Efficient Multicast Routing for Delay-Sensitive Applications

Quan Sun Horst Langendőrfer

Institute of Operating Systems and Computer Networks
Technical University of Braunschweig
Bűltenweg 74/75, 38106 Braunschweig, Germany
Email: {sun,langendf}@ibr.cs.tu-bs.de
FAX: +49 531 391 5936

Abstract

We give a multicast tree construction method for delay-sensitive applications by first trying to compute a constrained low cost tree spanning as many destination nodes as possible and then applying the shortest path algorithm. Experimental results through simulations show that this method greatly improves the multicast tree cost measure in comparison with the existing shortest path routing schemes, while it maintains a short computation time.

Key words: Multicast routing, Tree cost, Delay bound

1 Introduction

Multimedia communication often requires multicast support. Moreover, multimedia applications are often high bandwidth and delay sensitive. Therefore it is very important to consider the efficiency and quality of multicast delivery trees [1].

There are generally two types of multicast trees, namely group shared trees and source-based trees. Steiner trees [2] and core-based trees (CBT) [3] [4] are typical examples of group shared trees [1]. The main objective of constructing Steiner trees and core-based trees is to achieve optimal tree cost and to get favourable scaling characteristic respectively. Generally speaking, group shared trees are not appropriate for delay-sensitive multimedia applications because of the possible large end-to-end delays. Besides, group shared trees result in traffic concentrations [1] [5].

Sender-based shortest path trees (e.g. trees generated by DVMRP [6] and MOSPF [7]) are examples of source-based trees. Source-based trees usually contain a shortest delay path between source and each destination and are therefore appropriate for delay-sensitive applications. The main shortcomings of source-based trees are the poor scaling property and the possible high cost of the resulting tree. Recently Deering *et al* [5] proposed a new multicast routing scheme (PIM), attempting to improve the scaling for some particular cases while preserving the basic sender-based tree structure, but the routing scheme is very complex.

Cost, delay and traffic concentration are three important measures to judge the quality of a tree. In this paper we consider delay-sensitive applications. Because source-based trees generally have good delay and traffic concentration measures, we adopt the source-based tree structure. The aim of the paper is to construct low cost sender-based multicast trees with bounded delay constraints for each source-to-destination path in a short computation time.

In the past, the problem of constructing low cost trees with bounded end-to-end delay constraints was studied by Kompella *et al* [10] [11], Widyono [12], Zhu *et al* [13], Wi and Choi [14], etc. Some of these existing algorithms are centralized ones, while others compute route in a distributed way. For distributed routing

algorithms, it is not needed to maintain the whole network status in each node, but distributed algorithms are usually slow and complex. It should be pointed out that it is very important to consider dynamic groups for multicast routing schemes. In past years, the problem of multicast routing considering dynamic multicast groups was studied by Waxman [2], Doar *et al* [15], etc. But these existing dynamic algorithms (and their performance studies) have not explicitly considered the delay constraints. The first detailed, quantitative evaluation of various static multicast routing algorithms under realistic conditions was done most recently by Salama *et al* [16]. In that paper, various delay-constrained and -unconstrained algorithms were evaluated. From their work, it is easy to see that, on one hand, although most of the existing important constrained algorithms have similar good performance (they can construct low cost trees satisfying the given delay bound and can manage network resources efficiently), they are too slow , without modification, to be used in large networks. On the other hand, the unconstrained multicast routing algorithms can not be applied to real-time applications due to their inadequate delay performance. However, it is interesting to notice that one of the simple unconstrained algorithms, the Dijkstra's shortest path algorithm minimizing the cost of the path from the source node to each multicast group member individually, performs relatively well when considering the resulting tree cost, maximum end-to-end delay and computation time altogether.

In this paper, we give a method for constructing delay-constrained multicast trees based on the Dijkstra's shortest path algorithm by first trying to construct a constrained low cost tree and then computing the shortest delay path tree for those multicast group nodes which are not in the constructed constrained routing tree. Although our algorithm is a centralized one considering only static groups, it is expected to be a practical and useful algorithm because of its short computation time and good performance. In the following section we describe the proposed scheme in detail. In section 3 we give the empirical performance analysis.

2 The proposed multicast tree construction method

We use the similar way in [11] to describe the method. Given a directed graph[1] *G=(V,E)* with node set V and edge set E. Two weight functions are defined on edge e ($e \in E$), namely delay *D(e)* and cost *C(e)*. Given a source node s, a set of destination nodes S and a delay tolerance $\triangle$, a constrained spanning tree T is a tree rooted at s and spanning all the nodes in S such that for each node v in S, the delay on the path from s to v is bounded above by $\triangle$:

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

where *P(s,v)* is the path in T from s to v. The constrained Steiner tree can be described as a constrained spanning tree such that

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

This constrained Steiner tree problem is NP-complete. A heuristic with good performance for the problem is proposed by Kompella *et al* [11]. This heuristic takes $O(n^3 \triangle)$ time. In order to reduce the computation time, we propose the following method to construct the routing tree:

Step 1: Trying to compute a low cost spanning tree $T_1 = (V_1, E_1)$ rooted at s and spanning as many destination nodes in S as possible by using Dijkstra's shortest path algorithm [8] with the additional bounded delay $\triangle$ assurance condition for every generated path (See Appendix for detailed description).

Step 2: Computing a shortest delay path tree $T_2 = (V_2, E_2)$ that spans those destination nodes in S that are not in V_1 (by using Dijkstra's shortest path algorithm).

Step 3: Combining T_1 and T_2 to give a multicast routing tree. Note that loops may appear in combining T_1 and T_2. Fig.1 shows such an example. Because T_2 should be included in the resulting tree, some of the edges in T_1 must be removed when loops appear. Loops can be simply detected by checking if T_1 and T_2 contain edges that have one node in common. For example, consider edge *e12* in T_1 and edge *e20* in T_2. Both edges contain the same node 5. In this case, edge *e12* must be removed. Note that edge *e11* must also

[1] We consider graphs with asymmetric edges, i.e. the two edges in opposite directions between two nodes can have different costs and delays. [11] assumes only graphs with symmetric edges.

be removed, because node 4 is neither a multicast group node nor an intermediate node that leads to any other multicast group node after removing edge *e12*. In worst case, all edges in T_1 may be removed.

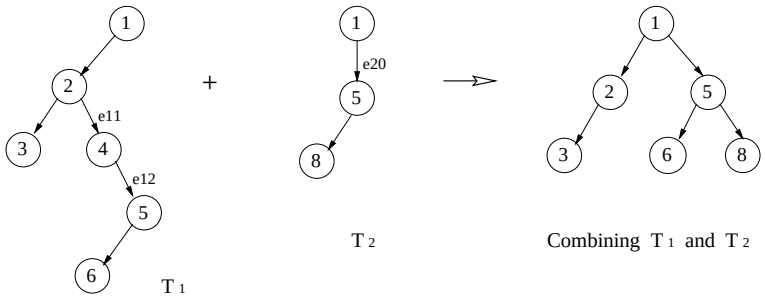

Figure 1: An example of combining T1 and T2

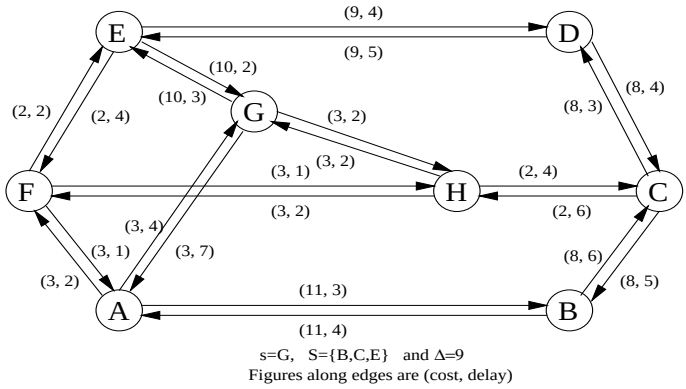

Figure 2: Example graph

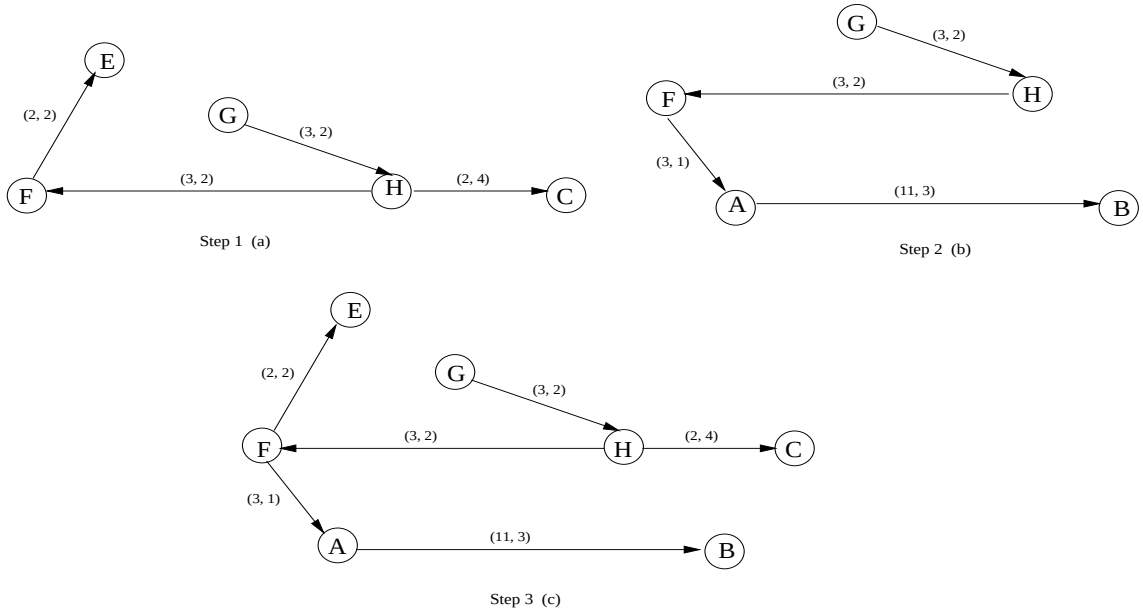

Figure 3: The results of step 1 (a), step 2 (b) and step 3 (c)

It takes $O(n^2)$ time in Step 2, therefore this routing scheme takes much less time than Kompella's routing algorithm to compute the multicast tree.

Fig.2 shows an example graph with source node $s = G$, destination nodes $S = \{B,C,E\}$ and delay bound $\triangle = 9$. Figures along edges are (cost,delay). Fig.3 shows the results of step 1,2 and 3.

3 Performance analysis

We describe the performance of the proposed routing scheme in terms of A_c:

$$A_c = \frac{\sum_{i=1}^{N}(T_{CON_i} - T_{CST_{C_i}})}{\sum_{i=1}^{N} T_{CST_{C_i}}}$$

where T_{CON_i} is the cost of the tree generated by using the proposed routing scheme in the ith run, $T_{CST_{C_i}}$ is the tree cost produced by the heuristic in [11] [2] in the ith run and N is the number of runs. The performance of the shortest delay tree generated by Dijkstra's shortest path algorithm is defined as follows:

$$A_s = \frac{\sum_{i=1}^{N}(T_{SPT_i} - T_{CST_{C_i}})}{\sum_{i=1}^{N} T_{CST_{C_i}}}$$

where T_{SPT_i} is the cost of the shortest delay path tree in the ith run.

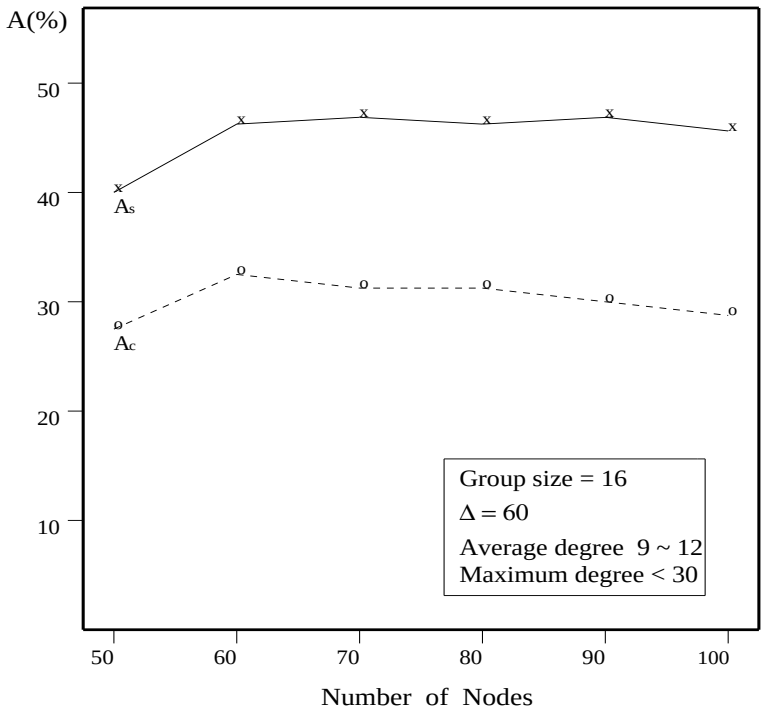

Figure 4: A versus number of nodes

In order to fairly evaluate the proposed routing scheme, we ran it like [11] also on randomly generated graphs with low average node degree, which is common for many point-to-point network topologies, such as NSFNET. The edges of the graph were randomly generated for a given number of nodes of the graph. We used unit edge costs and random edge delays generated uniformly from the set {1,2,3,4,5}. Each experiment generated a number of graphs with identical parameters: number of nodes, maximum degree of each node, delay bound and the size of the multicast group. Each graph was then checked to ensure that a solution existed. The confidence interval for A_c and A_s at the 95% confidence level is within 2% for all the simulation results shown (in our simulation N$\geq$200).

[2] In the simulation, we used graphs of symmetric edges with integer delays and delay bound, because this is assumed by Kompella's algorithm [11].

The performance comparisons between the proposed method and the shortest delay path tree algorithm are shown in Fig.4, Fig.5 and Fig.6. It is shown that the proposed routing scheme performs much better than the shortest delay path tree algorithm.

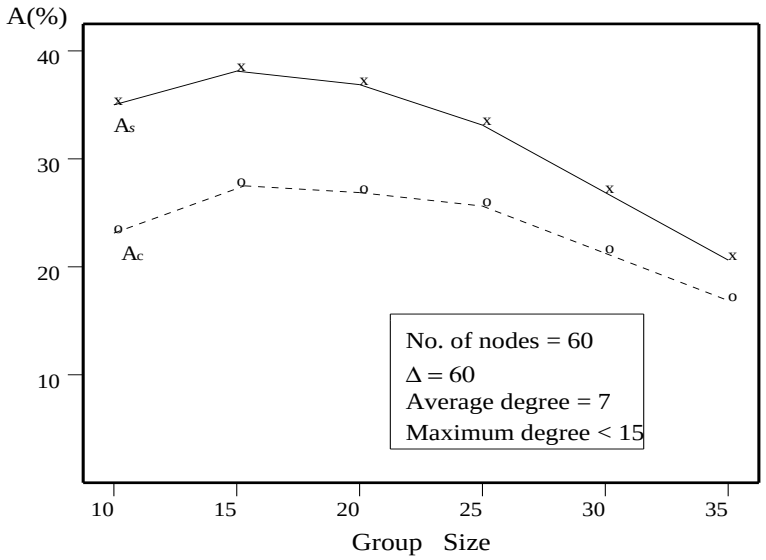

Figure 5: A versus group size

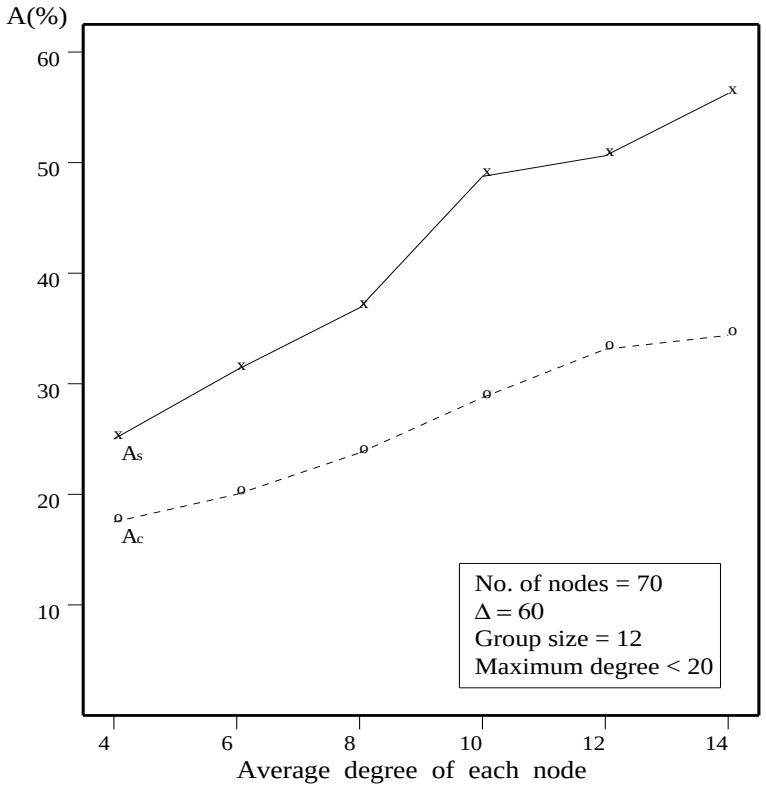

Figure 6: A versus average degree of each node

4 Conclusions

Cost, delay and traffic concentration are three important measures to judge the quality of a multicast routing tree. We give a multicast tree construction method for delay-sensitive applications. The proposed scheme takes much less time than the heuristic proposed by Kompella *et al*, nevertheless it performs much better than the shortest delay path routing and therefore turns out to be a practical routing algorithm for delay-sensitive applications.

5 Acknowledgment

We would like to thank Martina Zitterbart for the information of current Internet multicast routing schemes.

References

[1] Wei, L. and Estrin, D., The Trade-offs of Multicast Trees and Algorithms, Internet Draft (work in progress), March 1995

[2] Waxman, B.M., Routing of Multipoint Connections, *IEEE JSAC*, 6(9),pp.1617-1622, 1988

[3] Ballardie, T., et al., Core Based Trees (CBT) - An Architecture for Scalable Inter-Domain Multicast Routing, *ACM SIGCOMM*, 23(4),pp.85-95, 1993

[4] Ballardie, A.J., A New Approach to Multicast Communication in a Datagram Internetwork, Ph.D thesis, May 1995

[5] Deering, S., et al, An Architecture for Wide-Area Multicast Routing, *ACM SIGCOMM*, 24(4), pp.126-135, 1994

[6] Deering, S.E. and Cheriton, D.A., Multicast Routing in Datagram Internetworks and Extended LANs, *ACM Trans. on Computer Systems*, 8(2),pp.85-110, 1990

[7] Moy, J., Multicast extensions to OSPF, *RFC 1584*, March 1994

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

[9] Chachra, V., et al, Applications of graph theory algorithms, Elsevier North-Holland, 1979

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

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

[12] 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

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

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

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

[16] 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

Appendix

Given a cost matrix $C=\{C_{ij}\}$, a delay matrix $D=\{D_{ij}\}$ and a delay bound $\triangle$ for a directed graph $G=(V,E)$, the following algorithm computes a constrained spanning tree $T_1 = (V_1, E_1)$ rooted at the source node s and spanning some (or possibly all) of the nodes in the multicast group S by using the Dijkstra's algorithm with the additional bounded delay $\triangle$ assurance condition for every generated path.

For convenience, we follow the Dijkstra's algorithm described in [9]. The set of vertices is divided into two subsets T and W. The subset T is a vector containing vertices that have two permanent labels associated with them. One of the permanent labels of a vertex (called *cost label*) is the cost of the minimum cost path satisfying the delay bound $\triangle$ of the vertex from s, the other permanent label (called *delay label*) is the corresponding delay value of this path. The algorithm takes the following steps:

(1) Initially $T=\{s\}$ and $W=V$ - $\{s\}$. Two temporary cost and delay labels with infinite value are associated with every vertex in W.

(2) Determine the vertices in W that are adjacent to any vertex in T (namely the corresponding edge cost is finite). Assign each of these vertices two new temporary labels. The new temporary cost and delay labels of vertex V_i in W are denoted by $TC(V_i)$ and $TD(V_i)$ respectively and are given by:

$TC(V_i)$=min(permanent cost label of V_j + C_{ji}),
for those V_j's in T with finite value C_{ji} and satisfying
permanent delay label of V_j + $D_{ji} < \triangle$

$TD(V_i)$=permanent delay label of V_k + D_{ki},
where V_k is the vertex in T that leads to the resulting $TC(V_i)$.

Note that $TC(V_i)$ and $TD(V_i)$ remain unchanged (infinite value) if there is no V_j in T satisfying permanent delay label of V_j + $D_{ji} < \triangle$.

(3) make the smallest (finite value) temporary cost label and the corresponding delay label permanent. Transfer the corresponding vertex from W to T.

(4) Reset all temporary labels of the vertices in W to infinite. Repeat steps (2) and (3) until all nodes in the multicast group S are included in T, or all the (resulting) temporary cost labels in step (2) are infinite.

Note that the computed paths from the source node s to those nodes in T satisfy the delay bound $\triangle$. Tree T_1 consists of the paths from s to those nodes in T that are also in S.