QR \leq QD_1 \oplus QD_2$, and vice versa. Thus we have:

Lemma 2. Given QoS requirement QR, a path $S \to I$ with $QD(S \to I)$ and a path $I \to D$ with $QD(I \to D)$, QR is satisfied by the path $S \to I \to D$ if $QR \leq QD(I \to D)$ and $QR \ominus QD(I \to D) \leq QD(S \to I)$.

We also introduce a max operator on two QoS descriptors as

$$max[QD, QD'] =< ..., min[a_i, a'_i], ..., max[t_j, t'_j], ..., max[m_k, m'_k], ... >, \tag{9}$$

where QoS descriptor $QD =< ..., a_i, ..., t_j, ..., m_k, ... >$ and $QD' =< ..., a'_i, ..., t'_j, ..., m'_k, ... >$, and $a_i(a'_i)$ is an additive metric, $t_j(t'_j)$ is a transit metric and $m_k(m'_k)$ is a multiplicative metric. Intuitively, if max operator is applied on two QoS requirements, the result is a new QoS requirement which is at least as strict as both of the old requirements for any metric. Similarly we can define a min operator. By definition, we have

Lemma 3. If $max[QR_1, QR_2] \leq QD$, then both $QR_1 \leq QD$ and $QR_2 \leq QD$.

We now make modifications to the multicast tree construction problem in Section 2: instead of delay $d(e)$, each edge $e \in E$ now has QoS characteristics $QD(e)$; for the path (in the multicast tree) from s to a destination t, it has to satisfy the QoS requirement s.t. $QR(t) \leq QD(s \to t)$ where $QD(s \to t) = \sum_{e_i} QD(e_i)$ for all e_i on the path $s \to t$.

4 The Algorithm

4.1 Assumptions and Messages

We assume any QoS metric(requirement) needed to be considered is of one type of those defined above. In this section we describe how our algorithm work in a distributed fashion, the pseudo code for the algorithm can be found in[5]. We also assume the availability of link-state information at each node and a QoS-capable unicast routing algorithm.

Similar to PIM-SM(Protocol Independent Multicast Sparse Mode[4]), we assume a node keeps forwarding information such as source/group, incoming link and set of outgoing link(s). When a new member(receiver) wants to join the group, it sends an explicit **Join request** towards the source. Additionally, in a forwarding entry, each outgoing link is labeled as **active** or **pending**. Multicast traffic is only forwarded to outgoing link(s) marked as active. Two additional types of messages required are **Accept notification** and **Deny notification**. When the Join request is accepted(denied) at some node, an Accept(Deny) notification is sent downstream towards the new receiver. Accept notification will change the corresponding pending flags to active while a Deny notification will clear those pending entries. When a member leaves a group, it sends **Prune message** upstream as in PIM-SM. Additionally, through periodic(or triggered) **QoS Probing messages** from the source (or intermediate nodes), any node in the multicast tree will keep track of QoS characteristics (such as bandwidth reserved, delay, delay jitter and loss, depending on the QoS metrics concerned) on the current path in the multicast tree from the source to that node.

4.2 Algorithm Description

When a new receiver(say, R_3) wants to join the multicast group with QoS requirement $QR(R_3)$, it computes a join path utilizing the available link-state information and a QoS unicast routing algorithm. It then sends a Join message carrying the join path information towards the source.

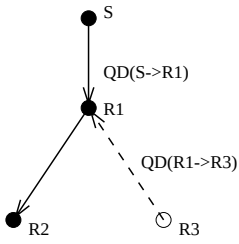

Fig. 2. Join request received by an intermediate node R_1.

An intermediate node will forward the Join message. It also stamps in the message the QoS characteristics of the link from which the message received, so that when a node(say, R_1) receives the Join message from R_3, the QoS characteristics on the reverse direction of the path it traveled so far is available as $QD(R_1 \rightarrow R_3)$. If the node receiving the message is not already in the tree, it creates a new multicast forwarding entry (group/source, in-link, out-link, destination). The out-link is the link from which it receives the message and the in-link is the link to which it should forward the message based on route information carried in the message. It then stamps in the Join message the QoS characteristics for the link from which it received the message and forwards that message to the next hop. It also marks the newly created routing entry as "pending" and starts a timer. The routing entry will be marked as active (route pinning) when an Accept message is received. The Accept message is then forwarded downstream. A "pending" routing entry is flushed if its timer goes off.

When a node receives a Join message and it is already part of the multicast distribution tree, say, node R_1 in Fig.2 (where S is the source), it will check current QoS characteristics $QD(S \rightarrow R_1)$ and QoS requirements $QR(R_3)$ and QoS characteristics $QD(R_1 \rightarrow R_3)$. If $QR(R_3) \leq QD(S \rightarrow R_1) \oplus QD(R1 \rightarrow R_3)$, by Lemma 1, QoS requirement for new receiver R_3 is satisfied by the path

$S \rightarrow R_1 \rightarrow R_3$, then the Join request can be accepted at node R_1. This is as illustrated in Fig.2. In Fig.2 and others to follow, a black node is an in-tree node of the multicast group, non-filled node (R_3) is the new joining receiver and a gray node may or may not be an in-tree node. A solid line represents an existing multicast forwarding path and a dashed line represents the path on which the Join request is sent. Multicast packets will be forwarded on the reverse direction of the dashed line if the Join request succeeds. Under the situation illustrated in Fig.2, the Join request is accepted at router R_1. No additional forwarding is required.

If this is not true, there are two possibilities: (1)the immediate upstream node R_0 of R_1 is the next hop specified by the Join request, (2)the next hop specified in the Join request is not R_0. In the first case, R_1 can simply add the interface from which it received the Join request to the corresponding multicast routing entry and mark it as pending, and then forward the request to R_0, which may accept the Join request or forward it further. This is illustrated in Fig.3.

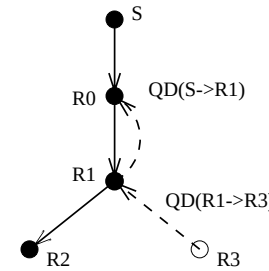

Fig. 3. $!(QR(R_3) \leq QD(S \rightarrow R_1) \oplus QD(R1 \rightarrow R_3))$.

In the latter case, the action to be taken by $R1$ is described as follows, which also has two possibilities: (1)$QD(S \rightarrow R_1) \leq QR(R_3) \ominus QD(R_1 \rightarrow R_3)$, (2)condition (1) is not satisfied. In the first case, let R' be the next hop specified by the Join request, if the Join request is accepted on the new join path $S \rightarrow R' \rightarrow R_1$, then $QR(R_3) \ominus QD(R_1 \rightarrow R_3) \leq QD(S \rightarrow R' \rightarrow R_1)$, thus the path $S \rightarrow R' \rightarrow R_1 \rightarrow R_2$ will also satisfy the QoS requirements for any downstream node R_2 of R_1 by Lemma 2, since we will have $QR(R_2) \ominus QD(R_1 \rightarrow R_2) \leq QD(S \rightarrow R_1)$ and $QD(S \rightarrow R_1) \leq QD(S \rightarrow R' \rightarrow R_1)$. So R_1 will forward Join request to R', and mark it as the preferred immediate upstream node. When an Accept message is received later, a QoS probing message is also generated with the current QoS characteristics $QD(S \rightarrow R_1)$ and multicasted to downstream nodes to trigger QD update. Pruning message is to be sent to its original parent node R_0 to prune itself off from the old branch once multicast traffic comes down from R'. This is illustrated in Fig.4. In the latter case ($!(QD(S \rightarrow R_1) \leq QR(R_3) \ominus QD(R_1 \rightarrow R_3))$), R_1 can compute a new join path with $max[QD(S \rightarrow R_1), QR(R_3) \ominus QD(R_1 \rightarrow R_3)]$ as the QoS requirement. In such a path can be found, QoS requirements for both R_3 and existing downstream nodes of R_1 can be satisfied by Lemma 3 and Lemma 2. Thus R_1 can send a Join request to setup that new path; once it succeeds, R_1 can accept the Join request from R_3 and switch to the new path. If such a path couldn't be found, then R_1 should reject the Join request. This is illustrated in Fig.5.

From the above description, it is easy see that:

Theorem 1. If a Join request is accepted, then the new member's QoS requirement is guaranteed to be met without breaking QoS requirement of any existing member.

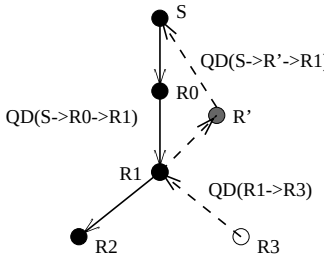

Fig. 4. $!(QR(R_3) \leq QD(S \rightarrow R_1) \oplus QD(R1 \rightarrow R_3)), QD(S \rightarrow R_1) \leq QR(R_3) \ominus QD(R_1 \rightarrow R_3)$.

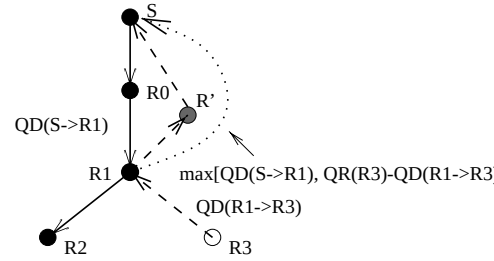

Fig. 5. $!(QR(R_3) \leq QD(S \rightarrow R_1) \oplus QD(R1 \rightarrow R_3))$, and $!(QD(S \rightarrow R_1) \leq QR(R_3) \ominus QD(R_1 \rightarrow R_3))$.

4.3 QoS Unicast Algorithms

Our algorithm relies on some available QoS unicast routing algorithm to compute join paths. It has been proved that, the shortest-path (or the min-cost path) problem subject to one or more additive or multiplicative constraints is NP-complete[7]. A number of polynomial-time heuristics have been proposed and studied [2]. In our simulation study, we use a simple greedy heuristic based on Bellman-Ford algorithm, more details and pseudo code can be found in [5]. This algorithm can be easily extended to handle multiple constraints and its time complexity is $O(|V||E|)$. Of course it does not guarantee a solution even if one exists, due to the NP-completeness nature of the problem.

4.4 Complexity Analysis

Let $n = |V|$ and $M = |E|$. At worst case, along the path for a new member to join the existing multicast tree, every intermediate node may need to compute a new join path, so we have:

Lemma 4. The computation complexity for a new member to join the group is $O(hf(n, M))$ at worst case, where $f(n, M)$ is the complexity of the unicast algorithm applied and h is the number of hops of the path.

Lemma 4 gives:

Theorem 2. The total complexity to setup a multicast group of m members with our algorithm is $O(mh_{max}f(n, M))$ at worst case, where h_{max} is the maximum number of hops of paths and $h_{max} < n$.

Algorithms mentioned in section 2 all assumed uniform QoS requirements for all members and only considered delay constraint. To compare our algorithm with them, we have to assume the same conditions:

Lemma 5. If there is only one uniform single additive or multiplicative metric constraint (in addition to a cost-minimization objective), the computation complexity for a new member to join the group is $O(f(n, M) + n)$, in a centralized implementation and in a distributed implementation assuming accurate link-state information (e.g., no stale information).

Proof: See [5].

Following Lemma 5, we have:

Theorem 3. When there is a single constraint of an additive or multiplicative metric, the total complexity to setup a multicast group of m members with our algorithm is $O(mf(n, M) + mn)$.

Using a constrained unicast routing algorithm with complexity $f(n, M) = O(nM)$ as mentioned earlier, our algorithm could build a multicast tree of m members with complexity $O(mnM)$. In most realistic networks, $M = O(n)$, thus the complexity is $O(mn^2)$. Table 1 gives a complexity comparison with several other algorithms:

Table 1. Comparison of computation complexity to build a multicast tree, using a $O(nM)$ unicast algorithm and assuming $|E| = O(|V|)$. For centralized algorithms, it assumed that all group members are known in advance and the algorithms build trees for all members at once.

algorithm	complexity
our algorithm	$O(mn^2)$
SPT (centralized)	$O(nlogn)$
SPT (distributed)	$O(mnlogn)$
Kompella's(centralized)	$O(n^3)$
BSMA(centralized)	$O(n^3logn)$

5 Simulation Results

Simulation has been used to study the behavior of our algorithm and to compare its performance with other schemes. In our first set of simulations, uniform bandwidth and delay constraints are considered, so we an compare our scheme with others. Three other algorithms are simulated: SPT(Shortest Path Tree) algorithm (single metric on delay), Kompella's algorithm([9]), and a greedy algorithm[1]. In the greedy algorithm, each time a node joins the group, it computes a delay-bounded min-cost path to all members in the existing tree. Then it chooses the one with lowest cost. The unicast QoS routing algorithm used for our algorithm is a delay-bounded min-cost heuristic algorithm based on Bellman-Ford algorithm[5].

We use randomly generated networks using the approach given in [13]. In simulations presented here, networks have a fixed size of 60 nodes chosen over a 30×30 grid. Parameters are chosen such that in average each node has a degree of 4 or 5. Geometric distance is used as delay on a link. To be as general as possible, a random cost between 0 to 1 is generated for each link. For simplicity, links are assumed to be bi-directional and symmetric. Furthermore, all links are assumed to have enough bandwidth, so bandwidth is not explicitly considered in the simulation.

Two sets of results for the comparison are shown here in Figure 6 (a) and (b). Each point in the figures represent the average from 100 instances. For each instance, we randomly pick a node as source node and a given number of

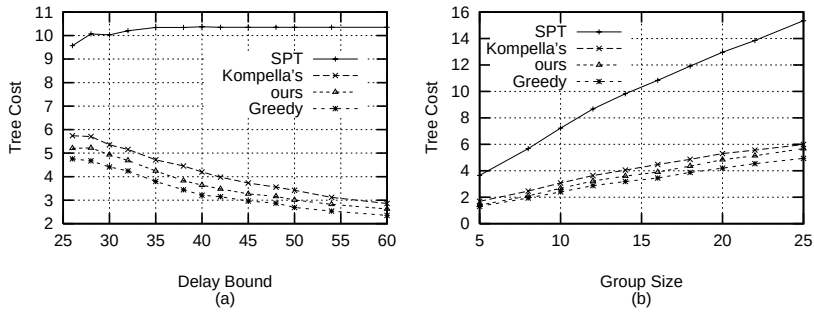

Fig. 6. (a)Tree cost vs. delay bound with fixed group size (=15); (b)Tree cost vs. group size with fixed delay bound(=40).

other nodes as receivers. In SPT and Kompella's algorithm, a tree is built for all members. In our algorithm and greedy algorithm, nodes join the group one by one in a random order. Fig.6(a) shows the tree cost vs. delay bound with fixed group size 15. Fig.6(b) shows the results for experiments with a fixed delay bound 40 and group size ranging from 5 to 25.

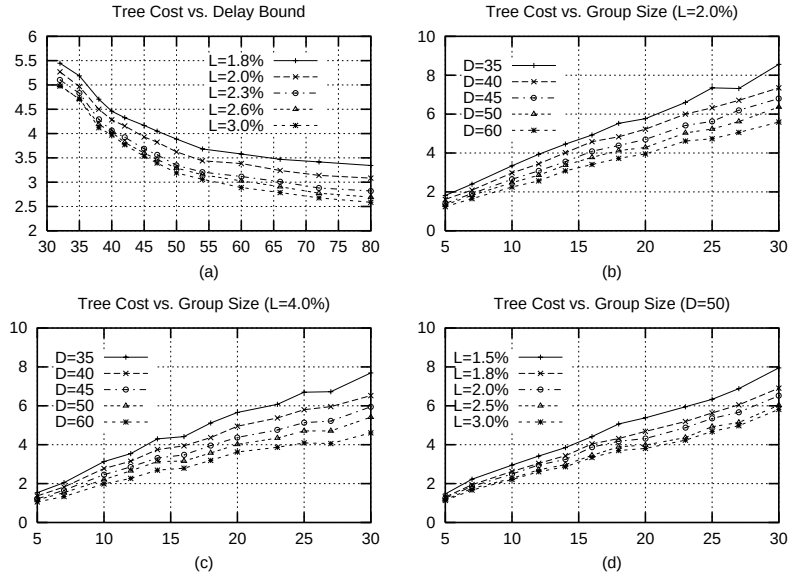

Fig. 7. Tree cost with varied delay bounds (D) and loss probability bounds (L).

As one can see, SPT always builds trees of higher cost than all others and almost don't change any with different delay. This is not surprising since SPT only builds a tree based on a single delay metric and makes no attempt to

optimize the tree in terms of cost. Of the remaining schemes, greedy algorithm always produce the lowest-cost tree. This is expected since it requires much more computation than others. Our algorithm works between Kompella's and greedy algorithms. Fig.6(a) shows all the three algorithms indeed can reduce the tree cost when delay bound is relaxed. In Fig.6(b), when group size grows, the cost of trees produced by all three algorithms increases at a rate lower than SPT. In summary, Kompella's, greedy and our algorithm produce trees of similar costs. However, compared with Kompella's algorithm, ours has the advantage of being fully distributed and allowing incremental tree build-up to accommodate dynamic joining members. Though greedy algorithm has the advantage of slightly lower cost, it is much more costly in terms of computation overhead.

To show our algorithm is effective when there are more than one constraint, we generate networks with one more random number(from 0 to 1%) associated with each link as loss probability. Such random loss can be considered to be caused by various reasons (transmission error, lossy wireless links, etc.). Because of the real-time nature of the applications, retransmission is not feasible and so we want to bound the total loss probability. Fig.7(a) shows tree cost vs. delay bound, similar to Fig.6(a), but now different curves are for different loss probability bounds. All curves have the same trend of decreasing with the relaxation of delay bound, as in Fig.6(a). It also shows that when loss probability bound is relaxed, the tree cost decreases along with. Fig.7(b),(c) and (d) present tree cost vs. group size with varied delay bounds and loss probability bounds. Fig.7(b) shows curves for different delay bounds with a fixed loss probability bound, Fig.7(c) shows curves of the same group of delay bounds but with a different fixed loss probability bound. Both demonstrate lower tree cost for more relaxed delay bound. Fig.7(c) has a more relaxed loss probability bound than (b), so one can see that each tree cost in (c) is lower than that in (b) with same group size and delay bound. Fig.7(d) shows a similar group of curves with a fixed delay bound but varied loss probability bounds. All these show that our algorithm is effective in lowering tree cost when it is allowed by the constraints.

6 Conclusions

In this paper, we have presented an incremental multicast tree construction algorithm. Assuming network information available at each node, it is possible to incrementally grow the tree through dynamic joining of group members. Our algorithm builds low-cost tree by utilizing unicast algorithm to compute a minimum-cost join path. Our algorithm specifies how to the adjust the tree when necessary to accommodate the QoS constraints of new members without breaking QoS of existing members, thus enhances the possibility of new member being accepted without recomputing the whole tree. In addition to its distributed nature and dynamic membership support, our mechanism can support heterogeneous receivers with any number of QoS metrics without any inter-metric dependency assumptions. Moreover, since a node doesn't need any explicit knowledge of existing receivers when it computes a join path, there is no requirement for

any node to maintain multicast tree topology information. Thus our approach can scale well to support multicast with large number of participants. Simulation results for delay-constrained multicast cast show that it has performance comparable to that of other schemes in terms of tree cost but with lower computation complexity. Additional simulation shows it performs consistently with more constraints. The key contribution of our work is an algorithm capable of handling multiple QoS constraints and supporting heterogeneous receivers with dynamic memberships.

References

1. D. Cavendish, A. Fei, M. Gerla, and R. Rom. On the maintenance of low cost multicast trees with bandwidth reservation. In Internet Mini-Conference with Globcom98, Australia, 1998.
2. S. Chen, and K. Klara. An overview of quality of service routing for next-generation high-speed networks: problems and solutions. In *IEEE Network Magazine*, pp.64-79, November/December 1998.
3. C. Diot, W Dabbous, and J. Crowcroft. Multipoint communication: a survey of protocols, functions, and mechanisms. In *IEEE Journal on Selected Areas in Communications*, Vol.15(3), pp.277-290, April 1997.
4. S. Deering, D. Estrin, D. Farinacci, et al.. Protocol independent multicast-sparse mode (PIM-SM): motivation and architecture. Internet draft: draft-ietf-idmr-pim-arch-05.txt{ps}, August 1998.
5. A. Fei, and M. Gerla. Receiver-initiated multicasting with multiple QoS constraints. *Technical Report No.990043*, Department of Computer Science, UCLA, 1999.
6. P. Ferguson and G. Huston. *Quality of Service*, John Wiley & Sons, Inc., 1998.
7. M. Garey, and D. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, 1979.
8. F. Hwang, D. Richards, and P. Winter. *The Steiner Tree Problem*, North-Holland, 1992.
9. V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. In *IEEE/ACM Transactions on Networking*, Vol.1(3), pp.286-292, June 1993.
10. V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Two distributed algorithms for the constrained Steiner tree problem. In *Proceedings of Computer Communication Networking*, June 1993.
11. D. Kosiur. *IP Multicasting*. John Wiley & Sons, Inc., 1998.
12. H. Salama, D. S. Reeves, and Y. Viniotis. Evaluation of multicast routing algorithms for real-time communications on high-speed networks. In *IEEE Journal of Selected Areas in Communications*, Vol.15(3), pp.332-344, April 1997.
13. B. M. Waxman. Routing of multipoint connections. In *IEEE Journal of Selected Areas in Communications*, Vol.6(9), pp.1617-1622, December 1988.
14. Q. Zhu, M. Parsa, and J. Garcia-Luna-Aceves. A source-based algorithm for delay-constrained minimum-cost multicasting. In *Proceeedings of IEEE Infocom'95*, pp.377-385, 1995.