Hyperbolic Embedding of Internet Graph for Distance Estimation and Overlay Construction

Yuval Shavitt, *Senior Member, IEEE*, and Tomer Tankel, *Student Member, IEEE*

***Abstract*—Estimating distances in the Internet has been studied in the recent years due to its ability to improve the performance of many applications, e.g., in the peer-to-peer realm. One scalable approach to estimate distances between nodes is to embed the nodes in some d dimensional geometric space and to use the pair distances in this space as the estimate for the real distances. Several algorithms were suggested in the past to do this in low dimensional Euclidean spaces.**

It was noted in recent years that the Internet structure has a highly connected core and long stretched tendrils, and that most of the routing paths between nodes in the tendrils pass through the core. Therefore, we suggest in this work, to embed the Internet distance metric in a hyperbolic space where routes are bent toward the center. We found that if the curvature, that defines the extend of the bending, is selected in the adequate range, the accuracy of Internet distance embedding can be improved.

We demonstrate the strength of our hyperbolic embedding with two applications: selecting the closest server and building an application level multicast tree. For the latter, we present a distributed algorithm for building geometric multicast trees that achieve good trade-offs between delay (stretch) and load (stress). We also present a new efficient centralized embedding algorithm that enables the accurate embedding of short distances, something that have never been done before.

I. Introduction

INTERNET distance estimation is important to improve performance of many applications, such as peer-to-peer application and application layer multicast. The network distance matrix can be compactly represented by mapping its nodes to a real geometric space. Such a mapping, called *embedding*, is designed to preserve the distance between any pair of network nodes close to the distance between their geometric images. A small subset of nodes, called *Tracers*, is embedded first, considering all inter Tracer distances. The coordinates of each of the other nodes is calculated by minimizing the distortion of the distances from this node to several or all Tracers. Euclidean embedding for predicting network distances was first suggested by Ng and Zhang [1], which named it Global Network Positioning (GNP). Lim *et al.* [2] suggest to use uncorrelated and orthogonal Cartesian coordinates to replace the minimization suggested at GNP. Tang and Crovella [3] suggest to use Lipschitz embedding, which ignore the distance between Tracers, and thus is less accurate.

Manuscript received January 21, 2005; revised December 14, 2005, and July 19, 2006; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z.-L. Zhang.

The authors are with the School of Electrical Engineering, The Iby and Aladar Fleischman Faculty of Engineering, Tel-Aviv University, Israel (e-mail: shavitt@eng.tau.ac.il; tankel@eng.tau.ac.il).

Digital Object Identifier 10.1109/TNET.2007.899021

Recently, we introduced BBS [4], a new numerical method for Euclidean embedding. Euclidean BBS had achieved the lowest embedding distortion with relatively low complexity compared to other numerical methods. In particular for AP embedding, defined hereon, our method was more accurate and far more scalable than DHS, the numerical method of GNP. While we achieved good embedding in [4] the results are far from perfect due to the Internet AS topology structure, which was shown to have a core in the middle and many tendrils connected to it [5], [6].

To understand the problem, consider embedding of the Internet in two dimensions. If the tendrils are placed with the correct distance from the core and are well spaced in all directions, the distance between them in the plane makes a shortcut not passing through the core and thus underestimates the real graph distance. Embedding in higher dimension space enables us to spread the tendrils tips farther apart, and thus improves the embedding, but at some point an increase in the number of dimensions gives us diminishing return. To overcome this effect and thus improve distance estimation accuracy, we introduced in [4] a threshold criteria above which simple triangulation is used. Although the threshold can be tuned, it does not reveal the geometric shape of the Internet structure.

In this paper we take a new and different approach for embedding the Internet graph in a geometric space. The core idea is to bend the line between two points in the tendrils to pass through the core and thus, follow the true Internet route. To make this happen we use hyperbolic geometric space where a distance unit decreases as one moves away from the origin. The calculation of distances in hyperbolic spaces is not significantly harder than in Euclidean spaces, which makes our approach practical. Our algorithm embeds the Internet graph into a hyperbolic space with preselected *curvature*, but was found to be insensitive to the exact curvature value. Therefore, we obtained the curvature to be used by an off-line iterative guessing algorithm. We were able to improve the performance of three applications: delay estimation (which can be used for QoS threshold estimation), server selection, and application level multicast.

A. Embedding Algorithms

An integrated embedding mechanism consists of three ingredients that distinguish it from its counterparts. The geometric space and the geometric distance function it defines is the first ingredient. The second ingredient is the algorithm of selecting one or more subsets of the pair distances in a given input metric. The third is the numerical method that calculates from the input of subset pair distances, the coordinates of each node, which minimize the symmetric distortion of these pairs. The *symmetric*

pair distortion is defined for each pair as the maximum of the ratio between the original and the geometric distance and its inverse. The three selection algorithms we are aware of are

1) All Pairs (AP)
2) Two phase (TP)
3) Log-Random and Neighbors (LRN)

The input to AP Euclidean embedding is the entire n-nodes metric, and these $n(n-1)/2$ distance pairs are embedded at once.

In TP Euclidean embedding, named GNP in [1], a small subset of $t \leq 15$ *Tracers* is embedded first, considering all $t(t-1)/2$ pair distances. The coordinates of other nodes are calculated from their distance to several or all Tracers by minimizing the symmetric distortion of these tn node-Tracer distance pairs.

We introduce here a third selection algorithm called **Log-Random and Neighbors (LRN)** embedding. Short distance pairs were largely overlooked by previous research, since their estimation errors were insignificant in some applications such as server selection. In [4] we compared distance estimation accuracy of GNP, TP Euclidean BBS, and IDMaps triangulation. For small distances, below 20% of the network diameter, all these methods yielded large relative estimation errors. TP embedding completely ignores the neighbor distances, whereas the input of AP embedding consists all $n(n-1)$ distance pairs, which is not a practical alternative. Aiming to increase neighbor distances accuracy, the LRN algorithm *concurrently* embed a subset of the entire metric, comprising the pairs whose distance is below a certain threshold, and the set R of randomly sampled pairs. The pairs in R are selected uniformly at random with probability $\log n/n$, and thus $|R| = n \log n$. The distance threshold is selected so that the number of distance pairs below the threshold is also $\Theta(n \log n)$. Thus, the total number of pairs in the embedded subset is $\Theta(n \log n)$.

Although the number of input pairs to LRN is similar to that of TP embedding, LRN cannot be calculated distributively since it embeds all the n nodes concurrently. The CPU complexity of the LRN calculation is higher than TP embedding, thus, BBS was the only scalable method that could handle medium to large LRN group sizes, $n = 200 \ldots 5000$.

B. Internet Embedding in Hyperbolic Space

As an Euclidean line, the hyperbolic line between two points is defined, as the parametric curve, connecting between the points, over which the integral of arc length is minimized. Unlike the Euclidean line, a hyperbolic line bends toward the origin point, $\mathbf{O}$, see Fig. 2. The amount of bent depends on the curvature of the hyperbolic space. As the space curvature increases, the bending becomes larger, and thus the hyperbolic distance between the points increases.

The Internet structure has been the subject of many recent works. Researchers have looked at various features of the Internet graph, and proposed theoretical models to describe its evolvement. Faloutsos *et al.* [7] experimentally discovered that the degree distribution of the Internet AS and router level graphs obey a power law. Barabási and Albert [8], [9] developed an evolutionary model of preferential attachment, that can be used for generating topologies with power-law degree distributions.

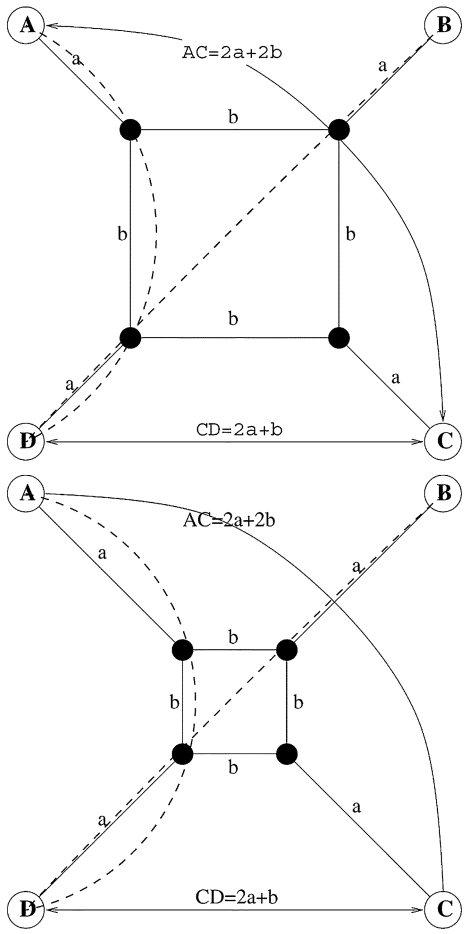

Fig. 1. An eight-node graph example in $\mathbb{D}^2$.

Around the core of the AS graph there are several rings of nodes all have tendrils of varying length attached to them [5], [6]. The average node degree decreases as one moves away from the core. Due to BGP policy based routing paths between peripheral nodes often pass through the core.

A very simplified example of the above structure is the eight node graph in Fig. 1. The shortest path distance between its four exterior nodes, denoted A, B, C, and D cannot be embedded without distortion in the two-dimensional Euclidean space. However, as we show below, there exist an embedding of these four points in a hyperbolic Poincaré disk with a specific curvature for which the hyperbolic distance matches the network distance between each of the node pairs. For this optimal curvature the ratio between shorter and longer pair hyperbolic distance, $|AB|/|AC|$, matches the corresponding network distances ratio. As can be seen in Fig. 1 the hyperbolic lines connecting the nodes A and B with D (dashed lines) indeed seem to pass through the core nodes.

In general, the **metric curvature** is defined as the *Gaussian curvature* of the geometric space in which this metric can be embedded with optimal embedding accuracy, that is with minimal embedding distortion. We embed the metric in a d-dimensional hyperbolic target space of varying curvature values, and deduct the optimal curvature by comparing their distance error results. If the longer original distances are underestimated, it indicates that the target space curvature is too small (see Fig. 1). We found that the sensitivity to the exact curvature value is mild.

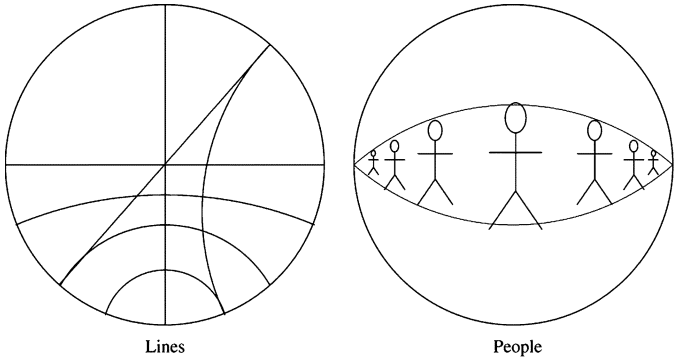

Fig. 2. Poincaré disk model.

TABLE I
GEOMETRIC SPACE AND ALGORITHM COMPARISON. THREE STARS (***) MARK THE METHOD WITH THE BEST PERFORMANCE, FEWER STARS MARK THE METHOD WITH DEGRADED PERFORMANCE, AND A DASH (—) INDICATES THE METHOD IS RULED OUT DUE TO ITS POOR PERFORMANCE OR COMPLEXITY

Space	Euclidean		Hyperbolic		
Algorithm	GNP	AP	TP	AP	LRN
Accuracy					
Average	**	**	**	***	**
Remote	*	*	**	**	*
Neighbors	–	–	–	–	**
Calculation					
Distributive	**	–	**	–	–
Scalable	***	–	***	–	**
Applications					
Server Sel.	**	–	***	–	**
Peer-2-Peer	*	–	**	–	*
Internet Map	–	*	–	*	**

Table I summarize the 5 combinations of embedding space and algorithms. In general, short distances are harder to estimate using all scalable methods, but as we see we are able to achieve a good enough estimation of these, as well. While the performance of all applications depends on the accuracy of the distance estimation, application level multicast is more sensitive to the accuracy of estimating short virtual links (distances) because these links are reused by many of the multicast tree paths. For server selection, the estimation accuracy of long distances, which we want to avoid, is more important.

The rest of this paper is structured as follows: In the next section we develop the hyperbolic space model and show how it is incorporated into the BBS numerical method. In Section III we present embedding results for the weighted and unweighted single instance of the AS topology, using different hyperbolic space curvatures, and compare the resulting distance distortion with GNP. We repeat this experiment in Section IV, for multiple BA generated graphs. Finally, in Section V we evaluate the performance of our hyperbolic embedding for the above mentioned three applications.

II. HYPERBOLIC EMBEDDING MODEL

In this section we discuss the embedding of network distances in hyperbolic spaces. First we review hyperbolic geometry models and the principles of the Poincaré disk model. Next we quote, in Section II-B1, the formulas of arc-length, distance and Gaussian curvature of this model, and demonstrate the curvature on hyperbolic embedding of the simple graph depicted in Fig. 1. Finally, we define in Section II-C1 the embedding potential function using the 'Loid model of hyperbolic space, and derive the field forces induced on BBS particle in Section II-C2.

A. Models of Hyperbolic Spaces

There are five models of hyperbolic spaces [10, Ch. 7]:
- H, the Half-space model;
- I, the Interior of, or Poincaré, disk model;
- J, the Jemisphere model;
- K, the Klein model;
- L, the 'Loid model (short for hyperboloid).

Our embedding solver described in Section II-C1 uses the 'Loid model. However, most of our analysis here utilizes the interior disk model, since it makes the derivation clearer. The distance formula for the 'Loid model, as well as the transformation between the two models, are detailed in the Appendix.

The interior of unit disk $\mathbb{D}^d$ in Euclidean space can be taken as a map of the d-dimensional hyperbolic space. In case $d = 2$ this disk becomes the unit circle depicted in Fig. 2. A hyperbolic line in this model (see Fig. 2 left pan) is any Euclidean circle that is perpendicular to the boundary $\partial\mathbb{D}^d$ of the unit disk. This model is conformally correct, i.e., hyperbolic angles agree with Euclidean angles. A hyperbolic circle maps to a Euclidean circle. Except when their center is at the origin, the two circles are not concentric. Distances in the hyperbolic space are greatly distorted, due to the element of arc length $|ds|$ given by

$$|ds_{\mathbb{D}}| = \frac{2}{1 - ||x||^2}|dx| \tag{1}$$

where $|dx|$ is the Euclidean arc length, and $||\cdot||$ is the Euclidean norm. Indeed, the Euclidean image of a hyperbolic object, Fig. 2 right pan, as it moves away from the origin, shrinks in size roughly in proportion to the Euclidean distance from $\partial\mathbb{D}^d$ (when this distance is small).[1]

Anderson [12] covers in details the upper half-plane model and has a chapter on the Poincaré disk model in case $d = 2$.

B. Analysis of Hyperbolic Space

In order to be able to embed an input metric in a geometric space, e.g., in the Poincaré disk model, we must first calculate the geometric distance determined by the element of arc-length defined for that space.

1) Distance, Metric, and Stretching: Consider the interior disk model with the canonical element of arc length given in (1) for the case of $d = 2$. The hyperbolic distance between $x, y \in \mathbb{D}^2$, denoted $d_{\mathbb{D}}(x, y)$, is given [12], 4.1 by

$$\frac{1}{2}(\cosh[d_{\mathbb{D}}(x, y)] - 1) = \varphi(x, y) \tag{2}$$

where

$$\frac{\varphi(x, y) = |x - y|^2}{(1 - |x|^2)(1 - |y|^2)} \tag{3}$$

With the contracted element of arc length

$$|ds^*_{\mathbb{D}}| = \frac{1}{\sqrt{\kappa}}|ds_{\mathbb{D}}|, \tag{4}$$

[1]This picture and the discussion of Poincaré disk, are taken from [11], 2.1.

the hyperbolic distance is also contracted by $\sqrt{\kappa}$, i.e.,

$$d^*_{\mathbb{D}}(x,y) = \frac{1}{\sqrt{\kappa}} d_{\mathbb{D}}(x,y). \tag{5}$$

Let (Δ_{ij}) denote an input metric, being embedded into a Hyperbolic space with the contracted element of arc length defined by (4). Consider a *stretched* metric, (Δ^*_{ij}), being embedded in hyperbolic space with canonical element of arc length, $ds_{\mathbb{D}}$. The canonical hyperbolic distance approximates the stretched metric, that is

$$d_{\mathbb{D}}(x_i,x_j) \approx \Delta^*_{ij} = \sqrt{\kappa}\Delta_{ij} \quad i,j = 1\ldots n \tag{6}$$

Dividing by $\sqrt{\kappa}$ and substituting (5) we find

$$d^*_{\mathbb{D}}(x_i,x_j) = \frac{d_{\mathbb{D}}(x_i,x_j)}{\sqrt{\kappa}} \approx \Delta_{ij}.$$

Thus embedding of the *stretched* metric, (Δ^*_{ij}), in space with canonical arc length, is equivalent to embedding of the input metric in space with the contracted element of arc length, $ds^*_{\mathbb{D}}$.

2) Hyperbolic Curvature: The Gaussian curvature of a metric induced by an element of arc length $ds_{\mathbb{D}} = g(x)|dx|$ is given by

$$\mathrm{curv}(g) = -\frac{\nabla^2 \log(g)}{g^2}, \tag{7}$$

where $\nabla^2(\,\cdot\,)$ denote the Laplacian

$$\nabla^2(f) \equiv \frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2}$$

For the interior disk model, the element of arc length given in (1) is $g = 2/1 - \|x\|^2$, yielding

$$\begin{aligned}\mathrm{curv}_{\mathbb{D}} &= \frac{1-\|x\|^2}{4}\left[\frac{\partial^2 \log(1-\|x\|^2)}{\partial x_1^2} + \frac{\partial^2 \log(1-\|x\|^2)}{\partial x_2^2}\right] \\ &= \frac{1-\|x\|^2}{2}\left[\frac{\partial}{\partial x_1}\frac{-x_1}{1-\|x\|^2} + \frac{\partial}{\partial x_2}\frac{-x_2}{1-\|x\|^2}\right] \\ &= -1. \end{aligned} \tag{8}$$

Similarly the Gaussian curvature for the contracted element of arc length (4) is given by

$$\mathrm{curv}^*_{\mathbb{D}} = -\frac{1}{(\frac{1}{\sqrt{\kappa}})^2} = -\kappa. \tag{9}$$

Namely, by contracting the element of arc length we can achieve any curvature in the Interior disk model.

3) Embedding Example in $\mathbb{D}^2$ Disk: Examine the eight node graph of Fig. 1 and consider the four exterior nodes, denoted A, B, C, and D. These four nodes measure the internodal distances among themselves. The induced metric is $\Delta_{AB} = \Delta_{BC} = \Delta_{CD} = \Delta_{DA} = 2a+b$ and $\Delta_{AC} = \Delta_{BD} = 2a+2b$. Dividing the two metric values we have

$$\frac{\Delta_{AC}}{\Delta_{AB}} = \frac{2a+2b}{2a+b} = \frac{2r+2}{r+2} \tag{10}$$

where $r = \frac{b}{a}$ is the ratio between the length of the inner (b) and outer (a) edges of the graph. Taking the limit as r approaches 0 or ∞ we have

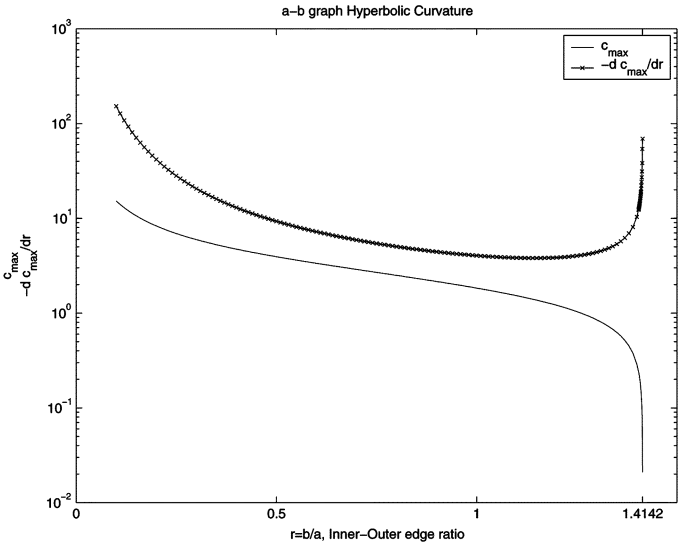

Fig. 3. The Hyperbolic curvature of the graph from Fig. 1.

$$\frac{\Delta_{AC}}{\Delta_{AB}} = \begin{cases} 1 \text{ for } r \to 0 \\ 2 \text{ for } r \to \infty. \end{cases} \tag{11}$$

Embedding of this metric in Euclidean plane must form a A-B-C-D square with diagonal length of $\Delta_{AC} = \sqrt{2}\Delta_{AB}$. Substituting in (10) and extracting, we see that only the ratio $r = \sqrt{2}$ can be exactly embedded in Euclidean plane.

However, in the Hyperbolic disk, the metric curvature $-\kappa$ can be adjusted to achieve an exact embedding of all r values. We normalize the multiplier $\sqrt{\kappa}$ by the maximal metric value, and define

$$c_{\max} \equiv \sqrt{\kappa}\max_{i,j}\Delta_{ij} = \sqrt{\kappa}\Delta_{AC} \tag{12}$$

Due to metric symmetry the four points must be placed on a circle centered at the unit disk origin. We can assume that the points reside on the XY axis at the four points $(\rho,0);(0,\rho);(-\rho,0);(0,-\rho)$. Substituting the stretched distance pairs $\sqrt{\kappa}AB$ and $\sqrt{\kappa}AC$ for $d_{\mathbb{D}}(x,y)$ in (2) we get

$$\cosh\left(c_{\max}\frac{\Delta_{AB}}{\Delta_{AC}}\right) - 1 = 2\frac{2\rho^2}{(1-\rho^2)^2} \tag{13}$$

$$\cosh(c_{\max}) - 1 = 2\frac{(\rho-(-\rho))^2}{(1-\rho^2)(1-(-\rho)^2)} \tag{14}$$

Multiplying (13) by 2, subtracting it from (14), and substituting (10) we obtain

$$\cosh(c_{\max}) - 2\cosh\left(c_{\max}\frac{r+2}{2r+2}\right) + 1 = 0. \tag{15}$$

This implicit function can be solved numerically, and the analytic derivative $dc_{\max}/dr(c_{\max}(r),r)$ can then be calculated. Fig. 3 depicts the resulting normalized curvature and its first derivative for the interval $0 < r \le \sqrt{2}$.

C. Hyperbolic Embedding Solver

Embedding of network distances in geometric space is a mapping between its n nodes to n points in the d-dimensional space,

such that the geometric distances between pairs of points approximates the input network distances metric $(\Delta_{ij})_{i,j=1...n}$.

1) BBS Embedding Method: For calculating this mapping we use the same minimization method we used earlier for Euclidean embedding [4], with adaptations to Hyperbolic space. This method, *Big-Bang Simulation* or BBS, minimizes the energy of a set of particles, traveling in the geometric space under the affect of a force field. Each of the network nodes is represented by a particle. We define the potential energy as the embedding error

$$E_T(v_1, v_2, \ldots, v_n) = \sum\nolimits_{i,j=1\ i>j}^{n} E_{ij}(v_i, v_j). \tag{16}$$

Here $v_k, k = 1, \ldots, n$ are vectors designating the coordinates of the n network nodes in the Hyperbolic space $\mathbb{D}^d$. The pair embedding error E_{ij} is the embedding error of the distance between a pair of particles.

Our embedding solver uses the 'Loid model of hyperbolic space, which averts the distance singularity on the boundary of the Poincaré disk. As in [4] we divide the embedding into four calculation phases. The phase pair embedding error function denoted by $E_{ij}^{\langle p \rangle}$, assumes the form

$$\begin{aligned} &E_{ij}^{\langle p \rangle}(v_i, v_j) = \mathcal{F}(d_S(v_i, v_j), \Delta_{ij}), \\ &\text{for } i \neq j \text{ and } v_i \neq v_j \end{aligned} \tag{17}$$

where $d_S(\mathbf{x}, \mathbf{y})$ is the Hyperbolic distance in S^d, the upper sheet of hyperboloid

$$S^d = \left\{ \xi : \xi_1^2 + \cdots + \xi_d^2 - \xi_{d+1}^2 = -1 \right\} \tag{18}$$

$$\xi = \left(\xi_1, \ldots \xi_d, \sqrt{1 + \sum_{i=1}^{d} \xi_i^2} \right). \tag{19}$$

For simplicity, we denote $\xi_{d+1} = \sqrt{1 + \sum_{i=1}^{d} \xi_i^2}$. At the end of each phase, the particles reach a least energy configuration. Finally, at the end of the last phase, each network node is mapped to the coordinates of the corresponding particle in the final low energy configuration.

2) Potential Field Force: The particle movement equations and their initial conditions were derived in [4], sec. 2 that discusses friction force and other implementation details. The potential force field in Hyperbolic space is different from the Euclidean space, since the two distance expressions differ. We thus redo here the calculation of potential force field for Hyperbolic case.

The field force $\vec{F}_{i_0}$ that is derived from the potential energy (16), is given by

$$\vec{F}_{i_0} = -\nabla_{v_{i_0}} E_T(v_1, \ldots, v_n) \tag{20}$$

$$= -\sum\nolimits_{j=1\ j\neq i_0}^{n} \mathcal{F}_x(x, \Delta_{i_0 j})|_{x=d_{i_0 j}^S} \nabla_{v_{i_0}} d_{i_0 j}^S, \tag{21}$$

where $d_{ij}^S \equiv d_S(v_i, v_j)$ denotes the pair hyperbolic distance between $\xi = v_i$ and $\psi = v_j$, and its gradient with respect to ξ is given by

$$\nabla_{v_i} d_{ij}^S \equiv \left(\frac{\partial}{\partial \xi_k} d_{ij}^S \right)$$

$$= \left(\xi_k \frac{\psi_{d+1}}{\xi_{d+1}} - \psi_k \right) \div \sin h d_{ij}^S. \tag{22}$$

III. Hyperbolic Embedding in Real Topologies

In Section III-B we use the Internet router topology extracted from Tel-Aviv University DIMES database [13] dated October 23–24, 2005. In Sections III-C and III-D we use the AS topology instance from the University of Oregon RouteViews database dated March 31, 2001.

A. Experiment Details and Legend

We use two measures to compare the accuracy

- **Symmetric Pair Distortion** Defined for each node pair as the maximum of the ratio of the measured to the geometric distance, and its inverse.
- **Directional relative error** Defined by [1], (4) as the ratio of the difference between the geometric and measured distances, to the minimum of the two distances.

The Symmetric pair distortion can be calculated by adding 1 to the absolute value of the directional relative error [4].

We experiment with different curvatures of the target hyperbolic space. In Section II-B2 we showed that embedding a given metric in hyperbolic space with curvature κ is equivalent to embedding the $\sqrt{\kappa} -$ stretched metric in canonical hyperbolic space. Before stretching we first divide the distances of each metric by

$$\begin{aligned} \eta_{\mathrm{AP}} &= \frac{1}{n^2} \sum\nolimits_{i,j=1}^{n} \Delta_{ij} \\ \eta_{\mathrm{TP}''+''} &= \frac{1}{t^2} \sum_{i,j \mathrm{Tracers}} \Delta_{ij} \\ \eta_{\mathrm{TP}''-''} &= \max_{i,j \mathrm{Tracers}} \Delta_{ij} \\ \eta_{\mathrm{LRN}} &= \max_{(i,j)\in E \cup R} \Delta_{ij}. \end{aligned} \tag{23}$$

The following legend notations were used in all the figures: "GNP" stands for DHS (which was used by GNP) and "HYP,#" stands for hyperbolic BBS with normalized stretch "#". Positive stretch stands for dividing by $\eta_{\mathrm{AP}}, \mathrm{and} \eta_{\mathrm{TP}''+''}$ for AP and TP, respectively, whereas negative stretch, that is legend "HYP,-#", stands for dividing by $\eta_{\mathrm{TP}''-''}$, and η_{LRN} for TP and LRN, respectively.

In each experiment we select a group of the router or AS nodes, called an overlay, and embed the *shortest-path* distances among these overlay nodes. The overlay from the routers topology in Section III-B consists of the 190 traced **access routers** of the DIMES agents used to measure the router topology instance. We combined the traceroute information For this relatively small overlay we evaluate the relative distance error of all pairs of overlay members. In Sections III-C and III-D, two overlays of 2000 low degree AS nodes are randomly selected from the AS topology. We select 400 of the overlay ASs, and evaluate for each one of them the relative error of all distance pairs from it to all other 1999 overlay members.

B. Embedding Measured Internet Distances

To generate the router level topology, we selected about 900 DIMES agents and performed UDP traceroute to each other.

After combining agents with the same first hop address (agents in the same LAN), and removing nodes that were either unreachable or could not perform UDP traceroute, we were left with 540 nodes. Due to the way the experiment was executed, we did not receive all the possible distances between node agent pairs. Thus, we removed all nodes that did not measure to enough other nodes and were left with 460 nodes. Among them we had almost 35 000 measured paths (we used the shortest of the two unidirectional paths between two nodes), which constitute 33% of the node pairs.

The simulated Tracers were selected randomly among the 31 nodes that could see over 90% of the other nodes, and no two Tracers from the same AS were selected. We end up with six nodes in the USA, one in Denmark, three in Great Britain, one in Norway, two in Israel, and two in Australia. Only two of the Tracers were inside universities, at UTDallas and TAU. 350 nodes were seen by all these Tracers, and the rest were seen by at least 10 of them.

We compare our TP hyperbolic embedding results with Euclidean Down-Hill-Simplex (DHS) embedding, the method used in Global-Network-Positioning (GNP) [1]. For this comparison we did not to compare hyperbolic BBS with Euclidean BBS, since for a small number or Tracers, $t \leq 15$, Euclidean BBS and GNP, yields similar intra-Tracers distortion [4], and are thus comparable. For TP embedding we selected the same $t = 15$ Tracers used in Section V-A, and embed all 15 node-Tracer distances.

The centralized step of the TP embedding, that is the embedding of the Tracers matrix holding $t(t-1) = 210$ Tracer-Tracer pairs (Fig. 4), is very efficient compared to other embedding algorithms. For instance the embedding in $d = 7$-dimensional space, took 0.5 and 1.75 CPU seconds for $(5\eta^{-1}_{\mathrm{TP}''+''})$-stretched and $(12\eta^{-1}_{\mathrm{TP}''+''})$-stretched, respectively, on a PIV-1.5 GHz. Euclidean DHS embedding of the same matrix took 0.33 seconds. Indeed with larger curvature HYP is considerably slower than GNP. Nevertheless, HYP's *absolute CPU time* is negligible, considering that the centralized step is performed infrequently.

Fig. 4 compares Euclidean and Hyperbolic embedding, for different embedding dimensions $d = 3 \ldots 10$. Fig. 4(c) depicts the 5, 25, 50, 75, and 95 percentile lines of the directional relative error. In Figs. 4(a) and (b) depict the accuracy statistics of embedding of the measured hop distance and delay, respectively. The mean and median relative error and its standard deviation are depicted on the left, center and, in reverse y-axis, on the right side respectively.

Fig. 4(a) indicates that GNP has slightly better median and average error but its variance is significantly worse. This fact can be clearly seen even for $d = 7$, where it performs best, in Fig. 4(c). Fig. 4(b) shows that there is no clear advantage for either method. The results above are depicted in more details in Section V-A. The surprising finding regarding delay, may be attributed to the strong correlation of Internet delay with geographic distance, while BGP policy routing is more dominant regarding hop distance.

C. Two-Phase Hop Distance Embedding

The distribution of the directional relative error, estimating hop distance in the AS topology, is depicted in Fig. 5. The results of five-dimensional TP embedding, with HYP and GNP, are depicted on the top part. We select the $t = 15$ Tracers randomly among overlay members and embed all 15 node-Tracer distances.

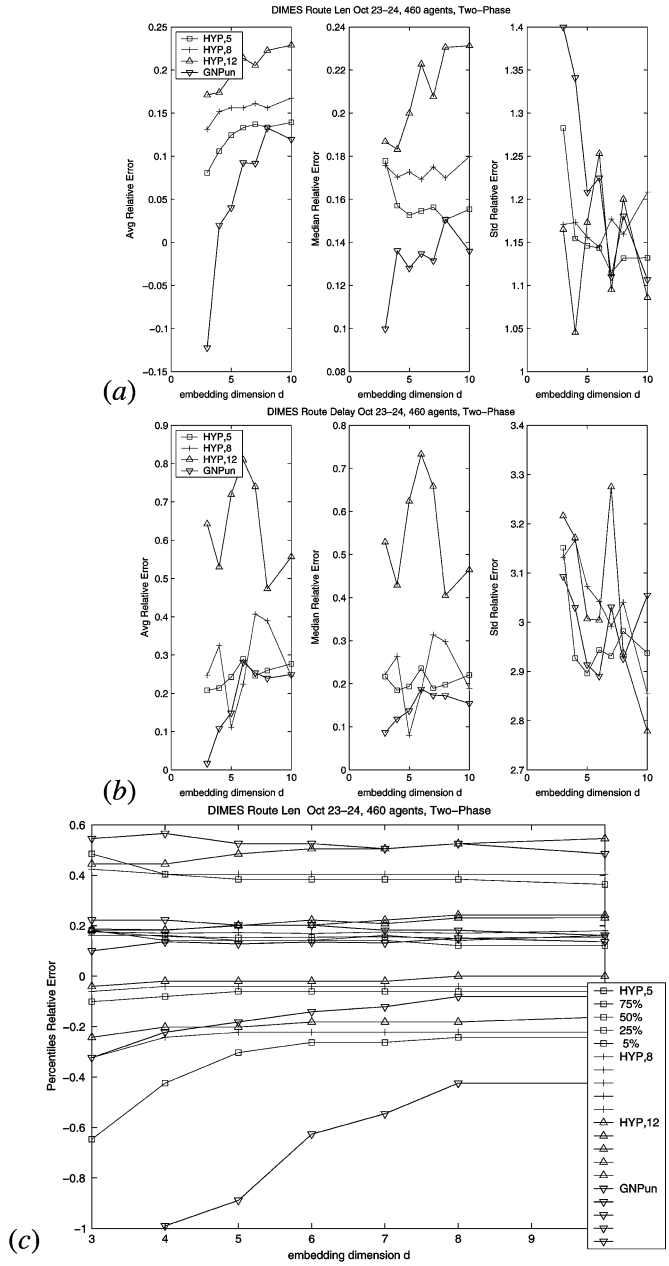

Fig. 4. DIMES Router Relative Error versus Embedding Dimension and Curvature: (a) & (c) hop count, (b) delay.

The frequency of pair distances is depicted by the thick black Δ curve. We group the directional relative errors for the same network hop distance pairs. The vertical lines correspond to integral hop distance, in the unweighted AS graph. The method marker is placed at the *average* directional relative error, and the star marker depicts the median. Each method line has whiskers at the 5, 25, 75, and 95 percentiles.

As we reported in [4], GNP underestimates longer hop distances, having negative relative errors. The $(10\eta^{-1}_{\mathrm{TP}''-''})$-stretched metric has the best relative hop error. For 4–6 hops distances the 5 to 95 HYP relative error percentiles

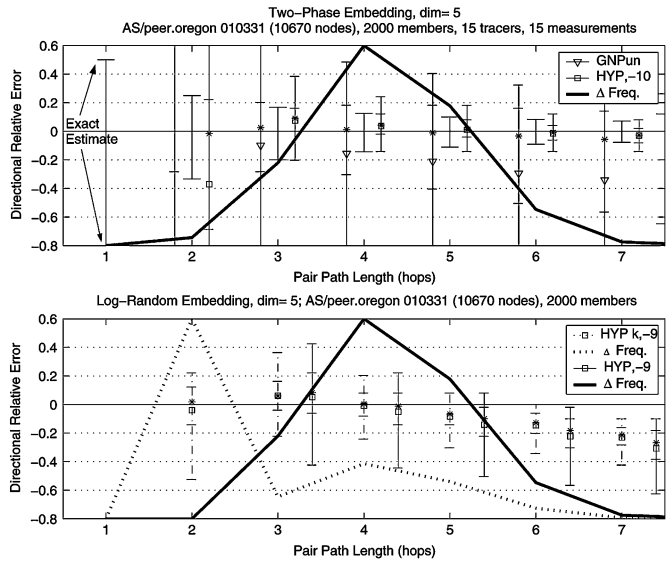

Fig. 5. AS Graph Relative Hops Error.

are inside $(-0.15 \ldots 0.25)$, compared to $(-1.5 \ldots 0.5)$ for GNP. In the two-hops distances the corresponding percentiles are $(-2.5 \ldots 0.8)$ for HYP, compared to $(-14 \ldots 1)$ for GNP.

The interval of relative error for which the rounded distance $\lfloor d_S(v_i, v_j) + .5 \rfloor$ is an *exact estimate* of the hop distance Δ_{ij}, is depicted in Fig. 5 between the GNP and TP percentile lines. For 3–7 hops distances the hop distance estimate of HYP coordinates is exact in 70 of the pairs!

D. LRN Hop Distance Embedding

The results of five-dimensional LRN embedding are depicted on the bottom of Fig. 5. The dotted black Δ curve depicts the frequency of the $O(n \log n)$ pairs embedded by LRN, whereas the solid black curve depicts the frequency of the other pairs. Each curve is scaled by its maximum frequency to enable plotting both curves together. The input to LRN embedding is identical to the one in Section III-C(two overlays of the AS topology, each containing $n = 2000$ nodes). As indicated by the two Δ curves, the group members are not directly connected to each other, as expected for stub ASs. Because the neighboring pairs threshold is a hard limiter, all two-hops pairs are embedded. There are approximately $25n$ two-hops pairs, while there are less than half random pairs $|R| = 11n$. As in the case of AP embedding, only BBS could handle 2000 members group and concurrently embed over 70 K distance pairs. The CPU time, running on PIV-1.5 GHz, was up to 300 seconds.

The dashed vertical lines, with "HYP k" legend, depict the directional relative error percentiles of embedded pairs, whereas the solid vertical lines depict percentiles of other pairs.

As expected, for short distance pairs, LRN is far more accurate than TP (and GNP). For two-hops distances the hop distance estimate of LRN is exact in over 75% of the pairs, and just 5% of all pairs are overestimated. This higher accuracy of $25n$ two-hops pairs, is gained by compromising accuracy for the rest of the $\Theta(n^2)$ pairs which are three-hops or more apart. Indeed, among four to six hops distances, the 5 to 95 HYP relative error percentiles are inside $(-0.5 \ldots 0.2)$, which is approximately double than the $5 \ldots 95$ gap of TP HYP with similar curvature.

IV. Hyperbolic Embedding of Generated Power-Law Graphs

We evaluate the hyperbolic embedding of five 1000 node Barabási–Albert (BA) topologies [9]. The n overlay nodes are selected at random from the group of low degree nodes. For $n = 20, 50$ we evaluate the relative embedding error of all $\frac{1}{2}n(n-1)$ pairs. For $n = 100 \ldots 800$, we select $\frac{3}{4}$ members, and evaluate for each one of them the relative error of all distance pairs from it to all other $(n-1)$ members.

To increase the confidence each experiment was conducted using 3 sets of random weights per generated topology. The weights drawn here, are i.i.d. random variables, distributed uniformly in the interval [1,1000]. From each random weights graph we embedded two random overlays as explained above. Namely each point in the comparison graph results from 30 embedding experiments, 6 per a generated BA topology.

Fig. 6(a) compares TP HYP and GNP with different embedding dimensions. We select the $t = 15$ Tracers randomly among overlay members and embed all 15 node-Tracer distances. The mean standard deviation of the relative estimation error are depicted on the left and, in reverse y-axis, on the right side, respectively. The accuracy of our method is far better than GNP for all HYP stretch values. The figure also demonstrate the insensitivity to the curvature: for $d \geq 4$, there is little difference between stretch values 6 to 9. Fig. 6(b) compares TP HYP and GNP, with embedding dimension $d = 7$, for different overlap sizes ranging from 20 to 800 members.

The standard deviation with optimal HYP stretch is minimal and the mean error is closest to 0. The common optimal stretch for all overlay sizes and larger embedding dimensions $d \geq 4$ is $6\eta^{-1}_{\mathrm{TP}''+}$, moreover, all values in the range $3\eta^{-1}_{\mathrm{TP}''+} - 9\eta^{-1}_{\mathrm{TP}''+}$ gives superb results.

V. Applications

In this section, we evaluate Two-Phase embedding with varying hyperbolic curvature, in three applications: delay estimation, server selection, and application level multicast. We present only TP results, since the TP embedding can be computed distributively, while LRN requires central calculation.

A. Delay Estimation

In this application we are interested in estimating the delay between a single source node and *all* other nodes of the graph. This can be used by a VoIP exchange that can connect its clients either through its (almost) free Internet connection, or if the delay is too long through the POTS system.

The distribution of the directional relative estimation error, for the DIMES Router topology of Section III-B, are depicted in Fig. 7. The measured Route Delay and Path Length results are depicted in Fig. 7(a) and (b), respectively. The delay and path length data were measured during two days, October 23–24, 2005, among 460 DIMES agents. The results of seven-dimensional TP hyperbolic embedding are compared with GNP. All

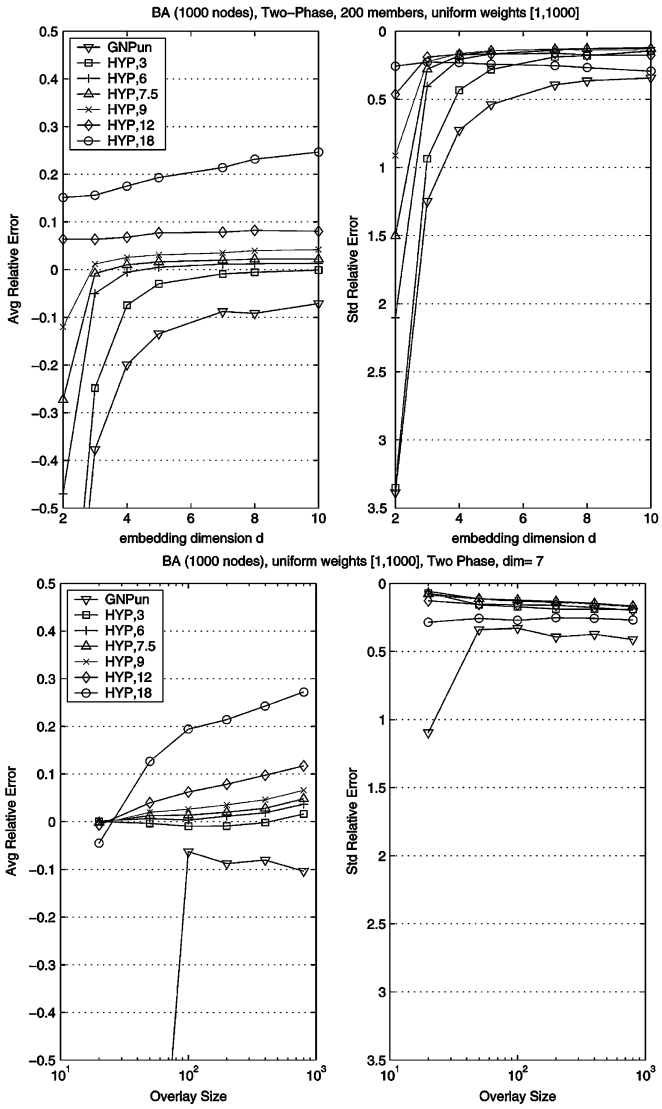

Fig. 6. Generated BA graphs with random weights. Relative error versus group size, embedding dimension and curvature.

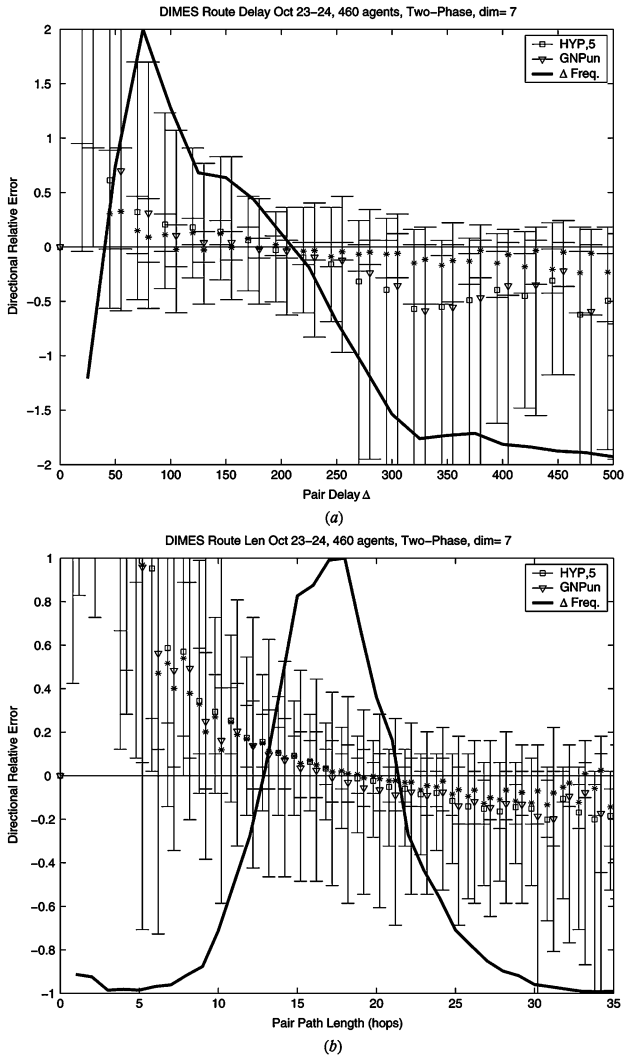

Fig. 7. DIMES router map relative error: (a) delay and (b) minimum hop.

HYP and GNP embedding depicted in Fig. 7 use $t = 15$ Tracers and 15 host-Tracer measurements.

In order to capture the distribution of the estimation error, we group the pair distances in 25 ms wide bins. Surprisingly, for this experiment the difference between HYP and GNP was negligible (see Fig. 7(a)). However, the picture is different for the hop distance as depicted in Fig. 7(b). While the average directional relative error is similar for both methods, GNP has a noticeable larger spread almost for the entire hop distance range.

B. Server Selection

This experiment used two Internet AS data sets, the University of Oregon RouteViews dataset and the combined RouteView with looking glass and router registry, as described in [14]. The nine couples of peering data sets were collected weekly starting March 2001. To increase the confidence each experiment was conducted using 3 sets of random weights per each of the peering topologies.

Following [15] we randomly selected 10 mirror servers and estimated the closet mirror to each of the rest of the graph nodes acting as clients. A client's decision is considered correct if it selects the mirror whose client-mirror distance is at most $1+\alpha$ times the optimal distance. We used $\alpha = 0.5$. For each mirror group, rank accuracy is defined as the percentage of correct client decisions.

Fig. 8 depicts the average cumulative distribution function (CDF) of rank accuracy for the IDMaps, GNP, and TP HYP methods. For all three methods, we used in this experiment $t = 15$ *low* degree Tracers and 8 measurements per node. Each mark is the average of the CDFs from the $9 \times 3 = 27$ simulated graphs, where each CDF consists of 300 mirror group experiments performed on a single graph. Marks denoted with the postfix $08lc$ depict the effect of the "leaf correction" procedure. In leaf correction, the distances from a degree-1 node are estimated by the geometric distance to the adjacent node, plus the distance between the degree-1 node and the corresponding neighbor.

Figs. 8(a) and (b) depicts the results for the RouteViews dataset, and for the combined dataset, respectively. IDMaps ranking performance are nearly perfect, achieving at least 98.5 correct answers in 99% of the mirror experiments. GNP however, has the worst ranking accuracy, due to underestimating of all the distances, and is thus ruled out as a practical method for

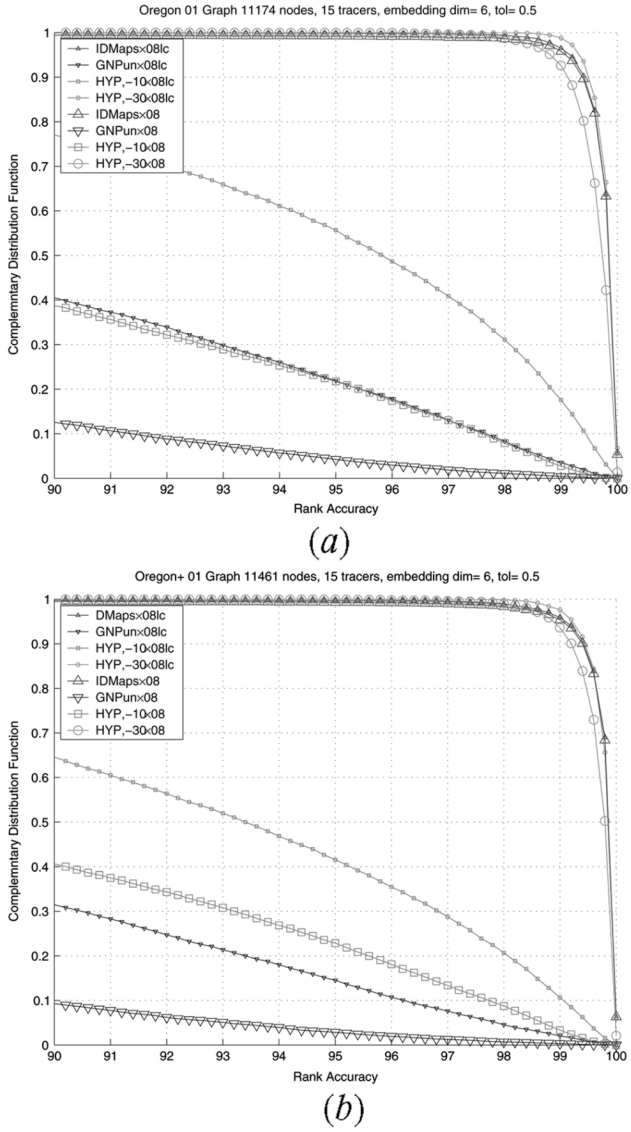

Fig. 8. RouteViews AS graph with random weights: mirror selection for various curvatures.

server selection with accuracy $\alpha = 0.5$. Our method ranking accuracy improves with increasing the embedding curvature, and is comparable with IDMaps performance for the stretch of $30\eta^{-1}_{\mathrm{TP}''\_''}$. For the RouteViews dataset, our performance even slightly supersedes IDMaps, achieving, at least, 99 correct answers in 99 of the mirror group experiments.

C. Application Level Multicast

In application level multicast [16], [17], we wish to build a multicast tree without network support. To make the tree efficient, we need to know the distances among the multicast group nodes. Otherwise, one may build a tree where the delay to some nodes is a large multiple of the unicast delay.

The first scalable approach for building application layer multicast trees was CAN [18], [19] and its derivatives [20], [21]. Due to the high accuracy of our embedding we are presenting smaller stretch factors for distances, i.e., the delays on our trees are shorter, while maintaining good stress factor distribution, namely, most of our tree links are not congested.

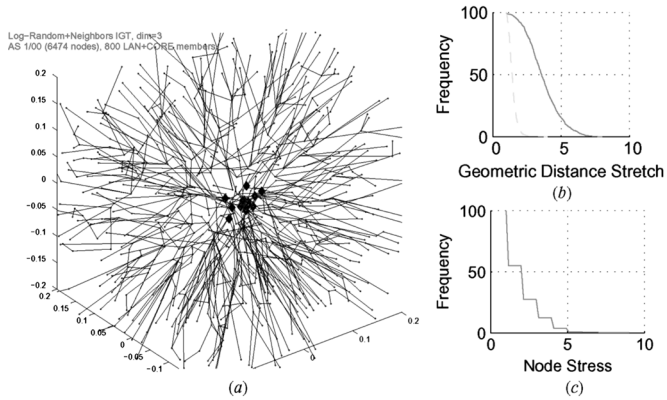

Fig. 9. RouteViews AS graph with random weights: 3-D multicast tree layout.

An alternative tree-first approach is NICE application multicast [22], which creates a hierarchy of clusters while selecting the same or adjacent cluster for all nodes that are "close by". CAN and NICE both have low, and thus scalable, link stress and control overhead. However, NICE incurs higher control load on the root node and its direct descendants.

We are given a multicast group, M, which is a subset of the graph nodes, a source node, s; and we construct $T(V_T, A_T)$, a multicast tree from s to all the nodes in M. *Latency Stretch* is defined per member $w \in M, w \neq s$ as the ratio of the path length along the tree T from s to w to the length of the direct unicast path. *Link Stress* is defined per link of the underlying topology and counts the number of identical packets sent between members of M over that link. This definition of stress, following [23], is from the network's perspective, rather than the application's.

Fig. 9(a) depicts the three-dimensional multicast tree constructed by our algorithm. Fig. 9(b) and (c) depict the hyperbolic distance stretch and node stress of that tree. The constructed tree follows the structure of the underlying "physical" network, and thus reduces the latency stretch. The tree roots are the 20 largest degree nodes, depicted by the ⧫ marker in the vicinity of the hyperbolic origin. From these roots the tree descends to medium degree nodes and then reaches the leaves that are the lowest degree nodes.

The **JoinNode**(i) procedure listed in Fig. 10 is run by nodes that join the multicast tree and by tree nodes with large latency stretch that wish to reduce this stretch. The procedure descends down the tree in BFS order searching for candidate parents for the given node i. In the first BFS iteration it scans the orphans group C_0. If no candidate parent is found in this iteration than i is added to the orphans group C_0.

Otherwise, each of the following iterations (lines 2-8) checks the children of the candidate parent nodes found in the previous BFS iteration. The number of candidate parents found in each iteration is bounded (line 5). Only the C_{Th} candidate parents that are the nearest to i are scanned in the next BFS iteration. After the last BFS iteration that does not find any more candidates, i assigns the nearest available candidate found in all iterations as its parent. A parent is considered available if the number of its children is smaller than $\mathrm{DEG}_{\mathrm{Max}}$ (line 8).

The **QueryBranch**(**Broot**,i) function listed in Fig. 11 checks the children of a given node Broot and returns the ones which

Function $newParent =$ **JoinNode**(i)

$Y \leftarrow \{0\},\ N \leftarrow \emptyset$
1. **while** $Y \neq \emptyset$
2. $X \leftarrow Y,\ Y \leftarrow \emptyset$
3. **for** $x \in X$ **do**
4. $Y \leftarrow Y \cup$ QueryBranch(x, i)
5. **if** $|Y| > C_{Th}$ **then**
6. $Y \leftarrow \{j_{1\ldots C_{Th}} \mid d(j_k, i) \leq \ldots d(j_{k+1}, i)\}$
7. $N \leftarrow N \cup Y$
8. $N \leftarrow N \cap \{l \leq m \mid |C_l| < DEG_{Max}\}$
9. **if** $N = \emptyset$ **then** $x_0 \leftarrow 0$
10. **else** $x_0 \leftarrow x;\ d(x, i) = \min_{l \in N} d(l, i)$
11. $C_{x_0} \leftarrow C_{x_0} \cup \{i\}$
12. $newParent \leftarrow x_0$

Fig. 10. The JoinNode function.

Function $nextParents =$ **QueryBranch**$(Broot, i)$

$nextParents \leftarrow \emptyset$
1. **for** $x \in C_{Broot}$ **do**
2. **if** $d(x, \mathbf{0}) < d(i, \mathbf{0})$ **and** $\frac{d(x,i)}{d(i,\mathbf{0})} < D_{Th}$ **then**
3. $nextParents \leftarrow nextParents \cup \{x\}$
4. **if** $d(x, \mathbf{0}) > d(i, \mathbf{0})$ **and** $\frac{d(x,i)}{d(x,\mathbf{0})} < D_{Th}$ **then**
≪ Replace current x's parent with i? ≫
5. **if** $P_x = 0$ **or** $d(x, i) < d(x, P_x)$ **then**
6. $C_{P_x} \leftarrow C_{P_x} \setminus \{x\},\ P_x \leftarrow i,\ C_i \leftarrow C_i \cup \{x\}$

Fig. 11. The QueryBranch function.

are candidate parents of i. For a node x to be a candidate parent of i, it must satisfy the two conditions in line 2

- x's $\mathbf{0}$-distance is less than i's $\mathbf{0}$-distance
- x's i-distance is small relative to i's $\mathbf{0}$-distance

Alternatively, if i and x satisfies the above two conditions with switched roles (line 4), then if $x \in C_0$ or the x's distance from its current parent ($P_x = \text{Broot}$) is less than its distance from i, then x assigns i as its parent (line 6).

The tree origin distance of an orphan node i is given by $\text{TrOrigDist}(i) = d(i, \mathbf{0}) \forall i \in C_0$, i.e., the hyperbolic distance to the origin. For the rest of the tree nodes $\text{TrOrigDist}(i) = \text{TrOrigDist}(P_i) + d(i, P_i)$, that is the distance along the tree path to their root ancestor plus ancestor's origin distance.

Our **Iterative-Geometric-Tree (IGT)** (see Fig. 12) assumes no topology or routing information from the underlying "physical" network, and uses only the hyperbolic coordinates, assigned by the embedding of each node. For clarity, the algorithm is presented as central. However, the algorithm can be easily distributed since all data structures used, except the orphans groups C_0, can be managed locally by each of the nodes.

Following are the parameter values for the IGT algorithm:

Name	[Fig#, line]	Value	Stretch $\times\ \eta_{\text{TP}''-''}$
C_{Th}	[10, 5]	2	
DEG_{Max}	[10, 8]	15	
D_{Th}	[11, 2 − 4]	.8	≤ 12
D_{Th}		1.0	≤ 18
D_{Th}		1.3	≤ 24
D_{Th}		1.5	> 24
$\text{Rewire}_{\text{Th}}$	[12, 7]	1.3	
UnRew_{Th}	[12, 7]	1.01	
AbortRewire	[12, 14]	.07	
RewireSweeps	[12, 3]	10	

Algorithm IGT $(V = \{v_1 \ldots v_m\})$

$C_0 \leftarrow \{1\}\ ;\ C_i \leftarrow \emptyset,\ P_i \leftarrow 0,\ R_i \leftarrow 0;\ i = 1 \ldots m$
≪ Join ≫
1. **for** $i = 2 \ldots m$ **do**
2. $P_i \leftarrow$ JoinNode(i)
≪ Rewire ≫
3. **for** $iSweep = 1 \ldots RewireSweeps$ **do**
4. $N_R \leftarrow 0;$
5. **for** $i \in \{1 \ldots m \mid R_i \leq 1\}$ **do**
6. $D_{\mathbf{0}} \leftarrow$ TrOrigDist$(i),\ R_i \leftarrow R_i + 1$
7. **if** $D_{\mathbf{0}} > d(i, \mathbf{0}) * Rewire_{Th}$ **then**
≪ Rewire ≫
8. $C_{P_i} \leftarrow C_{P_i} \setminus \{i\}$
9. $P\prime \leftarrow$ JoinNode(i)
10. **if** $P\prime \neq P_i$ **then**
11. **if** $D_{\mathbf{0}} \leq$ TrOrigDist$(i) * UnRew_{Th}$ **then**
≪ Undo Rewire ≫
12. $C_{P\prime} \leftarrow C_{P\prime} \setminus \{i\},\ C_{P_i} \leftarrow C_{P_i} \cup \{i\}$
13. **else** $P_i \leftarrow P\prime,\ R_i \leftarrow 0,\ N_R \leftarrow N_R + 1$
14. **if** $N_R < AbortRewire * m$ **then** break

Fig. 12. The Iterative Geometric Tree algorithm.

Initially, all nodes execute **JoinNode** once, and are assigned either to an existing parent or to the orphans group C_0. The constructed tree is then improved by several rewire sweeps. All recently rewired nodes, having $R_i \leq 1$, participate in the next rewire sweep. Each such node i compares (line 7, Fig. 12) its tree origin distance with its hyperbolic distance to the origin. If the tree distance is significantly larger than the hyperbolic distance the node executes the **JoinNode** function again. If the new parent reduces the tree distance significantly, then the node *rewires* to the new parent. The algorithm ends after $RewireSweeps$ sweeps or if the ratio of rewired nodes to $|M|$ is less then $AbortRewire$.

To evaluate our algorithm we performed the following experiment. Select members of M randomly among low-degree nodes of the graph. Use TP embedding with $t = 15$ Tracers and 15 measurements per node to embed these nodes in five-dimensional Hyperbolic space, and run our IGT algorithm from 40 different source nodes. We also use TP embedding in five-dimensional Euclidean space, and run the Geometric-Multicast-Tree algorithm of [24], Fig. 10 from these source nodes.

Fig. 13(a) depicts the calculation results on the AS topology instance from the University of Oregon RouteViews database dated January 2, 2000, for a group of $|M| = 800$ members. We use the same legend as above, which is "HYP,#" for hyperbolic BBS with normalized stretch "#". The legend "GNP" marks the Geometric Multicast Tree algorithm which run using GNP coordinates.

The complementary distribution function, depicted on the left hand side, was aggregated from latency stretches of all the nodes $w \in M$, from each of the 40 source nodes. The average stress frequency, depicted on the right hand side, is the total number of links having a given stress value, averaged over the 40 source node trees.

The multicast tree in application level multicast is constructed from shorter pairs, among which LRN estimation is more accurate. Nevertheless, TP performance is better since the two-hop pairs consist $O(n)$ of the distance pairs, which are negligible among $\Theta(n^2)$ pairs for which TP estimation is better.

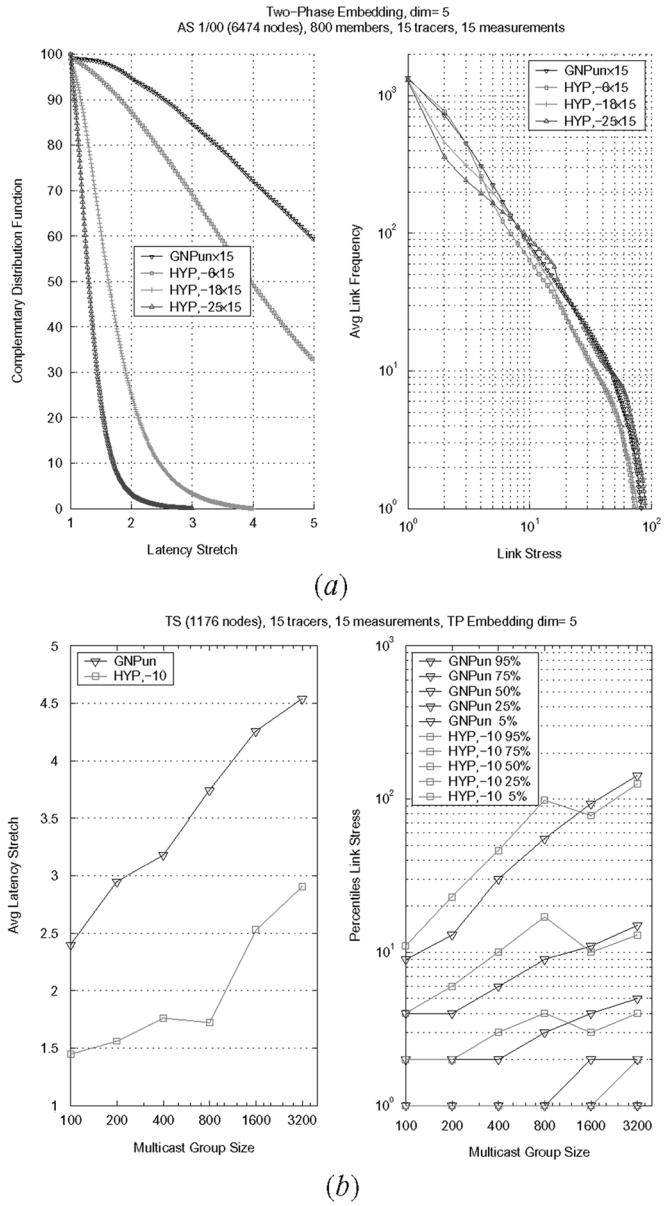

Fig. 13. Latency Stretch versus Stress and multicast group size: (a) Routeviews AS graph with random weights, (b) generated GA-Tech graph.

Fig. 13(a) shows a clear trade-off between stretch and stress. An increase in the HYP curvature yields smaller (better) stretch and larger (worse) stress. The stress of GNP is similar to "HYP,-25" stress, with normalized stretch $25\eta^{-1}_{\mathrm{TP}''\_''}$. However, the 95 percentile latency stretch of HYP with this curvature is 1.95, compared to 11.2 of GNP (omitted from plotted range).

For comparison with [19], [23] we performed the above experiment also for a Transit-Stub topology of 10000 nodes. The effect of multicast tree size on latency stretch is depicted in Fig. 13(b). With normalized stretch $10\eta^{-1}_{\mathrm{TP}''\_''}$ ($\square$ marker) our average latency stretch depicted on the left graph, is comparable with topology aware CAN [19], Fig 9. Note that results for CAN assume global and perfect knowledge of the topology.

VI. Concluding Remarks

Given the fact that Internet distances tend towards the core, we suggested to embed the Internet graph in hyperbolic space. We showed that this idea works well for generated power-law graphs (Fig. 6), and for calculated minimum hop distances of the RouteViews AS topology. Indeed, all three studied applications distance estimation, mirror selection, and application layer multicast (Figs. 7, 8, and 13, respectively) benefited from our approach.

However, for distances measured between endpoints in the Internet the picture is surprisingly different (Figs. 4 and 7). For the hop distance estimation, hyperbolic embedding is still better than Euclidean. Interestingly, for delay estimation there is no clear advantage for using the hyperbolic space. This puzzling point is the subject of our future research.

Appendix

The following formulas for hyperbolic distance in 'Loid model, are derived in [11], 2.6. This 'Loid space is embedded in the upper sheet of the hyperboloid (18), (19).

$$S^n = \left\{\xi : \xi_1^2 + \cdots + \xi_n^2 - \xi_{n+1}^2 = -1\right\}.$$

The Hyperbolic distance in 'Loid model $d_S(\xi,\psi)$ is given by

$$\cosh d_S(\xi,\psi) = -\xi\cdot\psi; \tag{24}$$

where $\cdot$ denotes the Minkowski inner product, defined as

$$\xi\cdot\psi = \sum_{i=1}^{d}\xi_i\psi_i - \xi_{d+1}\psi_{d+1}. \tag{25}$$

Here ξ_{d+1} and ψ_{d+1} are defined by (19). The hyperboloid S^n is isometrically transformed to the Poincaré disk $\mathbb{D}^n = \{\mathbf{x} : x_1^2 + x_y^2 + \ldots x_n^2 < 1\}$, via stereographic projection through the point $(0,0,\ldots 0,-1) \in \mathbb{R}^{n+1}$. The equations of this transformation are

$$\begin{aligned}
&\mathbf{x}\in\mathbb{D}^n \longrightarrow \left(\frac{2x_1}{1-|\mathbf{x}|^2},\ldots\frac{2x_n}{1-|\mathbf{x}|^2},\frac{1+|\mathbf{x}|^2}{1-|\mathbf{x}|^2}\right)\\
&\text{where,}\quad |\mathbf{x}|<1;\\
&\xi\in S^n \longrightarrow \left(\frac{\xi_1}{1+\sqrt{1+|\xi|^2}},\ldots\frac{\xi_n}{1+\sqrt{1+|\xi|^2}}\right)\\
&\text{where,}\quad |\xi|^2 = \sum_{i=1}^{n}\xi_i^2
\end{aligned} \tag{26}$$

In Poincaré Disk, $n=2$, the hyperbolic distance between each pair $x,y\in\mathbb{D}^2$, is given in [12], 4.1 by,

$$\cosh[d_{\mathbb{D}}(x,y)] = 2\varphi(x,y)+1, \tag{27}$$

(2), where φ is given by (3). Applying the Euclidean cosine law in the triangle $\triangle x0y$ and substituting $\cos\angle(x0y)$ with the normalized inner product yields

$$\varphi(x,y) = \frac{|x|^2+|y|^2-2x\cdot y}{(1-|x|^2)(1-|y|^2)}$$

$$\cosh d_{\mathbb{D}}(x,y) = \frac{-4x\cdot y + (1+|x|^2)(1+|y|^2)}{(1-|x|^2)(1-|y|^2)} = -\frac{2x}{1-|x|^2}\cdot\frac{2y}{1-|y|^2} + \frac{1+|x|^2}{1-|x|^2}\frac{1+|y|^2}{1-|y|^2}.$$

Substituting the images $\xi,\psi \in S^2$ of the projected $x,y \in \mathbb{D}^2$, respectively, as given in (26) we have

$$\cosh d_{\mathbb{D}}(x,y) = \xi_{n+1}\psi_{n+1} - \sum_{i=1}^{n}\xi_i\psi_i = -\xi\cdot\psi = \cosh d_S(\xi,\psi), \tag{28}$$

according to (24).

REFERENCES

[1] T. S. E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," *Proc. IEEE* , vol. 1, pp. 170–179, Jun. 2002.

[2] H. Lim, J. C. Hou, and C. H. Choi, "Provisioning of network distance service on the Internet," *IEEE/ACM Trans. Netw.*, vol. 13, no. 3, 2005.

[3] Tang and M. Crovella, "Virtual landmarks for the Internet," *ACM Internet Measurement*, Nov. 2003.

[4] Y. Shavitt and T. Tankel, "Big-Bang simulation for embedding network distances in Euclidean space," *IEEE/ACM Trans. Netw.*, pp. 993–1006, Dec. 2004, an earlier version appeared in IEEE INFOCOM 2003.

[5] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz, "Characterizing the Internet hierarchy from multiple vantage points," in *Proc. IEEE INFOCOM*, 2002, pp. 618–627.

[6] L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos, "A simple conceptual model for the Internet topology," *Global Internet*, Nov. 2001.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *ACM SIGCOMM 1999*, Boston, MA, Aug./Sep. 1999.

[8] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, Oct. 15, 1999.

[9] R. Albert and A.-L. Barabási, "Topology of evolving networks: Local events and universality," *Phys. Rev. Lett.*, pp. 5234–5237, Dec. 2000.

[10] J. W. Cannon, W. J. Floyd, R. Kenyon, and W. R. Parry, , S. Levy, Ed. , *Hyperbolic Geometry*. Cambridge, U.K.: Cambridge Univ. Press, 1997.

[11] W. P. Thurston, *The Geometry and Topology of Three-Manifolds*. Princeton, NJ: Princeton Univ. Press, 1997.

[12] J. W. P. Anderson, *Hyperbolic Geometry*. New York: Springer, 2001.

[13] Y. Shavitt and E. Shir, "DIMES: Let the Internet measure itself," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, Oct. 2005.

[14] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger, "The origin of power laws in Internet topologies revisited," in *Proc. IEEE INFOCOM*, New York, NY, Apr. 2002, pp. 608–617.

[15] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A global Internet host distance estimation service," *IEEE/ACM Trans. Netw.*, Oct. 2001.

[16] P. Francis, "YOID: Extending the Internet Multicast Architecture," 2000 [Online]. Available: http://www.icir.org/yoid

[17] S. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *3rd USNIX Symp. Internet Techn. Syst. (USITS'01)*, San Francisco, CA, Mar. 2001, pp. 49–60.

[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *ACM SIGCOMM*, Aug. 2001.

[19] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topology-aware overlay construction and server selection," in *Proc. IEEE INFOCOM*, Jun. 2002, pp. 1190–1199.

[20] Z. Xu and Z. Zhang, "Building low-maintenance expressways for p2p systems," no. HPL-2002-41, 2001.

[21] Z. Xu, C. Tang, and Z. Zhang, "Building topology-aware overlays using global soft-state," *Proc. ICDCS*, May 2003.

[22] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," *ACM SIGCOMM*, Aug. 2002.

[23] S. Jain, R. Mahajan, and D. Wetherall, "A study of performance potential of dht-based overlays," in *Usenix Symp. Internet Technologies (USITS)*, Mar. 2003.

[24] Y. Shavitt and T. Tankel, "On the curvature of the interent and its usage for overlay construction and distance estimation," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 374–384.

Yuval Shavitt (S'88–M'97–SM'00) received the B.Sc. degree in computer engineering (*cum laude*), the M.Sc. degree in electrical engineering, and the D.Sc. degree from the Technion, Israel Institute of Technology, Haifa, in 1986, 1992, and 1996, respectively.

From 1986 to 1991, he served in the Israel Defense Forces first as a System Engineer and the last two years as a Software Engineering Team Leader. After graduation, he spent a year as a Postdoctoral Fellow at the Department of Computer Science at Johns Hopkins University, Baltimore, MD. Between 1997 and 2001, he was a Member of Technical Staff at the Networking Research Laboratory at Bell Labs, Lucent Technologies, Holmdel, NJ. Since October 2000, he has been a Faculty Member in the School of Electrical Engineering at Tel-Aviv University, Tel-Aviv, Israel.

Dr. Shavitt served as a TPC member for INFOCOM 2000–2003 and 2005, IWQoS 2001 and 2002, ICNP 2001, IWAN 2002–2005, TRIDENTCOM 2005–2006, and others, and on the executive committee of INFOCOM 2000, 2002, and 2003. He was an Editor of *Computer Networks* from 2003 to 2004, and served as a Guest Editor for JSAC and JWWW. His recent research focuses on Internet measurement, mapping, and characterization and on QoS in networks.

Tomer Tankel (S'03) received the B.Sc. degree in scientific computation in 1986 and the M.Sc. degree in electrical engineering in 1992, both from Tel-Aviv University, Israel.

From 1992 to 2002, he worked as a Software Architect in subsidiaries of Comverse Tech. Since 2002, he has been a Ph.D. candidate in the School of Electrical Engineering at Tel-Aviv University, where he works in the computer network laboratory. His research focuses on Internet mapping and peer-to-peer networks.