The New Shortest Best Path Tree (SBPT) Algorithm for Dynamic Multicast Trees

Hiroshi Fujinoki and Kenneth J. Christensen
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
Email: {*fujinoki, christen*}*@csee.usf.edu*

Abstract

This paper presents the new Shortest Best Path Tree (SBPT) algorithm for multicast trees. The SBPT algorithm establishes and maintains dynamic multicast trees which maximize the bandwidth to be shared by multiple receivers and simultaneously guarantee the shortest paths for each receiver node. The SBPT algorithm is a distributed algorithm with cost in the same order as the sum of the shortest path tree (SPT) algorithm and the Greedy algorithm. The SBPT algorithm reduces bandwidth consumption by utilizing partial paths already established for other multicast receiver nodes. The SBPT algorithm finds such partial paths when multiple shortest paths exist. Simulation experiments comparing the SBPT and SPT algorithms show that the SBPT algorithm reduces bandwidth consumption by 5% to 17% when node utilization is greater than approximately 25% and always achieves the same shortest path lengths.

1. Introduction

Different from single-cast applications, the paths for a multicasting application are generally established as a tree, called a multicast tree, which contains a single sending node (the sender) and multiple receiving nodes (the receivers). A good multicast routing algorithm should be able to control end-to-end delay and bandwidth consumption in order to effectively support real-time video and audio applications (see, for example, [10], [11], and [17]). In addition, a good multicast routing algorithm should be implemented in a distributed fashion and be efficient in execution given rapidly changing populations of receivers (see [15]).

Three different types of multicast trees are described in [7]; Steiner trees, center-based trees, and source-based trees. Center-based and source-based trees categorize the place where the routing operation is initiated. These trees are generally represented by a Shortest Path Tree (SPT) (see [3], [6], and [12]) or a tree generated by the Greedy algorithm (see [14]). A Steiner tree, or Steiner Minimal Tree (SMT), is the multicast tree that minimizes total path cost. An algorithm to find an exact SMT belongs to the class NP-complete, hence heuristic algorithms have been studied (see [2], [13] and [16]). However, due to their high algorithm complexity, routing algorithms based on SMT heuristics are not popular in actual use.

One of the advantages of the SPT algorithm is that it always guarantees the shortest path from each receiver node to the sender. However, the SPT algorithm cannot explicitly route paths so that multicast paths are shared by as many receiver nodes as possible. The Greedy algorithm, however, does attempt to share paths to maximize bandwidth sharing. The Greedy algorithm develops a multicast tree by connecting each new receiver node to its nearest multicast receiver that has been already connected to the sender of the target multicast stream. However, the Greedy algorithm does not consider path length for each receiver node, resulting in longer path length than that of the SPT algorithm. This tradeoff between path length (minimized by the SPT algorithm) and bandwidth consumption (addressed by the Greedy algorithm) is solved in this paper by a new multicast routing algorithm called the Shortest Best Path Tree (SBPT) algorithm. Given multiple shortest paths between a receiver and a sender, the SBPT algorithm utilizes partial paths already established for other multicast receiver nodes. This guarantees that path length is the same as from the SPT algorithm, but bandwidth consumption is reduced.

The remainder of this paper is organized as follows. Section 2 describes the SPT and the Greedy multicast routing algorithms. A tradeoff relationship between the two algorithms is also described. In Section 3 the Shortest Best Path Tree (SBPT) algorithm is introduced. Section 4 describes simulation experiments comparing the SPT, Greedy, and SBPT algorithms. In Section 5, the experimental results are presented and observations are made based on the simulation results. The final section is a summary and is followed by references.

2. Existing Multicast Algorithms

It is assumed that multicast sessions have a single sender and that the root of a multicast tree is always the sender of a multicast session (therefore the root is not fixed at a specific node). The second assumption is made since it has been shown that a fixed root approach (i.e., where the root is not necessarily the sender) can be very inefficient (see [4]). Figure 1 is an example of such a multicast tree where the number adjacent to each edge represents a hop count and each node, except for the root of the tree, represents a receiver (labeled as *v1*, *v2*, … , *v13*). Each hop is assumed to consume a unit of bandwidth. We define B_{Total} to be the total amount of bandwidth consumed by a multicast tree for a multicast session at a given time. We define P_{Total} to be the total path length - the sum of the hops from all of the multicast receivers to the sender. A joining node is a node that is being connected to the sender. Neighbor nodes are multicast receivers that have already been connected to the sender. Neighbor nodes include the following two types of nodes; 1) the multicast receivers that are receiving a multicast stream from a sender, and 2) the multicast receivers that are not receiving the multicast stream but are on a path already created from a receiver node to the sender.

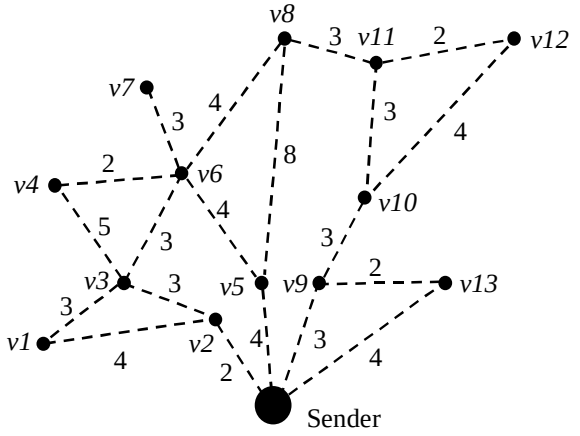

Figure 1 - Example network

The SPT algorithm is a multicast tree algorithm that minimizes path length by connecting each receiver node to the sender using the shortest path. Figure 2(a) shows an example of the resulting multicast tree after the SPT algorithm is applied to the network of Figure 1, assuming that nodes *v1, v2, v13, v9, v10, v12, v11, v8, v6, v7, v4, v3,* and *v5* are connected in that order. The resulting multicast tree has B_{Total} = 50, which is the sum of the hop counts of all the links used for connecting nodes. The total path length, P_{Total} = (6 + 2 + 5 + 10 + 4 + 8 + 11 + 12 + 3 + 6 + 9 + 10 + 4) = 90, with each term representing the path length for *v1* to *v13*, respectively. Using the SPT algorithm, *v7* may choose a shortest path to the sender through *v6* and *v5* based on *v7*'s shortest path calculation. The Greedy algorithm always connects a joining node to its nearest neighbor node. If such a neighbor node does not exist, or if the sender is the nearest neighbor node, then the joining node is connected to the sender using the shortest path. Figure 2(b) shows the resulting multicast tree after the Greedy algorithm is applied to the graph of Figure 1 and it has B_{Total} = 40 and P_{Total} = 153.

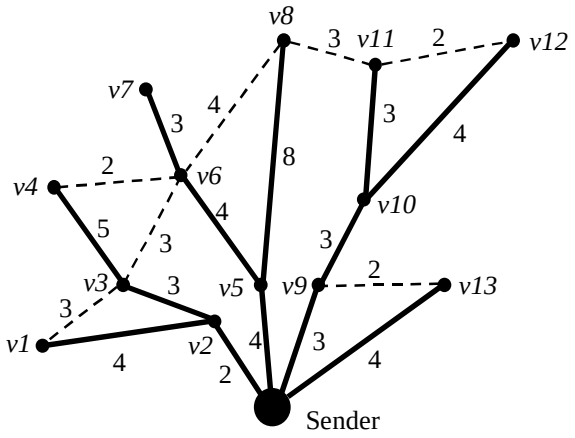

Figure 2(a) - Multicast tree from SPT algorithm

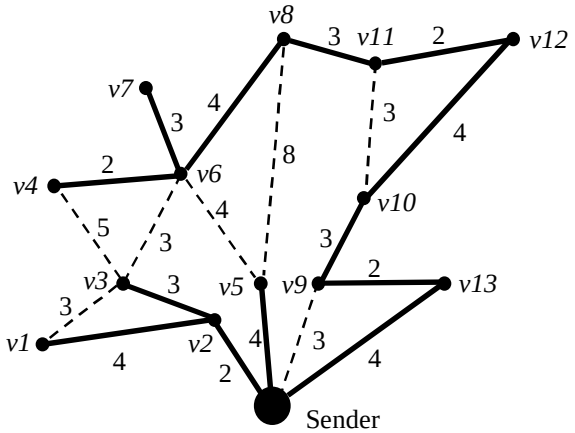

Figure 2(b) - Multicast tree from Greedy algorithm

We can see a tradeoff relationship between B_{Total} and P_{Total}. The multicast trees generated by the SPT algorithm generally have a larger B_{Total} than those created by the Greedy algorithm. This is because the SPT algorithm does not find paths that necessarily share

bandwidth by multiple receivers. The Greedy algorithm reduces the bandwidth consumed. However, a problem in the Greedy algorithm is that it routes paths based on the nearest neighbor node. Therefore, resulting multicast trees may not be efficient in path length. Waxman's weighted greedy algorithm has been proposed as a solution to the problem of excessive path length (see [7]). However, the weighted greedy algorithm requires the existence of special fixed nodes called "owner nodes" and the efficiency of its routing depends on how these owner nodes are assigned. There is no systematic method to assign these nodes.

3. The SBPT Algorithm

The Shortest Best Path Tree (SBPT) algorithm is proposed as a solution for the tradeoff problem discussed in Section 2 and is a middle ground between the SPT and Greedy algorithms. In this paper, distance between any two nodes is represented by the hop count between them. The following two variables are used in the SBPT algorithm.

- $M_{(i,s)}$ is the shortest distance from a node *i* to a sender *s*.
- $D_{(j,i)}$ is the distance of the shortest path between two nodes, a joining node *j* and a neighbor node *i*.

The SBPT algorithm is shown in Figure 3. The algorithm takes three input parameters; the node ID of the joining node, *j*, the node ID of the sender, *s*, and the directed graph *G*.

Assuming that the receivers are connected to the sender in the same order as in the example in Section 2 and Figure1, the resulting multicast tree after the SBPT algorithm is applied is shown in Figure 4. This graph is created as follows. First, node *v1* is connected to the sender using the shortest path through *v2*. Node *v2* has only one shortest path, so it is connected to the sender using the shortest path. Similarly, node *v13, v9, v10, v11*, and *v12* have only one shortest path and they are all connected by their shortest paths. When node *v8* is being connected, it sees *v11* as its nearest neighbor on one of its four shortest paths ($M_{(v8,s)} = D_{(v8,v11)} + M_{(v11,s)}$) and therefore, *v8* is connected to *v11*. When node *v7* is being connected, it sees *v8* as its nearest neighbor. However, *v8* is not on one of *v7*'s shortest path to the sender ($M_{(v7,s)} < D_{(v7,v8)} + M_{(v8,s)}$). The nearest neighbor node of *v7* on one of *v7*'s shortest paths is *v2* ($M_{(v7,s)} = D_{(v7,v2)} + M_{(v2,s)}$). Therefore, *v7* is connected to *v2*. Node *v6* sees itself as a neighbor node on its shortest path (i.e., $D_{(v6,v6)} = 0$), and it uses the shortest path established by *v7*. Node *v4* sees *v6* as its nearest neighbor on one of its three shortest paths and is connected to *v6*. Similar to node *v6*, node *v3* sees itself as a neighbor node and uses the shortest path to the sender through *v2*. Finally, *v5* has only one shortest path and is connected to the sender using this path. The resulting multicast tree has $B_{Total} = 41$ and $P_{Total} = 90$.

```
SBPT algorithm(j,s,G)
1.  NearestDistance = ∞
2.  Apply an SPT algorithm to graph
    G with as j its root to
    determine the shortest distance
    from j to each node v ∈ G
    and store the shortest distance
    for each v in D(j,v)
3.  for all i ∈ v
4.      if (i is a neighbor of j)
then
5.      if ( M(j,s) = D(j,i) + M(i,s) ) then
6.        if ( D(j,i) <Nearest) then
7.          NearestDistance = D(j,i)
8.          NearestNeighbor = i
9.        end-if
10.     end-if
11.   end-if
12. end-for
13. Establish shortest path between
    j and i
14. end-SBPT
```

Figure 3 - The SBPT algorithm

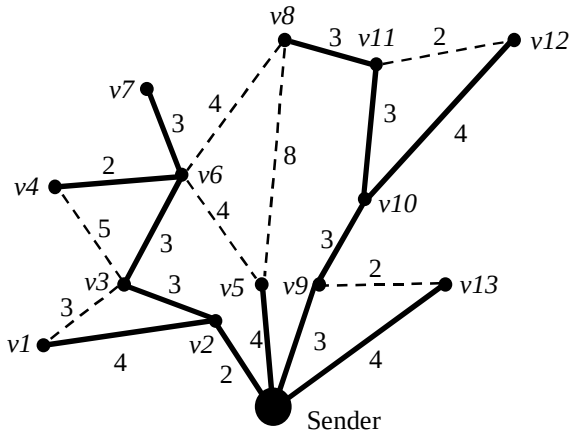

Figure 4 - Multicast tree from SPBT algorithm

The total cost of each algorithm can be subdivided into three categories. They are the costs for initialization, external operation, and internal operation. The cost for initialization is required in a network only once when a network is first configured. The cost for external operation is required at each joining node to get information from other nodes when a joining node tries to find its connecting point. The difference from the initialization cost is that the external cost is required at each joining node each time it joins a multicast session. The internal cost is the same as the external cost except that it is the cost of operations performed inside of a joining node to decide a joining node's connecting point based on the information collected by the external operation.

The SPT algorithm takes at least $O(n^2)$ time where n is the number of receiver nodes in a network, assuming the Bellman-Ford algorithm to determine the shortest path from the sender to each node in a network. There is no external operation and only a constant time for the internal cost is required. The Greedy algorithm does not require operations for initialization, it only needs to find the nearest neighbor node for each joining node. Using the Bellman-Ford algorithm to create a shortest path tree with a joining node as the root, it takes at least $O(n^2)$ time to determine the nearest neighbor node. The Greedy algorithm takes at least $O(n^2)$ time for the external cost to find neighbor nodes and their distance and $O(n)$ time for the internal operations to determine the nearest node in the found neighbor nodes (using a linear search). The SBPT algorithm takes $O(n^2)$ time for initialization since it requires determination of a shortest path for each node in a network. For external operation, the SBPT algorithm needs to locate neighbor nodes for a joining node and their shortest distance to the joining node. This operation is the same as in the Greedy algorithm (line 2 in Figure 3). The internal operation takes $O(n)$ time, where n is the number of the neighbor nodes (line 3 in Figure 3). Table 1 summarizes the costs for the three algorithms, which shows that the total cost of the SBPT algorithm is in the same order as the sum of the SPT and the Greedy algorithm.

	SPT	**Greedy**	**SBPT**
Initial cost	n^2	0	n^2
External cost	0	n^2	n^2
Internal cost	1	n	n
Total cost	n^2+1	$n+n$	n^2+n^2+1

Table 1 - Algorithm complexities

4. Evaluation of the SBPT Algorithm

The SPT, Greedy, and SBPT algorithms were compared using simulation methods. The simulation model was implemented as a time-based simulation where all statistics are collected at each discrete clock tick. All continuously distributed random time values are truncated to their corresponding integer values. The major variables in the simulation model are network size and multicast receiver behavior. The random network model introduced by Billharts et al. (see [3]) was used. This network model assumes a grid-patterned square network, where a node representing a multicast receiver site is located only at each intersection of the grid in a network. Thus, the network model is a non-hierarchical, flat structure. It is predicted that the future Internet will tend towards a non-hierarchical, flat network as the number of domains increases (see [9] for more details).

The number of nodes in a network, N^2, represents the network size where N is the number of nodes in each edge of a square network. To assign links between nodes, the link assignment model proposed by Waxman (see [14]) is applied in which the probability of there being an edge between nodes u and v is given by:

$$P(u,v) = \gamma \cdot e^{-d(\beta L)} \quad (1)$$

where d and L are the Euclidean distance and maximum possible link distance between the nodes u and v. The parameters, γ ($\gamma > 1$), and β ($\beta \leq 1$), control the average link distance between u and v.

A receiver's activities are assumed to be either a) joining a multicast tree, or b) disconnecting itself from the multicast tree. The period between the beginning of a connection and the subsequent disconnection is the session holding time. Poisson arrivals are assumed for session join requests and both exponential and Bounded-Pareto (BP) distributed session holding times are assumed. Data set numbers correspond to the experiment number. For data sets 1 through 5, exponentially distributed session holding times are modeled and for data sets 6 through 10 Bounded-Pareto distributed session holding times are modeled. For modeling session holding time, Almeroth suggests using the Zipf distribution (see [1]), which is the discrete version of Pareto distribution (see [5]). The BP distribution models "heavy tailed" session holding times. The probability mass function for $BP(p,k,\alpha)$ is,

$$f(x) = \frac{\alpha \cdot k^{\alpha}}{1 - \left(\frac{k}{p}\right)^{\alpha}} \cdot x^{-\alpha-1} \quad (2)$$

where k is the minimum duration of a session, p the maximum session holding time, and α a shape parameter. BP random variables, X, can be generated using the inverse transform method as,

$$X = \left(\frac{-\left(U \cdot p^{\alpha} - U \cdot k^{\alpha} - p^{\alpha}\right)}{p^{\alpha} \cdot k^{\alpha}} \right)^{-\frac{1}{a}} \quad (3)$$

where, U is uniformly distributed random variable from 0 to 1.

The simulation model measures the average bandwidth consumption per minute (A_{BWC}), and the average path length per connection (A_{PL}) for a simulation of T minutes,

$$A_{BWC} = \frac{1}{T} \sum_{c=0}^{T} b_c \quad (4)$$

and

$$A_{PL} = \frac{1}{T} \sum_{c=0}^{T} \left(\frac{1}{n_c} \sum_{i=0}^{n} l_i \right) \quad (5)$$

where b_c is the amount of bandwidth consumed in the network at clock tick c, l_i is the path length of the receiver node i at clock tick c ($l_i = 0$ if node i is not connected), n_c is the number of receiver nodes joined at the clock tick c, and n is the number of nodes in the network.

A range of network sizes, network connectivity, session holding times, and session initiation arrival rates were considered. For all the experiments, session requests were assumed to be Poisson. For each of the ten experiments, three different network sizes of N^2 = 100, 400, and 900 were used to model small, medium, and large networks. A simulation time of T = 28800 minutes (representing twenty days) was used for all experiments. Table 2 summarizes the key parameters for each data set. Average node utilization is defined to be the ratio of session holding time to total elapsed time at a receiver node. We classify each data set based on its average node utilization as "high utilization" is average node utilization over 75%, "medium utilization" is about 50%, "low utilization" is about 25%, and "very low utilization" is about 5%. Data sets 1 and 6 simulate long-lived receiver sessions. The parameters for these data sets are taken from statistics collected for the NASA STS-63 space shuttle audio sessions from February 3 to February 11, 1995 on the Mbone (Multicast Backbone), while those for the short-lived session (data set 2 and 7) are based on the statistics collected for Free BSD Lounge also on Mbone (see [1]). Data sets 3 and 8 are artificially created to see the effects from longer inter-session time (session request inter-arrival time is doubled from data set 1 and 6), while data sets 4 and 9 model shorter session holding time (session holding time halved from data set 1 and 6). Data sets 5 and 10 were created to see the effects of very low node utilization (session request interval time multiplied by six while session holding time halved from data set 1 and 6). The three parameters for BP distribution are chosen as follows. First, α was chosen to be 0.8 to achieve high variance in the resulting session holding time (standard deviations range from 95 minutes to 386 minutes in the five data sets). The maximum session holding time, p, was fixed to be 2880 minutes, assuming that a receiver will not stay connected for more than two days. The minimum session holding time, k, was then adjusted to achieve the same mean as its corresponding set in exponential distribution (see Table 2 for actual values). We chose k to control the mean where the mean for a BP random variable X is,

$$E[X] = \frac{\alpha \cdot k^{\alpha} \cdot \left(k^{1-\alpha} - p^{1-\alpha}\right)}{(\alpha - 1) \cdot \left(1 - \left(\frac{k}{p}\right)^{\alpha}\right)} \quad (6)$$

Data set	Session holding time (min)	Inter-session time (min)	Utilization (%)	BP shape parameter	(k)	(p)
1	35.0	67.5	43.8	------	------	-----
2	258.0	7.5	97.8	------	------	-----
3	35.0	135.1	25.3	------	------	-----
4	17.5	67.6	25.8	------	------	-----
5	17.5	405.4	4.2	------	------	-----
6	34.7	67.6	45.9	0.8	2.9	2880.0
7	258.0	7.5	97.8	0.8	48.2	2880.0
8	34.7	135.1	24.2	0.8	2.9	2880.0
9	17.2	67.6	23.9	0.8	1.2	2880.0
10	17.2	405.4	3.8	0.8	1.2	2880.0

Table 2 - Description of the experimental data sets

5. Experimental Results

Table 3 shows the results from experiments 1 through 5. The results from experiment 1 show that the Greedy algorithm saves bandwidth consumption by 3% to 9% from the SPT algorithm, while the average path length is 9% to 17% longer than those in the SPT algorithm. The SBPT algorithm saves bandwidth consumption by about 8% from the SPT algorithm while it achieves the same shortest average path length. Experiment 2 shows that the Greedy algorithm achieves the least bandwidth consumption, while its average path length is longer by 9% to 24%. The SBPT algorithm saves 11% to 17% of bandwidth consumption of the SPT algorithm, again with the shortest average path length. Experiments 3 and 4 show similar results, where the Greedy algorithm saves 2% to 17% of bandwidth from the SPT algorithm, while the increase in the average path length is about 6% to 15%. The SBPT algorithm saves bandwidth consumption by about 5% from the SPT algorithm. For experiment 5, the SBPT algorithm is about the same as the SPT algorithm, while the Greedy algorithm consumes more bandwidth than the SPT algorithm by 112% to 282%.

Figures 5, 6, and 7 show the relative difference of the three algorithms for experiments 1, 2, and 3. Experiments 6 through 10 show similar results as seen in experiments 1 through 5, and the results are shown in Table 4. The only exceptions are that they result in larger bandwidth consumption in a large network with low utilization and longer path length in a small network with high utilization.

		Bandwidth (%)			Path Lengths		
	Size	**100**	**400**	**900**	**100**	**400**	**900**
Set 1	SPT	100	100	100	100	100	100
	Greedy	97	91	92	117	113	109
	SBPT	91	92	92	100	100	100
Set 2	SPT	100	100	100	100	100	100
	Greedy	79	83	83	124	113	109
	SBPT	83	89	88	100	100	100
Set 3	SPT	100	100	100	100	100	100
	Greedy	117	106	102	114	113	107
	SBPT	95	95	94	100	100	100
Set 4	SPT	100	100	100	100	100	100
	Greedy	117	103	102	115	112	106
	SBPT	94	95	94	100	100	100
Set 5	SPT	100	100	100	100	100	100
	Greedy	382	237	212	110	110	106
	SBPT	100	100	99	100	100	100

Table 3 - The results from experiments #1 through #5

		Bandwidth (%)			Path Lengths		
	Size	**100**	**400**	**900**	**100**	**400**	**900**
Set 6	SPT	100	100	100	100	100	100
	Greedy	95	91	91	133	115	108
	SBPT	90	92	92	100	100	100
Set 7	SPT	100	100	100	100	100	100
	Greedy	77	84	82	120	113	108
	SBPT	84	89	88	100	100	100
Set 8	SPT	100	100	100	100	100	100
	Greedy	129	111	103	123	110	108
	SBPT	95	95	94	100	100	100
Set 9	SPT	100	100	100	100	100	100
	Greedy	112	110	104	112	110	106
	SBPT	95	95	94	100	100	100
Set 10	SPT	100	100	100	100	100	100
	Greedy	342	203	161	121	116	110
	SBPT	98	97	99	100	100	100

Table 4 - The results from experiments #6 through #10

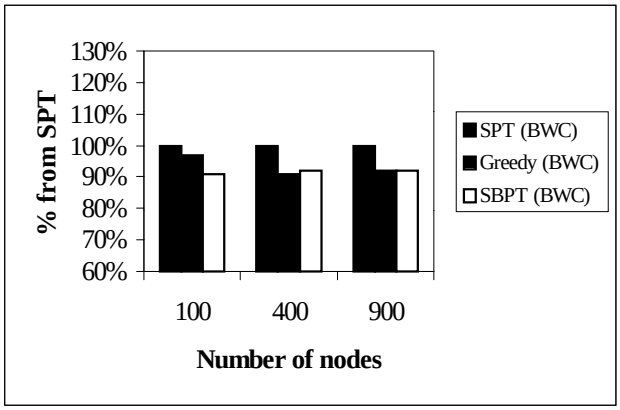

Figure 5(a) - Difference for experiment #1 (A_{BWC})

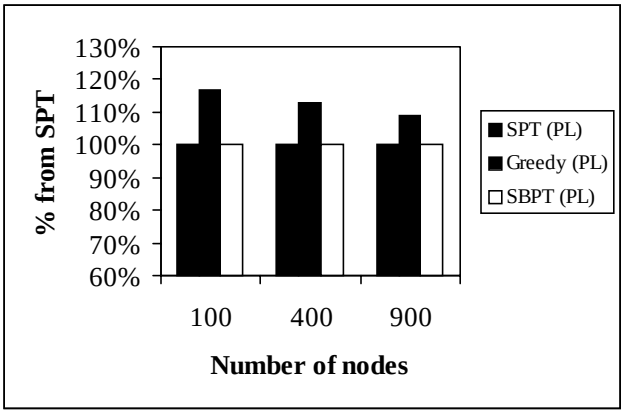

Figure 5(b) - Difference for experiment #1 (A_{PL})

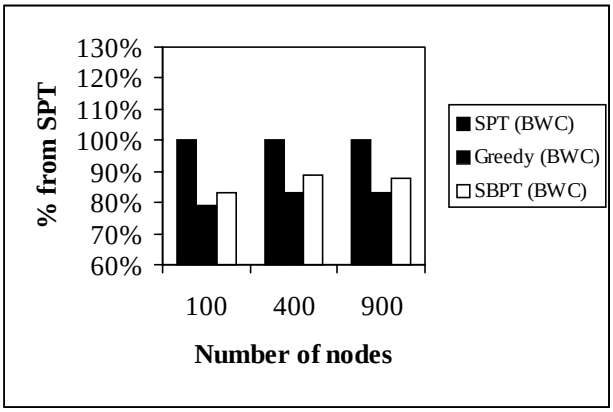

Figure 6(a) - Difference for experiment #2 (A_{BWC})

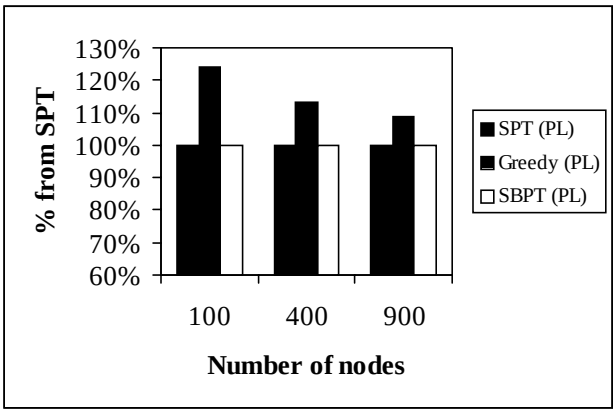

Figure 6(b) - Difference for experiment #2 (A_{PL})

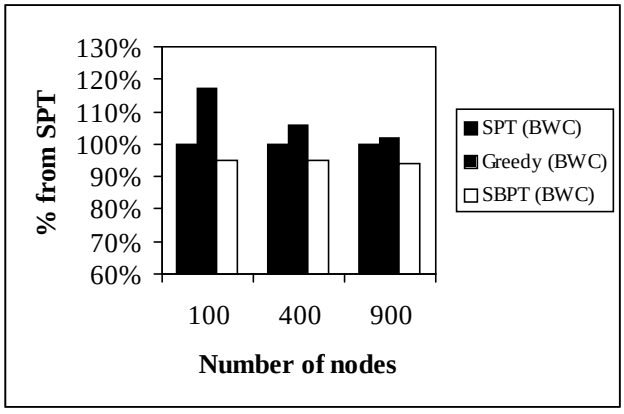

Figure 7(a) - Difference for experiment #3 (A_{BWC})

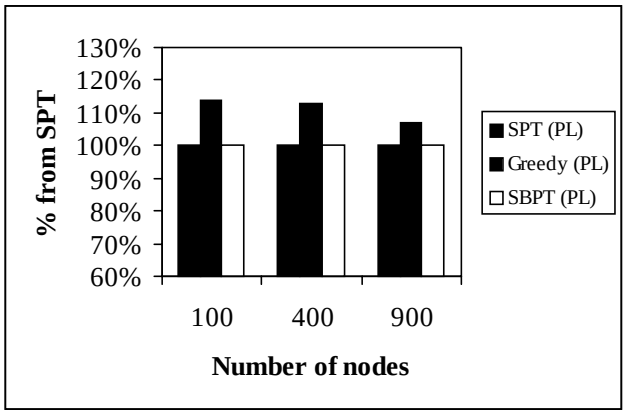

Figure 7(b) - Difference for experiment #3 (A_{PL})

The following observations are made from the experimental results for both the Exponential and Bounded-Pareto distributed session holding times:

1. Except in the networks with high utilization, the SBPT algorithm demonstrates the same shortest path lengths as the SPT algorithm and the least bandwidth consumption of the three algorithms.
2. For the network with high utilization, the SBPT algorithm works better than the SPT algorithm (less bandwidth consumption and the same shortest path length), but compared to the Greedy algorithm the SBPT algorithm consumes 4% to 6% more bandwidth. The SBPT algorithm results in path length shortened by 9% to 24% compared to the Greedy algorithm.
3. We observe the tradeoff relationship between the SPT and the Greedy algorithms in the network with high utilization: the Greedy algorithm consumes the least bandwidth, but results in the longest path length of the three algorithms.

4. When the utilization is very low, the performance of the SBPT algorithm is roughly equal to that of the SPT algorithm. The Greedy algorithm results in very inefficient paths at low utilization.
5. The performance of the three algorithms depends more on the average node utilization than on the absolute values of the average session holding time and the session initiation arrival rate.

6. Summary

In this paper, the Shortest Best Path Tree (SBPT) algorithm has been developed and evaluated. The SBPT algorithm is a distributed, polynomial time algorithm that solves a trade off problem between the SPT and Greedy algorithms. The SPT algorithm guarantees the shortest path for each receiver, while it generally results in larger bandwidth consumption than the Greedy algorithm. On the other hand, the Greedy algorithm generally results in larger average path length, but usually with less bandwidth consumption than the SPT algorithm. The SBPT algorithm utilizes partially established paths for other multicast receiver nodes to minimize bandwidth consumption. To find such partial paths, the SBPT algorithm uses two variables, *M* and *D*. The variable *M* is the length of a partial path, while *D* is the length of the new path to reach the partial path. By using these two variables the SBPT algorithm solves the tradeoff problem. From results obtained from simulation experiments, we can make the following conclusions: 1) for networks with medium and low utilization the SBPT algorithm consumes the least amount of bandwidth while it achieves the same shortest path length as achieved by the SPT algorithm, 2) the Greedy algorithm is efficient in bandwidth consumption only when the utilization is high, and 3) the SPT algorithm works best when the utilization is very low. As would be expected, for networks with very low utilization, the performance of the SBPT algorithm is about the same as that of the SPT algorithm. The complexity of the SBPT algorithm was shown to be the same order as that of combined cost of the SPT and Greedy algorithms.

References

[1] K. Almeroth and M. Ammar, "Multicast Group Behavior in the Internet's Multicast Backbone (Mbone)," *IEEE Communications Magazine*, Vol. 35, No. 6, pp. 124 - 129, June 1997.

[2] K. Bharath-Kumar, and J. Jaffe, "Routing to Multiple Destinations in Computer Networks," *IEEE Transactions on Communications*, Vol. COM-31, No. 3, pp. 343 - 351, March 1983.

[3] T. Billharts, B. Cain, E. Farry-Goudreau, B. Rieg, and S. Batsell, "Performance and Resource Cost Comparisons for CBT and PIM Multicast Routing Protocols," *IEEE Journal on Selected Areas in Communication*, Vol. 15, No. 3, pp. 304 - 315, April 1997.

[4] J. Cho, and J. Breen, "Analysis of the Performance of Dynamic Multicast Routing Algorithms," submitted to *ICCCN*, June 1998. URL: *http://cs-tr.cs.cornell.edu:80/Dienst/UI/1.0/Display/xxx.cs.NI/9809102*.

[5] M. Crovella, M. Harchol-Balter, and C. Murta, "Task Assignment in a Distributed System: Improving Performance by Unbalancing Load," *Technical Report BUCS-TR-1997-018*, October 1997.

[6] S. Deering and D. Cheriton, " Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems*, Vol. 8, No. 2, pp. 85 - 111, May 1990.

[7] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms," *IEEE Journal on Selected Areas in Communications*, Vol. 15, No. 3, pp. 277 - 290, April 1997.

[8] H. Fujinoki and K. Christensen, "A Routing Algorithm for Dynamic Multicast Trees with End-to-End Path Length Control," to appear in *Computer Communications*.

[9] R. Govindan and A. Reddy, "An Analysis of Internet Inter-Domain Topology and Route Stability," *Proceedings of IEEE INFOCOM*, pp. 850 - 857, 1997.

[10] M. Hofmann, "A Generic Concept for Large-Scale Multicast," *International Zurich Seminar on Digital Communication*, Broadband Communications: Lecture Notes in Computer Science, No. 1044, Editor: B. Plattner, Springer Verlag, February 1996.

[11] V. Kompella, J. Pasquale, and G. Polyzos, "Multicasting for Multimedia Applications,"

Proceedings of IEEE INFOCOM, pp. 2078 - 2085, 1992.

[12] C. Noronha and F. Tobagi, "Optimum Routing of Multicast Streams," *Proceedings of IEEE INFOCOM*, pp. 865 - 873, 1994.

[13] G. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," *Proceedings of IEEE INFOCOM*, pp. 353 - 360, 1996.

[14] B. Waxman, "Routing of Multipoint Connections," *IEEE Journal of Selected Areas in Communications*, Vol. 6, No. 9, pp. 1617 - 1622, December 1988.

[15] L. Wei and D. Estrin, "The Tradeoffs of Multicast Trees and Algorithms," *Proceedings of the Fourth International Conference on Computer Communications and Networks*, pp. 150 - 157, September 1995.

[16] P. Winter, "Steiner Problem in Networks: A Survey," *Networks*, Vol. 17, No. 2, pp. 129 - 167, 1987.

[17] Q. Zhu, M. Parsa, and J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrained Minimum-Cost Multicasting," *Proceedings of IEEE INFOCOM*, pp. 377 - 385, 1995.