Advanced Bounded Shortest Multicast Algorithm for Delay Constrained Minimum Cost

Moonseong Kim[1], Gunu Jho[2], and Hyunseung Choo[1,⋆]

[1] School of Information and Communication Engineering
Sungkyunkwan University, Korea
{moonseong, choo}@ece.skku.ac.kr
[2] Telecommunication Network Business
Samsung Electronics, Korea
jhogunu@daum.net

Abstract. The Bounded Shortest Multicast Algorithm (BSMA) is a very well-known one of delay-constrained minimum-cost multicast routing algorithms. Although the algorithm shows excellent performance in terms of generated tree cost, it suffers from high time complexity. For this reason, there is much literature relating to the BSMA. In this paper, the BSMA is analyzed. The algorithms and shortcomings are corrected, and an improved scheme is proposed without changing the main idea of the BSMA.

Keywords: Multicast Routing Algorithm, Delay-Bounded Minimum Steiner Tree (DBMST) Problem, and Bounded Shortest Multicast Algorithm (BSMA).

1 Introduction

Depending on the optimization goals, which include cost, delay, bandwidth, delay-variation, reliability, and so on, the multicast routing problem exists at varying levels of complexity. A Delay-Bounded Minimum Steiner Tree (DBMST) problem deals with the minimum-cost multicast tree, satisfying the delay-bounds from source to destinations. The Bounded Shortest Multicast Algorithm (BSMA) is a routing algorithm which solves the DBMST problem for networks with asymmetric link characteristics [1,2]. The BSMA starts by obtaining a minimum delay tree, calculated by using Dijkstra's shortest path algorithm. It then iteratively improves the cost, by performing, the delay-bounded path switching. The evaluation performed by the Salama *et al.* [3] demonstrates that the BSMA is one of the most efficient algorithms for the DBMST problem, in terms of the generated tree cost. However, the high time complexity presents a major drawback of BSMA, because a k-shortest path algorithm is used iteratively for path switching. There are also several approaches to improve the time complexity of BSMA [4,5]. However, among them, none can peer with the BSMA in terms of cost.

⋆ Corresponding author.

Y. Shi et al. (Eds.): ICCS 2007, Part I, LNCS 4487, pp. 668–675, 2007.

The subsequent sections of this paper are organized as follows. In Section 2, the BSMA is described. In Section 3, then the problems with the BSMA are described and the fact that the BSMA can perform inefficient patch switching in terms of delays from source to destinations without reducing the tree cost, are presented. A new algorithm is proposed, furthermore, to substitute for the k-shortest path algorithm, considering the properties of the paths used for the path switching. Finally, this paper is concluded in Section 4.

2 Bounded Shortest Multicast Algorithm

BSMA constructs a DBMST by performing the following steps:

1) Initial step: Construct an initial tree with minimum delays from the source to all destinations.
2) Improvement step: Iteratively minimize the cost of the tree while always satisfying the delay bounds.

The initial tree is minimum-delay tree, which is constructed using Dijkstra's shortest path algorithm. If the initial tree could not satisfy the given delay bounds, some negotiation would be required to relax the delay bounds of DDF (Destination Delay-bound Function). Otherwise, tree construction cannot succeed in satisfying the DDF.

BSMA's improvement step iteratively transforms the tree topology to decrease its cost monotonically, while satisfying the delay bounds. The transformation performed by BSMA at each iteration of the improvement step consists of a delay-bounded path switching. The path switching replaces a path in tree T_j by a new path with smaller cost, resulting in a new tree topology T_{j+1}. It involves the following:

1) Choosing the path to be taken out of T_j and obtaining two disjoint subtrees T_j^1 and T_j^2
2) Finding a new path to connect T_j^1 and T_j^2, resulting in the new tree topology T_{j+1} with smaller cost, while the delay bounds are satisfied.

A candidate paths in T_j for path switching is called a *superedge*. Removing a superedge from a multicast tree corresponds to removing all of the the tree edges and internal nodes in the superedge. From the definition of a superedge [1,2], a destination node or a source node cannot be an internal node of a superedge. This prevents the removal of destination nodes or the source node from the tree as a result of a path switching.

At the beginning of the improvement step, BSMA sets all superedges unmarked and selects the superedge p_h with the highest cost among all unmarked superedges. Removing the p_h in T_j breaks T_j into two disjoint subtrees T_j^1 and T_j^2, where $T_j = p_h \cup T_j^1 \cup T_j^2$. A delay-bounded minimum-cost path p_s between T_j^1 and T_j^2 is used to reconnect T_j^1 and T_j^2 to obtain the new tree topology T_{j+1}, where $T_{j+1} = p_s \cup T_j^1 \cup T_j^2$. The cost of p_s is not higher than that of p_h.

The search for the p_s starts with the minimum-cost path between the two trees. If the minimum-cost path results in a violation of delay bounds, BSMA uses an incremental k-shortest path algorithm [6] to find p_s. The k-shortest path problem consists of finding kth shortest simple path connecting a given source-destination pair in a graph. k-shortest path in BSMA is k-minimum-cost path between two trees and is equivalent to finding the k-shortest path between the two nodes. Because BSMA uses a k-shortest path algorithm for the path switching, its high time complexity is the major drawback. For this reason, the improvement algorithms are proposed in [4,5]. While these reduce the execution time, the performance loss in terms of tree cost is also happened.

3 Difficulties in BSMA

3.1 Undirected Graph Model for BSMA

It is not mentioned clearly in [1] whether the network model for BSMA is a directed or an undirected graph. Although the figures in [1] implicate that it is undirected, later version describing the BSMA [2] and other literatures [3,4,5] related to the BSMA simulation state that the network is modeled as a directed graph. But, we argue that it should be an undirected graph.

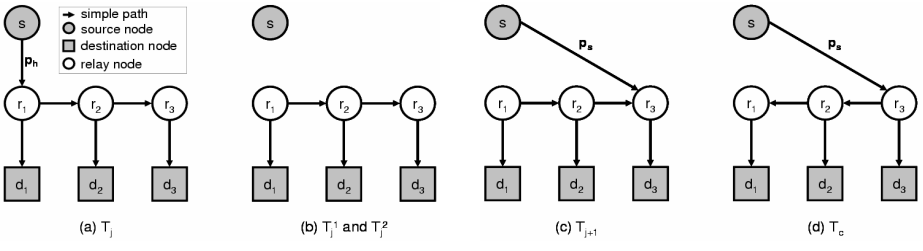


Fig. 1. A case that is able to happen at the step for delay-bounded path switching

The following case in Fig. 1 can be happened during the delay-bounded path switching of BSMA and shows that there could be a problem. Fig. 1(a) shows a tree T_j before the path switching, of which the highest cost superedge p_h is a path from s to r_1, as shown. There are two disjoint subtree T_j^1 and T_j^2 in Fig. 1(b) which are calculated by removing the highest cost superedge p_h in T_j. At this step of the path switching, a delay-bounded shortest path p_s is searched. By reconnecting the p_s (Fig. 1(c)), T_{j+1} is obtained.

As shown in the Fig. 1(c), T_{j+1} is a wrong tree, because the source s cannot send anything to destinations d_1 and d_2 using T_{j+1}. To convert T_{j+1} in Fig. 1(c) to T_c in Fig. 1(d), that is what the algorithm wants, it must be guaranteed that both path-delays and path-costs of the paths from r_1 to r_2 and from r_2 to r_3 are the same as those of paths from r_2 to r_1 and from r_3 to r_2, respectively. If the algorithm must guarantee this, it becomes overhead to check all the links in a subtree without source s that is T_j^1 or T_j^2 at every step it performs the

path switching. Simultaneously it can severely reduce the possible cases that can make the path switching as many as there are asymmetric links in the tree. Of course, there is no routine to handle this case in BSMA. To avoid this and to contribute to the main idea of the BSMA, the network model should be assumed as the undirected graph. From now on, we use the undirected links, so that $(u, v) = (v, u) \in E$ with the same link-delay and link-cost values.

3.2 Meaning of 'Unmark' the Superedge

The issue of this subsection is about the superedge. There are five superedges in Fig. 1(a), those are $p(s, r_1)$, $p(r_1, d_1)$, $p(r_1, r_2)$, $p(r_2, d_2)$, and $p(r_2, d_3)$. After the path switching, there are different superedges in Fig. 1(d), those are $p(s, r_3)$, $p(r_3, d_3), p(r_3, r_2), p(r_2, d_2)$, and $p(r_2, d_1)$. You can see that the superedges change, as the tree changes. The simple paths in Fig. 2(a), which redraws the Fig. 1(d), are all superedges. The tree in Fig. 2(b) is the result by another path switching. And the tree in Fig. 2(c) shows the same tree where the simple paths are all superedges.

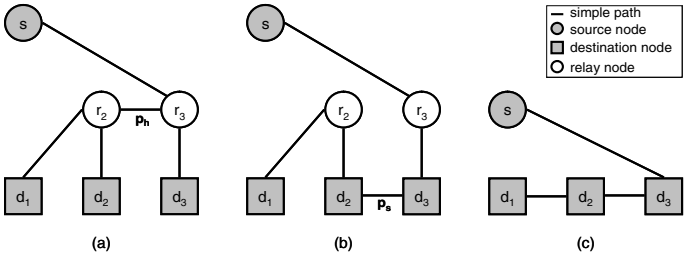


Fig. 2. Changes of superedges as the tree changes

BSMA marks the highest-cost unmarked superedge p_h when the superedge is on path switching. If the p_h is switched to a delay-bounded minimum-cost path p_s, BSMA *unmarks* all marked superedges [1,2]. If the flag value of a superedge is marked, it means there is no path to substitute for the superedge to reduce the current tree T_j. When p_h is switched to p_s, the tree is changed from T_j to T_{j+1}. Of course, BSMA must recalculate the superedges in the new tree T_{j+1} with initializing them as unmarked.

3.3 The Delay-Bounded Minimum-Cost Path p_s and the Function to Get the p_s Between T_j^1 and T_j^2

According to [1,2], one of two cases must happen when the delay-bounded minimum-cost path p_s is obtained:

1. path p_s is the same as the path p_h; or
2. path p_s is different from the path p_h.

If the first case occurs, the p_h has been examined without improvement of the tree cost. If the second case occurs, the tree T_{j+1} would be more cost-effective

tree than T_j. But it is possible in the second case to generate the tree T_{j+1} whose cost is the same as that of T_j and end-to-end delay between a source and each destination is worse than that of T_j. The former cost-effective tree T_{j+1} is what the algorithm wants. But the latter tree is not. This is because a path with the same cost as that of p_h could be searched as p_s.

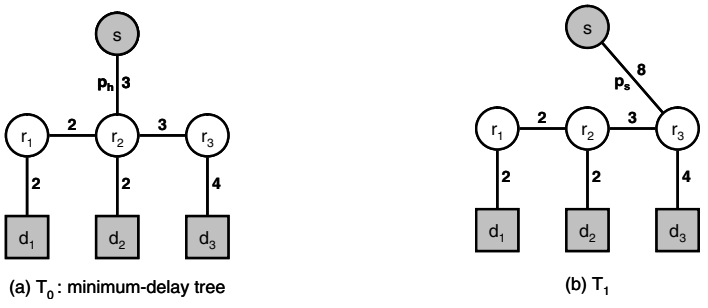


Fig. 3. Ineffective path switching in terms of end-to-end delay without any improvement of tree cost

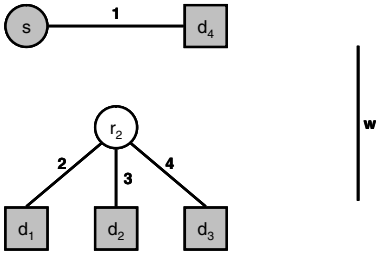
Fig. 3 shows the example for that. The values on the simple paths in the Fig. 3 are path-delays. The tree T_0 in Fig. 3(a) is a minimum-delay tree generated by using Dijkstra's algorithm. So, it is sure that the end-to-end delays between the source s and each destination d_1, d_2, and d_3 are the minimum values. The tree T_1 in Fig. 3 (b) is a tree after the path switching. The p_s is a simple path whose path-delay is 8 and path-cost is the same as that of p_h. As a result of the path switching, BSMA generates the ineffective tree T_1 in terms of end-to-end delay without any improvement of tree cost.

If BSMA considers only whether the p_s is equal to p_h or not, this ineffective path switching is always possible as long as there could be a path whose cost is the same as that of p_h while satisfying the delay-bounds. So BSMA must select a path with smaller cost than that of p_h, as the p_s. (We must note here that the ineffective path switching does not happen in *BSMA based on greedy heuristic*, since it performs the path switching when the *gain* is larger than zero.)

Additionally, we need to think about the procedure to determine whether a path is delay-bounded or not. Whenever BSMA finds the p_s, it has to determine whether a path is delay-bounded or not. That is to say, BSMA has to perform one of the followings:

1. construction of a tree from T_j^1, T_j^2 and a candidate path for p_s, for every candidate until finding the p_s; or
2. pre-calculation of end-to-end delays between a source and each destination for all cases that the p_s would connect T_j^1 with T_j^2.

The total cost of a tree can be calculated without considering how the nodes in the tree are connected by the links. Because it is the sum of link-costs in the tree, we only need information about which links are in the tree and how



(a) T_j^1 and T_j^2 (b) Candidate path for p_s

Connection points		Delay from source			
		d_1	d_2	d_3	d_4
s	r_2	2+w	3+w	4+w	1
s	d_1	w	5+w	6+w	1
s	d_2	5+w	w	7+w	1
s	d_3	6+w	7+w	w	1
d_4	r_2	3+w	4+w	5+w	1
d_4	d_1	1+w	6+w	7+w	1
d_4	d_2	6+w	1+w	8+w	1
d_4	d_3	7+w	8+w	1+w	1

(c) Pre-calculation

Fig. 4. Calculation of end-to-end delays in tree

much their costs are. But end-to-end delay between two tree nodes, such as a source and a destination, is only able to be calculated after considering the tree structure.

Fig. 4(a) shows subtrees on BSMA's path switching and (b) is a candidate path for p_s. If the values on the simple paths in Fig. 4(a) and (b) are the path-costs, the total cost of a tree T_{j+1} $(= T_j^1 \cup T_j^2 \cup p_s)$ would be determined when the path-cost w of the candidate is determined (that is $1+2+3+4+w$), without considering how the candidate connects T_j^1 with T_j^2. But if the values indicate the path-delays, it is impossible to determine the end-to-end delays between the source s and each destination d_1, d_2, or d_3 without considering the tree structures of T_j^1 and T_j^2.

Obviously there are two ways to determine whether the candidate is delay-bounded or not; one is to construct the tree by connecting T_j^1 and T_j^2 with the candidate; the other is to perform a pre-calculation such like Fig. 4(c). Of course, both must consider the structures of two subtrees.

3.4 Inefficient Use of *k*-Shortest Path Algorithm

As the literatures [4,5] mentioned, the k-shortest path algorithm [6] used for finding the p_s is the major drawback of BSMA. The time complexity of the k-shortest path algorithm is $O(kn^3)$. The k value can be set to a fixed value to reduce execution time of BSMA. However this also reduces the performance of BSMA in terms of generated tree cost. In this subsection we propose another algorithm to substitute the k-shortest path algorithm. The proposed algorithm finds candidate paths for p_s within some path-cost range and does not deterorate the performance of BSMA.

According to what we described in subsection 3.3, BSMA do not need the paths whose costs are equal to or larger than that of p_h while finding p_s. And obviously we cannot find any path with smaller cost than that of minimum-cost path calculated by Dijkstra's algorithm. Consequently, candidate paths for p_s are the paths with the cost range that is equal to or larger than that of the minimum-cost path and smaller than that of p_h. The following is the pseudo code of the proposed algorithm.

Description for internal variables and functions

$p[0..(|V|-1)]$, $q[0..(|V|-1)]$: an array containing the node sequence for a path
$index_p$: the index for p
$cost_p$: the path-cost of p
P: the set of searched paths
Q: queue containing paths on searching
PUSH(Q, p): insert p to Q
POP(Q): take a path out of Q

PROCEDURE PathSearch_SameCostRange(T_j^1, T_j^2, $minCost$,$maxCost$, G)

```
1. P ← ∅, Q ← ∅;
2. for each node i ∈ T_j^1 {
3.    p[0] ← i, index_p ← 0, cost_p ← 0;
4.    PUSH(Q, p);
5. }
6. while Q ≠ ∅ {
7.    p ← POP(Q);
8.    for each neighbor node n of p[index_p] {
9.       if (n is in the array p) then continue;
10.                          \\ Because we are looking for simple paths
11.      if (n ∈ T_j^1) then continue;
12.                          \\ Because we are looking for paths between T_j^1 and T_j^2
13.      if (cost_p+ link-cost of (p[index_p], n) ≥ maxCost) then continue;
14.      q ← p, cost_q ← cost_p, index_q ← index_p; \\ Copy p to q
15.      q[index_q + 1] ← n;
16.      cost_q ← cost_q+ link-cost of (q[index_q], n);
17.      index_q ← index_q + 1;
18.      if (n ∈ T_j^2 AND cost_q ≥ minCost) then P ← P ∪ {q};
19.      if (n ∉ T_j^2) then PUSH(Q, q);
20.   }
21. }
22. return P;
```

Although a number of candidates for p_s are searched, BSMA use only the one satisfying the delay-bounds with smallest cost, that is p_s. So BSMA do not need to find all the candidates at once. Therefore the arguments $minCost$ and $maxCost$ are not minimum path-cost between T_j^1 and T_j^2, and path-cost of p_h, respectively. The half closed interval [*minimum path-cost between* T_j^1 *and* T_j^2, *path-cost of* p_h) is divide into several intervals to be the $minCost$ and $maxCost$. BSMA iteratively increases the $minCost$ and $maxCost$ until either finding p_s or recognizing there is no path to substitute p_h.

When the network size is large with many links, the memory required to calculate the candidates is also heavy as well as the high time complexity. So, this is quite practical and dose not provide any limitation to BSMA's performance. The difference between $minCost$ and $maxCost$ can be adjusted according to

characteristic of modeled link-cost. (*i.e.* If the link-costs is modeled as integer values, $minCost$ and $maxCost$ can be the integer value x and $x+1$, where x is some starting point of divided interval and 1 stands for the characteristic.) In the next section, the characteristic is notated as the sys.

4 Conclusion

BSMA is very well-known one of delay-constrained minimum-cost multicast routing algorithms. Although, its performance is excellent in terms of generated tree cost, the time complexity is very high. There are many literatures related to BSMA for this reason. We have shown that BSMA has fallacies and ambiguities then, modified it. We start on the describing BSMA [2]. Then, we show that the BSMA has fallacies, and that the BSMA can perform inefficient patch switching in terms of delays from source to destinations without reducing the tree cost. Hence, we propose an algorithm to substitute for the k-shortest path algorithm considering the properties of the paths which are used for the path switching.

Acknowledgment

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment), IITA-2006-(C1090-0603-0046).

References

1. Zhu, Q., Parsa, M., Garcia-Luna-Aceves, J. J.: A Source-Based Algorithm for Delay-Constrained Minimal-Cost Multicasting. Proceeding of INFOCOM. IEEE (1995) 377-385
2. Parsa, M., Zhu, Q., Garcia-Luna-Aceves, J. J.: An Iterative Algorithm for Delay-Constrained Minimum-Cost Multicasting. IEEE/ACM Transactions Networking, Vol. 6, Issue 4. IEEE/ACM (1998) 461-474
3. Salama, H. F., Reeves, D. S., Viniotis, Y.: Evaluation of Multicast Routing Algorithms for Real-Time Communication on High-Speed Networks. Journal of Selected Areas in Communications, Vol. 15, No. 3. IEEE (1997) 332-345
4. Gang, F., Kia, M., Pissinoul, N.: Efficient Implementations of Bounded Shortest Multicast Algorithm. Proceeding of ICCCN. IEEE (2002) 312-317
5. Gang, F.: An Efficient Delay Sensitive Multicast Routing Algorithm. Proceeding of the International Conference on Communications in Computing. CSREA (2004) 340-348
6. Lawler, E.: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston (1976)