On the Curvature of the Internet and its usage for Overlay Construction and Distance Estimation

Yuval Shavitt and Tomer Tankel

***Abstract*— It was noted in recent years that the Internet structure resembles a star with a highly connected core and long stretched tendrils. In this work we present a new quantity, the Internet geometric curvature, that captures the above observation by a single number. We embed the Internet distance metric in a hyperbolic space with an optimal curvature and achieve an accuracy better than achieved before for the Euclidean space. This proves our hypothesis regarding the internet curvature. We demonstrate the strength of our embedding with two applications: selecting the closest server and building an application level multicast tree.**

I. Introduction

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.* [1] experimentally discovered that the degree distribution of the Internet AS and router level graphs obey a power law. Barabási and Albert [2], [3] developed an evolutional model of preferential attachment, that can be used for generating topologies with power-law degree distributions. The Internet AS structure was shown to have a core in the middle and many tendrils connected to it [4], [5]. A more detailed descriptions is that around the core there are several rings of nodes all have tendrils of varying length attached to them. The average node degree decreases as one moves away from the core.

In this paper we identified a new characteristic of the Internet graph, its *curvature*. We use this curvature to better represent the Internet distance map in a geometric space. Using this realistic representation we were able to improve performance of three applications: Delay estimation (which can be used for QoS threshold estimation), Server Selection, and Application Level Multicast.

The geometry of a distance matrix can be represented by mapping its nodes in 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. The *symmetric pair distortion* is defined for each pair as the maximum of the ratio between the original and geometric distance

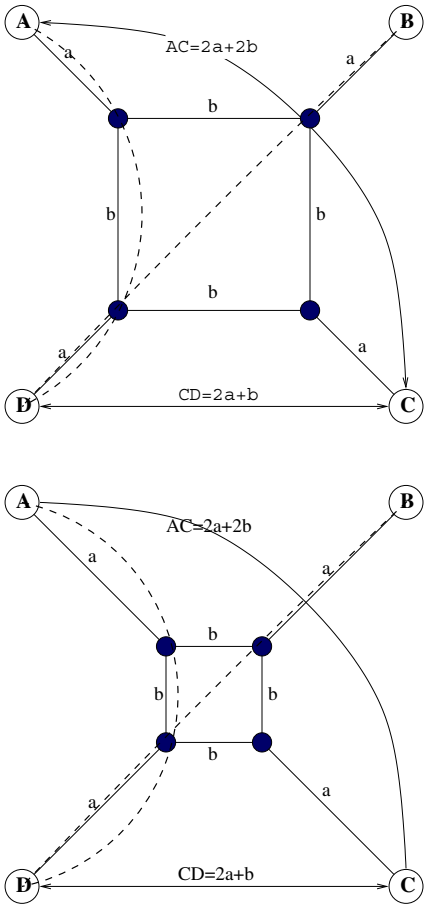

Fig. 1. a-b Graph in $\mathbb{D}^2$

and its inverse. An input metric can be embedded in two classes of algorithms

1) All pair (AP) embedding. The entire n-nodes metric, that is comprising $n(n-1)/2$ distance pairs, is embedded at once.
2) Two phase (TP) embedding. First, a small subset of $t \leq 15$ nodes, called *Tracers*, is embedded, considering all $t(t-1)/2$ pair distances. The coordinates of the rest of the nodes are calculated from their

distance to several *nearest* Tracers by minimizing the symmetric distortion of these node-Tracer pairs.

Using TP Euclidean embedding for predicting network distances was first suggested in [6], which named it Global-Network-Positioning (GNP), and was later improved in [7]. In [7] we applied Euclidean embedding to server selection application and compared its accuracy with IDMaps triangulation [8]. While we achieved good embedding in [7] the results are far from perfect due to the Internet topology structure, which consists of a core of well-connected nodes and many tendrils that are attached to it. 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've introduced in [7] a threshold criteria. Although this threshold can be tuned, it doesn't reveal the geometric shape of the graph.

In this paper we take a new and different approach for embedding the Internet graph in a geometric space. The 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. 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 bents towards the origin point, **O**. 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.

Consider for example the eight node graph in Fig. 1. The shortest path distance between its four exterior nodes, denoted A,B,C, and D can't be embedded with no distortion in 2-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.

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 target space curvature is too small (see Fig. 1). We found that the Internet AS graph (see Section III), *has the same normalized optimal metric curvature for any uniformly distributed weighting* as well as the unweighted hops graphs. A similar optimum was found for weighted Barabási-Albert model topologies, section IV.

The curvature, when selected properly (not necessarily optimally) is shown to improve the performance of all three applications mentioned above. With adequate curvature values, our method estimates all distances excluding the very short ones, while GNP suffers from underestimation for most distances and IDMaps suffers from overestimation for medium to short distances. 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. 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. For server selection, the estimation accuracy of long distances, which we want to avoid, is more important.

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 II-B.1, 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 II-C.1 the embedding potential function using the 'Loid model of hyperbolic space, and derive the field forces inducded on BBS particle in II-C.2.

A. Models of Hyperbolic Spaces

There are five models of hyperbolic spaces [9, 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 II-C.1 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 full version of this paper. 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 an 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

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

where $|dx|$ is Euclidean arc length, and $\|\cdot\|$ is 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).

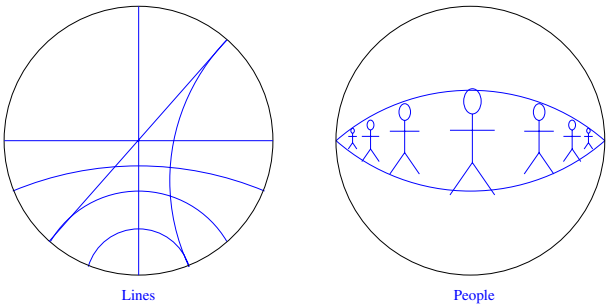

Fig. 2. Poincaré Disk Model [1]

Anderson [11] 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. 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 [11, 4.1] by

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

where,

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

With the contracted element of arc length

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[1] This picture and the discussion of Poincaré disk, are taken from [10, 2.1]

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

$$\frac{\Delta_{AC}}{\Delta_{AB}} = \frac{2a + 2b}{2a + b} = \frac{2r + 2}{r + 2} \qquad (10)$$

where $r = \frac{b}{a}$ is the ratio between the length of the inner (b) and outer (a) edges of the graph. 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

$$c_{\max} \equiv \sqrt{\kappa}\max_{i,j}\Delta_{ij} = \sqrt{\kappa}\Delta_{AC} \qquad (11)$$

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 coordinate $x = (\pm\rho, 0)$; $y = (0, \pm\rho)$. Substituting the stretched distance pairs $\sqrt{\kappa}AB$ and $\sqrt{\kappa}AC$ for $d_{\mathbb{D}}(x, y)$ in (2) we get

$$\cosh\left(c_{\max}\frac{\Delta_{AB}}{\Delta_{AC}}\right) - 1 = 2\frac{2\rho^2}{(1-\rho^2)^2} \qquad (12)$$

$$\cosh(c_{\max}) - 1 = 2\frac{(\rho-(-\rho))^2}{(1-\rho^2)(1-(-\rho)^2)} \qquad (13)$$

Multiplying (12) by 2, subtracting it from (13), and substituting (10) we obtain

$$\cosh(c_{\max}) - 2\cosh\left(c_{\max}\frac{r + 2}{2r + 2}\right) + 1 = 0\,. \qquad (14)$$

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

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 $(\Delta_{ij})_{i,j=1\ldots n}$.

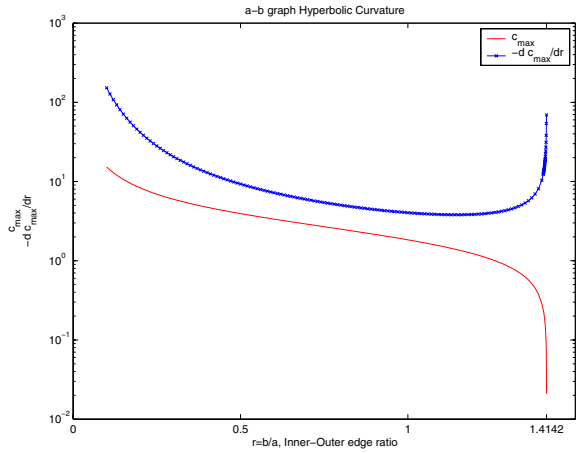

Fig. 3. a-b Graph Hyperbolic Curvature

1) BBS Embedding Method: For calculating this mapping we use the same minimization method we used earlier for Euclidean embedding [7], 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

$$E_T(v_1, v_2, \ldots, v_n) = \sum_{\substack{i,j=1 \\ i>j}}^{n} E_{ij}(v_i, v_j)\,. \qquad (15)$$

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 distance embedding error of a pair of particles, the 'pair embedding error' is denoted by E_{ij}.

Our embedding solver uses the 'Loid model of hyperbolic space, which averts the distance singularity on the boundary of the poincaré disk. As in [7] we divide the embedding into four calculation phases. The phase pair embedding error function denoted by $E_{ij}^{\langle p\rangle}$, assumes the form

$$E_{ij}^{\langle p\rangle}(v_i, v_j) = \mathcal{F}(d_S(v_i, v_j), \Delta_{ij})\,, \qquad (16)$$
$$\text{for } i \neq j \quad \text{and} \quad v_i \neq v_j$$

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

$$S^d = \left\{\xi\ : \xi_1^2 + \ldots + \xi_d^2 - \xi_{d+1}^2 = -1\right\} \qquad (17)$$

$$\xi = \left(\xi_1, \ldots \xi_d, \sqrt{1 + \sum_{i=1}^{d}\xi_i^2}\right)\,. \qquad (18)$$

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 [7, 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 (15), is given by

$$\vec{F}_{i_0} = -\nabla_{v_{i_0}} E_T(v_1, \ldots, v_n) \quad (19)$$

$$= -\sum_{\substack{j=1 \\ j \neq i_0}}^{n} \mathcal{F}_x(x, \Delta_{i_0 j})|_{x=d^S_{i_0 j}} \nabla_{v_{i_0}} d^S_{i_0 j} \ , \quad (20)$$

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

$$\nabla_{v_i} d^S_{ij} \equiv \left(\frac{\partial}{\partial \xi_k} d^S_{ij}\right) = \left(\xi_k \frac{\psi_{d+1}}{\xi_{d+1}} - \psi_k\right) \div \sinh d^S_{ij} \ . \quad (21)$$

III. The curvature of the AS graph

This section presents the AS topology curvature measured on a single[2] instance from U. of Oregon Route-Views database dated January 2nd 2000, by embedding its weighted and unweighted graph with different curvatures of the target hyperbolic space, and comparing the resulting distance distortion. The effect of increasing embedding dimension is also discussed. We compare our embedding results with two other Euclidean embedding methods

- Down-Hill-Simplex (DHS), the method used in Global-Network-Positioning (GNP) [6].
- Multi-Dimensional-Scaling (MDS), a method used in many fields [7].

We chose to compare only with GNP and not our Euclidean BBS, [7], although there we shown that BBS is more accurate and scalable than GNP. However [7] compared AP embedding, while in this paper we focus on TP embedding. For TP embedding the two methods produce similar results, since they both accurately calculate the coordinates of the $t \leq 15$ Tracers, while the coordinates of the rest nodes are calculated separately for each node in both methods.

[2]Since the curvature may slightly change in time we use only a single instance to show the stability of the curvature to link weights. In section V, where we present the performance of server selection application with our embedding, we use two sets of 9 different AS topologies each.

A. Experiment Details and Legend

We use the following measures to compare the embedding efficiency and accuracy

- **CPU Time** The CPU time to calculate an AP embedding, or with TP embedding, the CPU time to embed the distances among Tracers, and excluding the time consumed for triangulating the rest of nodes.
- **Symmetric Pair Distortion** Defined for each nodes pair as the maximum of the ratio of the measured to the geometric distance, and its inverse.
- **Directional relative error** Defined by [6, Eq. 4] as the ratio of the difference between the geometric and measured distance, to the minimum of the two distances.
- **95-5 percentile ratio** Defined as the ratio of the 95 percentile of the ratio between the geometric and the measured distance, to the 5 percentile of this ratio.

We experiment with different curvatures of the target hyperbolic space. In Section II-B.2 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 normalize the metric either by the mean of all distances or by their maximum that is the diameter of the metric.

The following legend notations were used in all the figures: HYP, MDS, and GNP where used to denote our hyperbolic embedding, MDS, and DHS (which was used by GNP), correspondingly. The number following HYP indicates the stretch where positive stretch values indicate normalization by the mean of all distances, and negative stretch value indicate normalization by the diameter of the metric.

In each experiment just a small group of the AS nodes were embedded. The groups we selected throughout the paper are all selected at random among the lowest degree nodes, or stub ASs.

B. Embedding Alternatives Comparison

The input to our calculation was the Jan-00 AS topology. To increase the confidence each experiment was conducted using 5 sets of random weights. The weights drawn here, and throughout the paper, are $i.i.d.$ random variables, distributed uniformly in the interval $[1, 1000]$. ¿From each random weights graph we embedded two random subsets of nodes. Namely each point in the comparison graph results from 10 embedding experiments

Fig. 4 compare the AP Embedding (top) and TP Embedding (bottom) for a subset of 150 nodes. The normalization for the AP Embedding was by mean distance, whereas the normalization for the TP Embedding was by

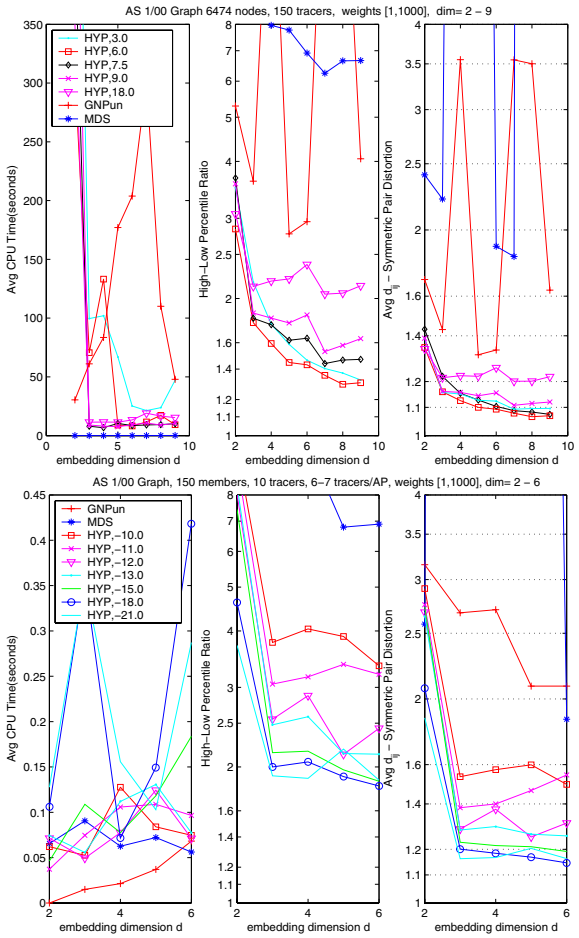

Fig. 4. AS Curvatures Distortion vs. Embedding Dimension

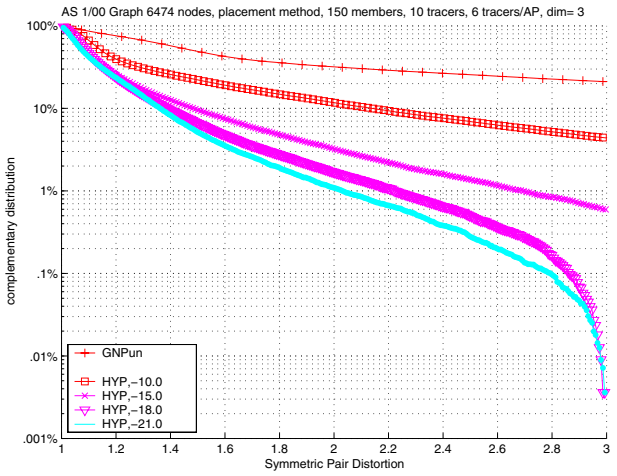

Fig. 5. AS Curvatures Distortion Histogram

D_{Tr}, the Tracers sub-graph diameter, due to the small number of Inter-Tracers distances. MDS doesn't support TP Embedding since it uses $O(n^2)$ distances, thus we use AP embedding for its both experiments. We used $t = 10$ Tracers and 6 measurements in the TP embedding for all dimensions, except for dimension 6 in which 7 measurements were used.

It is clear from Fig. 4 that HYP accuracy is much better than MDS and GNP for both embedding classes and for all tested stretch values. For both embedding classes an optimal metric stretch exists, which minimizes the average symmetric pair distortion and the 95-5 percentile ratio. The optimal stretching for AP embedding is $6/\overline{\Delta_{ij}}$, for the entire range of dimensions. In TP embedding the optimal stretch decreases with an increase in the embedding dimension. For $2, 3$ and 4-dimensional space the optimal metric stretch factor is $21/D_{Tr}$, while for 5 and 6 dimensional the optimum is $18/D_{Tr}$.

AP embedding yields better accuracy than TP embedding, as expected, however, the difference is not significant. On the other hand, TP embedding requires two orders of magnitude smaller CPU time, and achieves very good embedding for only 3 or 4 dimensions. This makes TP embedding the better choice in practice.

Fig. 5 depicts the symmetric pair distortion histogram, for 3-dimensional embedding using $t = 10$ Tracers with 6 measurements. The histogram is calculated over all pairs of nodes in the graph. For $(21/D_{Tr}) - stretched$ HYP, more than 99% of pairs, compared to 70% of GNP pairs, have symmetric pair distortion less than 2, which corresponds directional relative error in interval $(-1, 1)$. Moreover, 20% of GNP pairs, compared to $.003\%$ of optimal HYP pairs ($1 \div 6000$ of GNP pairs), have symmetric pair distortion larger than 3.

C. Hop Distance Estimation

The distribution of the directional relative error, estimating hop distance in the same AS topology, using HYP, GNP and IDMaps methods, is depicted in Fig. 6.

For all methods we select two subsets of $t = 10$ Tracers randomly, and used 6 measurements per each of 6474 nodes. Next we triangulated a randomly selected source node, and calculated the 5-dimensional geometric distance of this node from all the rest of the nodes in the AS graph. We repeated this for 100 different source nodes.

In order to better understand the distribution of the estimation error, we group the embedding pair distances 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 line has whiskers at the 5, 25, 75, and 95 percentiles.

As we reported in [7], GNP underestimates longer hop distances, having negative relative errors. Naturally IDMaps triangulation overestimates all distances, due to its additive estimation. The $(18/diameter) - stretched$ metric has the best relative hop error, with the 5 to 95 percentiles in $(-0.4 \ldots 0.5)$ for hop distances longer than 2. IDMaps estimation error is larger for small hop distances. For instance at 2-hops distance IDMaps have 5 to 95 percentile in $(1, 2)$, compared to $(-.7, .6)$ for optimally stretched metric.

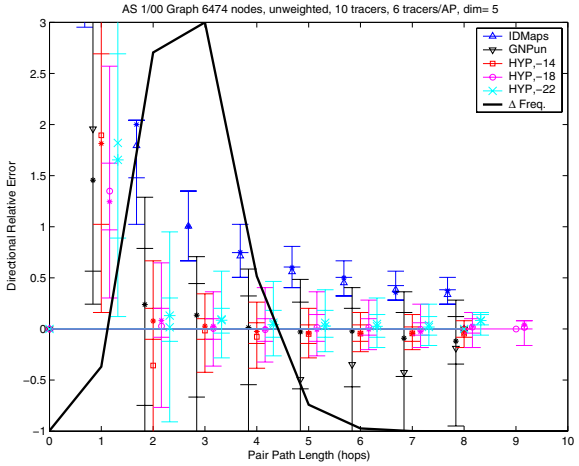

Fig. 6. AS Graph Relative Hops Error

IV. The Curvature of Generated Power-Law Graphs

We generated 5 random power-law topologies according to the Barabási-Albert (BA) algorithm [3]. To increase the confidence each experiment was conducted using 3 sets of random weights per each of the generated topologies. From each random weights graph we embedded two random subsets of nodes. Namely each point in the comparison graph results from 30 embedding experiments.

We used $t = 10$ Tracers and 6 measurements in the TP embedding for all dimensions, except for dimension 6 in which 7 measurements were used, and except MDS which uses AP embedding (see III-B).

Fig. 7 depicts TP embedding of the distances among a subset of 150 nodes.

The accuracy of our method is far better than MDS and GNP for all tested hyperbolic metric stretch values, marked as number following the HYP legend. The optimal curvature, which achieves the minimum pair distortion, decreases with increased embedding dimension. For 2 and 3-dimensional space the optimal metric stretch factor is $18/D_{Tr}$, while for 4, 5 and 6-dimensional the optimal value is approximately $12/D_{Tr}$.

V. Applications

In this section, we evaluate the effect of hyperbolic curvature, with TP embedding, on delay estimation, server selection, and application level multicast.

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. We compared our hyperbolic embedding with the optimal curvature (a stretch of $18/D_{Tr}$) in 5-dimensional hyperbolic space, with GNP and IDMaps estimations. In GNP and HYP we estimate the delay by the 5-dimensional geometric distance between the TP embedding coordinates of the node pairs. For all methods we randomly selected two subsets of $t = 10$ Tracers and have used 6 measurements per node. We repeated the above experiments with 100 source nodes, selected randomly from each of the 5 weighted AS graphs described in Section III-B.

In order to capture the distribution of estimation error, we group the pair distances from 75mS wide network distance intervals. The directional relative delay error, for weighted AS graph, are depicted on the left hand side of Fig. 8. The method marker is placed at the *average* directional relative error, and the star marker depicts the median. Each line has whiskers at the 5, 25, 75, and 95 percentiles. IDMaps highly overestimates distances, especially for delays below $1000ms$, due to its additive

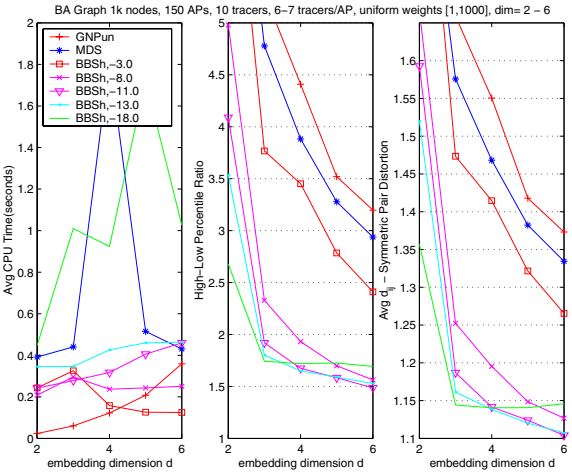

Fig. 7. BA Curvatures Distortion vs. Embedding Dimension

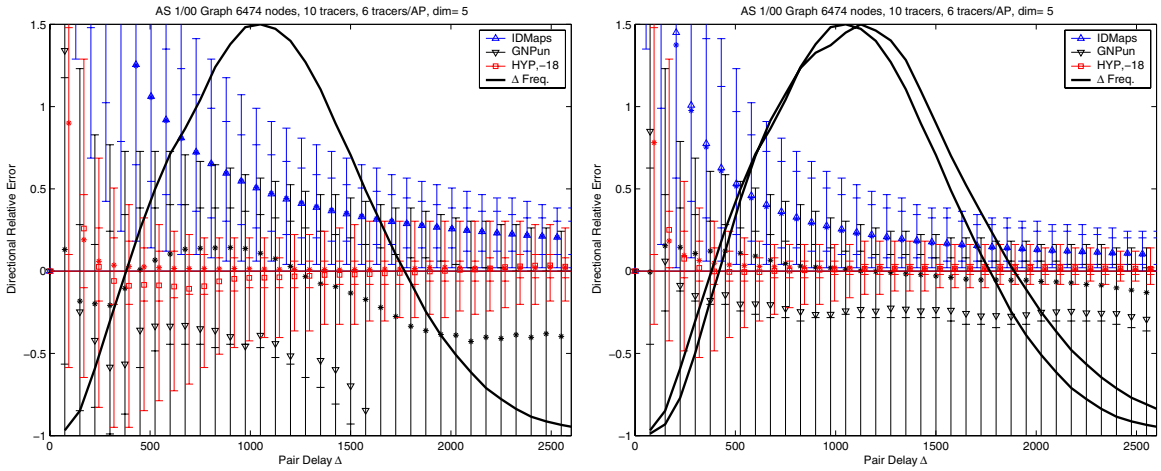

Fig. 8. AS Graph Relative Delay Error

estimation. As we reported in [7], GNP underestimates delays above $500ms$, due to the curvature of the AS graph. However when using a target space with optimal curvature, the median and average relative error of our embedding remains approximately 0, in all delays, except for very short delays below $200ms$.

In our simulations, $18/D_{Tr}$ proved to be the best stretch factor. Note that the same stretch value was also the optimal stretch for hop distance estimation discussed above (Section III-C).

By combining local information from the underlying network topology with the measurements to the Tracers, we can improve low-degree nodes embedding estimations. A relatively easy improvement is to calculate the geometric distance of degree-1 node using the coordinates of its degree-2 neighbor, and then add to the result the delay between the degree-1 node and its degree-2 neighbor. This leaf correction effect is depicted on the right hand side of Fig. 8. Clearly, IDMaps isn't affected by the correction due to its additive nature. GNP performance improves significantly, however, our optimal curvature embedding benefits most from this correction. For delays longer than $400ms$, more than 50% of our relative errors fall inside $(-.1, .1)$, and more than 90% fall inside $(-.2, .2)$ for delays longer than $600ms$[3].

B. Server Selection

This experiment used the Oregon route-views and the Oregon combined route-view plus looking glass and router registry, as described in [12]. The nine couples of peering data sets were collected weekly starting Mar. 2001. To increase the confidence each experiment was conducted using 3 sets of random weights per each of the peering topologies.

Following [8] 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. 9 depicts the average cumulative distribution function (CDF) of rank accuracy for IDMaps GNP and HYP methods. The number following HYP indicates the metric stretch factor. For all methods, due to the size of the Oregon topologies, we used in this experiment $t = 15$ 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 $08lc$ represent the usage of the leaf correction described in Section V-A.

The top graph depicts the results for the Oregon route-views, while the bottom one depicts the results for the combined 'Oregon+' views. IDMaps ranking performance are nearly perfect, achieving at least 98.5% correct answers in 99% of the mirror experiments. GNP method however has the least ranking accuracy, due to underestimating of all distances, and is thus ruled out as practical method for 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 $30/D_{Tr}$ (see inset). For Oregon route views, depicted on top inset, our

[3]These delay numbers are synthetic because the median delay with our weighting is at least twice than the one of real Internet

performance even slightly supercedes IDMaps, achieving, at least, 99% correct answers in 99% of the mirror group experiments.

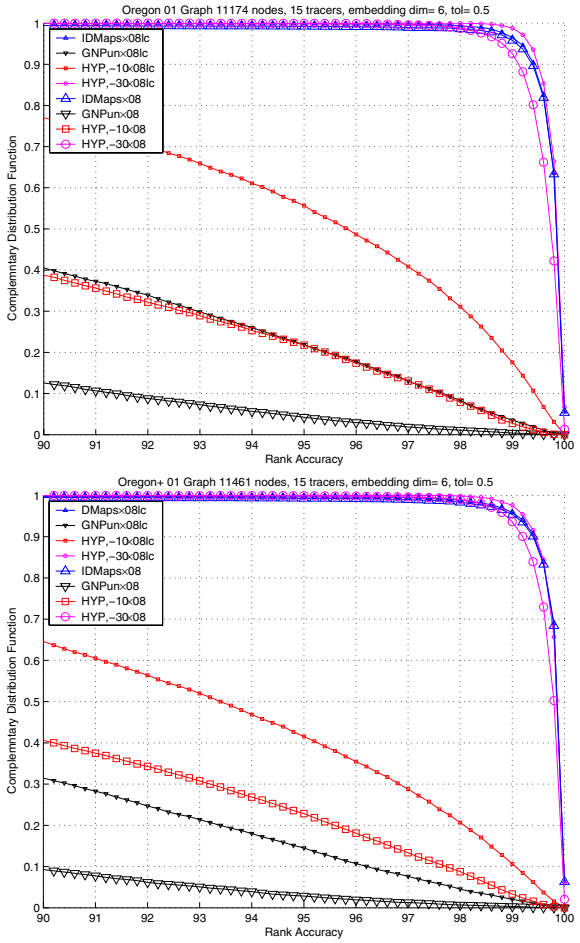

Fig. 9. AS Curvatures Mirror Selection

C. Application Level Multicast

In application level multicast [13], [14], [15], 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 [16], [17] and its derivatives [18], [19]. 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.

An alternative tree-first approach is NICE application multicast [20], 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 graph nodes, a source node, m, and construct $T(V_T, A_T)$, a multicast tree from m to all nodes in M. We assume no topology or routing information from the underlying 'physical' network. Our **Geometric-Multicast-Tree** (see Fig. 10) uses the geometric space coordinates, assigned by the embedding of each node, in order to make greedy geometric decisions. For given core ratio c, the core nodes radius r_c is selected such that

$$|\{v \in M : \|v\| <= r_c\}| = \quad c\,|M| \ ; \tag{22}$$

we used $c = .05$. Recall that an angle θ in a hyperbolic triangle with edge lengths a, b, c that is facing the edge a is given by the hyperbolic law of cosine **I** [11, 5.7]:

$$\cos\theta = \quad \frac{\cosh(b)\cosh(c) - \cosh a}{\sinh(b)\sinh(c)} \tag{23}$$

Estimation accuracy of short distances has significant effect on the output tree. Indeed, the algorithm elects tree children among unelected neighboring points of the current parent, satisfying the following two conditions

- Local Sparseness Rule. Refrain from electing as child of the current parent, a neighbor residing inside the θ_c-cones around other elected children of this parent.
- Global Expansion Rule. Our algorithm starts from the node nearest to the hyperbolic origin. It then spans neighboring core nodes with the local sparseness rule[4]. Upon discovering child nodes located further away from hyperbolic origin, the algorithm only elect a child if the corresponding origin-parent-child angles are larger then θ_p.

Latency Stretch is defined per member $m\prime \in M$, $m\prime \neq m$ and is the ratio of the path length along the tree T from m to $m\prime$ to the length of the direct unicast path. *Link Stress* is defined per link of the underlying topology and count the number of identical packets sent between members of M over that link. This definition of stress, following [21], is from the network's perspective, rather then the application's.

[4]In Euclidean space however, no special geometric region exists for the core, and thus we use a *local expansion rule*, (see footnote 5).

[5] For Euclidean space the expansion rule is $\angle u\,p\,p\prime \geq \theta_p$, where $p\prime$ denotes the parent of p in T,

Init: (different for Euclid/Hyp space)

Var.	Hyperbolic	Euclidean
P	$m_0 : \|\|m_0\|\| = \min_{v \in M} \|\|v\|\|$	m
T	$V_T = \{m\};\ A_T = \{(m, m_0)\}$	$\emptyset$

While $P \neq \emptyset$

1) $V_T \leftarrow V_T \cup P;\ C \leftarrow \emptyset$
2) For each point $p \in P$, selected at arbitrary order,
   a) $C_1 \leftarrow \emptyset$
   b) For $i = 1$ to N_0
      i) Select $u \in (M \setminus (V_T \cup C))$, the ith nearest point to p
      ii) Sparsness: $\forall v \in C_1$ holds $\angle u\, p\, v > \theta_c$
      iii) Expansion[5]: $[\angle u\, p\, \mathbf{0} \geq \theta_p] \vee [\|p\| \leq r_c]$
      iv) If ii and iii holds then $C_1 \leftarrow C_1 \cup \{u\};\ A_T \leftarrow A_T \cup \{(p, u)\}$
   c) $C \leftarrow C \cup C_1$
3) $P \leftarrow C$

Fig. 10. The Geometric-Multicast-Tree algorithm

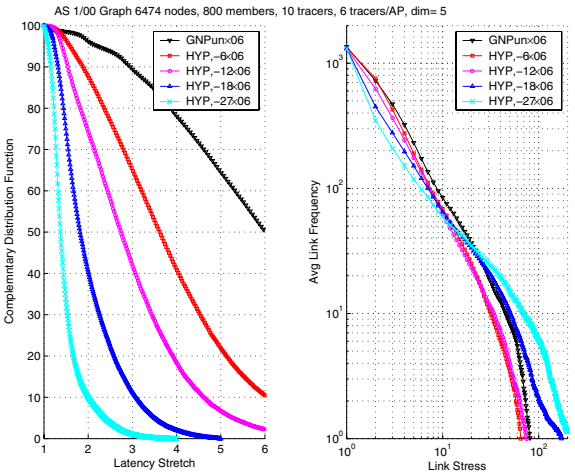

Fig. 11. Application Level Mulitcast Histogram

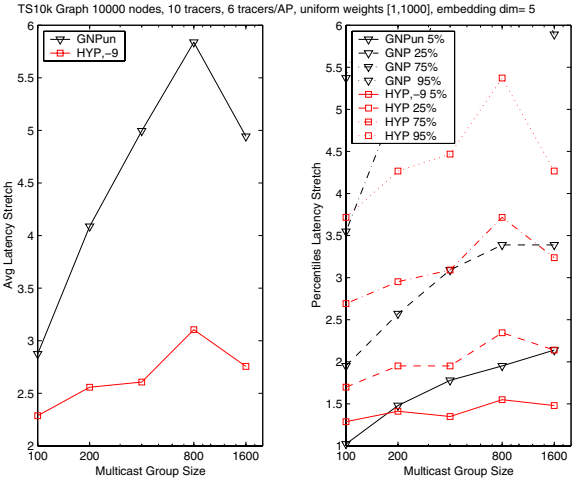

Fig. 12. Latency Stretch vs. Multicast Group Size.

To evaluate our algorithm we performed the following experiment. Select members of M randomly among low-degree, ≤ 2, nodes of the graph. Use TP embedding with $t = 10$ Tracers and 6 measurements per node to embed these nodes in 5-dimesional Hyperbolic and Euclidean space. Run[6] our tree algorithm from 40 different source nodes $m_1, m_2, \ldots$. Fig. 11 depicts the calculation results for $|M| = 800$ on the Jan-00 AS topology. We mark Hyperbolic and Euclidean space trees by HYP and GNP respectively. Also shown, in black, is the optimal minimum spanning tree of the with clique $O(|M|^2)$ network distances. The complementary distribution function, depicted on the left hand side, was aggregated from latency stretches of all the nodes $m\prime \in M$, 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.

Fig. 11 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 our stress with curvature factor $18/D_{Tr}$. However the 95 percentile stretch of HYP with this curvature is 3.5, compared to 11.4 of GNP.

For comparison with [17], [21] we performed the above experiment also for Transit-Stub topology of 10000 nodes. Fig. 12 depicts the effect of multicast tree size on stretch. With curvature factor $9/D_{Tr}$ ($\circ$ marker) our average stretch depicted on the left graph, is comparable with topology aware CAN [17, Fig 9]. Comparing with the more detailed data in [21], our high percentile stretch values (depicted by top line in the right graph), is similar to the 90th percentile with the best routing heuristic for topology aware CAN. Note that results for CAN assume global and perfect knowledge of the topology.

VI. Concluding Remarks

We introduced a new quantity, the Internet geometric curvature, and showed how embedding the Internet metric in hyperbolic space with adequate curvature results in superb accuracy. We believe that the curvature is an

[6]We used the tresholds $\cos\theta_c = .85;\ \cos\theta_p = .8$ and the neighbors limit $N_0 = 20$ in all Hyperbolic runs.

intrinsic characteristic of the Internet, and thus can be used to test network generators against true Internet topologies.

We demonstrated the quality of our embedding with three important applications. We are working on distributing our tree construction algorithm to enable its scaling to large peer-to-peer networks. For this end we will reverse the node attachment process, s.t., a node will search for the parent unlike the present algorithm where parents select children.

REFERENCES

[1] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *ACM SIGCOMM 1999*, Boston, MA, USA, Aug./Sept. 1999.

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

[3] R. Albert and A.-L. Barabási, "Topology of evolving networks: local events and universality," *Physical Review Letters*, pp. 5234–5237, Dec. 2000.

[4] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz, "Characterizing the internet hierarchy from multiple vantage points," in *Infocom*, 2002.

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

[6] T. Ng and H. Zhang, "Predicting internet network distance with coordinates based approaches," in *Infocom*, June 2002.

[7] Y. Shavitt and T. Tankel, "Big-Bang simulation for embedding network distances in Euclidean space," in *Infocom*, Mar. 2003.

[8] 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. on Net.*, Oct. 2001.

[9] J. W. Cannon, W. J. Floyd, R. Kenyon, and W. R. Parry, *Hyperbolic Geometry*, S. Levy, Ed. Cambridge University Press, 1997.

[10] W. P. Thurston, *The Geometry and Topology of Three-Manifolds*, 2002.

[11] J. W. P. Anderson, *Hyperbolic Geometry*. springer, 2001.

[12] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "The origin of power-laws in internet topologies revisited," in *IEEE Infocom 2002*, New-York, NY, USA, Apr. 2002.

[13] P. Francis, "Yoid: Extending the internet multicast architecture," 2000, http://www.icir.org/yoid.

[14] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01)*, San Francisco, CA, USA, Mar. 2001, pp. 49–60.

[15] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz, "Bayeux: An architecture for scalable and fault-tolerant widearea data dissemination," in *the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001)*, June 2001.

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

[17] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topology-aware overlay construction and server selection," in *Infocom*, June 2002.

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

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

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

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