A Decentralized Network Coordinate System for Robust Internet Distance Prediction*

Li-wei Lehman and Steven Lerman
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

Network distance, measured as round-trip latency between hosts, is important for the performance of many Internet applications. For example, nearest server selection and proximity routing in peer-to-peer networks rely on the ability to select nodes based on inter-host latencies. This paper presents PCoord, a decentralized network coordinate system for Internet distance prediction. In PCoord, the network is modeled as a D-dimensional geometric space; each host computes its coordinates in this geometric space to characterize its network location based on a small number of peer-to-peer network measurements. The goal is to embed hosts in the geometric space so that the Euclidean distance between two hosts' coordinates accurately predicts their actual inter-host network latency. PCoord constructs network coordinates in a fully decentralized fashion. We present several mechanisms in PCoord to stabilize the system convergence. Our simulation results using real Internet measurements suggest that, even under an extremely challenging flash-crowd scenario where 1740 hosts simultaneously join the system, PCoord with a 5-dimensional Euclidean model is able to converge to 11% median prediction error in 10 coordinate updates per host on average.

1 Introduction

Network distance, measured as round-trip latency between hosts, is important for the performance of many Internet applications. For example, nearest server selection and proximity routing in peer-to-peer networks rely on the ability to select nodes based on inter-host latencies. Network coordinate systems have recently been developed as a scalable mechanism to predict latencies among arbitrary Internet hosts [13, 11, 16, 22, 10, 20, 21, 4, 2, 8, 9]. The idea of a network coordinate system, first proposed by the GNP system [13], is for each host to derive a mapping of itself in some D-dimensional geometric space using a small set of sampled distances so that the actual inter-host network latencies can be estimated as a function of the nodes' geometric distances. GNP relies on a fixed set of landmark nodes in the Internet to provide reference coordinates. Hosts position themselves in the geometric space using measured distances to these landmark nodes. Using a fixed set of landmarks presents a potential performance bottleneck. Additionally, the accuracy of the GNP scheme is sensitive to the landmark placement.

This paper presents a fully decentralized network coordinate system, PCoord, for robust Internet distance prediction. PCoord does not rely on a fixed set of landmark nodes to construct coordinates. Instead, hosts compute their coordinates based on sampled distances and reference coordinates gathered from other participating peer hosts. PCoord differs from other decentralized coordinate systems, such as Vivaldi and PIC [4, 2], in that it introduces the following stability mechanisms to facilitate convergence to low system prediction errors:

- A sample weighting scheme that gives more weight to samples with higher prediction accuracy.
- A "friction" mechanism that helps to stabilize convergence.
- A "damping" mechanism to avoid instability and oscillation caused by noisy or faulty samples.

One major difference between PCoord and Vivaldi [3, 4] is that in PCoord a node computes its coordinates by optimizing a loss function over a *batch* of samples, whereas in Vivaldi a node adjusts its coordinates to minimize the prediction error one sample at a time by simulating movement in a physical spring system. Another important difference between the two schemes is that Vivaldi adjusts the amount of its coordinate movement based on an adaptive timestep defined in terms of local and remote errors. In contrast, a host in PCoord stabilizes convergence by introducing some amount of friction proportional to a node's confidence in its

*This research was supported in part by the Singapore-MIT Alliance.

own coordinates, and fine tunes its coordinate movement by an additional integrity check based on the goodness-of-fit of the entire batch of sampled distances and coordinates.

Through simulations using real network measurements, we examine the convergence behavior and prediction accuracy of PCoord, and compare its performance with existing network coordinate systems. Our results suggest that, under an extremely challenging flash-crowd scenario where 1740 nodes join simultaneously, PCoord can converge to prediction error as low as GNP within 10 seconds without any fixed landmarks, by using half as many samples as Vivaldi.

The rest of the paper is organized as follows. Section 2 describes the PCoord algorithm. In section 3, we present our simulation results and compare the performance of PCoord with existing schemes. In section 4, we describe related work. Finally, we present our conclusions and ideas for future work in Section 5.

2 The Algorithm

PCoord is a fully decentralized network coordinate system with each host updating its coordinates iteratively using a set of peer-to-peer latency measurements. Each host updates its coordinates to minimize a loss function that measures the difference between the actual and the geometric distances between itself and a small set of other hosts.

We previously introduced PCoord in [9] to refer to a general framework of peer-to-peer based decentralized network coordinates construction. Under the PCoord framework in [9], we evaluated several proof-of-concept coordinate mapping strategies and explored their performance characteristics under different peer sampling strategies. Our prior work in [9] did not provide any mechanisms to stabilize system convergence, nor did it address issues in dynamic join/leave. In this paper, we present the PCoord algorithm which introduces the following mechanisms.

- A weighted loss function that allows sampled coordinates with higher prediction accuracy to have a higher weight in the loss function.
- A friction mechanism with a weighted "resistance" factor in the loss function that helps to stabilize the convergence process.
- A damping mechanism that dampens the amount a node moves toward new coordinates by a factor that is inversely proportional to the fit error of the current batch of sampled peer nodes' coordinates and distances.

We present the PCoord algorithm in two steps. We first present a Simple algorithm, which describes the general PCoord framework without any of the stability mechanisms. We then present the actual PCoord algorithm, and describe the stability mechanisms in details.

2.1 A Simple Algorithm

In Simple PCoord, each node performs continuous update on its coordinates; each coordinate update consists of two phases: (1) the sampling and information exchange phase in which a node selects M reference points, gathers distance measurements and coordinates, and (2) the coordinate update phase, in which new coordinates are computed to minimize a loss function defined in terms of those M reference points. In this paper, we focus on phase two only, which is the coordinate update step. A variation of the Simple algorithm was previously presented in [9], which describes the peer sampling and probing strategies. In the Simple algorithm below, c_i is the coordinates of host i, d_{ij} is the measured RTT between i and j, and c_{origin} is the coordinates at the origin of the geometric space.

//i is the node that is running the procedure
SimplePCoord() {
  $c_i = c_{origin}$
  while (in the system) {
    $Samples$ = SamplePeers()
    c_i^{new} = **MinimizeError**($Samples$, c_i)
    $c_i = c_i^{new}$
  } //end while
}

//$Samples$ consist of M sampled peer nodes
//c_{guess} is the initial guess for the coordinates
MinimizeError ($Samples$, c_{guess}) {
  find c_i^{new} that minimizes E using
  c_{guess} as an initial guess, where
    $E = \sum_{j \in Samples} (d_{ij} - \|c_i^{new} - c_j\|)^2$
  return c_i^{new}
}

At each update iteration, each host i measures its round trip latencies to M other peer nodes, and obtains those M nodes' current coordinates. Host i then updates its coordinates to minimize the sum of squared differences between the measured and computed distances with those M peer nodes.

There are several potential problems with the above Simple algorithm. One problem is that it does not distinguish between nodes with coordinates that have different prediction accuracies. Another problem is that the algorithm determines the new coordinates entirely based on measurements from the current batch of reference points. The Simple scheme thus tends to react too quickly based on the measurements of the current batch of reference points.

2.2 The PCoord Algorithm

Constructing network coordinates in a fully decentralized manner is difficult, because hosts are making parallel and independent updates of their network coordinates based on a small number of samples which potentially contain noisy and faulty information. From an individual node's perspective, the key question is whether and by how much it should react and move its own coordinates based on sampled information, and whether such incremental local optimization steps will ultimately lead to a low-error global configuration. The PCoord algorithm introduces the following three stability mechanisms to facilitate convergence to a low error configuration.

2.2.1 Weighted Loss Function

To avoid reacting too quickly to bad reference points, we propose a weighted loss function, in which the loss each reference point contributes is weighted by the confidence index of each reference point's coordinates so that nodes with more accurate coordinates are given more weight. In order to associate a confidence index with a node's coordinates, each node i maintains a weighted moving average of its past relative prediction error, e_i^p, where $0 < e_i^p < 1$. Node i computes the relative prediction error defined as $MIN(1, \frac{|\|c_i - c_j\| - d_{ij}|}{d_{ij}})$, where j is another node sampled by i in the past. The confidence index of node i, denoted a_i, is defined as $1 - e_i^p$. Each node continuously updates the confidence index of its own coordinates as a function of the weighted moving average of its past prediction error. The weight is assigned to each sample j as follows: $w_j = \frac{a_j^2}{\sum_{k \in Samples} a_k^2}$.

2.2.2 The Friction Mechanism

In order to reduce oscillation, we introduce an additional friction mechanism into the loss function so that a node with accurate coordinates will not overly react to reference points with less accurate coordinates. When computing c_i^{new}, node i adds itself as the $(M+1)$th node in its reference points set, and thus introduces c_i into the loss function as a resistance factor that penalizes movement of c_i^{new} to a new location. This term is weighted by the relative prediction accuracy (relative to other reference points of i) of node i's coordinates, so that the more confident a node is about the accuracy of its own coordinates, the more resistance the term introduces. For a newly joined node, the weight to this friction term is initialized to zero. Let w_i be the weight of node i. The coordinate update procedure now becomes a problem of finding c_i^{new} that minimizes the weighted loss $\mathcal{E}$, where $\mathcal{E}$ is defined as follows.

$$\mathcal{E} = w_i(d_{ii} - \|c_i^{new} - c_i\|)^2 + \sum_{j=1}^{M} w_j(d_{ij} - \|c_i^{new} - c_j\|)^2$$

where $d_{ii} = 0$, $0 \leq w_i \leq 1, 0 \leq w_j \leq 1$, and $w_i + \sum_{j=1}^{M} w_j = 1$.

2.2.3 The Damping Mechanism

In this section, we introduce a mechanism that allows a node to adjust how much it moves its coordinates in response to a particular batch of samples based on the goodness-of-fit index of the current batch of samples. The goodness-of-fit is a confidence measurement associated with an entire batch of samples for one coordinate update iteration. While the weighted loss function relies on the other nodes to supply their confidence indexes, the damping mechanism provides an additional "sanity" check on the integrity of the samples. The idea is that if the batch of samples contain any "inconsistent" or "mis-behaving" distances or coordinates, it will likely yield higher residual error than good batches of samples. To avoid reacting to a batch of samples with bad fit, each PCoord node maintains a weighted moving average of the fit error over time. A node decides how much it should react to the current batch of samples based on the the goodness-of-fit of current batch of samples relative to the past batches of samples. More precisely, if the fit error of the current batch of samples exceeds the average fit error, then the node dampens the amount it moves toward the new coordinates by a factor ρ which is the ratio between the average and current fit error. The PCoord pseudocode is summarized as follows.

PCoord() {
    $c_i = c_{origin}$
    while (in the system) {
        $Samples$ = SamplePeers()
        //add node i's own coordinates to the samples
        $Samples$ = $Samples$.add(i)
        c_i^{new} = **MinimizeWeightedError**($Samples, c_i$)
        $c_i = c_i^{new}$
    }
}

MinimizeWeightedError ($Samples$, c_{guess}) {
    for each node k in $Samples$ {
        //assign weight to each node in $Samples$
        $w_k = \frac{a_k^2}{\sum_{j \in Samples} a_j^2}$
    } //end for

    find c_i^{new} that minimizes
        $\sum_{k \in Samples} w_k(d_{ik} - \|c_i^{new} - c_k\|)^2$

//e_i^f is the weighted moving average of i's fit error
//e_i^{newf} is the residual error after minimization
$e_i^{newf} = \sum_{k \in Samples} w_k (d_{ik} - \|c_i^{new} - c_k\|)^2$
$\rho = MIN(\frac{e_i^f}{e_i^{newf}}, 1)$
//move ρ fraction of the way toward the new solution
$c_i^{new} = c_i + (\rho * (c_i^{new} - c_i))$
return (c_i^{new})
} //end MinimizeWeightedError

3 Evaluation of PCoord

3.1 Evaluation Methodology

We evaluate PCoord through simulations using real network measurements. We compare the performance of PCoord with Vivaldi, and the original GNP scheme in terms of pairwise distance prediction accuracy.

3.1.1 Performance Metrics

We define the *prediction error (PE)*, or simply error, of a link as the absolute difference between the predicted RTT and the actual RTT. Following the conventions in Vivaldi [4], we define the error of a node as the median of the link errors for links involving that node. The error of the system is defined as the median of the node errors for all nodes in the system. We compare PCoord, Vivaldi, and the GNP scheme using the relative error (RE) metric. For each pair of nodes,i and j, their relative error is defined as $\frac{|\|c_i - c_j\| - d_{ij}|}{d_{ij}}$.

3.1.2 Data Collection

We evaluate our scheme using the following network measurements.

- The King data set from Vivaldi [4], which involves the round-trip latencies among 1740 Internet DNS servers. The median RTT of the King data set is 159 ms.
- The PlanetLab [17] all-pairs-ping data set among 127 nodes collected on May 10, 2004.
- The AMP [6] data set from January 30, 2003 which measures the round-trip ping time among 104 nodes.
- The RON2 data set [18, 1], which measures the RTTs among 15 Internet hosts.

We only present the King results here due to space constraints. The results from other data sets are qualitatively similar and can be found in [7].

3.1.3 Simulation Setup

We have simulated the execution of PCoord using p2psim [14], an event-driven, packet-level network simulator. We use the Simplex Downhill algorithm to minimize the loss function at each coordinate update. We measure the processing cost of the Simplex Downhill operation on a Sun UltraSparc with 150 MHz CPU, and use the measured median processing time in our PCoord simulation. The median processing time is 10 ms per coordinate update when 10 reference points are used.

Each node proceeds in its coordinate update independent of other nodes' update progress. Asynchronous communication is used when gathering samples from M peer nodes in each coordinate update step. Both PCoord and Vivaldi randomly sample peers from the global population.

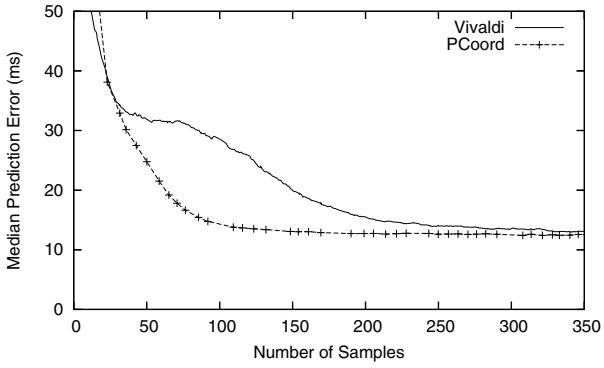
We ran the PCoord simulations with various sample batch sizes M to explore their effects on convergence and prediction accuracy. Our results suggest that using 10 reference points at each update step ($M = 10$) yields quick convergence to low error and achieves a good tradeoff between communication and computation overhead.

All Vivaldi simulations presented in this work use the adaptive time step mechanism described in [4] and implemented in the p2psim [14]. In Vivaldi, a constant C_c ($0 \leq C_c \leq 1$) is used to control how much a node reacts to each new sample. We set C_c to 0.25, which is reported to yield both quick error reduction and low oscillation [4].

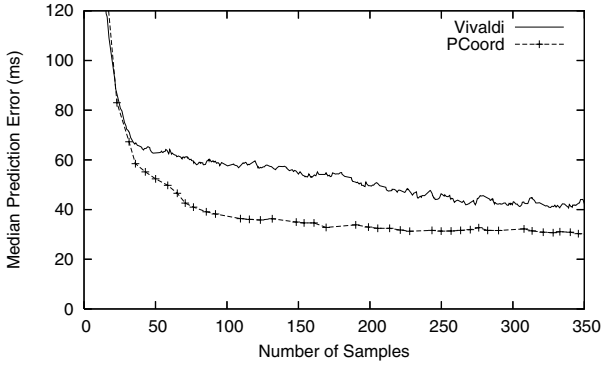
3.2 Convergence Behavior of PCoord and Vivaldi

In this section, we compare PCoord and Vivaldi in terms of the number of samples required for convergence using the King data set under a flash-crowd scenario when all 1740 hosts join simultaneously. Figure 1 plots the median, 95th and 5th percentile error of PCoord and Vivaldi as a function of average number of samples used per host. PCoord converges to 14 to 13 ms median prediction error with approximately 100 samples per host, and to 12 ms error range with more samples; the convergence time is less than 10 seconds, with each host performing approximately 10 coordinate updates using 10 reference points per update. Vivaldi requires over twice the number of samples per host as PCoord to reach the same error range. We have also simulated Vivaldi with $C_c = 1$, in which case, Vivaldi converges faster but to a higher error range.

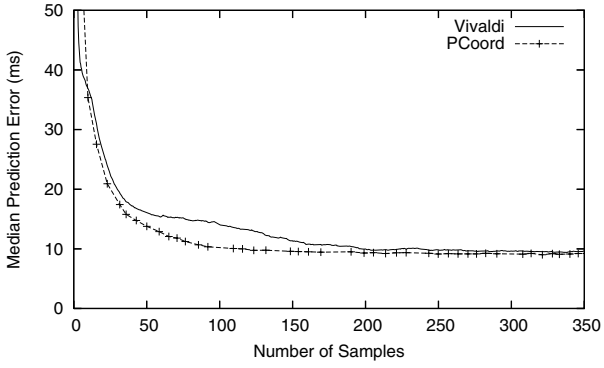
Next, we compare PCoord and Vivaldi in terms of relative error distribution using GNP as the benchmark. The GNP results were generated using the GNP software [12] with 10 fixed landmarks randomly drawn from the King data set. Twenty different randomly-generated landmark configurations were used, and the GNP result with the best performance (with 10% median error) is reported.



(a) Median error



(b) 95th Percentile Error



(c) 5th Percentile Error

Figure 1. Convergence of PCoord (M=10) and Vivaldi (C_c=0.25). King data.

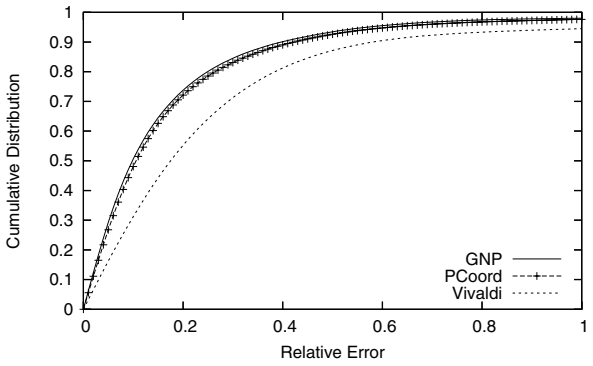


Figure 2. Comparing PCoord, Vivaldi and GNP in terms of relative error after PCoord and Vivaldi used 100 samples per host on average. King, D = 5, Vivaldi C_c = 0.25.

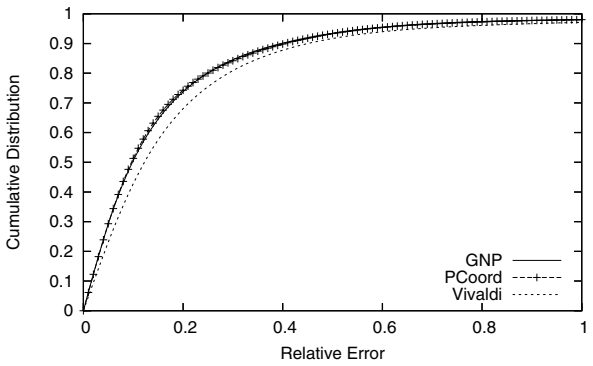


Figure 3. Comparing PCoord, Vivaldi and GNP in terms of relative error after PCoord and Vivaldi used 200 samples per host on average. King, D = 5, Vivaldi C_c = 0.25.

Figures 2 and 3 show the cumulative distribution of relative error after PCoord and Vivaldi hosts have updated their coordinates using on average 100 and 200 samples per host. We note that PCoord's median relative error is 11% after 100 samples, which is comparable to that of the GNP scheme. After 100 samples, Vivaldi's median relative error is about 18%. After 200 samples, PCoord's and Vivaldi's median relative errors are 10% and 12% respectively.

In comparing Vivaldi and PCoord, we note that one advantage of Vivaldi is that each coordinate update is lightweight in terms of the computation cost. PCoord computes coordinates by minimizing a loss function numerically over a batch of samples, and thus incurs more computation overhead. However, we believe that such local computation overhead, at 10 ms per coordinate update on a 150 MHz CPU, is rather modest. For systems with large numbers of

participating hosts, we believe that the number of samples required for the system to converge to low-error coordinates is a more important performance metric.

We have compared PCoord's and Vivaldi's convergence behavior using other data sets. Our results suggest that when the number of hosts in the system is small, it takes PCoord and Vivaldi approximately the same number of samples to converge to low-error coordinates. However, as the number of hosts in the system increases, it takes Vivaldi more number of samples to converge to low-error coordinates in comparison to PCoord. For example, using the RON data set, with 15 hosts in the system, it takes both schemes approximately the same number of samples to converge to a low median error range. Using the PlanetLab data with 127 hosts, PCoord requires 30% less samples to converge than Vivaldi. Using the King data set with 1740 hosts, it takes PCoord half the number of samples required by Vivaldi to reach low-error coordinates.

We have also simulated PCoord in an incremental join scenario, where a node joins after the rest of the system has converged. On average, the median prediction error of a newly joined node can decrease to the 12 ms range within two coordinate updates using 10 reference points per update (i.e., within 20 samples).

3.3 Performance under High Churn

In this section, we examine PCoord's performance under churn, i.e., when the system experiences continuous membership changes as a result of nodes joining and leaving. We examine the following questions. How robust is PCoord under high churn? At what point do we begin to observe significant performance degradation as the join/leave rate increases?

Under the dynamic join and leave mode, each node alternately leaves and re-joins the system throughout the entire simulation period. The time interval a node stays in and out of the system is exponentially distributed with a mean t. Recent studies suggest that the median session duration of hosts in peer-to-peer systems is approximately one hour [19]. We have chosen to use shorter time intervals, and thus higher churn rates, in our simulations in order to examine PCoord's performance under extreme conditions. We have experimented with t equal to 2, 5, 10, 20, 30 and 40 seconds, with a total simulated time of 300 seconds. When a node re-joins the system, its coordinates are re-initialized to the origin.

Figure 4 plots the average prediction accuracy (averaged over time) as a function of the mean host session life time, t. The average prediction accuracy represents the steady-state median prediction error of the system averaged over time using statistics gathered after 60 seconds of simulated time. Figure 4 shows that when join/leave interval t is 10 seconds or greater, the prediction error stays in the range of 12 ms, indicating that the churn has very little effect on the prediction accuracy of PCoord. This suggests that PCoord is likely to do well under the dynamic membership changes of existing peer-to-peer systems, which were reported to have a median session duration on the order of sixty minutes [19].

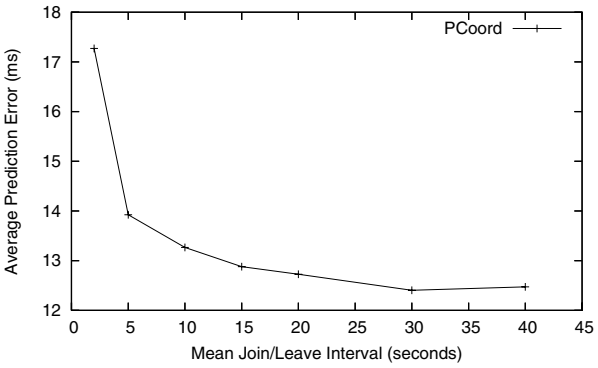


Figure 4. PCoord prediction accuracy as a function of mean join/leave intervals. King, N = 1740, M = 10, and D = 5.

4 Related Work

The IDMaps [5], GNP [13], NPS [11], and King [15] are all architectures for a global distance estimation service. In contrast, PCoord's goal is for peer nodes in an overlay network to estimate their locations relative to other nodes in the same overlay using purely peer-to-peer measurements.

Several schemes [16, 22, 10] allow hosts to use different subsets of landmarks to construct the coordinate system. Other works focus on the geometric models and coordinates computation using global distance measurements [20, 21]. These schemes did not address issues in decentralized coordinates constructions. In [23], an approach that builds *network distance maps* based on hierarchical clustering is proposed. The Mithos [24] system embeds the network into a multi-dimensional space. The focus of their work is on overlay construction and efficient lookup forwarding.

One of the main differences between PIC [2] and PCoord is that in PIC, coordinate update at a node is completely determined by current batch of sampled distances; PIC does not provide mechanisms to retain information learned from previous iterations. This results in a system that reacts too quickly to current measurements.

5 Conclusions

In this paper, we have designed and evaluated a fully-decentralized coordinate system called PCoord. Our sim-

ulation results suggest that, in a 1740 nodes peer-to-peer system, PCoord can converge to a low error configuration within 10 seconds, with each node performing on average 10 coordinate updates using 10 reference points per update. When the system has already converged, a newly joined node can compute low-error coordinates using approximately 20 samples. PCoord is robust under high churn. Our results indicate that, when the host's mean session life time is 10 seconds or longer, dynamic join and leave have limited effect on PCoord's prediction accuracy.

Fast convergence to low-error coordinates is an important property of a large-scale decentralized network coordinate system on the Internet, where flash-crowd appears and demands for services can arise instantaneously. Our results suggest that when the system size is large, PCoord requires less number of samples to converge to low-error coordinates than Vivaldi. Under a simultaneous join scenario with 1740 hosts, PCoord can converge to a 11% median prediction error using half the number of samples required by Vivaldi. As part of our future work, we plan to investigate mechanisms to detect and cope with corrupted and faulty measurements, including those introduced by network routes anomalies and malicious peer nodes.

References

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symp. on Operating Systems Principles (SOSP)*, Banff, Canada, October 2001.

[2] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet coordinates for distance estimation. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, Tokyo, Japan, March 2004.

[3] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Practical, distributed network coordinates. In *Proceedings of HotNets-II*, November 2003.

[4] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of SIGCOMM'04*, August 2004.

[5] P. Francis, S. Jamin, C. Jin, Y. Jin, V. Paxson, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A global Internet host distance estimation service. In *Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.

[6] T. Hansen, J. Otero, T. Mcgregor, and H.-W. Braun. Active measurement data analysis techniques. http://amp.nlanr.net/, 2002.

[7] L. Lehman. *PCoord: A Decentralized Network Coordinate System for Internet Distance Prediction*. PhD thesis, Massachusetts Institute of Technology, 2005.

[8] L. Lehman and S. Lerman. PALM: Predicting Internet network distances using peer-to-peer measurements. Technical report, appeared in Annual Singapore-MIT Alliance Symposium, January 2004.

[9] L. Lehman and S. Lerman. PCoord: Network position estimation using peer-to-peer measurements. In *Proceedings of IEEE International Symposium on Network Computing and Applications (NCA'04)*, pages 15–24, Boston, MA, August 2004.

[10] H. Lim, J. Hou, and C.-H. Choi. Constructing Internet coordinate system based on delay measurement. In *Proceedings of Internet Measurement Conference(IMC'03)*, October 2003.

[11] T. Ng and H. Zhang. A network positioning system for the Internet. In *Proceedings of USENIX 2004 Annual Technical Conference*, pages 141–154, Boston, MA, June 2004.

[12] T. E. Ng. Global Network Positioning (GNP) software. http://www.cs.cmu.edu/∼eugeneng/research/gnp/, 2003.

[13] T. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of INFOCOM*, 2002.

[14] p2psim. http://www.pdos.lcs.mit.edu/p2psim/.

[15] K. P.Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary Internet end hosts. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop(IMW'02)*, November 2002.

[16] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. Lighthouses for scalable distributed location. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, February 2003.

[17] PlanetLab. http://www.planet-lab.org.

[18] Resilient overlay networks. http://nms.lcs.mit.edu/ron/.

[19] S. Saroiu, K. P. Gummadi, and S. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Systems Journal*, 9(2):170–184, August 2003.

[20] Y. Shavitt and T. Tankel. Big-bang simulation for embedding network distances in Euclidean space. In *Proceedings of IEEE INFOCOM'03*, April 2003.

[21] Y. Shavitt and T. Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *Proceedings of IEEE INFOCOM'04*, April 2004.

[22] L. Tang and M. Crovella. Virtual landmarks for the Internet. In *Proceedings of Internet Measurement Conference(IMC'03)*, October 2003.

[23] W. Theilmann and K. Rothermel. Dynamic distance maps of the Internet. In *Proceedings of IEEE INFOCOM'00*, New York, June 2000.

[24] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. In *Proceedings of the First Workshop on Hot Topics in Networks (Hotnets-I)*, Princeton, NJ, October 2002.