QDMR: An Efficient QoS Dependent Multicast Routing Algorithm*

Liang Guo Ibrahim Matta
College of Computer Science
Northeastern University
Boston, MA 02115
{guol, matta}@ccs.neu.edu

Abstract

Many real-time applications, such as video conferencing, require the transmission of messages from a sender to multiple receivers subject to Quality-of-Service (QoS) delivery constraints (e.g. bounded delay). This requires the underlying multicast protocol to find a QoS-constrained minimum-cost communication path (tree). However, finding such a tree is known to be computationally expensive. In this paper, we present a fast heuristic, called QDMR, for generating delay-constrained low-cost multicast routing trees. A salient feature of QDMR is that it dynamically adjusts its low-cost tree construction policy based on how far the current on-tree node is from violating the QoS delay bound. This QoS dependent (adaptive) tree construction, together with the capability to merge least-delay paths into the low-cost tree in case of stringent delay requirements, lead to the following properties: (1) QDMR guarantees to find a feasible multicast tree if such tree exists; (2) this delay-bounded multicast tree is very rapidly generated; and (3) the tree has low cost. Through analysis and extensive simulations, we confirm the premise of QDMR by comparing it to many existing multicast algorithms.

1. Introduction

Distributed real-time applications, such as audio- and video-conferencing, collaborative environments and distributed interactive simulation, by and large, involve a source sending messages to a selected set of destinations with varying Quality-of-Service (QoS) delivery constraints. This requires the underlying network to provide multicasting (group communication) and QoS capabilities to efficiently support these applications.

The routing of multicast traffic requires the construction of a tree, referred to as the *multicast path*. When a source sends a message, this message is forwarded along the multicast path replicating the message only when the paths to different destinations diverge, thereby taking advantage of common path segments.

There are many algorithms to construct multicast paths. They can be broadly categorized as QoS-oblivious algorithms and QoS-sensitive algorithms. **QoS-oblivious algorithms** (e.g. [12, 18, 3, 1]) build multicast trees in a best-effort way *without explicitly* accounting for the requested QoS. Some minimize replication and bandwidth cost by building a so-called "Steiner tree" [9] that spans all group members, and the sum of its link costs is minimum. Others build a tree of shortest (least-delay) paths rooted at the source in order to minimize end-to-end delays to individual destinations. Shortest path trees can be efficiently computed, while Steiner trees are very expensive to compute (i.e., NP-complete [5]). For this reason, some QoS-oblivious algorithms try heuristically to achieve a balance between cost and delay by building a tree of shortest paths rooted at some (heuristically chosen) node in the center of the group [19, 1].

These latter QoS-oblivious algorithms may not, however, provide feasible trees that satisfy the requested QoS in terms of the end-to-end delay along the individual paths from the source to each of the destination nodes. To overcome this limitation, a number of QoS-sensitive algorithms have been proposed. These **QoS-sensitive algorithms** (e.g. [10, 21, 13, 22, 17, 16]) try to heuristically construct a low-cost tree *subject to* a given upper bound on end-to-end delay. However, some of these heuristics may fail to provide a low-cost tree as they assume that network links are symmetric. Furthermore, the time required to construct such a

*This work was supported in part by NSF grants CAREER ANIR-9701988 and MRI EIA-9871022, and NEU grant RSDF-377090.

tree may be prohibitive, especially for large networks, as they employ a brute-force approach to search for low-cost delay-bounded paths.

In this paper, we propose an efficient algorithm for rapidly generating a low-cost multicast tree subject to end-to-end delay constraints in large networks with asymmetric link costs and delays. Our algorithm is based on Shaikh and Shin's Destination-Driven MultiCasting (DDMC) algorithm [15]. DDMC is an efficient algorithm for generating low-cost *unconstrained* multicast trees. The basic idea of DDMC is to give the low-cost path going through a destination node priority over other paths to be extended in order to add a new node to the current tree. This helps reduce the tree cost as the cost incurred to go to one destination node can be leveraged to reach other destination nodes. However, this may result in a tree with long paths connecting a "chain" of destination nodes, and hence the end-to-end delay QoS requirement could be violated. We extend DDMC to overcome this problem by adjusting *dynamically* (on the fly) the low-cost tree construction policy so as to take into account how far the current destination node is from violating the QoS (delay) requirement. Furthermore, in the case where the requested QoS is still violated for some destination node, we merge the least-delay path into the tree. Thus, the new algorithm guarantees to find a feasible path if there is one, while keeping the tree cost low. We call this algorithm *QoS Dependent Multicast Routing* (QDMR for short). To summarize, QDMR has the following features compared to other algorithms: (1) a tree generation process that is QoS delay-dependent so as to minimize tree cost while *at the same time* keeping an eye on how far it is from violating the delay bound; (2) a very fast algorithm for generating low-cost delay-bounded multicast trees because of its QoS-dependent (adaptive) tree generation process; (3) it guarantees that a feasible tree will be constructed if one exists by merging least-delay paths into the low-cost tree if necessary. The merging process of least-delay paths stops as soon as the low-cost tree is encountered and the delay bound is satisfied, thus keeping the tree cost as low as possible.

Through extensive simulations, we show that QDMR yields trees that have comparable cost to other existing heuristics. Furthermore, these low-cost trees are very rapidly generated; the execution time of QDMR is several orders of magnitude lower than that of other heuristics, including the fastest heuristic recently proposed in [17]. For lack of space, proofs of correctness and time complexity are omitted and can be found in [7].

The rest of the paper is organized as follows. Section 2 defines the delay-constrained multicast routing problem. Section 3 describes our QDMR algorithm. Section 4 surveys some recently proposed heuristics against which we compare QDMR via simulations in Section 5. Section 6 concludes the paper with future work.

2. The Delay-Constrained Steiner Tree Problem

We represent the network by a directed graph $G = (V, E)$, where V is the set of all vertices (nodes) representing routers or switches, E is the set of edges (links) representing physical or logical connectivity between nodes. Each link is bidirectional, i.e., the existence of a link $e = (u, v)$ from node u to node v implies the existence of another link $e' = (v, u)$ for any $u, v \in V$. Any link $e \in E$ has a cost $C(e) : E \mapsto \mathcal{R}^+$ and a delay $D(e) : E \mapsto \mathcal{R}^+$ associated with it. The function $C(.)$ defines the measure we want to optimize (minimize). The function $D(.)$ defines the measure we want to constrain (bound). Due to the asymmetric nature of computer networks, it is possible that $C(e) \neq C(e')$ and $D(e) \neq D(e')$.

We denote a multicast group by the set $\{\{s\} \cup R\} \subseteq V$, where s is a source node which will send messages to a set of receivers denoted by R. A multicast tree $T(s, R) \subseteq E$ is a tree rooted at s and spanning all members of R. We denote by $P_T(r_i) \subseteq T$ the set of links in T that constitute the path from s to $r_i \in R$. The total (path) delay from s to r_i, denoted by $Delay[r_i]$, is simply the sum of the delay of links along $P_T(r_i)$, i.e.,

$$Delay[r_i] = \sum_{e \in P_T(r_i)} D(e)$$

The total cost of the tree, denoted by $Cost(T)$, is defined as the sum of the cost of all links in the tree, i.e.,

$$Cost(T) = \sum_{e \in T} C(e)$$

The application may assign an upper bound δ_i to $Delay[r_i]$, and the upper bound can be different for each destination r_i. For simplicity, in our network model, we assume that the upper bound for all destinations is the same, and is denoted by $\Delta = \delta_i, \forall r_i \in R$. We are currently extending this model to consider multiple delay bounds.

Given these definitions, we can formally present the *Delay-Constrained Steiner Tree (DCST) Problem* as follows:

The DCST Problem. *Given a network G, a source node s, destination node set R, a link delay function $D(.)$, a link cost function $C(.)$, and a delay bound Δ, the objective of the* Delay-Constrained Steiner Tree (DCST) Problem *is to construct a multicast tree $T(s, R)$ such that the delay bound is satisfied, i.e.,*

$$Delay[r_i] \leq \Delta \qquad \forall r_i \in R$$

and that the tree cost $Cost(T)$ is minimized.

Proposition 1 *The DCST problem is NP-complete.*

3. Our QoS Dependent Multicast Routing Algorithm

3.1 Motivation and Mechanics

Our QoS Dependent Multicast Routing (QDMR) algorithm is based on the Destination-Driven MultiCasting (DDMC) algorithm recently proposed by Shaikh and Shin [15]. The idea of DDMC comes from the fact that the well-known algorithms of Prim's minimum spanning tree and Dijkstra's shortest path tree [2] both use essentially the same greedy strategy. In Prim's algorithm, a minimum-cost tree that spans all nodes is constructed by augmenting the current tree with the minimum-cost edge emanating from some on-tree node (*not necessarily the source*). In Dijkstra's algorithm, at each step, a new node with the minimum path cost *from the source* is added to the current tree. Thus, the difference between the two algorithms lies in the choice of the cost function used at each step. In Prim's algorithm, the cost of a new (not-on-tree) node to be added to the current tree is simply the cost of the minimum-cost edge to it, whereas in Dijkstra's, this cost is the minimum total cost from the source to the node. The DDMC algorithm defines the cost of a new node $v \notin T$ via node $u \in T$ to be:

$$Cost[v] = I_D(u)Cost[u] + C(u, v)$$

where the "indicator function" $I_D(u) : V \mapsto \{0, 1\}$ is defined as

$$I_D(u) = \begin{cases} 0 & \text{if } u \in R \\ 1 & \text{otherwise} \end{cases}$$

The algorithm thus tries to make the destination nodes appear as new "sources". This gives preference to the paths going through destination nodes since the path to a new node via a destination node is likely to have a lower cost and thus be added to the tree. Since we have to absorb the cost of reaching a destination node anyway, the DDMC algorithm can multicast to all destination nodes on a low-cost tree.

Modifications to DDMC

Because the DDMC algorithm treats each destination node as a new "source" after that node is added to the tree, the finally constructed tree can easily have some very long branches which look like a "chain" of destination nodes. This enables DDMC to efficiently generate a low-cost multicast tree. However, this may also lead to some destination nodes violating the delay bound. We illustrate the above observation by an example in Figure 1. For ease of presentation, we assume here that links are symmetric.

Figure 1(a) shows the tree generated by the DDMC algorithm. This tree runs through nodes S, D_1, D_2, D_3, and D_4, because at each step, the path ending at a destination node will always be selected since it has a lower cost. This gives a tree cost of 7, as opposed to a cost of 14 if a tree of least-delay paths (LDT) is constructed as shown in Figure 1(b). However, with a DDMC tree, the path delays for nodes D_3 and D_4 are 7 and 8 respectively, and hence for both destinations the delay bound of 6 is violated. To satisfy the delay bound, a solution is to then replace the paths to D_3 and D_4 by the least-delay paths (LDP) as shown in Figure 1(c). We call such a tree DDMC+LDP. Now the cost of the finally formed tree becomes 11, which is not the lowest cost possible (as we show below).

To efficiently obtain delay-bounded low-cost trees, we modified the original DDMC algorithm so as it *dynamically* adjusts its tree construction policy according to how far a destination node is from the delay bound—our QoS requirement; the further the destination node is away from the delay bound, the more priority it has in being selected as parent of the newly added node. In other words, if we are still far from reaching the delay bound, our QDMR algorithm would behave as DDMC, i.e. it can produce "less bushy" trees with possibly long paths connecting a "chain" of destination nodes. This is desirable so as to minimize the cost of the tree. However, if the delay bound is about to be violated, our QDMR algorithm would give less priority to destination nodes and result in "bushier" trees so as to reduce the *likelihood* of violating the delay bound. A salient feature of QDMR is that it is simple and hence computationally inexpensive by adjusting its tree construction policy dynamically *as* nodes are added to the tree. This is in sharp contrast to other algorithms that exhibit higher computation cost because they pre-compute multiple paths between the source and each destination and then select those paths that are least costly and do not violate the delay bound.

Because it is easy to maintain the delay from the source to each on-tree node, our QDMR algorithm

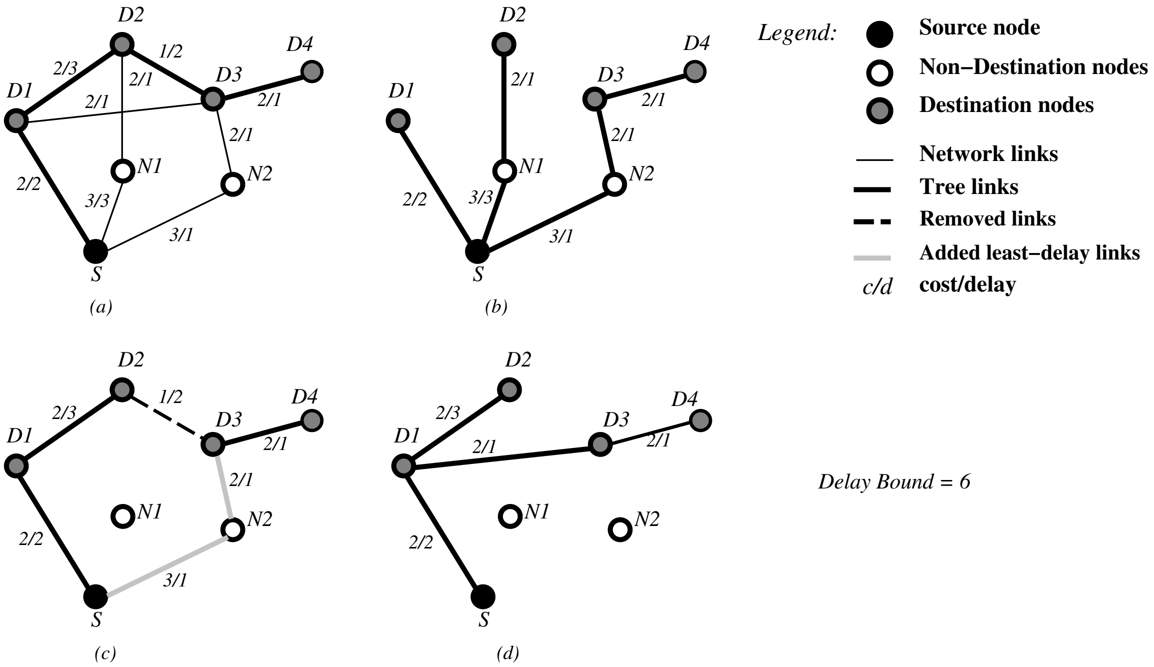


Figure 1. Example to show difference between DDMC and QDMR: (a) Example network and DDMC tree; (b) Least-delay tree (LDT); (c) DDMC+LDP tree; (d) QDMR tree.

achieves its delay dependent tree construction objective by replacing the simple indicator function $I_D(u) : V \mapsto \{0,1\}$ used in DDMC with a new indicator function defined as the ratio of the node delay to the delay bound,[1] i.e.,

$$I_D(u) = \begin{cases} Delay(u)/\Delta & \text{if } u \in R \\ 1 & \text{otherwise} \end{cases}$$

Using this new function, when QDMR constructs the multicast tree (cf. Figure 1(d)), node D_2 will have a lower priority to be added to the tree than node D_1, because D_2 is farther away from the source (or equivalently its delay is closer to the delay bound) and so has a higher cost than D_1.[2] On the other hand, D_2 still has a little higher priority than nodes N_1 and N_2 because it is a destination node.[3] Since D_3 has lower cost via D_1 than via D_2[4], the finally constructed tree would have a branch from node D_1 to D_3 and D_4. This tree satisfies the delay bound and has a cost of 8, which is lower than the cost of the DDMC+LDP tree shown in Figure 1(c).

We note that even after applying the new indicator function, our QDMR algorithm may still fail to find a feasible path to a destination node (i.e., a path that satisfies the delay bound) even if such path does exist. The reason for this is that we are using a greedy strategy to minimize tree cost when selecting the next not-on-tree node to be added. Thus we might miss some feasible path because of its high cost. Therefore, we add an additional phase in QDMR to fall back on using the least-delay path if a feasible low-cost path is not found for any destination node. In this case, we include the destination node in the tree by merging the least-delay path from the source to that node into the partial QDMR tree as follows: we start from each such (destination) node, and trace back the least-delay path for it until some on-tree node (another destination node, a relay node or the source) is encountered. If the merged path satisfies the delay bound, the merging process terminates for that destination. However, if the merged path does not satisfy the delay bound, we continue tracing back the least-delay path. In the worst case, the merging succeeds at the source node. In this case, the least-delay path in its entirety becomes part of the tree. This merging process is illustrated in Figure 2.

[1] Other definitions of $I_D(u)$ are also possible, but we found through simulations that this simple definition provides sufficiently good results.

[2] $Cost[D_1] = I_D(s)Cost[s] + C(s, D_1) = 1 \times 0 + 2 = 2$. $Cost[D_2] = I_D(D_1)Cost[D_1] + C(D_1, D_2) = \frac{2}{6} \times 2 + 2 = \frac{8}{3}$.

[3] $Cost[N_1] = I_D(s)Cost[s] + C(s, N_1) = 1 \times 0 + 3 = 3 = Cost[N_2]$.

[4] $Cost[D_3] = min\{I_D(D_1)Cost[D_1] + C(D_1, D_3), I_D(D_2)Cost[D_2] + C(D_2, D_3)\} = min\{\frac{2}{6} \times 2 + 2, \frac{5}{6} \times \frac{8}{3} + 1\} = min\{\frac{8}{3}, \frac{29}{9}\}$.

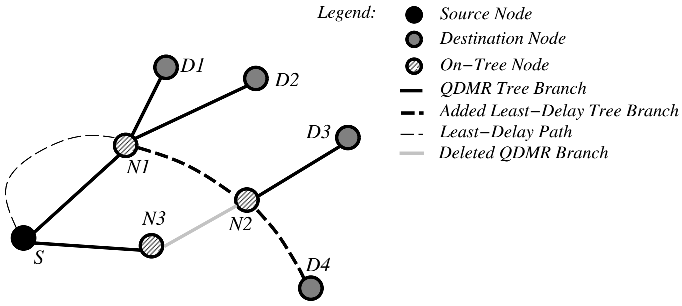


Figure 2. Merging process.

In this example, node D_4 can not be included in the QDMR tree in the *initial construction phase* since QDMR fails to find a feasible path to D_4. The *merging phase* then starts to merge the least-delay path (shown as dashed line) into the tree by tracing it back to the source until the on-tree node N_2 is encountered. However, at this point, it finds that the delay bound still can not be satisfied, i.e., $Delay[N_2] + D(N_2, D_4) > \Delta$. It then continues the merging process, and updates the parent pointer of node N_2 from node N_3 to node N_1 (cutting off the branch (N_3, N_2) from the tree). At node N_1, the merging process finds that $Delay[N_1] + D(N_1, N_2) + D(N_2, D_4) \leq \Delta$, so the subpath (N_1, D_4) is added as a new branch in the tree and now the destination node D_4 is included in the tree. Finally, a *pruning phase* is necessary to prune (s, N_3) since node N_3 is a leaf node that is not a destination.

The pseudo-code of our QDMR algorithm is given in Figure 3. Steps 0-13 show the initial tree construction phase. Steps 14-23 show the merging phase. Steps 24-25 show the pruning phase.

QDMR $(G(V, E), s, R, \Delta, D, C)$

0. /* Initial Tree Construction Phase */
1. Call Dijkstra's algorithm $DJK(G, s, R, D)$ to compute the least-delay tree (LDT) to find out the lowest possible delay bound $\Delta_{min} \leftarrow min_{r_i \in R}\{Delay[r_i]\}$

2. **if** $\Delta_{min} > \Delta$
3.   **Return** $FAILED$ /* a feasible tree does not exist */
4. $Cost[s] \leftarrow 0, Delay[s] \leftarrow 0$
$Cost[u] \leftarrow \infty, Delay[u] \leftarrow \infty$ for $u \neq s$
5. $T \leftarrow \emptyset$, $Q \leftarrow V$
6. **while** $Q \neq \emptyset$ and $R - T \neq \emptyset$ do
7. $u \leftarrow$ **Extract-Min**(Q) /* pick next node */
8. $T \leftarrow T \cup \{u\}$
9.   **for** each vertex $v \in Adj[u]$ /* for each adjacent node */
10.   **if** $Delay[u] + D(u, v) < \Delta$ and $v \notin T$
11.   **if** $Cost[v] > I_D(u)Cost[u] + C(u, v)$
12. $Cost[v] \leftarrow I_D(u)Cost[u] + C(u, v)$
13. $\pi[v] \leftarrow u$ /* u is the parent */

14. /* Merging Phase */
15. **if** $R - T \neq \emptyset$ /* some destinations haven't been reached*/
16.   **for** each $u \in R - T$ do
17. $d_{cumm} \leftarrow 0$, $p \leftarrow \pi_{LDP}[u]$ /* merge LDP */
18.   **while** $u \neq s$
19. $\pi[u] \leftarrow p$, $d_{cumm} \leftarrow d_{cumm} + D(p, u)$
20. $T \leftarrow T \cup \{u\}$
21.   **if** $Delay[p] + d_{cumm} \leq \Delta$
22.   go to 16
23. $u \leftarrow p$, $p \leftarrow \pi_{LDP}[u]$

24. /* Pruning Phase */
25. return **PruneLeaves(T)**

Figure 3. Our QDMR algorithm.

3.2 Correctness and Time Complexity of QDMR

Theorem 1 *The path generated by QDMR is loop-free.*

Theorem 2 *The QDMR algorithm always constructs a delay-bounded multicast tree if such a tree exists.*

Lemma 1 *The time complexity of QDMR is* $O(|E| \log |V|)$.

4. Related Work

In this section, we briefly discuss some recently proposed delay-constrained Steiner tree heuristics. (Reference [14] provides a good survey.) The KPP [10] and the CKMB [17] heuristics both extend KMB, an unconstrained Steiner tree heuristic [11]. KMB first converts the original network graph into a complete graph connecting multicast group members, where a (logical) edge represents the least-cost path between two members. Then KMB constructs a minimum-cost (spanning) tree, and finally replaces each logical edge in the tree with the corresponding (physical) least-cost path.

KPP extends KMB to compute delay-bounded paths assuming the delay bound and link delays are all integers. This assumption limits its applicability although an approximate solution was proposed that transforms real delay values into integer values. This approximation, however, may compromise the accuracy of the algorithm by failing to find a feasible tree even though one exists. The time complexity of KPP is $O(\Delta|V|^3)$. CKMB also extends KMB without assuming integer delay values. However, CKMB, like KPP, builds a minimum-cost tree of *logical* edges. Thus they may fail to exploit common *physical* edges, leading to high tree cost. The time complexity of CKMB is $O(|R||V|^2)$.

The CDKS [16] algorithm differs from QDMR in that the cost of a node is simply the total cost from the source, *i.e.* unlike QDMR, it does not attempt to reduce tree cost by using a dynamic cost function. Furthermore, if the delay bound is violated for some destination node, the least-delay path is simply used in its *entirety*, leading to further increase in tree cost. The time complexity of CDKS is $O(|V|^2)$.

BSMA [22] is another multicast algorithm that is considered the best in terms of tree cost [14]. BSMA iteratively replaces the edges in the tree until the tree cost can not be further reduced. This makes the algorithm computationally very expensive to be used for large-scale networks. The *average* time complexity of BSMA is $O(k|V|^3 \log |V|)$, where k is the average number of lower cost paths examined to replace a lower delay path.

Another algorithm is CAO [21]. It incrementally adds new tree branches by merging in delay-constrained low-cost *unicast* paths leading to new destination nodes. The time complexity of CAO is $O(t(\mathcal{A}) \times |R|)$, where $\mathcal{A}$ is the algorithm used to find unicast paths. The original version of CAO uses a constrained breadth-first search algorithm, thus its time complexity can be exponential in some cases. However, even with an efficient unicast routing heuristic (e.g. [8]), it may still need a much longer time than QDMR to form a tree.

In Section 5, we compare the various heuristics in terms of *actual* execution time via extensive simulations. Our results show that QDMR is the fastest in generating multicast trees that are competitive in terms of tree cost. This indicates that QDMR is an attractive algorithm for real-time multicasting in large networks.

5. Simulation Model and Results

To evaluate the efficiency of our QDMR algorithm, we use the multicast routing simulator $MCRSIM$ [4] developed at North Carolina State University. $MCRSIM$ allows the simulation of arbitrary networks and supports multiple traffic types, as well as background traffic on each link. $MCRSIM$ already implements many existing Steiner tree heuristics, including BSMA [22], CAO [21], KPP [10], CDKS [16], and CKMB [17]. We implement our QDMR algorithm in the $MCRSIM$ simulator and compare it to these other heuristics.[5]

[5]In our simulation experiments, for the KPP algorithm, we map real delay values in $[0, \Delta]$ to integer values in $[0, 10]$.

We consider randomly generated network topologies using a modified version of the graph generation algorithm described in [20]. The average degree of a node is set to 4, and the capacity of each link is 155 Mbps. The link delay function $D(e)$ is defined as the propagation delay of the link plus a fixed switching delay of 30 μsec.[6]

The link cost function, $C(e)$, is defined as the current total bandwidth reserved on the link[7]. We conducted two sets of experiments. We present each next.

Experiments 1

In this first set of experiments, we investigate the quality of the multicast tree generated by each heuristic and their computation requirement. This was done by measuring for an arriving group the cost of the tree and the actual execution time to generate the tree under each heuristic. All simulations were running on a SUN SPARCstation-10, and the codes for all the simulated algorithms were not optimized for speed. For each run, the cost of a link is the amount of bandwidth reserved on the link for some background traffic. This reserved bandwidth is a random variable uniformly distributed between B_{min} and B_{max}. We set B_{min} to 10 Mbps and B_{max} to 120 Mbps. The difference between B_{min} and B_{max} reflects the *asymmetry* of the link loads when the multicast group arrives. The group members are also randomly selected. We obtain results for different network sizes, multicast group sizes and different delay bounds. For a specific network/group configuration, each algorithm is executed. This was repeated until confidence intervals of less than 5% (using 95% confidence level) were achieved for all measured quantities.

Figure 4 shows the tree cost and the execution time for varying network size. For clarity, execution times are shown in log-scale. We fix the group size at 10, and we choose a stringent delay bound of 30 ms. Figure 4(a) shows that as the size of the network increases from 20 nodes to 100 nodes, the difference between the sub-optimal BSMA heuristic and other heuristics becomes larger. CAO always yields the closest tree cost to that of BSMA. The remaining three heuristics perform very close to each other (less than 4%); CKMB has relatively lower tree cost, followed closely by KPP then QDMR. Figure 4(b) shows that BSMA and KPP are the most computationally expensive algorithms. QDMR is the fastest algorithm, followed by CKMB,

[6]It is assumed that queueing delays are negligible given enough bandwidth is reserved on the link for each multicast group, and that transmission delays are very small.

[7]This definition reflects the relationship between cost and link utilization; a highly loaded link usually costs more. Other definitions are also possible.

then CAO. QDMR is orders of magnitude faster than all other algorithms.

Figure 5 shows the performance measures versus group size for a 50-node network and a delay bound of 30 ms. It can be seen from Figure 5(a) that the performance of CAO in terms of tree cost degrades as the group size increases. Other heuristics have similar performance; in general QDMR gives slightly higher tree cost (less than 4%) than KPP and CKMB.

Figure 5(b) shows that the execution times of BSMA and KPP are very large, almost 3 orders of magnitude higher than that of QDMR. As the group size increases, the execution times of both CAO and CKMB grow. On the contrary, the execution time of QDMR remains almost the same with increasing group size, and it can be up to 1 order of magnitude lower than that of CKMB. Thus **QDMR scales very well to large networks and multicast groups. It can produce low-cost trees at a significantly higher speed.**

Figure 6 shows the performance of different heuristics for varying delay bound, for a 50-node network and a group size of 15. Figure 6(a) shows that all heuristics can find a lower cost tree as the delay bound increases. CKMB is slightly more sensitive to the delay bound than others. When the delay bound is not so stringent (i.e. large), both CKMB and QDMR perform better than KPP. Figure 6(b) shows that the execution time of all heuristics is not very much affected by the delay bound.

Since CDKS was suggested to be a suitable algorithm for large networks [14], Figure 7 compares our QDMR algorithm to CDKS as well as CKMB for relatively large networks. We also show results for DDMC+LDP. The only difference between DDMC+LDP and QDMR is in the definition of the indicator function used to give priority to paths going through destination nodes; $\{0, 1\}$ is used in DDMC+LDP, whereas QDMR uses the linear delay-dependent function (cf. Section 3.1). We vary the network size from 200 nodes to 600 nodes, and set the group size to be 10% of the network size and the delay bound to be 30 ms. Figure 7(a) shows that CKMB yields the lowest tree cost. QDMR has a slightly higher tree cost (always less than 2%). QDMR always has a lower tree cost than CDKS. Figure 7(b) shows that QDMR is significantly faster. Both QDMR and CKMB have a lower rate of increase in execution time than CDKS as the network size increases. Note that even for a very large network (600 nodes), the time required for constructing a QDMR tree is less than 1 second on a SUN SPARCstation-10. (Recall that the code for the various algorithms under study was not optimized for speed.) It is also important to note that QDMR significantly outperforms DDMC+LDP in terms of tree cost due to its use of a QoS dependent tree construction process, while maintaining its fast execution feature. This indicates that QDMR is a promising QoS multicast routing algorithm for large networks.

Experiments 2

In this second set of experiments, we compare the efficiency of different heuristics in terms of how well they distribute arriving multicast groups (traffic) over the network so as to increase network utilization (or revenue). In each run, we start with an empty network and generate successive arrivals of multicast groups. A multicast group is randomly generated. The source of each arriving group is assumed to be a variable bit rate (VBR) video source. The QoS requirement is guaranteed by reserving some amount of bandwidth (or the *equivalent bandwidth* of the source [6]) on each link of the multicast tree. In these experiments, the equivalent bandwidth of each multicast session is 0.5 Mbps. Each group requests a delay bound of 30 $msec$.

A group is *not* admitted into the network (i.e. rejected) if the total bandwidth reserved on a link would exceed 85% of the link's capacity. The run is terminated when the total group rejection rate exceeds 15%. We note that an arriving group could be rejected either because of failure to reserve bandwidth or because a feasible tree could not be found[8]. Figure 8 shows the number of admitted multicast groups on a 20-node network for different group sizes. Again, 95% confidence intervals were computed for the results. We observe that QDMR performs as well as BSMA, which was suggested in [14] to be the most efficient in distributing sessions over the network and thus increasing the likelihood of an arriving group being admitted into the network. We also note that an algorithm that simply builds a tree of least-delay paths (LDT) performs the worst as it produces costly trees.

6. Conclusions and Future work

We presented an efficient algorithm for obtaining delay-constrained low-cost multicast trees. Our algorithm extends a recently proposed unconstrained algorithm [15] so as to rapidly generate a low-cost tree while adapting (on the fly) the generation process to account for application's end-to-end delay demands. We showed through extensive simulations that the proposed algorithm generates low-cost trees with a com-

[8]KPP is the only simulated algorithm which may not find a feasible tree even though one exists, because of its real-to-integer conversion of delay values.

putation overhead that is several orders of magnitude lower than that of other algorithms.

Since the problem of finding a delay-constrained minimum-cost path is NP-complete, we resorted to a simple heuristic in order to keep the computational cost low. Like other existing algorithms (such as CKMB and KPP), our algorithm may fall back on using least-delay paths (or segments of them) so as to find a feasible tree. A more sophisticated search may yield lower cost paths that are also feasible. To that end, we will investigate the use of our recently proposed heuristic to find low-cost QoS-constrained (unicast) paths [8]. An interesting question here is whether the gain in tree cost will justify the additional computational cost.

It is worth mentioning that our QDMR algorithm, as well as all the other algorithms presented in this paper, assumes that information regarding the state of each link (i.e. link cost and delay) and the location of group members is available at each node. This is the same link-state approach used in current Internet protocols like MOSPF [12]. If the network provides best-effort service, then the delay bound satisfied by the computed tree is not guaranteed (i.e. may be violated for some packets). For all packets to satisfy the delay bound, resources must be reserved over the computed tree to guarantee the associated link delays.

Other future work include investigating other QoS constraints (e.g. delay jitter) and applications where group members join and leave the multicast group at will. Such dynamic group membership makes it very hard, if not impossible, to maintain all the time an optimal cost multicast tree that also satisfies given performance (delay) constraints. One possible solution to this problem is to, whenever a new group member joins or an existing member becomes out-of-bound, add or replace the old path with a new low-cost delay-bounded (unicast) path. QDMR could be applied regularly to restore optimality of the whole multicast tree. We will investigate this approach in our future work.

References

[1] A. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees. In Proc. *SIGCOMM '93*, San Francisco, California, September 1993.

[2] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company, 1990.

[3] S. Deering, D. Estrin, D. Farrinacci, V. Jacobson, C. Liu, and L. Wei. Protocol Independent Multicast (PIM): Protocol Specification. Internet Draft, 1995.

[4] H.F. Salama et al. MCRSIM simulator source code and Users' Manual. Center for Advanced Computing and Communication, North Carolina State University, Raleigh, 1995. Available by anonymous ftp at ftp.csc.ncsu.edu:/pub/rtcomm.

[5] M.R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[6] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Networks. *IEEE J. Select. Areas Commun.*, SAC-9(7):968–981, September 1991.

[7] L. Guo and I. Matta. QDMR: An Efficient QoS Dependent Multicast Routing Algorithm. Technical Report NU-CCS-98-05, College of Computer Science, Northeastern University, August 1998.

[8] L. Guo and I. Matta. Search Space Reduction in QoS Routing. Technical Report NU-CCS-98-09, College of Computer Science, Northeastern University, October 1998.

[9] F.K. Hwang and D.S. Richards. Steiner tree problems. *IEEE/ACM Transactions on Networking*, 22:55–89, January 1992.

[10] V.P. Kompella, J.C. Pasquale, and G.C. Polyzos. Multicasting for Multimedia Applications. In Proc. *IEEE INFOCOM'92*, pages 2078–2085, 1992.

[11] L. Kou, G. Markowski, and L. Berman. A Fast Algorithm for Steiner Trees. *Acta Informatica*, 15:141–145, April 1981.

[12] J. Moy. Multicast Extensions to OSPF. Internet draft, Network Working Group, September 1992.

[13] G.N. Rouskas. Multicast Routing with End-to-End Delay and Delay Variation Constraints. *IEEE J. Select. Areas Commun.*, 15:346–357, April 1997.

[14] H.F. Salama, D.S. Reeves, and Y. Viniotis. Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks. *IEEE J. Select. Areas Commun.*, 15:332–346, April 1997.

[15] A. Shaikh and K. Shin. Destination-Driven Routing for Low-Cost Multicast. *IEEE J. Select. Areas Commun.*, 15:373–381, April 1997.

[16] Q. Sun and H. Langendoerfer. Efficient Multicast Routing for Delay-Sensitive Applications. In Proc. *Second Workshop on Protocols for Multimedia Systems (PROMS)*, pages 452–458, October 1995.

[17] Q. Sun and H. Langendoerfer. An Efficient Delay-Constrained Multicast Routing Algorithm. Technical Report Internal Report, Institute of Operating Systems and Computer Networks, TU Braunschweig, Bueltenweg 74/75, 38106, Braunschweig, Germany, January 1997.

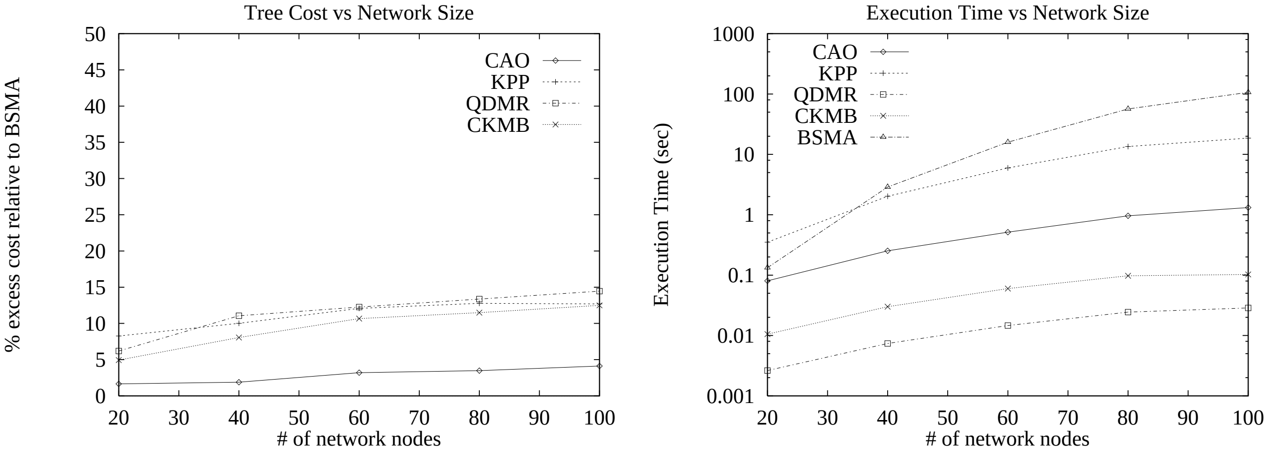
[18] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. Request for Comments RFC-1175, November 1988.

[19] D.W. Wall. *Mechanisms for Broadcast and Selective Broadcast.* PhD thesis, Stanford University, Department of Computer Science, 1982.

[20] B.M. Waxman. Routing of multipoint connections. *IEEE J. Select. Areas Commun.*, pages 1617–1622, December 1988.

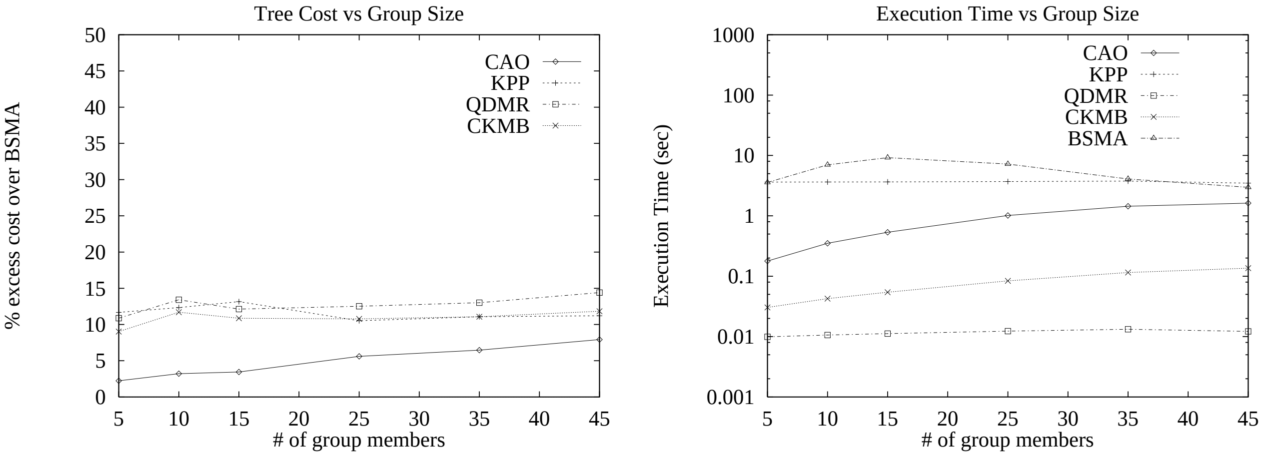
[21] R. Widyono. The Design and Evaluation of Routing Algorithms for Real-Time Channels. Technical Report ICSI TR-94-024, International Computer Science Institute, U.C. Berkeley, June 1994.

[22] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves. A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting. In Proc. *IEEE INFOCOM'95*, pages 377–385, 1995.



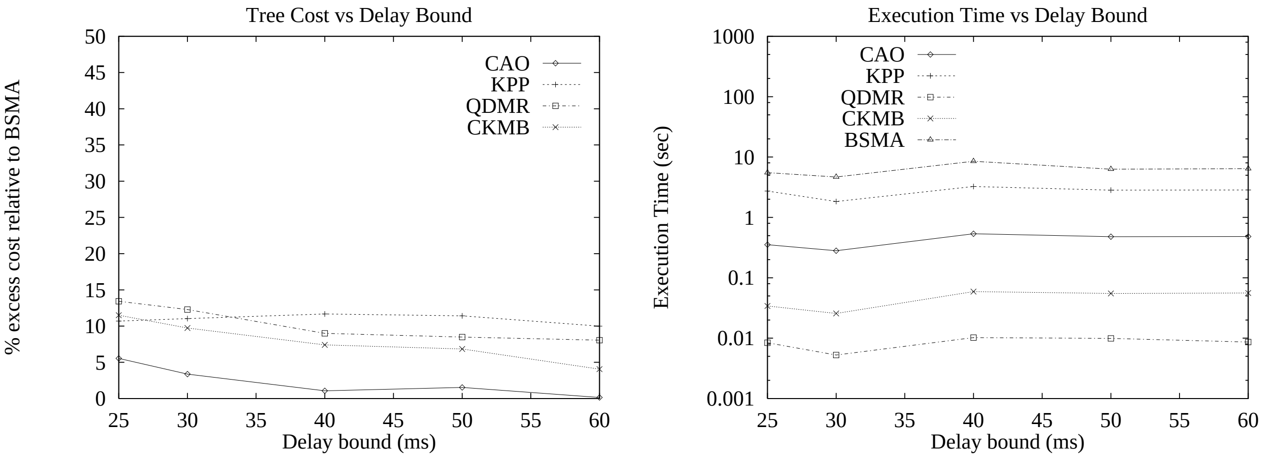
(a) (b)

Figure 4. Tree cost and execution time versus network size: group size=10, delay bound=30 ms.



(a) (b)

Figure 5. Tree cost and execution time versus group size: network size=50, delay bound=30 ms.



(a) (b)

Figure 6. Tree cost and execution time versus delay bound: network size=50, group size=15.

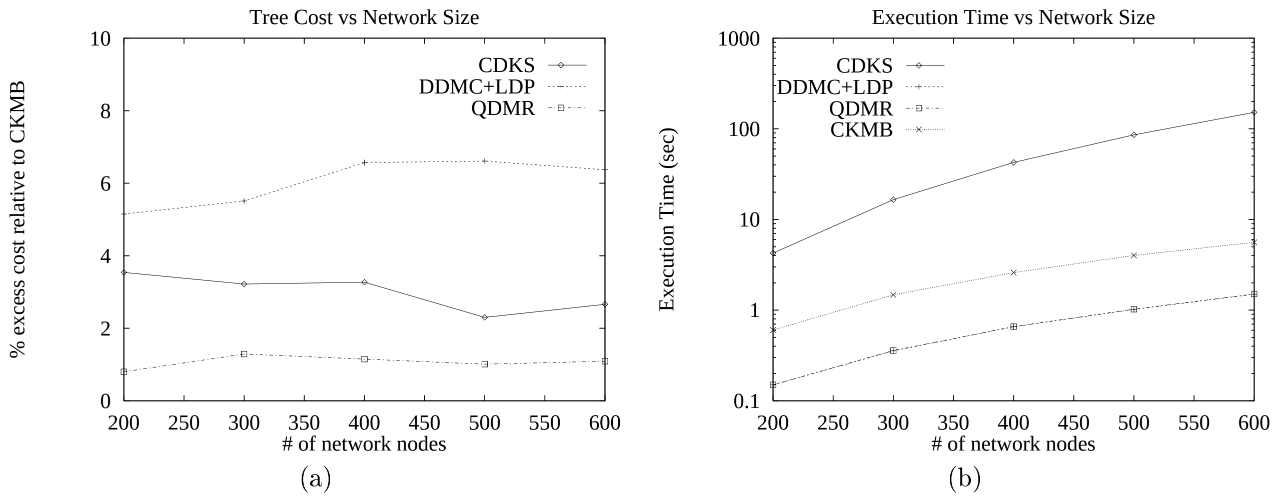


Figure 7. Tree cost and execution time vs network size: group size=10% of network size, delay bound=30 ms.

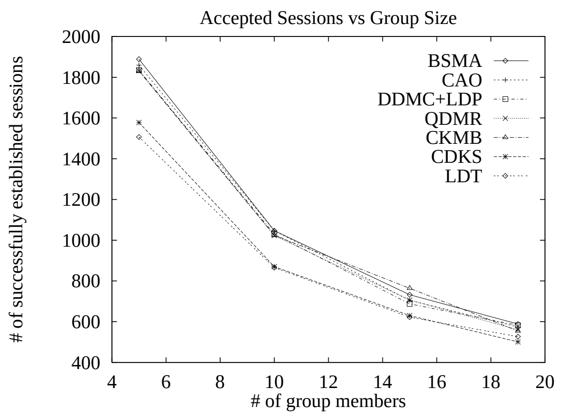


Figure 8. Number of admitted sessions versus group size: network size=20, delay bound=30 ms.