RC23043 (W0312-113) December 23, 2003
Computer Science

IBM Research Report

Overlay Multicast Trees of Minimal Delay

Anton Riabov
Columbia University
IEOR
500 West 120 Street
New York, NY 10027

Zhen Liu, Li Zhang
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Overlay Multicast Trees of Minimal Delay

Anton Riabov
Columbia University, IEOR
500 West 120 Street
New York, NY 10027
e-mail: avr11@columbia.edu

Zhen Liu and Li Zhang
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
e-mail: {zhenl,zhangli}@us.ibm.com

***Abstract*— Overlay multicast (or application-level multicast) has become an increasingly popular alternative to IP-supported multicast. End nodes participating in overlay multicast can form a directed tree rooted at the source using existing unicast links. For each receiving node there is always only one incoming link. Very often, nodes can support no more than a fixed number of outgoing links due to bandwidth constraints. In this paper, we describe an algorithm for constructing a multicast tree with the objective of minimizing the maximum communication delay (i.e. the longest path in the tree), while satisfying degree constraints at nodes. We show that the algorithm is a constant-factor approximation algorithm. We further prove that the algorithm is asymptotically optimal if the communicating nodes can be mapped into Euclidean space such that the nodes are uniformly distributed in a convex region. We evaluate the performance of the algorithm using randomly generated configurations of up to 5,000,000 nodes.**

I. Introduction

In many applications, such as Internet-based content distribution networks, a desirable means of delivering information is *multicast*, delivering information simultaneously to a chosen group of hosts. Currently a set of standards exists for supporting multicast in IP networks. However, *overlay* (application-layer) multicast ([6], [13]) has become an increasingly popular alternative to network-supported IP multicast. While IP multicast is not universally available on the Internet, and requires allocation of globally unique IP address for each communicating group, overlay multicast can be easily implemented over existing infrastructure, and no global group identifier is required. The interested reader is referred to [7] for a comprehensive survey on this topic.

In simplest multicast scenario, a dedicated source host delivers information to a group of receiving hosts. Overlay multicast is implemented in application layer, and all the data is transmitted via unicast delivery supported in the underlying network. Because of bandwidth limitations, it may not be possible to simultaneously send data from the source to each receiving host via unicast. An implementation of overlay multicast uses receiving hosts to forward information to other receivers. If the data stream intensity does not change, it is natural to assume that each participating host has a fixed bound on the number of hosts to which it can communicate. Bandwidth capacity constraints of this kind translate into degree constraints on the nodes of the multicast tree. In this case, to initiate overlay multicast, one needs to construct a degree-constrained spanning tree in a complete graph, where the nodes correspond to the hosts, and the edges correspond to the unicast communication paths.

An important practical problem in this context is to determine how to construct a multicast tree which minimizes the largest communication delay observed by the receiving hosts during a multicast. Various studies have been conducted whose primary focus is protocol development for efficient overlay tree construction and maintenance. Examples are Narada [6], Yoid [8], ALMI [13], Host Multicast [20], NICE [3], Delauney graph [10]. Some other work in peer to peer network is related to the tree construction in application level multicast; see e.g. Chord [18] and CAN [14]. Most of such studies have been experimental in nature; see e.g. [5], [6], [4], [9], [13], [1]. In particular, Chu et. al. [5] use a heuristic called *Bandwidth-Latency* to build the multicast overlay tree. This heuristic, described in more detail in [19], selects paths by choosing those with the greatest available bandwidth (i.e., maximum possible fanout).

We note that this tree construction problem corresponds to a graph-theoretic problem of constructing a rooted spanning tree of minimum radius with degree constraints. Various versions of the problem have been studied in the literature. The famous Traveling Salesman Problem [2] is a special case. In general, the degree-constrained spanning tree problem is harder than the TSP. In [17], and later [16] and [15], the authors describe an NP-hard *minimum diameter, degree-limited spanning tree problem (MDDL)*, and propose heuristics for solving it. In the minimum-diameter version they consider, the objective is to minimize largest communication delay between any pair of participating nodes. However, the quality of the heuristic solution observed in simulations described in [17] decreases as the number of nodes increases.

In [11], Malouch *et. al.* introduce the radius minimization version, where the distance to the root is minimized. The authors prove that the problem in general is NP-hard, and show that in the special case of unit node-to-node delays the problem can be solved optimally in polynomial time. For the case of general distances a set of heuristics is described.

For all the proposed heuristics, the scalability issue remains open. Namely, the worst-case delay bound proven for these algorithms may grow quickly with the size of the system.

In this paper we assume that each node can be mapped to a point in Euclidean space, and node-to-node delays can be approximated by Euclidean distances between these points. Under this assumption, we describe an algorithm for constructing a degree-constrained spanning tree, and show that it arrives at asymptotically optimal solution. The asymptotic optimality result holds if points are uniformly distributed inside a convex region in Euclidean space, and at least 2 outgoing links are allowed at each node. This result easily extends to the non-uniform distribution case, with the only requirement that density function is strictly more than some constant $\epsilon > 0$ inside the convex region, and is zero everywhere else.

The method of mapping hosts to points in Euclidean space, with delays corresponding to Euclidean distances, is often used in the analysis of overlay networks. For example, [16] and [10] use geographical locations of computers to create a mapping of hosts to the two-dimensional plane. The advantage of this method is that no actual network delays need to be measured to construct the mapping, and subsequently the multicast tree. Another approach, proposed in work by the Global Network Positioning group [12], achieves higher accuracy by measuring some of the delays, and mapping hosts into Euclidean spaces of dimension 3 and above. In our work, we assume that the mapping has already been done, for example, using one of the methods above. We will concentrate on constructing the degree-constrained spanning tree with minimal radius.

We organize our presentation into four parts. First, we present a simple constant-factor approximation algorithm for solving the problem in Euclidean space. We use a constant-factor algorithm as a subroutine of the asymptotically optimal algorithm, to connect points inside cells of a polar grid. Next, we describe our asymptotically optimal algorithm for the special case of out-degree at least 6 at each node, and with points uniformly distributed in a two-dimensional disk, and prove asymptotic optimality. We follow by describing how to extend the algorithm to work in higher dimensions, with general degree constraints, and with general convex regions. Finally, we analyze algorithm performance through simulation.

II. Constant Factor Approximation

Before we can describe the asymptotically optimal algorithm, which is the main focus of this paper, we need to introduce a subroutine for connecting points within cells of a polar grid. This subroutine in itself is an approximation algorithm. It creates a valid degree-constrained spanning tree for a given set of points in Euclidean space. The length of the longest path in the tree is within a constant factor of the best solution among all the possible degree-constrained spanning trees. This constant approximation factor is independent of the number of points in the region. Although it is easier to describe a version of the algorithm for a square, we will describe a polar version, which is more suitable for this paper's setting.

Consider ring segment shown on Figure 1 a), with inner radius r, outer radius R, and angle a. Suppose all the points are contained within this segment. Assume that the source point, which is the root of the tree, is also specified. The algorithm proceeds recursively as following:

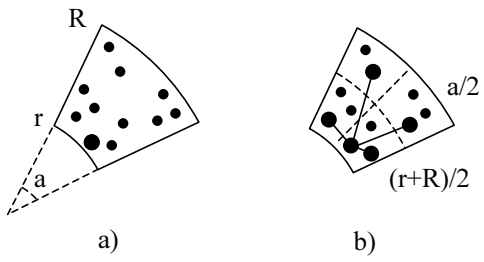

Fig. 1. Constant factor approximation algorithm.

Bisection Algorithm

1) Divide the segment into 4 sub-segments, by splitting it with an arc of radius $(R+r)/2$ and a ray dividing angle a into two halves.
2) Pick a representative point in each non-empty sub-segment, such that its' radius in polar coordinates is closest to the radius of the source node. Connect the source to all the representatives. See Figure 1 b).
3) Repeat the procedure within each non-empty sub-segment, to connect the points inside the sub-segment, using the representative point as a local source.

The algorithm constructs a spanning tree in which each node has at most 4 children. We observe that each path always moves monotonically along the radius axis. The steps along the angle axis at each level can be bounded by the angle of the sub-segment. Therefore, the length of each path l_p can be bounded from above using the triangle inequality as follows:

$$l_p \leq \max(R-q, q-r) + Ra + Ra/2 + Ra/4 + ... \leq$$
$$\leq \max(R-q, q-r) + 2Ra, \quad (1)$$

where q is the radius of the source node.

We will now show that this algorithm can be used to construct a constant factor approximation for a given set of nodes. We first construct a ring segment to cover all these points. We pick the center of the ring to be very far, so that the angle a is small, ($\sin a > 5/6a$), and both R and r are large, such that $r > 0.6R$. Pick R and r such that $R - r$ can not be reduced, without leaving some nodes out of the ring segment. Similarly, assume that a can not be reduced. Then, since any path must connect to extreme nodes, and using triangle inequality, it is easy to see, that for the optimal longest path OPT the following holds:

$$OPT \geq \max(R-q, q-r),$$
$$OPT \geq r \sin a \geq {}^1/_2 Ra.$$

Combining this with (1), for any tree path p, we obtain:

$$l_p \leq 5 \cdot OPT,$$

and therefore this algorithm can be used to produce solutions within a constant factor of the best possible.

It is not difficult to modify the algorithm to produce a spanning tree with out-degree 2. To do this, during each recursive call, connect the source to two points from the same segment. Points should be chosen to have a radius closest to the source. Then each of the two points can be used to connect 2 of the 4 sub-segments, so that all sub-segments are connected. In this case, the upper bound on the solution doubles the angle term, since on each level of the path we now use 2 links instead of one:

$$l_p \leq \max(R-q, q-r) + 4Ra. \tag{2}$$

Theorem 1: The Bisection Algorithm provides a solution within a factor of 5 times optimal for the minimum radius problem when maximum out-degree is restricted to be 4. The approximation factor becomes 9 if the maximum out-degree is restricted to be 2.

III. Asymptotically Optimal Algorithm

In this section we describe our hierarchical algorithm to recursively build multicast trees. We will prove that it is asymptotically optimal. To simplify the presentation, we will make several assumptions. These assumptions will be lifted in the next section.

We assume that the n points corresponding to the communicating hosts are uniformly distributed inside a disk of radius 1 and the source is located at the center of the disk. We assume each node can forward transmission to at least 6 down-stream links. The main idea of the algorithm is to divide the disk into a hierarchy of smaller and smaller grid cells. The algorithm builds a tree based on the hierarchy to connect the points in the grid cells. At a high level, our grid partitioning algorithm proceeds in three stages:

Algorithm Polar_Grid

1) Create a grid of equal area cells, partitioning the disk.
2) Connect the cells, using cell representatives, and form a core network.
3) Connect points within the cells, using the constant factor approximation algorithm.

We describe the details of each step of the algorithm in the following subsections. We then evaluate the performance of the algorithm and prove the asymptotic optimality results.

A. Constructing the polar grid.

First, the algorithm creates a polar grid covering the unit disk. This grid must have the following properties:

1) All cells of the grid have the same area.
2) Cells are organized in rings. Each containing ring has twice more cells than the ring immediately inside it.
3) There is at least one point in each cell of the grid, except for the cells in the outermost ring.

For a fixed number of rings k, we construct the grid by dividing the unit disk using k circles with the same center, and radius

$$r_i = 1/\sqrt{2}^{k-i}, \quad 0 \leq i \leq k-1. \tag{3}$$

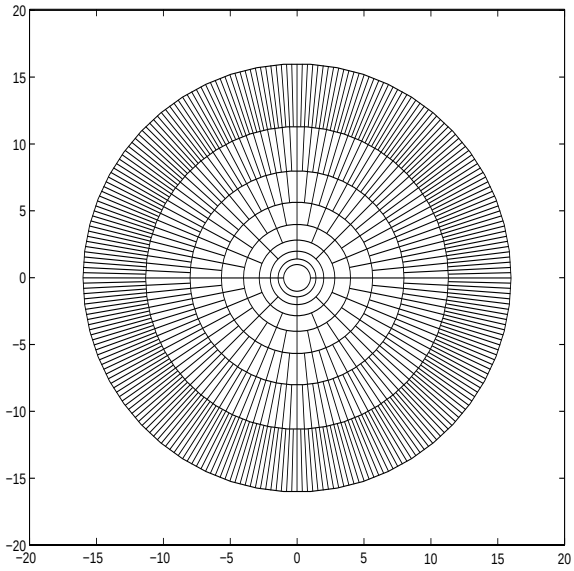

Fig. 2. Dividing the disk into ring segments of equal area.

We further divide each ring i into 2^i equal segments such that each cell segment on level i is aligned with 2 segments on level $i+1$ (see Figure 2).

Since the radius of ring i is chosen such that $r_i = \sqrt{2}r_{i-1}$, the area of disk bounded by circle i is twice the area of disk bounded by circle $(i-1)$. If we imagine that there are two cells inside circle 0, then it is easy to see that for each i, circle i contains twice as many cells as circle $(i-1)$, and therefore property 1) holds.

Given a set of points, we choose the number of rings k as large as possible, such that property 3) is satisfied. In the analysis section we will show that k increases as the number of points n increases.

B. Connecting the cells.

According to property 3) of the grid, each ring segment contains at least one point (except for the outermost segments). We can choose a point within each segment to be the representative of the segment. If there is more than one point in a segment, choose the point that is closest to the center on the inner arc of the segment. Cell representatives are connected in a binary tree, rooted at the source node in the center of the unit disk. Each representative is connected to two representatives of next ring cells, aligned with its cell. Outermost ring cells that do not have any points are ignored.

C. Connecting remaining points within cells.

Finally, in each cell that contains more than one point, we run the constant factor algorithm described in the previous section. The algorithm connects all the remaining points, and the distribution tree is completely constructed.

The constant factor algorithm requires out-degree 4, and additional out-degree 2 can be used at the representative node to connect to next level cells, and therefore the resulting

spanning tree will have maximum out-degree 6. We will improve on this estimate in the next section.

D. Lemmas.

In order to prove asymptotic optimality of the solution, we need to show that k increases as a function of number of nodes n. To do this, we need to introduce the following two lemmas.

Lemma 1: If each of n balls is uniformly and independently assigned to one of n^α buckets (for some fixed α), the probability $p_\alpha(n)$ of having at least one empty bucket after the assignment is complete satisfies

$$p_\alpha(n) \le n^\alpha e^{-n^{1-\alpha}}. \tag{4}$$

Proof: The probability of having at least one bucket empty is bounded from above by the sum of probabilities of having each of the buckets empty. Therefore,

$$p_\alpha(n) \le n^\alpha \left(1 - \frac{1}{n^\alpha}\right)^n.$$

Note that $1 - x \le e^{-x}$ for any x, and inequality (4) follows. ∎

The Corollary below follows immediately.

Corollary 1: If $\alpha < 1$, then $p_\alpha(n) \to 0$ as $n \to \infty$.

Since we are interested in deriving an asymptotic result, Corollary 1 would suffice for our analysis. However we would like to know the values of α that can give useful results even for small n. The following lemma gives some insight.

Lemma 2: If $\alpha \le 1/2$, then $p_\alpha(n) \le e^{-1}$ for all $n \ge 1$.

Proof: Consider $f_\alpha(x) = x^\alpha e^{-x^{1-\alpha}}$. Assume that $0 < \alpha < 1$ and $x \ge 0$. Observe that in this case $f_\alpha(x)$ is a concave function of x. By taking derivative, we can show that it reaches its maximum at

$$x^*_\alpha = \left(\frac{\alpha}{1-\alpha}\right)^{1/(1-\alpha)}.$$

Notice that x^*_α is increasing in α and $x^*_{1/2} = 1$. Therefore if $\alpha \le 1/2$, the maximum is attained at some $x^*_\alpha \le 1$, and hence for $x \ge 1$ function $f_\alpha(x)$ is non-increasing. Furthermore, for any α, $f_\alpha(1) = e^{-1}$. The lemma follows from equation (4), i.e. $p_\alpha(n) \le f_\alpha(n)$. ∎

Therefore, since in k-ring grid there are 2^{k+1} cells, with high probability we can say that if we require at least one point in each cell, then

$$\sqrt{n} \le 2^{k+1},$$

and therefore,

$$k \ge 1/2 \log_2 n. \tag{5}$$

In our analysis we will assume that n is sufficiently large and $k \ge 1$.

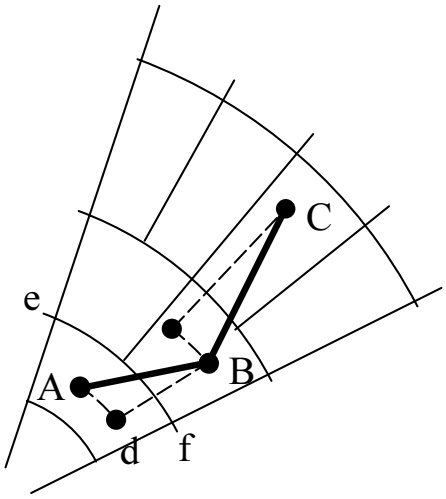

Fig. 3. Proof of upper bound on longest path.

E. Solution analysis.

We can now evaluate the solution quality based on the uniform distribution assumptions. It is easy to see that, as the number of nodes n increases, the lower bound of the optimal solution cost (the longest distance from disk center to any point) approaches 1 from below. To complete the proof, we need to show that an upper bound on solution obtained by the algorithm approaches 1 from above.

Any path P in the constructed spanning tree consists of two parts: the sub-path p connecting cell representatives, and the sub-path q between the points in the last cell, constructed by the constant factor approximation algorithm:

$$l_P = l_p + l_q.$$

Making use of (1), we can write

$$l_q \le \max(R - q, q - r) + 2Ra,$$

for some R, r, a, q, defined by the last cell of path P.

Using the polar version of the triangle inequality, the length of the path can be bounded from above by computing the radius and arc components separately. The path which follows the cell boundaries is an upper bound. For example, in Figure 3, the length of AB is less than $Ad + dB$, and arc Ad can be upper-bounded by arc ef. The total length of all the ray segments (similar to dB) is at most 1 minus the radius of the disk. The $\max(R - q, q - r)$ component of l_q can be included in this estimate as well, since we pick the least-radius point to be our cell representative.

Thus,

$$l_P \le 1 + 2Ra + S_k, \tag{6}$$

where S_k is the sum of arc lengths for inner $(k-1)$ circles of k-ring grid.

Let Δ_i be the length of an arc segment on circle i in the polar grid:

$$\Delta_i = 2\pi \frac{1}{\sqrt{2}^{k-i}} \cdot \frac{1}{2^i} = \frac{2\pi}{\sqrt{2}^{k+i}}, \quad 0 \le i \le k.$$

In our estimate of S_k only the inner arcs are involved, i.e. arcs 1 through k-1. Hence,

$$S_k = \sum_{i=1}^{k-1} \Delta_i = \frac{2\pi}{\sqrt{2}^{k+1}} \cdot \frac{1 - 1/\sqrt{2}^{k-1}}{1 - 1/\sqrt{2}}.$$

Recall that Ra in (6) is an arc length as well, for some ring j:

$$Ra \leq \Delta_j.$$

We can rewrite (6) as following:

$$l_P \leq 1 + 2Ra + S_k \leq 1 + 2\Delta_j + S_k. \tag{7}$$

We can show that the right-hand side of inequality (7) approaches 1 from above as n approaches infinity. Here is the precise argument. Both Δ_j and S_k are infinitesimal as k goes to infinity. For any arbitrary small $\epsilon > 0$, there exists a K such that when $k > K$, the delay value corresponding to the solution l_p is less than $1 + \epsilon/2$. Based on Corollary 1, for any arbitrary small $\delta > 0$, there exists an N_1, such that when $n > N_1$, the probability of having at least one point in each cell is larger than $1 - \delta/2$. It is also easy to show that there exists an N_2 such that when $n > N_2$, the probability of having a point in the ring between the circle of radius $1 - \epsilon/2$ and the unit circle is larger than $1 - \delta/2$. This implies the minimum radius is at least $1 - \epsilon/2$. Therefore, with probability at least $1 - \delta$, when $n > \max\{N_1, N_2\}$ the minimum radius is at least $1 - \epsilon/2$. At the same time there is at least one point in each of the grid cells, which implies $l_p < 1 + \epsilon/2$. Under this condition, the length of the longest path in this tree is within ϵ plus the value of the optimal solution. This completes the proof for the asymptotic optimality of Algorithm Polar_Grid:

Theorem 2: For any small $\epsilon, \delta > 0$, there exists an N such that with probability greater than $1 - \delta$, when the number of points n is larger than N, the length of the longest path in the tree produced by Algorithm Polar_Grid is within ϵ plus the optimal solution.

IV. Generalization

In the previous section, in order to simplify the presentation, we assumed that points are uniformly distributed inside a disk, the out-degree of at least 6 is allowed, and the points belong to two-dimensional space. The algorithm can be adjusted accordingly to remove these constraints. We describe the necessary changes in this section.

A. Out-Degree 2

There is a version of the asymptotically optimal algorithm, in which at least out-degree 2 must be allowed at every node. In other words, it is possible to construct a binary tree with the same asymptotic optimality property.

We have discussed how to adjust the constant factor approximation algorithm in Section II. A few changes have to be made to the algorithm that chooses and connects the cell representatives. In each cell, three cases are possible:

1) There is only one point in the cell. Make it a cell representative, and use it to connect to the two cells in the next ring.
2) There are two points in the cell. Choose a point closest to the center of the disk as the cell representative. Connect the representative directly to the other point. Then connect the second point to the two cells in the next ring.
3) There are three or more points in the cell. Choose the point closest to center point as the cell representative. Pick one of the remaining cells to be the center for connecting other points in the cell. Choose another point for connecting cells in the next ring. The two special points are connected directly to the representative point.

The asymptotic analysis and the constant factor approximation analysis are very similar. The only difference is that the contributions from the arcs need to be doubled. This is the case because now two links are used in each cell, instead of just one link. Since this contribution is infinitesimal for large n, the constant multiplier can be ignored, and the same proof can be used to show asymptotic optimality.

B. Higher Dimensions

The algorithm can be adjusted to work in dimensions higher than two. The most important component of the proof is the creation of the polar grid, which satisfies properties 1)-3). The grid can be created similarly, in polar coordinates, by splitting the d-dimensional sphere into segments. The radius of each subsequent ring should equal to previous ring radius, multiplied by $\sqrt[d]{2}$ (so it has twice the volume). Each cell is split into two along a splitting axis. The splitting axes are chosen to cycle through all the axes. Although the details of equal volume split become tedious, a similar proof can be constructed.

C. General Convex Region

Proving asymptotic optimality for a circle (d-sphere), with the source in the center, implies asymptotic optimality in any convex region with arbitrary source placement inside the region. The algorithm constructs the smallest ring covering all points and centered at the source, and proceeds similarly as the circle case. The analysis is very similar. In this case, the lower bound on the longest path approaches the outer ring radius from below.

V. Experiments

In this section, we provide some experimental results to illustrate the quality, running time and other properties of our heuristic algorithms, for problems of different sizes. For each problem size, we have generated 200 random sets of points, uniformly distributed inside the unit disk. The average maximum delay and other parameters of solution trees are computed. We have tested both the out-degree 6 and out-degree 2 versions of the algorithm. We have also evaluated the performance of the three-dimensional version of the algorithm to connect points uniformly distributed inside a unit sphere.

TABLE I

EXPERIMENT RESULTS.

		Out-Degree 6					Out-Degree 2				
Nodes	Rings	Core	Delay	Dev	Bound	CPU Sec	Core	Delay	Dev	Bound	CPU Sec
100	3.61	1.53	1.852	0.20	7.18	0.002	2.21	2.634	0.31	10.74	0.0015
500	5.26	1.22	1.420	0.08	4.92	0.01	1.61	1.876	0.15	6.96	0.01
1,000	6.06	1.13	1.302	0.05	4.09	0.02	1.40	1.622	0.11	5.66	0.02
5,000	8.01	1.00	1.142	0.02	2.65	0.08	1.12	1.285	0.04	3.44	0.08
10,000	8.97	0.99	1.102	0.02	2.20	0.17	1.06	1.202	0.03	2.76	0.17
50,000	11.00	0.94	1.049	0.01	1.61	0.96	0.98	1.095	0.01	1.88	1.02
100,000	11.98	0.95	1.034	0.00	1.43	2.01	0.97	1.067	0.01	1.63	2.13
500,000	14.00	0.92	1.016	0.00	1.22	11.06	0.93	1.031	0.00	1.32	11.84
1,000,000	15.00	0.93	1.012	0.00	1.15	22.99	0.94	1.022	0.00	1.22	24.52
5,000,000	17.00	0.91	1.005	0.00	1.08	132.34	0.91	1.009	0.00	1.11	142.08

The experiments are run on an Intel Pentium II 400 Mhz computer with 128 megabytes of RAM.

All the data obtained in our experiments on unit disk is shown in Table I. Column one contains n, the number of nodes to be connected. The Second column, "Rings", is the average value of k that is, the number of rings for this problem size. Columns 3 and 8, "Core", contain the average *core delay* – the longest portion of the path between cell representative nodes. Columns 4 and 9, "Delay", show the average longest delay observed in the solution tree. Columns 5 and 10, "Dev" display the standard deviation of the longest delay. The lower bound on the delay is close to 1, so the closer delay is to 1, the better. "Bound" columns show the value of the upper bound given by equation (7), evaluated at $j = 0$. The reason to pick $j = 0$ is because $\Delta_0 \geq \Delta_j$ for all j. In the formula for upper bound, the coefficient of Δ_j should be doubled for out-degree 2 trees. Finally, the "CPU Sec" column contains the computation times.

To illustrate our results, we have included a set of plots, based on data shown in Table I. The results demonstrate that the algorithm converges very quickly.

Figure 4 shows the maximum sender-to-receiver delay, together with the delay bound and the core delay. The horizontal axis representing the number of nodes is in logarithmic scale. This is also the case for plots 5 and 6. The bound used in the analysis of the algorithms significantly over-estimates the delay for problems with a small number of nodes. The bound becomes better and better as the number of nodes increases. The difference between the core and the total delay does not diminish. This is because the difference depends on the radius of the outermost ring, which remains constant as the number of nodes increases.

Figure 5 combines the plots on Figure 4, and compares the maximum delay for degree 2 and degree 6. The delay overhead of degree 2 trees is almost 2 times the overhead of degree 6 trees. This is intuitive, since there is the same relationship between the bounds on the lengths of the paths. As the number of nodes increases, the degree of each particular node becomes less and less important, and the two curves all converge to the best possible delay of one.

Figure 6 shows how the number of rings, k, in the grid created by the algorithm changes with the number of nodes, n. The node axis is again in logarithmic scale. The points follow almost a straight line. This indicates that there is a logarithmic dependence, which is implied by (5).

Figure 7 shows how the running time of the program increases with the number of nodes. The small insert plot shows the details for problems with nodes between 100 and 10,000. The plot allows us to evaluate the general trend of the algorithm complexity. In our experiments we observed that running time increases almost linearly, which makes it possible to run the algorithm for networks with very large sizes. We remark that our straightforward implementation of the algorithm can probably be fine tuned and improved. Furthermore, in practice, the running time will depend on the hardware and software environment used.

In fact, during the assignment of points to the grid cells, our algorithm inspects each point only once, which requires $O(n)$ operations. Then, the bisection algorithm must divide ring segments, and enumerate all the points within each segment. Given m points, the bisection algorithm will create at most m non-empty segments. In the worst case, the number of operations at this stage can be estimated as $O(m^2)$, since each point may be inspected during the processing of each segment. Since the distribution of points is uniform, the total running time of our algorithm will be linear in n with high probability. This can be intuitively explained by the following argument. Since points are distributed uniformly between cells, the average number of points in each cell is $n/2^k$. Our experiments confirm that the relationship between k and n stated in (5) holds, i.e., k is a logarithmic function of n (see Figure 6). Because of this relationship, the number of points per cell remains constant on average, independent of n. Therefore, the running time of bisection in each cell is also roughly constant. Since we require at least one point to be contained in each cell, the total number of cells is at most $O(n)$. Therefore, the total number of calls to the bisection procedure is at most $O(n)$, leading to an overall number of operations which is $O(n)$.

Finally, in Figure 8 we demonstrate algorithm convergence results in the three-dimensional unit sphere. Similar to the unit disk case, we run 200 experiments for each problem

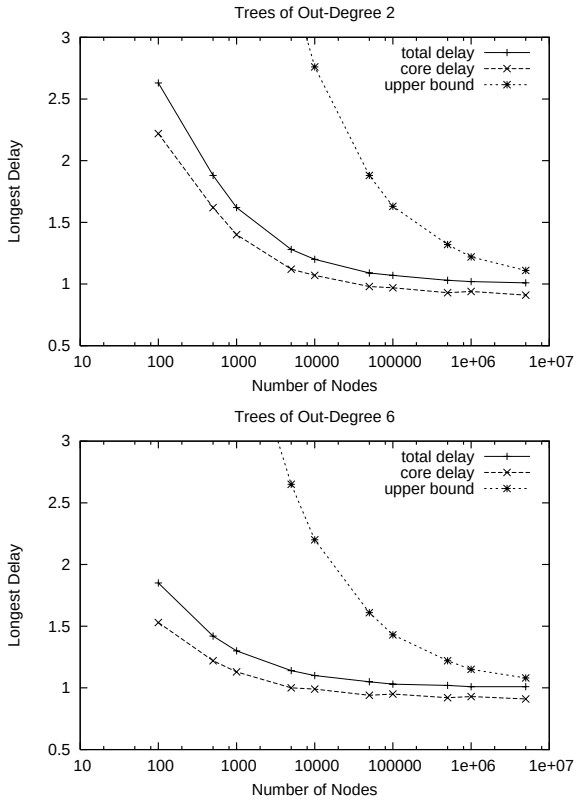

Fig. 4. Average maximum delay compared to bounds.

size, and computed the average longest path length. For three dimensions, the straightforward extension of our algorithm builds a tree of out-degree 10. In the bisection algorithm, each cell representative node uses 2 links to connect to cells in the next ring, and uses at most 8 links to connect to points inside the cell. As in two dimensions, we modify the algorithm to construct trees of out-degree no more than 2. In both cases, the longest path length converges to the lower bound of 1.

Similar to the longest path results on unit disk, shown in Figure 5, the difference in three dimensions between out-degree 2 and out-degree 10 trees becomes less noticeable as the number of nodes increases. Although the asymptotic optimality holds in general multi-dimensional Euclidean space, Figure 8 shows that the largest delay in 3 dimensions is higher than in 2 dimensions, for problems with the same number of nodes. This can be explained by the increase in the average distance between uniformly distributed points, as the dimensionality of the unit sphere increases and number of points remains constant.

VI. Conclusion

We have investigated the problem of constructing an overlay multicast tree that minimizes the largest sender-to-receiver delay, and satisfies bandwidth constraints by limiting the out-degree of nodes in the tree. We approach the problem by creating a mapping of communicating hosts to points in multi-dimensional Euclidean space. Using methods described in [12] or [16] and [10], the mapping approximates the unicast communication delays between hosts to the distances between points in the Euclidean space.

In this setting, we describe a simple bisection algorithm to construct a tree with maximum delay within a constant factor of optimal for any set of nodes. Next, we assume that the communicating points are randomly distributed inside a two-dimensional disk centered around the sender. We describe another approximation algorithm to build a tree with maximum out-degree 6. The algorithm uses the bisection method as a subroutine in each cell of a polar grid. We prove that the algorithm creates a tree with maximum delay that asymptotically approaches the best possible, as the number of nodes increases. This result implies that as the number of communicating hosts grows, it becomes possible to construct better and better overlay multicast trees. In particular, for many remote leaf nodes, the communication delay from the source will decrease.

Next, we describe how to extend our grid-based algorithm to construct trees of out-degree no more than two, to work with points in more than two dimensions, and to work with general convex regions, not limited to spheres. We show that asymptotic optimality is preserved during all modifications we make.

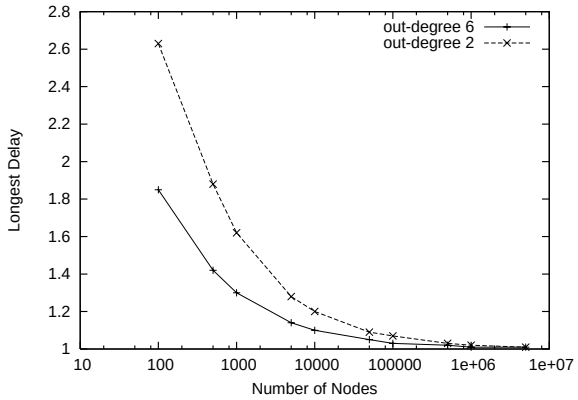

Fig. 5. Comparison of average maximum delay for out-degrees 2 and 6.

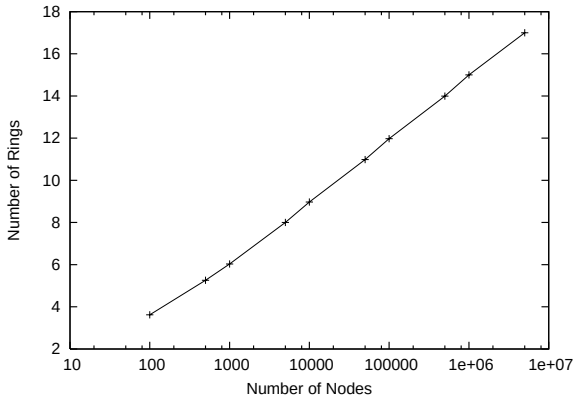

Fig. 6. Average number of rings in polar grid.

Finally, we implement the algorithm, and perform simulation experiments to analyze its performance. Experimental results show that, in practice, the algorithm converges much faster than the theoretic bound we derived. Furthermore, the experiments confirm that the running time of the algorithm grows almost linearly with the number of nodes, and the algorithm is reasonably scalable.

Our algorithm can also be applied to the minimum diameter version of the problem, described in [17] and other papers. The bisection algorithm provides a larger factor approximation for the minimum diameter problem. But the asymptotic optimality of our algorithm can not be guaranteed for any convex region in Euclidean space, although the result still holds for points uniformly distributed in a sphere. To construct an optimal solution in the sphere, an artificial root node should be chosen among nodes closest to the sphere center. In general convex regions, the algorithm will only find a tree with delay within factor of 2 of the optimal as the number of nodes becomes large.

Since for all mapping methods, there is usually a discrepancy between the Euclidean distances and the actual transmission delays, it is interesting to see how well the algorithm performs in combination with the mapping. We leave this for future work. Also, we note that in practice, there is interest in a decentralized version of the algorithm.

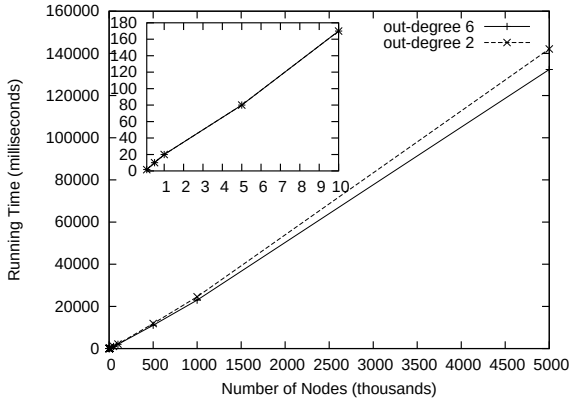

Fig. 7. Algorithm running time.

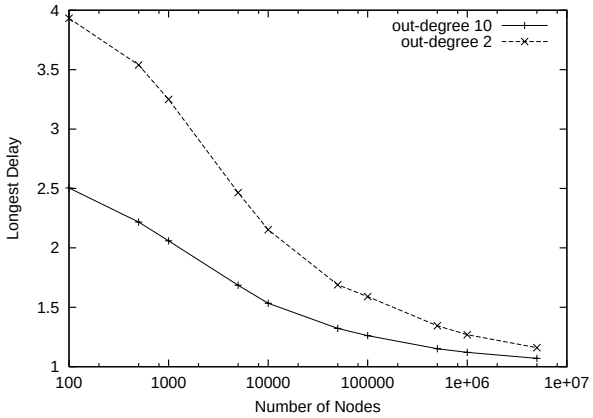

Fig. 8. Average maximum delay in three-dimensional unit sphere.

REFERENCES

[1] Akamai Corporation. Internet Bottlenecks: The Case for Edge Delivery Services, 2000. Akamai whitepaper.

[2] S. Arora. "Polynomial-time Approximation Schemes for Euclidean TSP and other Geometric Problems", Journal of the ACM 45(5) 753-782, 1998.

[3] S. Banerjee, B. Bhattacharjee and C. Kommareddy, *Scalable Application Layer Multicast*, in Proceedings of ACM Sigcomm 2002.

[4] Y. Chawathe, S. McCanne, and E. A. Brewer, *RMX: Reliable Multicast for Heterogeneous Networks*, in Proceedings of IEEE Infocom, 2000.

[5] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA, August 2001.

[6] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in Proceedings of ACM Sigmetrics, June 2000.

[7] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network Magazine*, Jan/Feb 2000.

[8] P. Francis, *Yoid: Extending the Internet Multicast Architecture*, http://www.icir.org/yoid/docs/yoidArch.ps.gz (April 2000).

[9] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, *Overcast: Reliable Multicasting with an Overlay Network*, in Proceedings of the 4th Symposium on Operating Systems Design and Implementation, Oct. 2000.

[10] J. Liebeherr and M. Nahas. "Application-layer Multicast with Delaunay Triangulations", Global Internet Symposium, IEEE Globecom 2001, November 2001.

[11] N. M. Malouch, Z. Liu, D. Rubenstein and S. Sahu. "A Graph Theoretic Approach to Bounding Delay in Proxy-Assisted, End-System Multicast", Tenth International Workshop on Quality of Service (IWQoS 2002).

[12] T. S. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches", INFOCOM'02, New York, NY, June 2002.

[13] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure", in 3rd Usenix Symposium on Internet Technologies systems (USITS), March 2001.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A scalable content-addressable network* in Proceedings of ACM SIGCOMM (August 2001).

[15] S. Shi. "Design of Overlay Networks for Internet Multicast". Ph.D. Thesis, Washington University in St. Louis, August 2002.

[16] S. Shi, J. S. Turner. "Routing in Overlay Multicast Networks", IEEE INFOCOM, New York City, June 2002.

[17] S. Shi, J. S. Turner and M. Waldvogel. "Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks", The 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2001), Port Jefferson, New York, June, 2001.

[18] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. *Chord: A scalable peer-to-peer lookup service for Internet applications*, in Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, 2001, pp.149–160, Diego, California, United States.

[19] Z. Wang and J. Crowcroft. Bandwidth-delay Based Routing Algorithms. In *IEEE Globecom'95*, November 1995.

[20] B. Zhang, S. Jamin, L. Zhang, *Host Multicast: A Framework for Delivering Multicast To End Users*, in Proceedings of IEEE Infocom (2002).