An Efficient Delay-Constrained Multicast Routing Algorithm

Quan Sun   Horst Langendőrfer

Institute of Operating Systems and Computer Networks

Technical University of Braunschweig

Bűltenweg 74/75, 38106 Braunschweig, Germany

Abstract

An efficient delay-constrained multicast routing algorithm computing low cost delay-constrained multicast trees with small algorithm complexity is proposed in this paper. The proposed algorithm can be used to compare cost performances of the existing or future multicast routing algorithms that provide quality of service (QoS) guarantees, especially in multicast routing simulation environments which simulate relatively large networks and consider realistic network conditions. The proposed algorithm is also useful for routing real-time multicast data in high-speed networks if adequate global information is available to the source.

Keywords: Multicast Routing Algorithm, Delay Constraint, Tree Cost

1 Introduction

Many networking applications, e.g. multimedia applications, are delay-sensitive and involve multiple participants. In order to support these applications efficiently, multicast routing algorithms satisfying stringent delay bound constraint are needed. The first multicast routing algorithm satisfying a given end-to-end delay constraint was proposed by Kompella *et al* in 1992 [1]. Since then large number of delay-constrained multicast routing algorithms have been proposed (see [2]

- [9]). Because many more such algorithms are also expected to appear in the future [13], we need an efficient delay-constrained multicast algorithm for performance comparison purpose. In the past, the efficient KMB algorithm [19] has often been chosen as the basis for comparing with other unconstrained algorithms (e.g. in [10] [11] [12]), however, an efficient algorithm is lacking for comparing the cost performances of delay-constrained multicast routing algorithms. Most existing delay-constrained multicast algorithms that can compute low-cost delay-constrained multicast trees require too much computation time. For example, in the multicast routing simulation environment MCRSIM [14], the bounded shortest multicast algorithm [7] needs an average running time of more than one minute to compute a delay-constrained multicast tree in a network with only 80 nodes on our local SUN SPARCstation 5, other existing algorithms like [2] [5] have also speed problems to simulate networks with more than 100 nodes (see the simulation results in section 4 of the paper).

In this paper, we propose an efficient delay-constrained multicast routing algorithm. The proposed algorithm can be used to compare cost performances of the existing or future delay-constrained multicast routing algorithms, especially in multicast routing simulation environments (like MCRSIM) which simulate relatively large networks and consider realistic network conditions. The algorithm may be also useful for routing multicast data in the network if adequate information is available to the source because of its adequate computation time. The rest of the paper is arranged as follows. The problem definition is given in section 2. Section 3 presents our proposed algorithm. We have integrated our proposed algorithm into the simulation environment MCRSIM and conducted extensive simulation experiments to compare the performances of most of the delay-constrained multicast algorithms proposed so far (including ours). The simulation results are presented in section 4. Section 5 concludes the paper.

2 Network Model and Problem Definition

A network is modeled as a connected, directed graph $G=(V,E)$ where V is the set of network nodes and E is the set of links. In addition, we have two real value functions associated with each link e ($e \in E$): delay $D(e)$ and cost $C(e)$. The link delay $D(e)$ is the delay a data packet experiences on the corresponding link, the link cost $C(e)$ is a measure of the utilization of the corresponding link's resources. Links are asymmetrical, namely the cost and delay for the link $e=(i,j)$ and the link $e' = (j, i)$ may not be the same.

The delay-constrained multicast routing problem can be stated as follows. Given a source node s ($s \in V$), a set of destination nodes S ($S \subseteq V - s$) and a delay tolerance $\triangle$, find a tree T ($T \subseteq G$) rooted at s and spanning all of the nodes in S such that for every node v in S:

$$\sum_{e \in P(s,v)} D(e) < \triangle \qquad (A)$$

where $P(s,v)$ is the unique path in T from s to v, and that the tree cost

$$\sum_{e \in T} C(e) \text{ is minimized.}$$

Throughout this paper, we let $n=|V|$ and $m=|S|$. The nodes in S are often called multicast *group members* and m is then the multicast group size. $\triangle$ is also called *delay bound*. When $\triangle$ is ∞, the delay-constrained multicast routing problem reduces to the standard Steiner tree problem [25] and therefore is *NP*-complete. When $m=1$, it reduces to the delay-constrained unicast routing problem and was shown also to be *NP*-complete [17].

A tree rooted at s, spanning all the nodes in S *and* satisfying (A) is called a *constrained* multicast tree. A heuristic solution to the delay-constrained multicast routing problem should always find a constrained multicast tree, if one exists. Besides, an efficient heuristic algorithm for this problem should make sensible tradeoffs between the cost of the tree produced and the computation time used to produce the tree.

Readers may have noticed that a centralized delay-constrained multicast routing algorithm can be easily extended to compute multicast trees that satisfy *both* a given end-to-end delay constraint *and* a given link bandwidth constraint. We will not describe and evaluate the performances of such delay- and bandwidth-constrained algorithms in this paper.

3 The Proposed Algorithm

Because our proposed algorithm follows the idea of the famous heuristic algorithm KMB for the minimum Steiner tree problem [19] to compute multicast trees satisfying a given end-to-end delay *constraint*, we simply call the proposed algorithm **CKMB** in the following.

3.1 The Constrained Routing Algorithm (CRA)

Before presenting CKMB, we first describe a Constrained Routing Algorithm (CRA), a key component in CKMB. CRA was proposed by us in *PROMS'95* [8]. For a directed network with link costs C, link delays D and a delay bound $\triangle$, given a source node s and a set of destination nodes S,the CRA algorithm tries to compute a low cost tree rooted at the source node s and spanning some (or possibly all) of the destination nodes in S subject to the delay bound constraint. The CRA$(s, S, \triangle, C, D)$ procedure is given in Fig.1.

```
Procedure CRA(s, S, △, C, D)
/*
s is the source node, S is the set of destination nodes. △ is the bounded delay. The network is represented
by its cost adjacency matrix C with C(i,j) being the cost of the link (i,j) and by its delay adjacency matrix
D with D(i,j) being the delay of the link (i,j). Both C(i,j) and D(i,j) will be set to ∞ in case the link
(i,j) is not in E. For i=j, C(i,j) and D(i,j) are set to 0. The nodes of the network G(V,E) are numbered
1 through n.
*/
boolean S(1:n); real CST(1:n), DLY(1:n);
/*
The set of nodes V is divided into two subsets, namely the set of already computed nodes and the set of
nodes to be computed. If S(i)=1, node i is then an already computed node and the cost and delay of the
computed path from the source s to i are given in CST(i) and DLY(i) respectively. S(i)=0 means node
i has not been computed yet.
*/
integer u, i, w; real vc;
begin
/* initialize the set of already computed nodes to empty */
for i ← 1 to n do
    S(i) ← 0;
    if D(s,i) < △
        then CST(i) ← C(s,i)  and  DLY(i) ← D(s,i);
        else CST(i) ← ∞
    end if
end for
/*s is the first computed node */
S(s) ← 1; CST(s) ← 0; DLY(s) ← 0;
/* compute paths */
while there are nodes in S which have not been computed yet  do
    choose u such that CST(u) = min{CST(w)} for all 1 ≤ w ≤ n and S(w)=0;
    if CST(u) is ∞ then break; end if
    /*u is a new computed node */
    S(u) ←1;
    /* update costs and delays */
    for  all w with S(w)=0  do
        if DLY(u) + D(u, w) < △ then
            vc ← CST(u) + C(u, w);
            if vc < CST(w) then
                CST(w) ← vc;
                DLY(w) ← DLY(u) + D(u, w);
            end if
        end if
    end for
end while
prune those leaf nodes in the resulting tree that are not in S;
end
```

Figure 1: The *CRA* procedure

If readers are familiar with the Dijkstra's shortest path algorithm [18] [21], the procedure shown in Fig.1 can be easily understood. Therefore, instead of explaining the procedure step by step here, we give in Fig.3 an example to illustrate the application of CRA to a 8-node graph shown in Fig.2, with the source node $s = 7$, destination node set $S = \{2, 3, 5\}$ and $\triangle$=9.0. In *(a)* of Fig.3 the source node *7* is the first computed node (see line (9) of the CRA procedure). For *(b)* - *(f)* each step generates a new node to which the path from the source node *7* through (some) already computed nodes has the least cost and satisfies the delay bound constraint among all the paths from node *7* to all the nodes to be computed that satisfy the delay bound constraint (see lines (11) - (22) of the CRA procedure). The resulting tree after pruning the link (7,1) is shown in *(g)* of Fig.3.

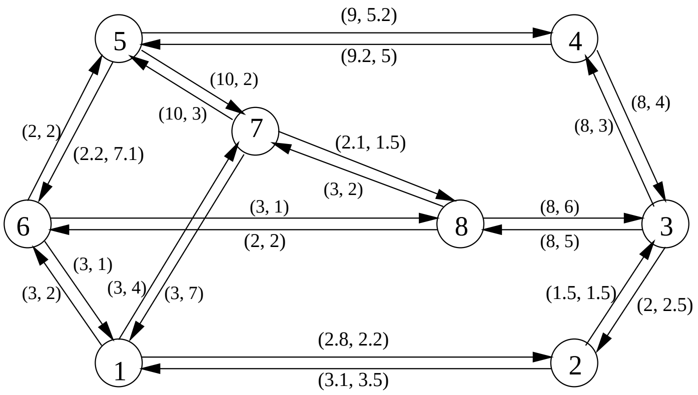

Figure 2: Example graph

Lemma 1: The time complexity of CRA is O(n^2).

Proof: The running time of *CRA* is dominated by the **while** loop of line (10) (see Fig.1), which is executed at most n - 2 times. Each execution of this loop requires *O(n)* time at line (11) to select the next node and again at lines (14)-(22) to update costs and delays. Therefore, the time complexity of *CRA* is O(n^2). □

Lemma 2: The tree produced by CRA may not span all of the destination nodes.

The example in Fig.3 shows the correctness of this lemma. In Fig.3, the resulting tree has not spanned the destination node *2*, although there exists a path from node *7* to node *2* satisfying the given delay constraint.

Lemma 3: If $\triangle$ is ∞, the unique path from the source node to each destination node along the

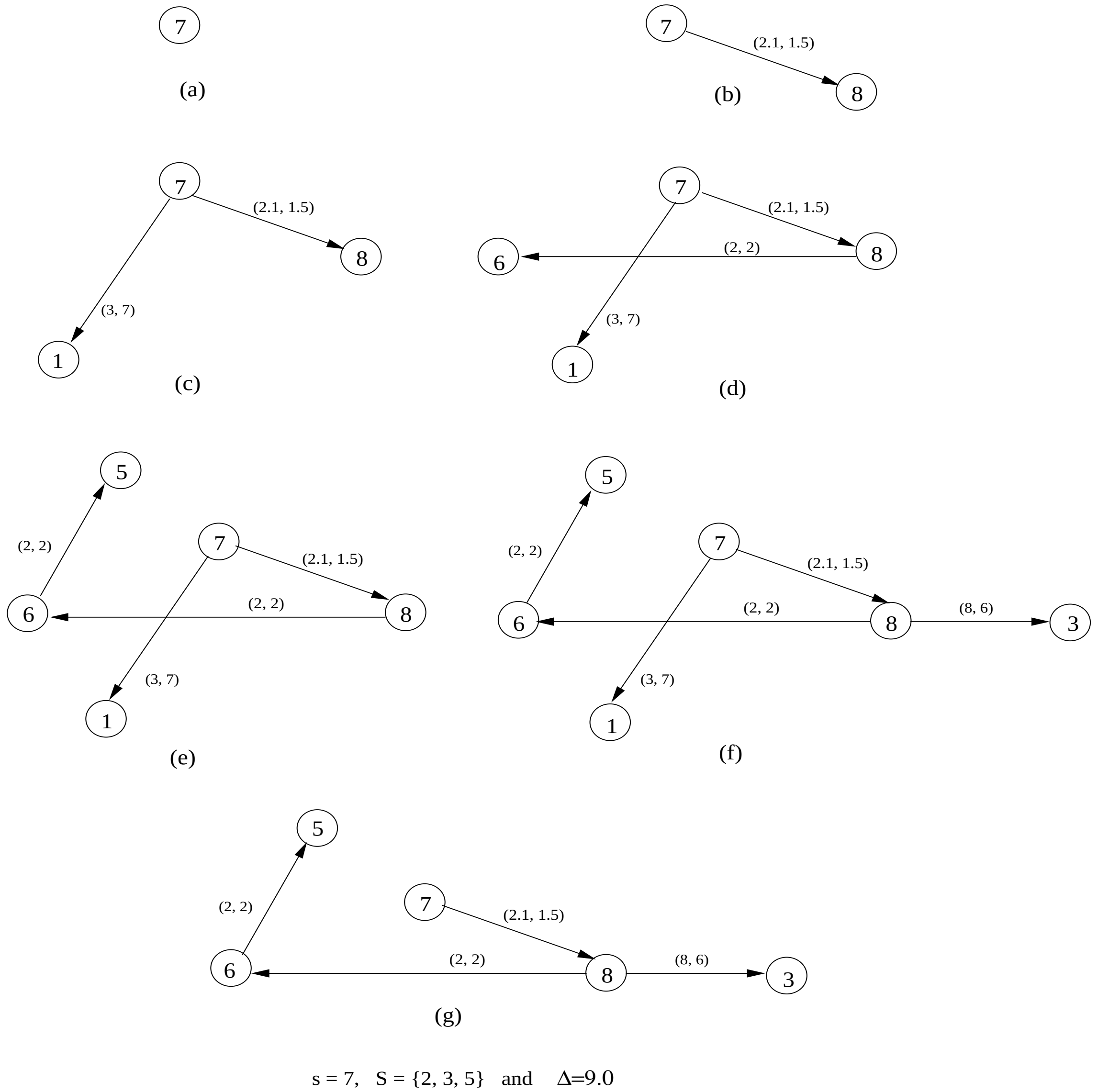

Figure 3: The application of CRA to the example graph

resulting tree produced by CRA is the least-cost path in the network between the two nodes.

Proof: If $\triangle$ is ∞, the CRA procedure in Fig.1 becomes essentially the shortest path algorithm shown in [21] if line (24) of CRA in Fig.1 is not considered. Therefore, CRA computes a least-cost path (spanning) tree before executing line (24). The proof completes. $\square$

3.2 The Proposed Algorithm (CKMB)

We now describe how CKMB works. Given a source node s, a set of destination nodes S ($s \notin S$) and a delay tolerance $\triangle$ for a directed network *G(V,E)* with link costs C and link delays D, CKMB takes the five steps shown in Fig.4 to compute the delay-constrained multicast tree rooted at s and spanning all the nodes in S. CKMB works very much like KMB [19]. KMB is an efficient unconstrained minimum Steiner tree heuristic for *undirected* networks. For save of space, we do not describe KMB in any detail here, readers are referred to [19] or [25] for the necessary information.

First, we know that in KMB [19] a closure graph composed of the source node and all destination nodes must be constructed. Step 1, 2 and 3 in Fig.4 try to do similar work. Step 1 of CKMB computes a low cost tree T_1 rooted at s and spanning a set of destination nodes $S_1 (S_1 \subseteq S)$ subject to the delay bound $\triangle$ constraint using CRA. If T_1 has not spanned all of the destination nodes in S, a least delay tree T_2 spanning all those nodes in $S - S_1$ is computed in step 2 using Dijkstra's shortest path algorithm [18][1]. Step 1 and Step 2 compute a directed path for every destination node $i \in S$ from the source node s. The delay $d_s(i)$ of the path from s to i (for all $i \in S$) is bounded above by $\triangle$.

In step 3, we first get a simplified network G_1 by deleting s and all links connecting s in G. In G_1, a set of trees rooted at each destination $v \in S$ and spanning some (or possibly all or possibly none) of other destination nodes S - $\{v\}$ are then computed using CRA. Each time we select the destination node v with

$$d_s(v) = min\{d_s(i)\} \text{ for } i \in S$$

and compute a tree rooted at this node v with the delay constraint of $\triangle - d_s(v)$ using CRA if $d_s(v) < \triangle$. After this, we adjust $d_s(i)$ for all $i \in S$ according to the method described in step 3. Step 3 attempts to compute a set of paths from every destination node to all other destination nodes subject to a delay bound constraint which is less than $\triangle$ by using CRA.

Step 1, 2 and 3 generate a directed graph G' with the nodes being $\{s\} \cup S$. The links of this graph represent the corresponding paths generated. Because paths between some destination nodes may

[1] According to Lemma 2, T_1 may not span all nodes in S. The importance of Step 2 will be seen in Lemma 4. In [8], we combine T_1 and T_2 to give a constrained multicast routing tree, whose cost is unfortunately still large.

not exist (i.e. these paths can not be generated in step 3), graph G' may not be a closure graph.

Next, we recall that in KMB a minimum spanning tree of the closure graph must be computed. In our case, however, (a) G' may not be a closure graph and (b) even if G' is a complete graph we can not find an optimal algorithm computing the minimum cost delay-constrained spanning tree for G' within a reasonable computation time because the problem of finding a delay-constrained minimum spanning tree in a directed graph is also NP-complete [16]. Therefore, we must design an efficient heuristic to compute a delay-constrained spanning tree of G'. Fortunately, CRA can be easily modified to fulfil this purpose (please see Step 4 in Fig.4 for detail).

Finally, we remember that in KMB the links of the spanning tree of the closure graph are expanded into original paths in the network to form a solution. In step 5 of our algorithm, the links of the constrained spanning tree are expanded into the corresponding paths they represent to give a constrained multicast tree. Step 5 of our algorithm is exactly the same last step of the delay-constrained multicast heuristic proposed in [2]. Note that loops may appear in the expansion process, however, it is very easy to remove these loops. In fact, this step needs only O(mn) time to expand links and remove loops (please see also [2]).

Steps taken by $CKMB(s, S, \triangle, C, D)$:
/*
s is the source node, S is the set of destination nodes. $\triangle$ is the bounded delay. C and D are link costs and link delays of the network G respectively.
*/

Step 1: Call $CRA(s, S, \triangle, C, D)$ to compute a low cost tree T_1 rooted at s. Let $S_1 (S_1 \subseteq S)$ denote the set of destination nodes spanned by T_1 and $d_s(i)$ the delay of the path from s to the node $i \in S_1$ in T_1.

Step 2: If $S_1 \subset S$, compute a least delay tree T_2 rooted at s and spanning all the nodes in $S - S_1$. Let $d_s(i)$ denote the delay of the path from s to the node $i \in S - S_1$ in T_2.

Step 3: Simplify G to G_1 by deleting s and all links connecting s in G. Let C_1 and D_1 denote link costs and link delays of G_1 respectively.
Find node v such that $d_s(v) = min\{d_s(i)\}$ for $i \in S$;
while $d_s(v) < \triangle$ **do**
    call $CRA(v, (S - \{v\}), (\triangle - d_s(v)), C_1, D_1)$ to compute a tree $T_v(V_v, E_v)$;
    if E_v is not empty, then
        let $d_v(i)$ denote the delay of the path in T_v from v to $i \in S - \{v\}$;
        for all $i \in S - \{v\}$ with $d_s(i) < \triangle$
            if $d_s(i) > d_s(v) + d_v(i)$ then $d_s(i) \leftarrow d_s(v) + d_v(i)$;
    set $d_s(v) \leftarrow \triangle$;
    find node v such that $d_s(v) = min\{d_s(i)\}$ for $i \in S$;
end while

/*
Step 1, 2 and 3 generate a directed graph G' with the nodes being $\{s\} \cup S$. The link (v_1, v_2) in G' represents the corresponding path from v_1 to v_2 generated from step 1 to step 3. Note that G' may not be a complete graph.
*/

Step 4: Construct a spanning tree of G' satisfying the delay constraint $\triangle$:
Let C' and D' represent the link costs and link delays of G' respectively.
Replace line (16) of the procedure CRA (see Fig.1):

$$vc \leftarrow CST(u) + C(u, w);$$

with

$$vc \leftarrow C(u, w);$$

Let this changed version of CRA denoted by CRA'.
Call $CRA'(s, S, \triangle, C', D')$ to compute a spanning tree of G' rooted at s.

Step 5: Expand the links of the constructed constrained spanning tree into the corresponding paths they represent (and remove any loops that may be caused by this expansion) to give the constrained multicast routing tree solution.

Figure 4: The proposed algorithm CKMB

Fig.5 illustrates the application of the CKMB to the same graph in Fig.2, with s=*7*, S={*2,3,5*} and $\triangle$=9.0. The tree in *(a)* of Fig.5 is generated by step 1 of CKMB (see Fig.3). Because node *2* is not in *(a)* of Fig.5, a least delay tree from node *7* to node *2* is computed in step 2 (see *(b)*). Step 1 and step 2 generate the paths from the source node *7* to all of the destination nodes *2,3,5* satisfying the delay constraint:

path *7-8-6-5* from node *7* to node *5* with path delay $d_7(5) = 5.5$ (from Step 1)

path *7-8-3* from node *7* to node *3* with path delay $d_7(3) = 7.5$ (from Step 1)

path *7-8-6-1-2* from node *7* to node *2* with path delay $d_7(2) = 6.7$ (from Step 2)

In Step 3, because $d_7(5) = min\{d_7(2), d_7(3), d_7(5)\}$ we first try to compute a tree rooted at node *5* and spanning {*2,3*} with a delay bound of $\triangle - d_7(5)$=3.5 using CRA. It is found soon that no such tree exists. $d_7(5)$ is then set to $\triangle$. Now because $d_7(2) = min\{d_7(2), d_7(3), d_7(5)\}$ we try to compute a tree rooted at node *2* and spanning {*3,5*} satisfying a delay bound of $\triangle - d_7(2)$=2.3 using CRA. The computed tree contains only the link *(2,3)*. Because $d_7(2) + d_2(3) = 6.7 + 1.5 > d_7(3)$ we need not adjust $d_7(3)$. $d_7(2)$ is also set to $\triangle$. Finally we try to compute a tree rooted at node *3* and spanning {*2,5*} satisfying a delay bound of $\triangle - d_7(3)$=1.5. No such tree can be produced by CRA. The result of step 3 is shown in *(c)* of Fig.5. Step 1, 2 and 3 generate a directed graph G' with nodes of {*7,2,3,5*} and the links shown in *(d)* of Fig.5. In step 4, a spanning tree of G' rooted at node *7* is generated using a modified version of CRA (see *(e)*). The resulting constrained multicast tree with the cost of 13.4 is produced in step 5 (see *(f)*).

Lemma 4: CKMB can always find a constrained multicast routing tree in the network G if one exists.

Proof: If there exists a constrained multicast routing tree in G, step 1 and step 2 of CKMB can always find a path for each multicast group member from the source node subject to the delay bound constraint because of the least delay tree computed in step 2. In this case, step 4 will construct a tree spanning the source node and all of the multicast group nodes subject to the delay bound constraint (in worst case the links from the source node to each of the multicast group members are chosen in G' in step 4), therefore, step 5 will give a constrained multicast routing tree solution from the computed constrained-spanning tree rooted at the source node (note that although possible loops may appear in the process of expanding the links of G', Step 5 can easily

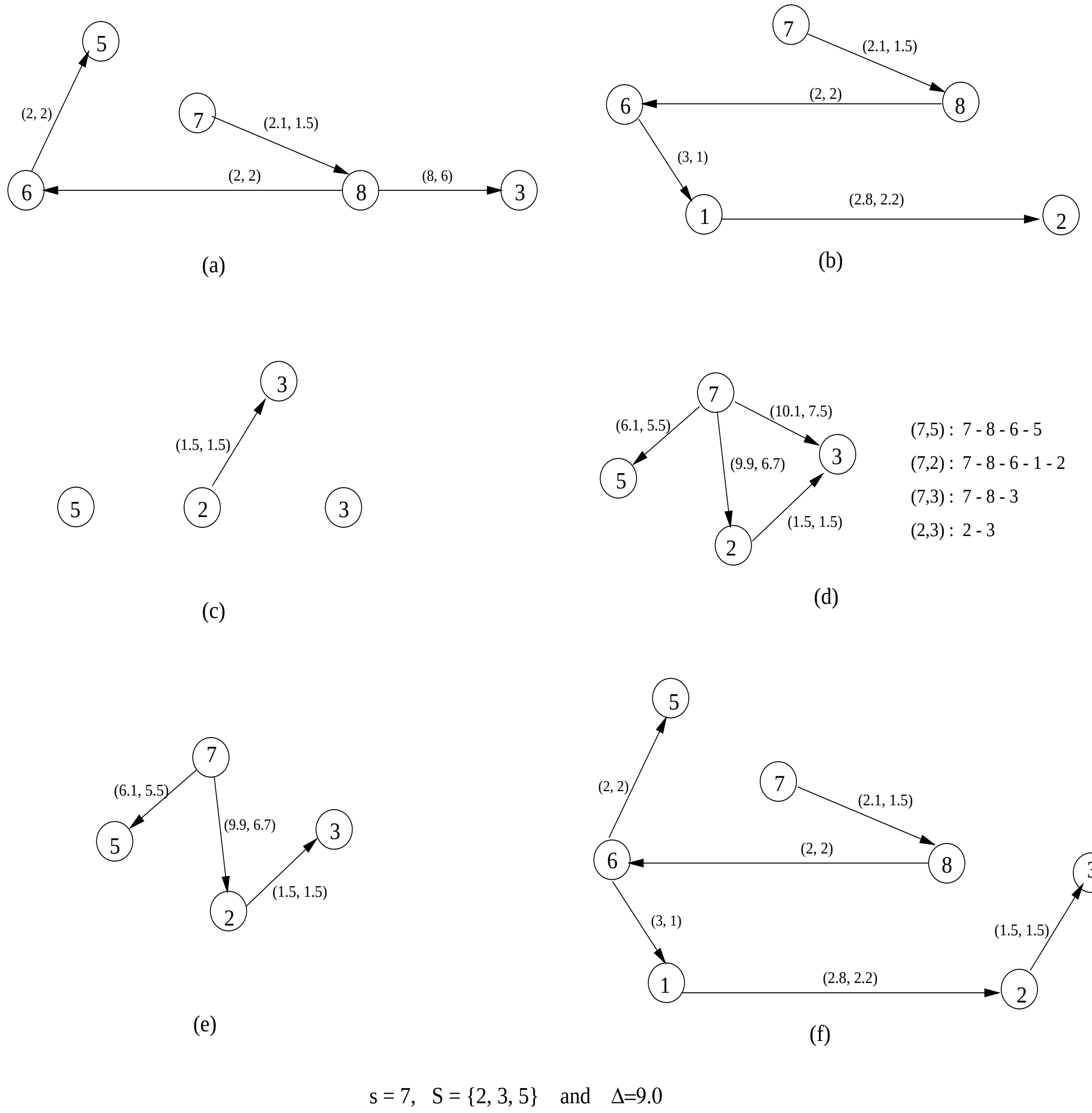

Figure 5: The application of CKMB to the example graph

break such loops. In fact, it takes only O(mn) time for Step 5 to expand links and break loops (see also [2]). For save of space, we do not give the algorithm of Step 5 in this paper). □

Lemma 5: The time complexity of CKMB is O(mn^2).

Proof: Step 1 needs O(n^2) time because CRA is called. Step 2 also has a time complexity of O(n^2) because the least delay tree can be computed using Dijkstra's shortest path algorithm [18]. The running time in step 3 of CKMB is dominated by the *while* loop, which is executed m times. Each execution of this loop requires O($(n-1)^2$) time because CRA is executed for a simplified network with $(n-1)$ nodes. Step 4 needs O(n^2) for executing the modified version of CRA. Step

5 takes O(mn) time to expand the links of the constructed constrained spanning tree *and* break any possible loops (see also [2]). Therefore, for CKMB the majority of the time is spent in step 3, which needs time O(mn^2). □

Theorem 1: For *undirected* and connected networks, the cost of the minimum spanning tree for the closure graph computed by KMB[2] and the cost of the spanning tree computed by CKMB in Step 4 are the same if $\triangle$ is ∞.

Proof: According to Lemma 3, Step 1 and Step 3 of CKMB will construct a closure graph G' where the edge *(u,v)* from u to v in G' is the least cost path from u to v in the original undirected and connected graph if $\triangle$ is ∞. Obviously, CKMB and KMB compute the same closure graph (see the KMB algorithm in [19]). In Step 4 of CKMB, we begin with the source node and then each time generate a new node so that the cost of the link from one of the already computed nodes to the new node has the least value among all the links from any already computed node to any other node to be computed (note that link delays do not play any role in Step 4 if $\triangle$ is ∞). According to the principle of computing the minimum spanning tree for undirected networks in [22], Step 4 of CKMB will generate a minimum spanning tree of G'. The proof completes. □

Lemma 5 and Theorem 1 show that if $\triangle \rightarrow \infty$ CKMB can construct very low cost trees[3] [4] for undirected and connected graphs within a short computation time. As will be shown in the next section, CKMB is also a very efficient algorithm for computing delay-constrained multicast trees for directed networks.

In concluding this section, we shortly compare CKMB with the delay-constrained multicast algorithm proposed by Kompella *et al* [2]. In the following we denote Kompella's algorithm as KPP. KPP is also based on KMB, however, KPP assumes $\triangle$ and link delays are all integer values and uses a dynamic programming approach similar to Floyd's shortest path algorithm [24] to compute the closure graph in O($\triangle n^3$) time. If $\triangle$ and link delays take real values, the computation of the

[2] Note that in KMB only one link metric, namely link cost, is considered.

[3] In practice, as also specified in [23], simply expanding the links of the minimum spanning tree usually gives a tree (without loops) for undirected networks, in this case CKMB and KMB give the same multicast tree cost.

[4] In this paper we consider each network link has cost and delay without any relationship between them, theorem 1 has shown that CKMB can perform much better than the shortest path algorithm even if the link cost and link delay are the same when the end-to-end delay constraint is relatively large for undirected networks.

closure graph takes exponential time. In contrast, CKMB computes G' based on CRA and has no assumptions on the values of $\triangle$ and link delays. The other difference between KPP and CKMB is that CKMB uses simply a modified version of CRA to compute the constrained spanning tree of G', while KPP uses a greedy approach to compute the constrained spanning tree of the closure graph. These differences mean that CKMB runs much faster than KPP, nevertheless, the cost performance of CKMB is about the same as that of KPP, as will be shown in the next section.

4 Performance Studies

Unlike unconstrained algorithms, delay-constrained multicast algorithms are very difficult to derive their cost performance through *analytical* method (if not impossible). However, if $\triangle = \infty$, delay-constrained algorithms become essentially unconstrained algorithms. As already specified in the last section, the cost performance of CKMB will approach that of KMB in undirected networks if $\triangle \rightarrow \infty$. In this section, we use simulation experiments to evaluate the performances of various delay-constrained multicast algorithms for moderate $\triangle$ values in directed networks. We have integrated our proposed algorithm into the multicast routing simulator MCRSIM [14] developed at North Carolina State University and used this simulator to test most of the existing delay-constrained multicast routing algorithms proposed in [8] (called CDKS in the following), [6] (called MSC), [2] (called KPP), CAO [5] and BSMA [7]. Both CDKS and MSC have a time complexity of O(n^2). KPP takes O($\triangle n^3$) time if link delays and $\triangle$ take integer values. However, if the link delays and $\triangle$ take real values, it is necessary to limit the granularity of the computation [5] [15] and in this case the computation time of KPP is very large. CAO has an exponential computation time, but CAO's running time is moderate for small networks and small average network node degree. BSMA takes O($kn^3\log n$) where the value of k depends on the network size and node degrees (see [7] for detail). The Dijkstra's shortest path algorithm [18], called LD, was also included for the performance comparison. LD takes O(n^2) time.

In the simulation, full duplex ATM networks with homogeneous link capacities of 155 Mbps (OC3) were used. The positions of the network nodes were kept in a rectangle of size 3000 * 2400 Km^2, roughly the area of the USA. A random graph generator based on Waxman's generator [25] is used to create networks in MCRSIM. 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 any two nodes is a function of the distance between these two nodes. The created networks are always connected (see [14]), and the degree of each network node is ≥ 2. Most of our experiments were conducted on networks with an average node degree of 4 through adjusting the parameters of the random graph generator[5].

Each node in the created network represented a non-blocking ATM switch. Each link was assumed to have a small output buffer. The propagation speed through the links was taken to be two thirds the speed of light. The propagation delay was dominant under these conditions, and the queuing component was neglected when calculating the link delay $D(e)$. In the simulation, the link cost $C(e)$ was set to be the total currently reserved bandwidth on the link, which is a suitable measure of the utilization of both the link's bandwidth and its buffer space. A link can be used in a new multicast session and reserve bandwidth for it only if the sum of the total currently reserved bandwidth 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 purposes of our study of the multicast routing algorithms.

Real-time applications usually have a delay requirement of less than 150 *ms* [26]. Most of our simulation experiments used a value of 50 ms for $\triangle$ which represents only an upper bound on the end-to-end propagation time across the network. We think this value is appropriate for the evaluation of the different constrained multicast routing algorithms.

For each run of the experiment we generated a given number of nodes at random locations and a random set of links to interconnect these nodes using the graph generator described above (for comparison, the experiments conducted in [15] were only for networks with a random set of links connecting nodes at *fixed* locations). Random background traffic for each link was also generated. The equivalent bandwidth of each link's background traffic is a random variable uniformly distributed between $B\text{min}$ and $B\text{max}$. As the difference between $B\text{min}$ and $B\text{max}$ increases, the asymmetry of the link loads also increases. The routing algorithm was applied to create a con-

[5]We used the same parameters specified in the source file *router1.c* of MCRSIM (see [14])

strained multicast tree for a random source generating video traffic with an equivalent bandwidth of 0.5 Mbps and a group of randomly chosen destination nodes. We measured the cost of the created tree and the running time of the routing algorithm. The experiment was run repeatedly until confidence intervals of less than 5%, using 95% confidence level, were achieved for the mentioned measured quantities. Because BSMA has a near optimal performance regarding the resulting tree cost [7], unless otherwise specified, we will show the percentage increase in the total cost of other constrained algorithms relative to the total cost of BSMA for the tree cost measure. For the (average) execution time measure it should be noted that the code used for the algorithms was not optimized for speed [15]. In implementing CKMB, we carefully checked the source code of KPP in MCRSIM and used similar data structures in KPP. CKMB even used the same source code of KPP in MCRSIM in the last step to expand the links of the spanning tree and break any possible loops (see section 3). Like [15] we chose a granularity of $\triangle/10$ for KPP in all the simulation experiments. Because the execution time of BSMA is much larger than all other algorithms (except for small networks and small group size) in the simulation experiments, we will not show BSMA's execution time measure in the following.

Fig.6 [6] shows the mentioned measures versus network size for Bmin=10 Mbps, Bmax=120 Mbps, average node degree=4.0, delay bound=50 ms and a fixed group size of 15. It is observed that CAO yields a cost close to that of BSMA. KPP and CKMB have about the same cost performance. When network size increases, CKMB performs slightly better than KPP. The costs of CKMB and KPP are approximately 8% and 10% worse than BSMA at the network size of 100 nodes respectively. The cost performances of CDKS, MSC and LD are significantly worse than BSMA. The costs of CDKS, MSC and LD are over 25%, 48% and 65% worse than BSMA respectively. The running

[6] We have run the simulation experiments two times. For the first time we have also measured other quantities like average end-to-end delay, maximum end-to-end delay, number of tree links etc. besides cost and average execution time for the specified confidence interval. In the meantime, the simulation software was extended by us to include also other (unicast and dynamic multicast) routing algorithms for future evaluation purposes. We have again run the simulation and measured only the cost and execution time measures for the second time. The cost performances of all constrained algorithms for both times are exactly the same, the execution time growth rate remains also about the same. We show the results of simulation experiments conducted for the second time in this paper. All simulation experiments were conducted on a SUN SPARCstation 5.

times of CAO and KPP are much larger than that of CKMB, MSC, CDKS and LD. When the network size increases, CKMB has the slowest growth rate.

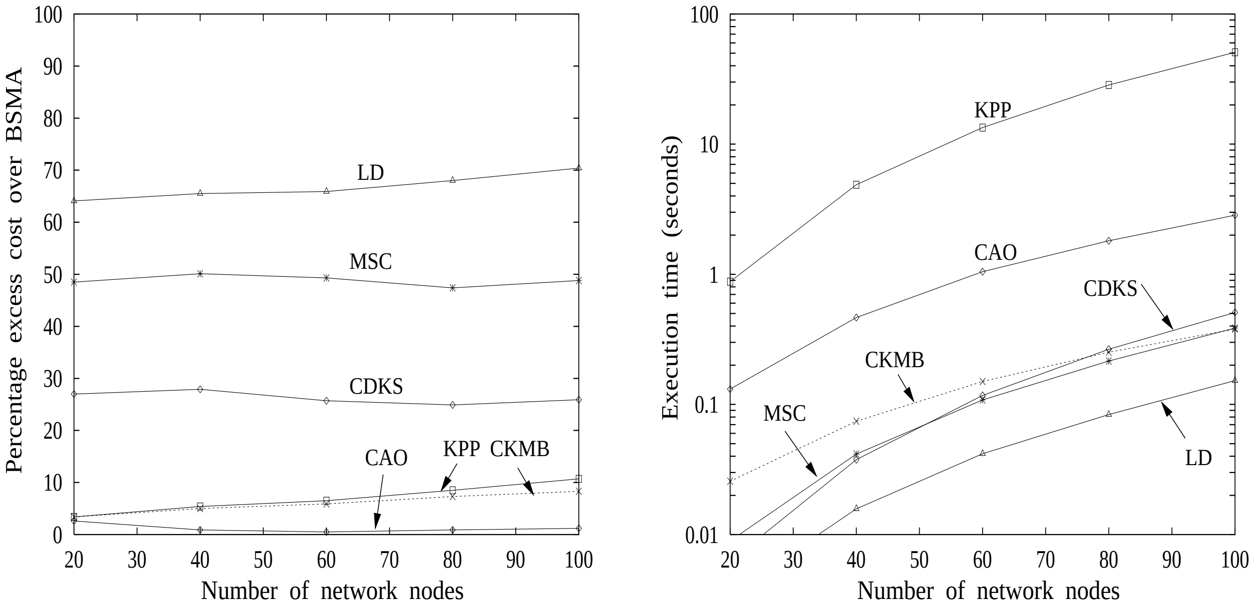

Figure 6: Percentage excess cost (over BSMA) and execution time versus number of network nodes, for *Bmin*=10 Mbps, *Bmax*=120 Mbps, average node degree=4.0, delay bound=50 *ms* and number of group members=15

Fig.7 shows the cost measure and the execution time measure versus group size, for *B*min=10 Mbps, *B*max=120 Mbps, a network size of 60 nodes, average node degree=4.0 and delay bound=50 *ms*. When group size increases, CAO's cost performance becomes slightly worse. CAO's costs are about 7% worse than BSMA at the group size of 57. For small group sizes CKMB performs better than KPP. At the group size of 3 (namely 5% of the total number of network nodes), CKMB's costs and KPP's costs are 2.7% and 7.7% worse than BSMA respectively, while the costs of CDKS, MSC and LD are about 13%, 35% and 53% worse than BSMA respectively. When group size increases, however, the cost performances of CDKS, MSC and LD are up to 29%, 50% and 70% worse than BSMA respectively, while the costs of both CKMB and KPP are less than 10% worse than BSMA. The running times of CAO and KPP are very large and are up to about 10 and 100 times of that CKMB respectively. When group size increases, the running times of both CAO and CKMB also grows, while CDKS, MSC, LD and KPP remain almost unchanged. LD's execution time is smaller that that of MSC and CDKS. The running times of MSC and CDKS are about the same.

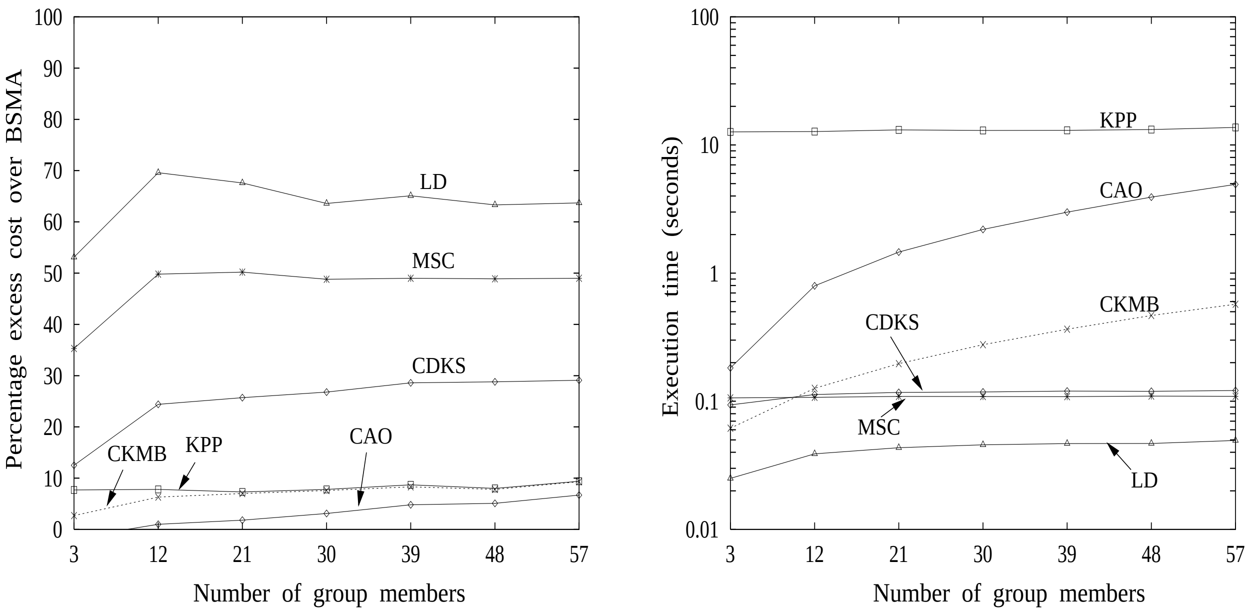

Figure 7: Percentage excess cost (over BSMA) and execution time versus group size, for *Bmin*=10 Mbps, *Bmax*=120 Mbps, number of network nodes=60, average node degree=4.0 and delay bound=50 *ms*

Fig.8 shows the two performance measures versus the delay bound for *B*min=10 Mbps, *B*max=120 Mbps, number of network nodes=60, average node degree=4.0 and number of group members=15. For the very stringent delay bound of 20 *ms* (a value close to the largest propagation delay between the nodes in the network), we observed that the costs of CDKS, MSC and LD are only about 16%, 18% and 30% worse than BSMA respectively. the costs of CAO, KPP and CKMB are about 6%, 8% and 11% worse than BSMA respectively. Under this very small delay bound condition the probability that the resulting trees generated by CRA in CKMB span all of the group members becomes small, therefore CKMB's cost performance is slightly worse than KPP but is still better than CDKS. It is interesting to see that for this small delay bound the execution time of CKMB is very small. When delay bound increases, CKMB performs better than KPP. For delay bounds greater than 40 *ms* CKMB's costs are only $\leq$ 8% (only 3.8% for the delay bound of 120 ms) worse than BSMA , but it runs about 10 and 45 times faster than CAO and KPP respectively! When delay bound is greater than 60 *ms*, the cost performances of CDKS, MSC and LD are over 27%, 50% and 68% worse than BSMA respectively.

Fig.9 shows the cost and the running time measures versus average node degree for *B*min=10

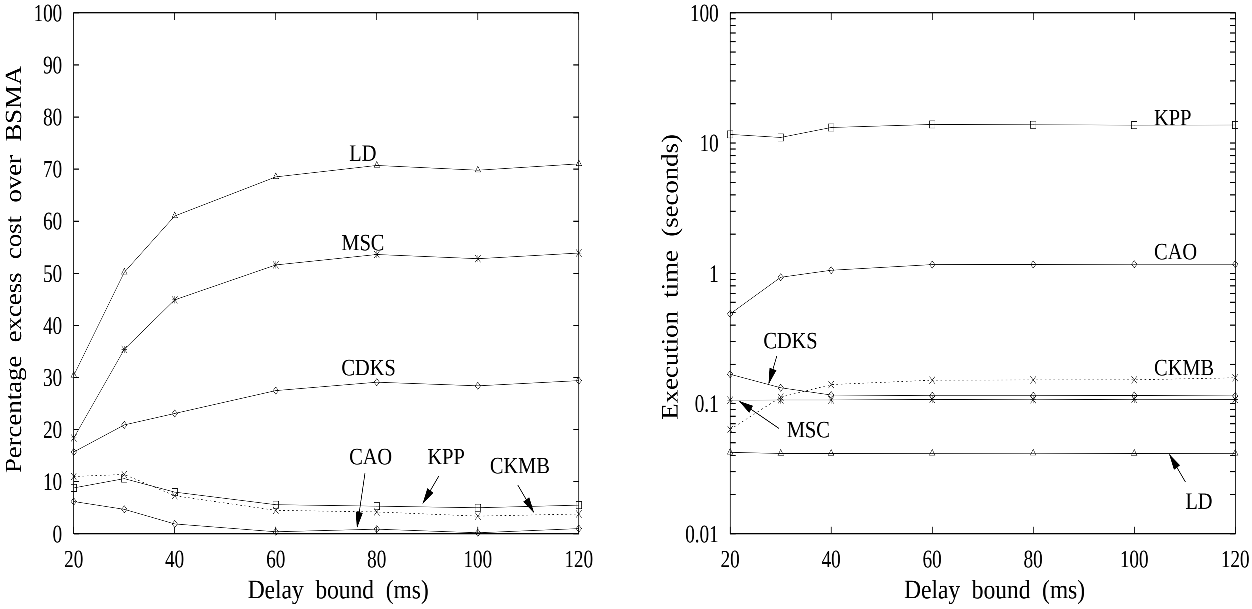

Figure 8: Percentage excess cost (over BSMA) and execution time versus delay bound, for $Bmin$=10 Mbps, $Bmax$=120 Mbps, number of network nodes=60, average node degree=4.0 and number of group members=15

Mbps, Bmax=120 Mbps, number of network nodes=60, delay bound=50 ms and number of group members=15. We see that the cost measures of CAO, CKMB and KPP remain about the same as the average node degree changes. However, the cost performance of CDKS, MSC and LD become worse, especially the percentage excess cost of LD and MSC over BSMA grows sharply as the average node degree increases. The average running times of CDKS, MSC, CKMB and BSMA remain almost unchanged for node degrees greater than 4. The running times of LD and KPP increases slightly as the average node degree increases. CAO's running time increases almost linearly as the node degree increases.

Finally, we will see how well CKMB performs for relatively large networks. Because of the large simulation time, we have run simulation experiments only for the algorithms CKMB and LD. Fig.10 shows the cost performance for Bmin=10 Mbps, Bmax=120 Mbps, average node degree=4.0, delay bound=50 ms and group size equal to 10% of the number of network nodes. Fig.10 shows that CKMB performs still much better than LD in large networks.

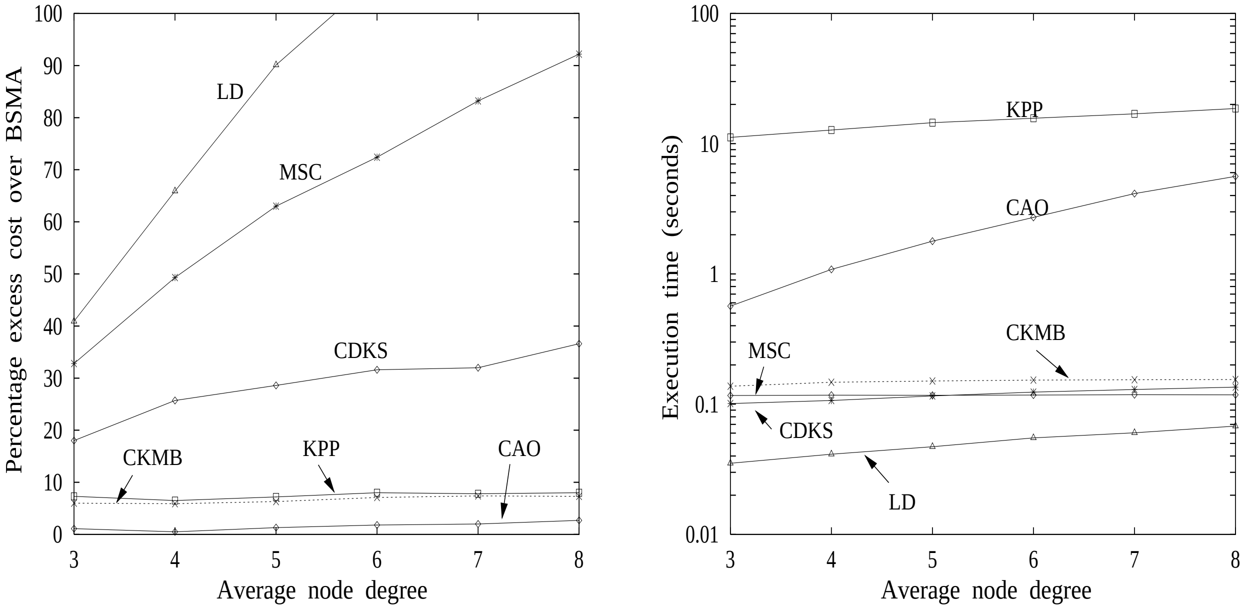

Figure 9: Percentage excess cost (over BSMA) and execution time versus average node degree, for *Bmin*=10, *Bmax=120*, number of network nodes=60, delay bound=50 *ms* and number of group members=15

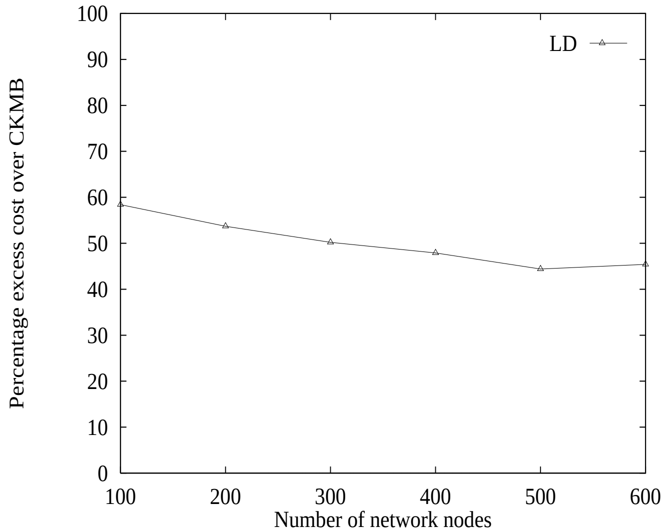

Figure 10: Percentage excess cost over CKMB versus number of network nodes, for *Bmin*=10 Mbps, *Bmax*=120 Mbps, average node degree=4.0, delay bound=50 *ms* and number of group members equal to 10% of the number of network nodes

5 Conclusions

Multicast routing algorithms satisfying stringent delay bound constraint are becoming increasingly important. We proposed an efficient delay-constrained multicast algorithm in this paper. Compared with the existing delay-constrained algorithms, the proposed algorithm results in a very good tradeoff between optimality and complexity.

The proposed algorithm may be very well suited for comparing cost performances of the existing or future multicast routing algorithms providing end-to-end delay guarantee in multicast routing simulation environments that simulate relatively large networks and consider realistic conditions. The proposed algorithm is also useful for routing real-time multicast data in high-speed networks if adequate global information is available to the source node.

Acknowledgment

The authors would like to thank H. F. Salama, D. S. Reeves and Y. Viniotis at North Carolina State University for making their MCRSIM simulation software available. We especially thank H. F. Salama for sending us the source code of CDKS and some useful discussions about MCRSIM.

Availability

The source code of CKMB can be found at URL:

`http://www.cs.tu-bs.de/~sun/routCST.c`

References

[1] Kompella, V. P., et al, Multicasting for Multimedia Applications, *IEEE INFOCOM'92*, pp.2078-2085, 1992

[2] Kompella, V.P., et al., Multicast Routing for Multimedia Communication, *IEEE/ACM Tansactions on Networking*, 1(3),pp.286-292, 1993

[3] Kompella, V.P., et al., Two Distributed Algorithms for the Constrained Steiner Tree Problem", in *Proceedings of the Second International Conference on Computer Communications and Networking*, pp.343-349, 1993,

[4] Wi, S. and Choi, Y., A Delay Constrained Distributed Multicast Routing Algorithm, *12th International Conference on Computer Communication, ICCC'95*, pp.833-838, 1995

[5] Widyono, R., The Design and Evaluation of Routing Algorithms for Real-Time Channels. Technical Report TR-94-024, Tenet Group, Department of EECS, University of California at Berkeley, 1994

[6] Waters, A.G., A new heuristic for ATM multicast routing, *Proceedings of 2nd IFIP Workshop on Performance Modelling and Evaluation of ATM Networks*, pp.8/1 - 8/9, July 1994

[7] Zhu, Q., et al, A Source-Based Algorithm for Near-Optimum Delay-Constrained Multicasting, *IEEE INFOCOM'95*, pp.377-385, 1995

[8] Sun, Q. and Langendőrfer, H., Efficient Multicast Routing for Delay-Sensitive Applications, in *Proceedings of 2nd international Workshop on Protocols for Multimedia Systems (PROMS'95)*, pp.452 - 458, 1995

[9] Waters, A.G. and Crawford, J.S., Low-cost ATM Multicast Routing with constrained Delays, Internal Report, Computer Laboratory, University of Kent at Canterbury, UK, Jan. 1996

[10] Doar, M. and Leslie, I., How Bad is Naive Multicast Routing, in *proceedings of INFOCOM'93*, pp.82-89, 1993

[11] Wei, L. and Estrin, D., The Trade-offs of Multicast Trees and Algorithms, Internet Draft, draft-ietf-idmr-mtree-01.ps, March 24, 1995

[12] Biersack, E. and Nonnenmacher J., WAVE: A New Multicast Routing Algorithm for Static and Dynamic Multicast Groups, in *proceedings of 5th Workshop on Network and Operating System Support for Digital Audio and Video*, 1995

[13] Rajagopalan, B. and Nair, R., Quality of Service (QoS)-based Routing in the Internet - some Issues, Internet draft, Oct. 1996

[14] Salama, H.F., et al, MCRSIM simulator source code and MCRSIM User's Manual, Center for Advanced Computing and Communication, North Carolina State University, Raleigh, 1995 (available by anonymous ftp from *ftp.csc.ncsu.edu/pub/rtcomm*)

[15] Salama, H.F., et al, Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks, in *High Performance Networking VI, IFIP 6th International Conference on High Performance Networking*, pp.27-42, 1995 (An extended version to appear in *IEEE Journal of Selected Areas in Communications* in 1997)

[16] Salama, H.F., et al, The Delay-Constrained Minimum Spanning Tree Problem, Technical Report TR-96/17, Center for Advanced Computing and Communication, North Carolina State University, USA, June 1996

[17] Wang, Z. and Crowcroft, J., QoS Routing for Supporting Resource Reservation, Tech. Doc., Dept. of Computer Science, University College London, London WC1 6BT, UK, 1994

[18] Dijkstra, E.W., A note on two problems in connection with graphs, *Numerische Mathematik*, vol. 1, pp.269-271, 1959

[19] Kou, L., et al, A fast algorithm for Steiner trees, *Acta Informatica*, No.15, pp.141-145, 1981

[20] Lawler, E., Combinational Optimization: Networks and Matroids, Holt, Rinehart and Winston, 1976

[21] Horowitz, E. and Sahni, S., Fundamentals of Computer Algorithms, Computer Science Press, Inc. pp.183-188, 1978

[22] Sedgewick, R., Algorithms, 2. ed., Addison-Wesley, 1988

[23] Esbensen, H., Computing Near-Optimal Solutions to the Steiner Problem in a Graph Using a Genetic Algorithm, *Networks*, Vol.26, pp.173-185, 1995

[24] Floyd, R.W., Algorithm 97: Shortest Paths", *Commun. ACM*, vol.5, pp.345, 1962

[25] Waxman, B.M., Routing of Multipoint Connections, *IEEE Journal of Selected Areas in Communications*, 6(9),pp.1617-1622, 1988

[26] Rampal, S. and Reeves, D.S., Routing and Admission Control Algorithms for Multimedia Traffic, Dept. of Computer Science, North Carolina State University, Raleigh, 1995 (available by anonymous ftp from *ftp.csc.ncsu.edu/pub/rtcomm*)