PCoord: A Decentralized Network Coordinate System for Internet Distance Prediction

by

Li-wei H. Lehman

Submitted to the Department of Civil and Environmental Engineering in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in the Field of Information Technology

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2005

© Massachusetts Institute of Technology 2005. All rights reserved.

Author .... [illegible] ....
Department of Civil and Environmental Engineering
March 16, 2005

Certified by ....
Steven Lerman
Class of 1922 Professor of Civil & Environmental Engineering
Thesis Supervisor

Accepted by .... [illegible] ....
Andrew Whittle
Chairman, Department Committee on Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
MAY 3 1 2005
LIBRARIES

BARKER

PCoord: A Decentralized Network Coordinate System for Internet Distance Prediction

by

Li-wei H. Lehman

Submitted to the Department of Civil and Environmental Engineering
on March 16, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in the Field of Information Technology

Abstract

Several recently emerged Internet services make use of application-level or overlay networks. Examples of such services include overlay multicast, structured peer-to-peer lookup services, and peer-to-peer file sharing. Many of these services could benefit from enabling participating end hosts to estimate their relative network locations within the overlay. In this thesis, we present PCoord, a peer-to-peer network coordinate system for overlay topology discovery and distance prediction. The goal of PCoord is to allow participating peer nodes in an overlay network to collaboratively construct an accurate geometric model of the overlay network topology in a completely decentralized peer-to-peer fashion. We evaluate the PCoord approach through extensive simulations using both real network measurements and simulated topologies. Our simulation results indicate that PCoord can embed hosts in a low dimensional Euclidean model with a small median prediction error.

Thesis Supervisor: Steven Lerman
Title: Class of 1922 Professor of Civil & Environmental Engineering

Acknowledgments

First and foremost, I would like to express my deepest appreciation to my advisor Professor Steven Lerman for his mentoring and guidance over the years. I have been extremely fortunate to be one of Steve's students. His technical guidance, constant encouragement, support and incredible patience helped me through the most challenging times during graduate school. I would not have completed my Ph.D. program without such a caring advisor. I thank Professor Hari Balakrishnan and Professor Kevin Amaratunga for serving on my committee and for providing invaluable insights to my work.

I am grateful for many people at CECI for their support and encouragement over the years, in particular Jud Harward and Phil Bailey. I would also like to thank Kirky for maintaining the computing and networking resources I use for my research.

I would like to thank my department for awarding me with the Schoettler Fellowship during the first year of my graduate study. It has been a great honor to be part of such a dynamic department, with so many exciting research projects that span so many areas of studies.

I thank Dr. David Tennenhouse and Dr. David Wetherall for initiating me into the field of networking research during the early stage of my graduate study. I would also like to thank several others in the TNS/SDS and the ANA group, including Henry Houh, David Murphy, Shankar Rahul, Suchitra Raman, Vanu Bose, Xiaowei Yang and Dina Katabi for making graduate school a stimulating and enjoyable experience. I would like to thank several others at LCS/CSAIL for their help and support. I thank Frank Dabek and Dave Andersen for answering questions on the King and RON data set. Thanks go to Mor Harchol-Balter for being a valued friend and an excellent teacher.

I am grateful to be involved in 1.00 as a teaching assistant during my graduate career. It has been a rewarding experience to interact with so many wonderful faculty members, TAs, graders, and students in 1.00. I have enjoyed the company from many 1.00 TAs, including Salal Humair, David Zhang, Bharath Krishnan, Shaomin

Wang, Petros Komodromos, Nicolas Aplincourt, Vinay Yadappanavar, Peilei Fan, Anil Gupta, Wen Xiao, and many others.

In addition to people who are directly involved in my academic life, there are also many others that have provided me with great support. In particular, I would like to thank Dr. Yan for his support and guidance over the years. Many other people have provided me with friendship and encouragement, including Ming Dao, Jianjuen Hu, Xiaoyun Guo, Yuchun Guo, and Yajun Fang to just name a few.

My most important acknowledgment goes to my family who have been the source of my resilience and comfort over the years. First and foremost, I would like to thank my husband Brad for his love, support and care during the most challenging times of my graduate career, for being such an incredibly dedicated husband and father, for constantly encouraging me to pursue my graduate study, and for always believing in me, even when I did not. Not only is he my best companion in life, but he also provides insightful research and career advice which is criticial to the successful completion of my Ph.D. study. I especially thank my two children, Ryan and Eric, for being so wonderful and supportive during the long process of my Ph.D. study. Two of them were the best cheer leading team that brought great joy and strength in my life, and kept me motivated, especially during the final stage of my Ph.D. study. Last but not least, I would like to thank my parents and my sisters Jackie and Jill for always being there for me and for all their help and support over the years.

Li-wei H. Lehman

Boston, March 2005

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Several recently emerged Internet services make use of application-level or overlay networks. Examples of such services include distributed content delivery services, overlay multicast [7, 4, 25], structured peer-to-peer lookup services [48, 53, 44, 41], and peer-to-peer file sharing. Topological information about the relative locations of hosts within these overlay networks improve many of these services. To help with the performance of these services, much research has been done to allow end hosts to discover network topology and accurately predict network distances in a scalable and timely fashion. The key challenge is to predict inter-host network distances with as few measurements as possible.

This thesis is about the design and evaluation of a decentralized network coordinate system for Internet distance prediction and location estimation. The idea of a network coordinate system, first proposed by the GNP system [32], is to model the Internet as a D-dimensional geometric space; hosts compute their coordinates in this space to characterize their locations on the Internet. The goal is for each host to derive a mapping of itself in the geometric space using a small set of sampled distances so that the actual inter-host network distances can be estimated as a function of the nodes' geometric distances.

There are many advantages and applications to a geometric model of the network distances. One advantage is that coordinates efficiently summarize inter-host distances: once a node computes its coordinates, it can estimate its distance to any other nodes in the coordinate system without making explicit measurements to those nodes. Instead of storing and communicating $O(N^2)$ distances, N vectors, each with D dimensions, suffice to summarize the topological relationships among hosts, where N is the number of hosts in the system. Potential applications of such a network coordinate system include:

- Peer-to-peer (P2P) file sharing: the coordinate information can be used to enable a host to download a file from the "closest" peer node that has a copy of the file.
- Content distribution: the coordinates can be used by a redirection service that directs clients to the "closest" content server to reduce response latency.
- Efficient logical topology construction in P2P networks and application-level multicast: in many of these applications, each host is logically connected to a small number of other participating peers to form an overlay network. Communication among peers usually follows the logical links, each of which could traverse multiple physical IP hops. Coordinate information can help improve the performance by avoiding logical links over high latency IP hops.

1.2 The Problem and the Challenges

Earlier network coordinate systems rely to some extent on distance measurements to a common set of reference nodes. For example, the Global Network Positioning (GNP) system [32] uses a host's distance measurements to a fixed set of landmarks to compute absolute coordinates to characterize the host's location on the Internet. Using fixed landmarks introduces potential bottlenecks. Additionally, the performance of GNP is sensitive to landmark placement. More recently proposed coordinate-based approaches allow a subset of end hosts to be used as landmarks [37, 49, 26, 31]. However, most of these schemes are not fully decentralized.

In contrast to the earlier landmark-based approaches, the goal of this research is for participating peer nodes in an overlay network to collaboratively construct an accurate geometric model of the overlay network topology in a fully decentralized, peer-to-peer fashion. There are many challenges in designing such a decentralized coordinate system for large-scale Internet applications:

- Scalability. The system must be able to support large-scale Internet distributed applications. Our goal is for each host to estimate its coordinates by probing a small number of other peers.
- Decentralization and fault tolerance. The system should operate without relying on any common reference points, infrastructure support, or global information. Each node must estimate its "global" network position based on a small samples of distances.
- Adapting to dynamic network conditions and membership changes. Once the coordinates have been generated, they must be continuously maintained and updated to reflect changes in network conditions and peers joining and leaving.
- Timeliness and efficiency. One of the main applications of the coordinate system is to improve on end-to-end response time based on the location information. Thus, the coordinates must be computed in a timely fashion for them to be of practical values.

1.3 Approach Overview

In this thesis, we present PCoord, a fully decentralized network coordinate system for overlay topology discovery and distance prediction. In PCoord, there are no specially designated landmark nodes; peers measure latencies to a small number of other peer nodes to estimate their own network locations in the overlay.

Additionally, we present a proof-of-concept distributed coordinate framework named PALM (or Peers as Landmarks). The PALM framework was proposed and evaluated as a proof-of-concept for a decentralized coordinate system.

In this section, we briefly describe both frameworks below. We then describe the network distance measurement and the geometric model we use for our coordinate system, and provide some rationale for our choices.

1.3.1 System Overview

PALM

The focus of the PALM study is not to define a complete system, but rather to investigate whether it is possible to build a network coordinate system in a peer-to-peer fashion, and to help understand issues involved in building distributed network coordinate systems. PALM (or Peers as Landmarks), is a direct extension of the original GNP framework in a decentralized, peer-to-peer environment.

In PALM, any host which has already derived its coordinate can be selected by another peer node as "landmarks". Within the PALM framework, we explore two different peer sampling strategies: RandPalm and Island. In RandPalm, a peer node randomly selects from existing peer nodes to function as its reference points. In Island, each peer node selects its reference points by exploiting the topological information derived based on existing peer nodes' coordinate values. We have simulated PALM using both real network and simulated topologies. Details of PALM can be found in Appendix A.

PCoord

PCoord is a fully decentralized network coordinate system with each host updating its coordinate iteratively to refine the prediction accuracy of its estimated position. 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. PCoord does not require any special bootstrap process – each node goes through an iterative calibration process to refine its coordinates. PCoord has the following features and components:

- A weighted loss function to distinguish between nodes with high and low errors and a "resistance" factor in the loss function that helps to stabilize the convergence and avoid oscillation.

- A threshold-based mechanism to dampen the amount a node moves toward new coordinates based on the confidence associated with the current batch of sampled coordinates and RTTs.
- A message exchange protocol that enables fast discovery of nearby peers.

We evaluate the prediction accuracy and convergence behavior of PCoord under several factors, including peer distance distribution, dimensionality of the geometric space, and the degree of triangle inequality violations in the data set. Through extensive simulations using both real network measurements and simulated topologies, we compare the performance of PCoord with Vivaldi and the original GNP scheme (referred to as the FixedLM scheme from now on).

1.3.2 Network Measurement

There are many different ways to measure network distances among hosts on the Internet. Some of the measurements include: ping-based RTT probing, 10KB TCP probing, bottleneck bandwidth probing, Internet administrative system (AS) hop counts, IP-level router hop counts, and geographic distances [30, 16, 33].

Recent studies show that RTT-based latency measurement not only has the advantage of being relatively easy to obtain on an end-to-end basis, but it also translates well into end-to-end performance such as throughput and response latency [30, 33]. In particular, Ng *et al.* have shown in [30] that round-trip ping time (RTT) can in fact effectively identify peers with high TCP throughput for media file sharing. Their results suggest that short RTTs often translate to high TCP throughput.

Since the goal of PCoord is primarily to provide location information to improve end-to-end application performance, we use round-trip time (RTT) as our distance measurement in this thesis. Our problem is then to find configuration of nodes in a D-dimensional space so that the geometric distances between pairs of hosts match their round-trip latency as closely as possible.

1.3.3 Choice of the Geometric Model

PCoord models the network as a D-dimensional Euclidean space, where the geometric distance function is simply the L_2 norm. We chose the Euclidean space due to its simplicity and accuracy. Other existing research has explored various options for geometric space, including spherical, cylindrical [32], and hyperbolic [47], and Vivaldi's height vectors [10]. It has been shown that the Euclidean space does better than the spherical or the cylindrical model [32, 10]. This is probably due to the fact that Internet latencies are largely dominated by geographic distances, and the routing paths through the Internet do not often "wrap around" the Earth as a spherical (or cylindrical) model would assume [10]. While other models such as the hyperbolic space [47] and the height vector [10] have been shown to model Internet distances more accurately, it is not clear whether their performance improvement justifies the additional complexity.

1.3.4 The Coordinate Computation Method

In PCoord, coordinates computation is cast as a non-linear optimization problem, with each host computing its own coordinates that minimizes the total sum of squared differences between the actual sampled distances and the node's Euclidean distances to those sampled nodes. We use the Simplex Downhill method to solve the optimization problem, since it has been shown to yield good prediction accuracy for Internet distances with modest computation cost [32]. Many other works, some concurrent to ours, have proposed other methods to compute coordinates [10, 46, 37]. Most notably, Vivaldi solves the same minimization problem over a similar sum of squared error function by minimizing the energy in a physical spring network. Neither spring relaxation nor Simplex Downhill guarantees finding the global minimal solution.

While the Simplex algorithm has been criticized for its computational cost, our experience in PCoord suggests that most coordinate updates using Simplex Downhill to minimize a loss function involving 10 reference points can be done in less than 10 milliseconds on a 150 MHz Sun UltraSparc. Since each PCoord host computes its own coordinates, we believe that such computation overhead is very modest.

1.4 Contributions

In this thesis, we have designed a fully-decentralized coordinate system PCoord, and evaluated it using extensive simulations under various real and artificial network topologies. We compared the performance of PCoord with another decentralized network coordinate system Vivaldi, and the original GNP scheme using fixed landmarks.

We have examined PCoord's convergence behavior using simulations in several different scenarios: (1) the simultaneous-join scenario with all nodes joining the coordinate system at approximately the same time, (2) the incremental join scenario, in which we evaluate the number of samples required for a newly joined node to converge to a low prediction error when the rest of the system has already converged, and (3) the high churn scenario, in which the system experiences continuous membership changes with dynamic node join and leave.

Our simulation results indicate that PCoord can achieve competitive prediction accuracy in comparison to the GNP scheme without relying on a fixed set of landmark nodes. Our results also suggest that, though PCoord incurs a higher computation overhead in comparison to Vivaldi, it can converge to a lower prediction error using fewer samples than Vivaldi. Our simulation results suggest that PCoord is robust under high churn when the system experiences continuous membership changes, and can effectively guard against faulty coordinate information.

PCoord fills in one of the missing pieces not addressed in Vivaldi – i.e., how a peer discovers and samples other peers. While Vivaldi's simulations in [10] simply assume that nodes will have access to a list of its nearest peers, we provide an efficient peer discovery mechanism using triangulated distances. We believe PCoord provides a competitive alternative to Vivaldi as a decentralized coordinate system due to the following novel features:

- A weighted loss function to distinguish between nodes with high and low errors and a "resistance" factor in the loss function that helps to stabilize the convergence and avoid oscillation.

- A threshold-based mechanism to dampen the amount a node moves toward new coordinates based on the confidence on the current batch of samples.

- A message exchange protocol that enables fast discovery of nearby peers using triangulated distances.

In addition to PCoord, we have proposed and evaluated a proof-of-concept coordinate system named PALM, and demonstrated the feasibility of a decentralized approach in building network coordinate systems by using distance measurements to *any* subset of hosts. Through simulation-based evaluations, we show that the PALM based approaches have rather different performance characteristics than the fixed landmarks based approach, such as GNP. We believe that many of our findings with respect to the performance characteristics of PALM provide valuable insights for designers of decentralized network coordinate systems or peer-to-peer location systems in general.

1.5 Thesis Overview

The rest of this thesis is organized as follows. In Chapter 2, we outline related work. Chapter 3 describes the PCoord algorithm. We evaluate PCoord and compare its performance with that of Vivaldi using simulations in Chapter 4. Chapter 5 studies the sources of prediction errors of PCoord in more details and examine the impact of triangle inequality on prediction accuracy. Chapter 6 examines PCoord's convergence behavior and error characteristics using variations of a small 15-node topology. Finally, we present our conclusions and ideas for future work in Chapter 7. We present the design and evaluation of the PALM-based schemes in Appendix A.

Chapter 2

Related Work

Much research has been done in Internet topology discovery and distance prediction. Earlier approaches in this area place more emphasis on predictions based on infrastructure support [12, 32], which involves the use and deployment of dedicated nodes in the Internet to provide for the distance prediction service. Later works, such as PALM [24], PCoord [23], Vivaldi [9, 10], and PIC [8], focus more on a distributed approach that work in a peer-to-peer overlay environment. In this chapter, we review existing works in the area of Internet distance prediction, wireless network location estimation, and theoretical work done in metric embeddings. We also discuss the similarities and differences between PCoord and several other decentralized network coordinates systems, such as Vivaldi [10] and PIC [8].

2.1 Internet Distance Prediction and Location Estimation

The IDMaps [12] and GNP [32] are both architectures for a global distance estimation service. Both IDMaps [12] and GNP [32] rely on the deployment of infrastructure nodes. King [36] proposes direct online measurements using the DNS infrastructure to predict network latencies between arbitrary Internet end hosts. NPS [31] proposes a hierarchical network position architecture that enables decentralized coordinate computation. The goal of these systems is mainly to develop public infrastructure that provides distance information between *any* two arbitrary points on the Internet. 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; PCoord predicts network distances using purely peer-to-peer measurements without relying on the infrastructure services.

Several works provide network proximity or location estimates using the distance measurements to a set of well-known landmarks. For example, the GeoPing algorithm [35] uses latency measurements to a set of well-known landmarks to determine end hosts' geographic locations. In [15], Hotz proposes a triangulated heuristic to give a bound on the network distance between any pair of hosts by using their distances to a common set of base nodes. Internet Iso-bar [6] performs clustering on hosts based on the similarity in their distance to a small set of sites. The distances between hosts are estimated using inter- or intra-cluster distances. In CAN [41] and the binning scheme [42], distance measurements to landmarks are used to support proximity routing in a structured peer-to-peer network. The location of an end host in their scheme is characterized by the ordering of landmarks in terms of their distances to the host. These schemes, in contrast to ours, do not attempt to model Internet hosts using absolute coordinates.

In [50], an approach that builds *network distance maps* is proposed. It clusters hosts hierarchically into network regions. Cluster representatives measure and maintain distance information among each other. Two hosts can then estimate their inter-host distance as a function of the distance between their corresponding cluster representatives. In contrast, our work does not require nodes to maintain any cluster structure, which maybe difficult to implement in a peer-to-peer environment with dynamic membership changes.

To avoid the fixed landmark problem in GNP, several schemes [37, 49, 26] have been proposed that allow hosts to use different subsets of landmarks to construct a local coordinate system, which are then transformed to a global coordinate system. For example, the Lighthouse scheme [37] uses multiple local bases to allow a host to determine its coordinates relative to any set of landmark nodes. Virtual Landmarks (VLM) and Internet Coordinate System (ICS) both use principal component analysis (PCA) to extract topological information. The above schemes, however, are not fully decentralized in that they still require distance measurements to a common set of nodes to compute coordinates.

Several other works focus on the modeling and coordinates computation issues. For example, the Big Bang Simulation (BBS) [46] solves the embedding problem by simulating an explosion of particles under a force field. Shavitt and Tankel [47] recently proposed a hyperbolic coordinate space to model the Internet, which achieves better accuracy than the Euclidean embedding. The focus of these two works, however, is on computation methods and geometric models that yield low embedding error assuming global distance measurements are available. It is not clear how they can be applied in a decentralized environment.

The Mithos [51] system also embeds the network into a multi-dimensional space; it uses a spring relaxation technique for coordinate computation. The focus of their work is more on overlay construction and efficient lookup forwarding and less on network distance prediction.

Meridean [52] provides a framework for hosts to lookup their nearest peers in an overlay network. Each Meridian node keeps track of a small, fixed number of other hosts that are organized and maintained as a multi-resolution ring structure with exponentially growing ring radii. A node's query for its nearest peer can then be forwarded along the ring structure, which exponentially reduces the distance to the target at each query hop. In contrast to our approach, Meridian builds many local coordinate systems instead of an absolute coordinate system; their work focuses more on overlay construction and lookups, rather than distance prediction.

2.2 Decentralized Network Coordinate Systems

Similar to our work, Vivaldi [9, 10] is a fully decentralized coordinate system. Compared with Vivaldi, PCoord uses a more aggressive message exchange protocol for fast near peer discovery. Vivaldi piggybacks the latency probes on application-level traffic, and does not address the peer discovery issue. Vivaldi uses a spring relaxation algorithm to solve for the coordinates. Spring relaxation is more efficient than the Simplex algorithm in terms of computation overhead at each node. However, our results suggest that the additional computation time incurred in PCoord due to the Simplex algorithm does not significantly affect its overall system convergence time. One major difference between PCoord and Vivaldi is that in PCoord a node computes its coordinate by optimizing a loss function over a *batch* of samples, whereas in Vivaldi a node adjusts its position to minimize the error one sample at a time. We believe that PCoord's batch-based approach gives it a faster system convergence time than Vivaldi in terms of number of samples needed for convergence.

Similar to our work, the PIC [8] system also proposes a distributed coordinate system that uses the Simplex algorithm for coordinate computation. One of the main differences is that PIC requires a set of peer nodes to compute the bootstrap coordinates. In contrast, PCoord is fully decentralized and does not require a set of peer nodes to carry out the bootstrap process. Additionally, in PIC, coordinates update at a node is completely determined by current batch of sampled distances; it does not provide mechanism to retain information learned from previous iterations. This could result in a system that reacts too quickly to current measurements. Finally, PIC uses a different strategy to locate nearby peer nodes: PIC uses estimated coordinates to locate near peers, whereas PCoord uses triangulated distances to infer near peers. In PIC [8], each peer node performs a greedy walk to locate a nearby peer using the node's current coordinates to guide the walk. Although the strategy has been shown to work well in an MSPastry framework where each node points to a mix of near and far away nodes, it is unclear whether the strategy would be effective in an unstructured overlay, where a node's logical neighbors may not have such a convenient mix of near and far nodes. PCoord's near peer discovery mechanism does not make assumptions about the types of logical connections maintained at each node.

2.3 Location Systems in Wireless Networks

There has been a large number of works in wireless, sensor network location systems [40, 11, 5, 1, 14]. In a sensor network environment, the location problem is usually one in which a small fraction of network nodes have known locations, while the rest of the nodes must estimate their locations using distance measurements to these reference nodes. Many of the applications focus on estimating physical locations of devices. Proximity measurements among nodes are measured using either received signal strength or time-of-arrival between themselves and neighboring nodes. A fundamental difference between the wireless and the Internet environments is that, the distances in the former are largely dominated by geographic proximity. Another difference is that the distance measurements in a wireless environment are limited by the transmission range of the reference points, which is a problem not encountered in the Internet environment.

2.4 Metric Embeddings

Embedding of arbitrary distance matrices into geometric spaces is a problem faced in many applications, ranging from computer vision to protein sequence analysis in bioinformatics. Much theoretical work has been done on metric embeddings [28, 29, 17, 27, 21, 3, 22, 18]. The emphasis is usually on embeddings of finite metric spaces into normed spaces with the least possible distortion, which is a measure of how much the geometric distance differs from the actual distance.

There are fundamental differences between the theoretical approaches to the embedding problem and the Internet coordinate-based approaches. Works in metric embeddings generally assume that distance matrices satisfy the triangle inequality, which is often violated in Internet distances. Perhaps a more important difference is that the goal of the metric embedding algorithms is to generate embeddings with least possible distortions assuming knowledge of global pair-wise distances; in a decentralized network coordinate system, the goal is to estimate a node's position in the geometric space using a small set of sampled distances.

Work by Kleinberg *et al.* [19] is perhaps the first to present a theoretical framework that proves performance guarantees for Internet-based network coordinate systems. While general metrics cannot be embedded in Euclidean spaces with constant distortions, by introducing the notion of a *slack*, Kleinberg *et al.* is able to prove performance guarantees for the Internet coordinate-based embedding algorithms by allowing a certain fraction of all distances to be arbitrarily distorted.

Kleinberg's work is complementary to ours in the sense that, the extensive simulation results presented in this thesis can be used to support the theoretical framework presented in [19]. Additionally, our work presents empirical data that quantifies the impact of the degree of triangle inequality violations in Internet distances on prediction accuracy of the embedding, which is an issue not addressed in [19].

Chapter 3

The PCoord Algorithm

In this chapter, we present PCoord, a fully decentralized network coordinate system for overlay topology discovery and distance prediction. In PCoord, there are no specially designated landmark nodes; peers measure latencies to a small number of other peer nodes to estimate their own network locations in the overlay. Nodes initialize their coordinates to the origin, and go through an iterative calibration process to refine their coordinates. In order to distinguish them from the GNP "landmarks", which are fixed nodes embedded in the network, we call the set of peers selected by a PCoord host for computing their coordinates reference points (RPs).

PCoord has the following features and components:

- A weighted loss function that allows sampled coordinates with higher prediction accuracy to have a higher weight in the loss function.
- An additional weighted "resistance" factor in the loss function that helps to stabilize the convergence process.
- A threshold-based mechanism to dampen 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.
- A message exchange protocol that enables fast discovery of nearby peers using triangulated distances, and a peer sampling strategy that includes both near

peers and randomly sampled nodes in each calibration step.

Our simulation results suggest that calibrating coordinates using samples that span a wide range of nodes allows the algorithm to converge faster than always calibrating coordinates with the same set of nodes at each iteration. In order to allow peers to calibrate their coordinates with a large set of peer nodes, PCoord uses a gossip-based protocol to enable peer discovery. At each iteration, PCoord hosts discover other peers in the same overlay by exchanging a list of peers they know of with their reference points.

To improve the coordinates' accuracy in modeling short distances, each PCoord peer includes a mixture of near and far nodes in its reference set at each coordinate update. PCoord includes a message exchange protocol that enables fast discovery of nearby peers using triangulated distances.

In this chapter, we first present two PCoord based algorithms: a simple version using an unweighted loss function for coordinate update, and the actual PCoord algorithm with a weighted loss function and other additional mechanisms to ensure convergence to a lower prediction error. Both versions of the PCoord algorithms use the same message exchange protocol for fast discovery of nearby peers. We present the message exchange protocol last to complete the algorithm description.

3.1 A Simple Algorithm

In PCoord, each node performs continuous update on its coordinate. Each of the coordinate update consists of three phases: (1) the sampling and information exchange phase in which a node selects M reference points, gathers distance measurements and coordinates, and exchanges peer list information with those M reference points, (2) the coordinate update phase, in which a new coordinate is computed to minimize a loss function defined in terms of those M reference points, and (3) the near peer probing phase in which each host refines its search for its nearby peers by probing other hosts based on their triangulated distances.

We first summarize the notations below and then present the algorithm in more detail. In the algorithm description, i indicates the node which is running the procedure.

M = Number of reference points for each coordinate update

R = Peer list with known RTT

T = Triangulated peer list

$P = R \cup T$

Y = Set of peers selected as reference points

c_i = Coordinates of host i

C_Y = Set of coordinates of hosts in Y

d_{ij} = RTT between i and j

c_{origin} = Coordinates at the origin of the D-dimensional geometric space

The simple form of the PCoord algorithm is as follows:

```
//$i$ is the node that is running the procedure
**SimplePCoord() {**
    $c_i = c_{origin}$
    $R = NIL$
    $T = NIL$
    while (in the system) {
        //SamplePeers() method selects and returns $M$ samples from peer list $R$ and $T$
        $Samples$ = SamplePeers($R$, $T$)
        $c_i^{new}$ = MinimizeError($Samples$, $c_i$)
        $c_i = c_i^{new}$
        ProbeNearNeighbors($R$, $T$)
    } //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 latency 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. More specifically, let c_i be the coordinates currently assigned to node i, and c_i^{new} be the new coordinates node i wishes to solve for using a new batch of samples. Let d_{ij} be the measured round trip latency between nodes i and j. The computation of the coordinates for node i then involves finding c_i^{new} that minimizes the following loss function.

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

The global minimization problem can be approximately solved using many generic multi-dimensional minimization algorithms, such as the Simplex Downhill method, which we use in this thesis. The Simplex Downhill method solves the minimization problem numerically by forming a simplex based on an initial guessed solution. Instead of the normal procedure of using randomly generated values as initial guess, we use the current coordinate c_i to form the initial simplex used to solve for coordinates c_i^{new}.

Vivaldi uses a more computationally efficient method to solve the minimization problem by simulating the process of minimizing the potential energy of a network of physical springs. Neither Simplex Downhill nor the spring simulation is guaranteed to find the global minimum.

3.2 The PCoord Algorithm

There are several potential problems with the above simple version of the algorithm. One problem is that it does not distinguish between nodes with coordinates that have different prediction accuracy – it trusts the coordinates of newly joined nodes as much as old coordinates with high prediction accuracy. 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 prediction accuracy of each reference point's coordinates. The weight is computed based on the relative prediction accuracy of each reference point so that the nodes with more accurate coordinates will have more influence on the solution than the less accurate ones.

Another problem is that the algorithm determines the new coordinate entirely based on measurements from the current batch of reference points. There is no mechanism for the coordinates generated using previous samples to cast any vote on the the position of the new coordinate in the current update. The simple scheme thus tends to react too quickly based on the measurements of the current batch of reference points, and leads to potential oscillation. This is in particular a problem when the sample size of each batch is small relative to the dimensionality of the geometric space.

In order to reduce oscillation, we introduce an additional "resistance" factor into the loss function so that a node with highly 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 resistance term is initialized to zero. Each node continuously updates the confidence index of its own coordinates as a function of the weighted moving average of its current and past prediction error.

3.2.1 Weighted Error Function with a Resistance Factor

More precisely, the weighted loss function with the resistance factor is as follows. 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.

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

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

As described earlier, the weight is computed as a relative confidence on the prediction accuracy of each node's coordinates. The weighted loss function requires each node to obtain a confidence index on the accuracy of its current coordinates. For this purpose, each node maintains a weighted moving average of its relative prediction error.

Update Prediction Error

The procedure for maintaining the weighted moving average of the relative prediction error is described in this section. This procedure is invoked at each coordinate update iteration after the sampling phase. Vivaldi nodes also use a similar procedure to maintain a weighted moving average of prediction error, but use it in a different way than PCoord in coordinate computation.

e_i^p = weighted moving average of relative prediction error at node i

e_{ij}^p = relative prediction error for distance between node i and j

α = weight for computing weighted moving average of prediction error

```
//$i$ is this node
UpdatePredictionError($Samples$) {
    for each $j$ in $Samples$ {
        $e_{ij}^p = \frac{|||c_i - c_j|| - d_{ij}|}{d_{ij}}$
        $w = \frac{(e_i^p)^2}{(e_i^p)^2 + (e_j^p)^2}$
        $e_i^{newp} = e_{ij}^p * w + e_i^p * (1 - w)$
        $e_i^p = \alpha * e_i^p + (1 - \alpha) * e_i^{newp}$
        $e_i^p = MIN(1, e_i^p)$
    } //end for
}
```

The following pseudocode fragment describes how the weight is assigned based on the relative prediction error. τ is a small constant added to one to define e_{TOP}^p for weight computation. When $\tau = 0$, nodes with relative prediction error of one have zero weight in the loss function. Setting $\tau \geq 0$ allows nodes with relative error of one to have a non-zero weight. In this study, τ is set to 0.05.

Weight Assignment

```
    $\tau$ = a small constant added to 1 for weight computation, $\tau \geq 0$
    $e_{TOP}^p = 1 + \tau$
    //assign weight to each sample in $Samples$, which includes the "resistance" term
    for each node $j$ in $Samples$ {
        $a_j = e_{TOP}^p - e_j^p$
        $w_j = \frac{a_j^2}{\sum_{k \in Samples} a_k^2}$
    }
```

3.2.2 Adjusting Amount of Coordinate Updates Based on Goodness-of-Fit

The idea of an Internet distance prediction scheme is based on the assumption that measurements of distances to a few nodes on the Internet can be used to infer distances to a significant fraction of other nodes on the Internet with some small error. To achieve good prediction accuracy, ideally a node should position itself in the geometric space using sampled distances that are "representative" of its distance relationships to all other nodes in the Internet. Using samples with "non-representative" distances to predict the position will likely lead to coordinates with high prediction error. For example, inter-host distances that violate the triangle inequality constraint either due to measurement error or Internet routing will likely bias the newly generated coordinates toward the "un-representative" distances.

The weighted loss function described above helps to reduce the negative effect of reference points with high prediction error. However, the prediction error is only an estimate of the overall prediction accuracy of a node's coordinates; it does not necessarily reflect whether a particular pair-wise distance between nodes i and j serves as a good sample to predict the position of the two nodes. In particular, a node i which has coordinates with high overall prediction accuracy may be connected to another node j using a direct high-speed Internet link; if j were to use distance to i as a sample in the above weighted loss function, j will be generate coordinates much closer to i than it should, since i's high confidence index will put a heavy weight on its corresponding term in the loss function.

In general, it is difficult to estimate the "predictive power" of a particular sample on its own merit. The effectiveness of a sampled distance between node i and j can be evaluated in the context of how well it matches the distance relationships between node i and j to other nodes.

In this section, we introduce a mechanism that allows a node to adjust how much it should move its coordinates in response to a particular batch of samples based on the goodness-of-fit index. The goodness-of-fit is a confidence measurement associated with an entire batch of samples. The idea is that a batch of samples containing "un-representative" distances 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 assigns a weight to each batch of samples as a function of the ratio between the average and current fit error, and then decides how much it should react to the current batch of samples based on the weight. 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 coordinate by a factor ρ which is the ratio between the average and current fit error. We call this weight asssociated with each batch of samples, the "batch" weight, to distinguish from the previously described "sample" weight associated with each sample in the loss function.

Let e_i^f be the weighted moving average of fit error of node i, and e_i^{newf} be the fit error after minimizing the weighted loss function using the new batch of samples. When the new fit error e_i^{newf} exceeds the average e_i^f, PCoord only moves ρ fraction of the way toward the new coordinates, where $\rho = MIN(e_i^f / e_i^{newf}, 1)$, and $0 \leq \rho \leq 1$. The following pseudocode describes the procedure in more details.

e_i^f = Weighted average of fit error of node i

e_i^{newf} = The fit error of the new batch of data

β = Weight for computing weighted moving average of e_i^f

ρ = Fraction of movement toward the new coordinates, $0 \leq \rho \leq 1$

Update Fit Error

The procedure for maintaining the weighted moving average of the fit error is as follows. It is invoked at each iteration after the coordinate update phase.

```
//$i$ is this node
**UpdateFitError(** $e_i^{newf}$ **) {**
    //update the weighted moving average of fit error
    $e_i^f = \beta * e_i^f + (1 - \beta) * e_i^{newf}$
}
```

Compute Fraction of Coordinate Movement

```
    //$e_i^{newf}$ is the residual error after the minimization step above
    $e_i^{newf} = \sum_{k \in Samples} w_k (d_{ik} - ||c_i^{new} - c_k||)^2$
    //figure out how much to move toward new coordinates based on goodness of fit
```

```
$\rho = MIN(\frac{e_i^f}{e_i^{newf}}, 1)$
//now, only move $\rho$ fraction of the way toward the new solution
$c_i^{new} = c_i + (\rho * (c_i^{new} - c_i))$
```

3.3 The Peer Discovery and Selection Process

In this section, we present the first and third phase of the PCoord algorithm, namely the sampling and the near peer search phases.

In PCoord, each host selects its M reference points independently of other hosts. These M peers can be any other peers in the system. We assume that each peer node is initially connected to K logical neighbors in the overlay. In the initial iteration, a host selects its reference points randomly from its K logical overlay neighbors. In order to allow peers to calibrate their coordinates with a large set of peer nodes, PCoord uses a gossip-based protocol to enable peer discovery. At each iteration, PCoord hosts discover other peers in the same overlay by exchanging a list of peers they know of with their reference points. Each node i maintains two lists: R, a list of peers whose RTTs to i are known to i, and T, a list of peers which i has triangulated distances for so far. To improve the coordinates' accuracy in modeling short distances, each PCoord peer includes a mixture of near and far nodes: half of the reference points are the nodes with shortest measured RTTs in a host's R list, and the other half are randomly selected from the combined R and T lists.

SamplePeer

```
//select $M$ samples from peer list $R$ and $T$
SamplePeer($R$, $T$) {
    if ($R == NIL$) {
        $Samples$ = random $M$ nodes from default logical neighbors
    } else {
        $Y1$ = $numNN$ peers with shortest RTT to $i$ in $R$
        $Y2$ = Randomly select $(M - numNN)$ peers from $R \cup T$
        $Samples = Y1 \cup Y2$
    }
//Ping for RTTs and exchange peer list $R$ with reference points
    for each $j$ in $Samples$ {
        ping $j$ to get $d_{ij}$
        Send$_{dest=j}$ $(d_{ij}, R_i)$
        //for the Simple PCoord algorithm, $j$ does not send $e_j^p$
        Receive$_{sender=j}$ $(c_j, R_j, e_j^p)$
        $R$.add($j$)
        $T$.add($R_j$)
    } //end for each $j$
    return $Samples$
}
```

Next, we describe PCoord's algorithm in discovering nearest peers based on triangulated distances. A peer node i can compute its triangulated distance to another peer j if they both have measured latency to a common node k. In particular, the distance between i and j is lower bounded by $L = |d_{ik} - d_{jk}|$ and upper bounded by $U = d_{ik} + d_{jk}$. There are three ways to estimate two peers' triangulated distance: upper bound U, lower bound L, and their average A $(= \frac{L+U}{2})$ [32]. A peer node periodically probes nodes in its triangulated peer list T to discover its nearest peer node. In particular, it probes the peers with minimum triangulated distances in its peer list T. The probed nodes and their corresponding RTTs are then stored in the peer list R. The following is the psuedocode used by each peer node to update its peer list R at each iteration to find its nearest peer nodes.

ProbeNearNeighbors

ProbeNearNeighbors(R, T) {

```
    $j_1 = T$.remove(peer with min upper bound $U$)
    $j_2 = T$.remove(peer with min lower bound $L$)
    $j_3 = T$.remove(peer with min average $A$)
    ping($j_1$, $j_2$, $j_3$)
    R.add($j_1$, $j_2$, $j_3$)
}
```

The ping(j_1, j_2, j_3) operation involves measuring the round-trip ping times to nodes j_1, j_2 and j_3.

3.3.1 Summary: the PCoord Algorithm

In this section, we pull the above pieces together and summarize the PCoord pseudocode below, which incorporates fragments of pseudocode from the three mechanisms described above: (1) weighted loss function (2) a resistance factor in the loss function (3) adjusting amount of coordinates updates based on goodness-of-fit of a batch of samples. We will describe how to set some of the following parameters in Chapter 4.

M =	Number of reference points for each coordinate update
$numNN$ =	Number of nearest neighbors in a sample batch
α =	weight for computing weighted moving average of prediction error
τ =	a small positive constant added to 1 for weight computation
β =	weight for computing weighted moving average of fit error

```
//$i$ is this node
**PCoord() {**
    $c_i = c_{origin}$
    $R = NIL$
    $T = NIL$
    while (in the system) {
        //select $M$ samples from peer list $R$ and $T$
        $Samples$ = SamplePeers($R$, $T$)
        UpdatePredictionError($Samples$)
        //add node $i$'s own coordinates to the samples
        $Samples = Samples$.add($i$)
        $(c_i^{new}, e_i^{newf})$ = MinimizeWeightedError($Samples$, $c_i$)
        $c_i = c_i^{new}$
        UpdateFitError( $e_i^{newf}$ )
        ProbeNearNeighbors($R$, $T$)
    }
}


**MinimizeWeightedError** ($Samples$, $c_{guess}$ ) {
    $e^p_{TOP} = 1 + \tau$
    for each node $k$ in $Samples$ {
        //assign weight to each node in $Samples$
        $a_k = e^p_{TOP} - e^p_k$
        $w_k = \frac{a_k^2}{\sum_{j \in Samples} a_j^2}$
    } //end for

    //now find new coordinate to minimize weighted sum of squared error
    find $c_i^{new}$ that minimizes
        $\sum_{k \in Samples} w_k(d_{ik} - ||c_i^{new} - c_k||)^2$

    //$e_i^{newf}$ is the residual error found after the minimization step above
    $e_i^{newf} = \sum_{k \in Samples} w_k(d_{ik} - ||c_i^{new} - c_k||)^2$
    //figure out how much to move toward new coordinates based on goodness-of-fit
    $\rho = MIN(\frac{e_i^f}{e_i^{newf}}, 1)$
    //now, only move $\rho$ fraction of the way toward the new solution
    $c_i^{new} = c_i + (\rho * (c_i^{new} - c_i))$
    return ($c_i^{new}$, $e_i^{newf}$ )
} //end MinimizeWeightedError
```

The SamplePeer() and ProbeNearNeighbor() procedures are described in section 3.3. They are the same procedures used by the Simple algorithm, with the exception that, in PCoord, the SamplePeer() phase in the PCoord scheme requires peers to exchange their relative prediction error for the purpose of weight computation.

Chapter 4

Evaluation of PCoord

4.1 Evaluation Methodology

We evaluate the PCoord approach extensively through simulations using both real network measurements and simulated topologies. We compare the performance of PCoord with the Vivaldi scheme in terms of pairwise distance prediction accuracy.

4.1.1 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. More precisely, the link error between node i and j is $|\|c_i - c_j\| - d_{ij}|$, where d_{ij} is the measured RTT between i and j, and c_i and c_j are the assigned coordinates of i and j respectively. Following the conventions in Vivaldi [10], 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.

The *directional prediction error (DPE)* of a link between i and j is simply the difference between the predicted RTT and the actual RTT, or $(\|c_i - c_j\| - d_{ij})$. Thus, a positive DPE value indicates an over-prediction, and a negative value indicates an under-prediction of the actual distance.

We also use the *prediction error ratio (PER)* as our performance metric. The

Table 4.1: Latency Data Statistics (in ms)

Data Set	N	Mean	Median	Std	Min	Max
AMP	104	60.12	45	48.48	1	744
PlanetLab	127	108.18	95.79	74.30	0.069	382.84
King	1740	181.74	158.53	132.35	1.01	799.99

error ratio of a link is defined as $\frac{|\|c_i - c_j\| - d_{ij}|}{d_{ij}}$. The *directional prediction error ratio (DPER)* is defined as $\frac{\|c_i - c_j\| - d_{ij}}{d_{ij}}$.

Finally, we use the absolute relative error (RE) as our performance metric when comparing with the GNP scheme. For each pair of nodes, their absolute relative error is defined as $\frac{|\|c_i - c_j\| - d_{ij}|}{MIN(d_{ij}, \|c_i - c_j\|)}$.

4.1.2 Data Collection

We evaluate our scheme using three data sets containing real network measurements.

- We use the AMP [13] data set on July 16, 2002 (110 nodes) and January 30, 2003 (104 nodes), which measure the round-trip ping time among 110 and 104 nodes respectively.

- We use the PlanetLab [38] all-pairs-ping data set collected on May 10, 2004. After postprocessing to eliminate missing data, we derived end-to-end latency data among 127 nodes.

- We use the King data set from Vivaldi [10], which involves the round-trip latency among 1740 Internet DNS servers.

Table 4.1 lists the mean and median RTT for each data set. Figure 4-1 plots the RTT distribution of the King and PlanetLab data sets. Figure 4-2 plots the cumulative RTT distribution of the data set. We present only the results from the King and PlanetLab data sets.

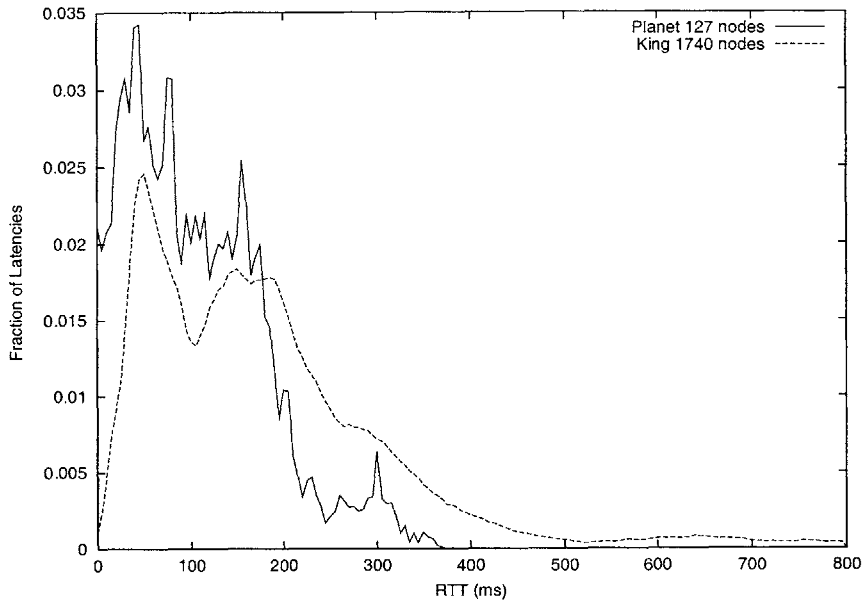


Figure 4-1: PlanetLab (127 nodes) and King (1740 nodes)

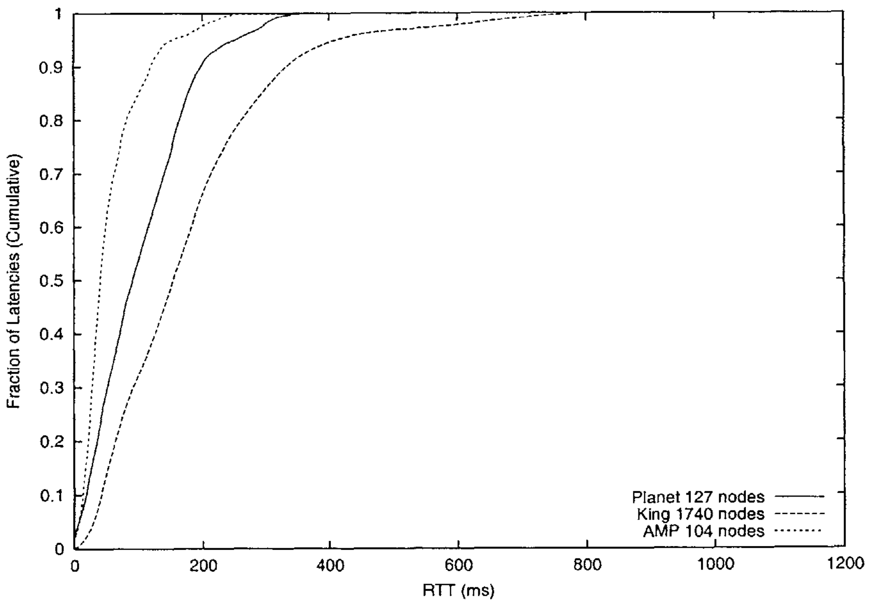


Figure 4-2: Cumulative RTT Distribution

4.1.3 Simulation Setup

We have simulated the execution of PCoord using p2psim [34], an event-driven, packet-level network simulator. We measure the processing cost of the simplex downhill operation on a Sun UltraSparc (with 150 MHz CPU and 4096 Megabytes of memory), and use the measured median processing time in our PCoord simulation. We ran the PCoord coordinate update procedure and measure the time it takes to run the simplex downhill minimization using 10 reference points at each update process. A total of 34,800 measurements (1,740 nodes running the coordinates updates for 20 iterations) were obtained. We repeated the measurements using 20 and 30 reference points. The median processing time is 10 ms per coordinate update when 10 reference points are used, and 20 ms when 20 and 30 reference points are used in each update. Our results are consistent with the measured Simplex algorithm processing time in GNP [32], which reported that computing an ordinary host's coordinates using 15 landmarks takes on the order of ten milliseconds on a 866 MHz Pentium III. The landmark operation, which minimizes loss function involving pair-wise distances among landmarks, takes on the order of a second [32]. PCoord, however, does not perform the more expensive landmark operation step.

In the simulation, the coordinate update of each PCoord node is a two-step process. First, a node pings a set of peers using asynchronous (non-blocking) communication to gather the RTTs and current coordinates of its reference peer nodes. After the node hears back from all the reference peers, it proceeds to the next step to compute its updated coordinates using the RTTs and coordinates gathered in its current period; it also performs the peer exchange protocol with each of the reference peers, again using asynchronous (non-blocking) communication. After the node has completed the second step (and the simulated clock is advanced by the maximum time it takes to update coordinate and time it takes to complete the message exchange phase), it proceeds to the next coordinate update iteration. Each node proceeds in its coordinate update independent of other nodes' update progress.

Our simulation assumes that the latencies among hosts are fixed throughout the simulation period, and we do not consider packet loss. In practice, a node would need to implement a timeout mechanism to deal with packet loss.

4.1.4 PCoord Parameter Settings

We ran the PCoord simulations with various sample batch size M to explore its effect on convergence and prediction accuracy. The rest of the parameters are set as follows.

$numNN = \frac{M}{2}$

$\alpha = 0.95$

$\tau = 0.05$

$\beta = 0.6$

We set the number of nearest neighbors in a sample batch to be half the sample batch size in order to generate a sample set with a mixture of both near and far nodes. The parameter α is the weight ($0 \leq \alpha \leq 1$) used for computing the weighted moving average of prediction error in the weighted loss mechanism. Our results are generated with α of 0.95.

The parameter β is the weight ($0 \leq \beta \leq 1$) used for computing weighted moving average of fit error in the damping mechanism. The fit errors of a new node tend to be the highest in the initial iterations of coordinate updates, and drop to significantly lower values after a small number of iterations. Setting β to a high value will cause the weighted moving average to put more emphasis on past fit errors rather than the more recent ones. As a result, a higher β implies a longer period of time (from when the node first joins) before the damping mechanism takes effect, since it takes more iterations for the initially high weighted moving average to catch up to the lower instantaneous fit error. An extreme example is that when β is 1, the weighted moving average e_i^f will always be the initial fit error, and thus the damping mechanism will likely never take effect.

A low value of β places more weight on the more recent fit errors. An extreme example is that when β is zero, the damping mechanism uses the instantaneous fit error from the previous iteration as the threshold to determine whether to trigger the damping mechanism in the current iteration.

We have experimented with PCoord using β in a range of values between 0.2 to 0.9 using the PlanetLab and King data set. Our results suggest that different values of β in the above range do not change the overall system prediction accuracy significantly. At steady state, the differences in median prediction error when using different β values are less than 2 ms; a β of less than 0.7 performs slightly better. The simulation results presented in this thesis is when β is set to 0.6.

At each coordinate update, a peer includes its nearest peer discovered so far (i.e., the peer with minimum RTT to itself in the peer list R) in its reference point set. To refine its search for its nearest neighbor, each peer probes peers with shortest triangulated distances in its T list at each calibration phase. Results presented here are generated by having each peer probe six peers from its T list in each calibration phase.

The default dimension of the geometric space used is five for the King data set and three for the PlanetLab data set unless otherwise noted. We do not limit the size of peer list T and R stored at each peer. However, the size of the peer list that peers exchange is limited to five for the PlanetLab data and thirty for the King data set. We use different parameter settings for King and PlanetLab in order to compare PCoord's performance with that of Vivaldi's under different parameter settings that reflect different cost/performance tradeoffs.

Simplex Downhill Settings

Our implementation of the Simplex algorithm is adapted from the GNP software distribution, which is in turn based on the code in *Numerical Recipes in C* [39]. In order to obtain high quality solutions, the Simplex algorithm usually restarts the minimization routine after it claims to have found a solution. For each restart, it uses the claimed minimum from the previous run as the initial guess for the next run. We use *restart* to indicate the number of times the minimization procedure is restarted for each coordinate update step. Within each restart, the terminating criteria are usually specified by some tolerance $ftol$ and some threshold $NMAX$, which denotes the maximum allowed function evaluations.

In GNP, the landmarks' coordinates were generated after repeating the minimization procedure for 300 iterations, and the normal hosts' coordinates were computed after repeating the procedure for 30 iterations [32]. The GNP [32] authors reported that in general 3 iterations are sufficient to obtain a fairly robust estimate. The default $NMAX$ value in the GNP software distribution is 500,000.

In PCoord, the Simplex algorithm is run in a "light-weight" mode. In particular, we skip the restart step ($restart = 0$) in the Simplex algorithm and limit the maximum function evaluations to 1000 ($NMAX = 1000$), and achieve similar prediction accuracy in comparison to when using a more heavy-weight parameter settings, for example, $restart = 3$ and $NMAX = 500,000$. The PCoord results presented in this thesis are generated using the light-weight parameter setting described above. More specifically, within each coordinate update step, at most 1000 evaluations of the objective function are performed, and we do not restart of the minimization procedure within each coordinate update step.

We believe PCoord is able to perform well using the light-weight Simplex procedure because PCoord itself maps the coordinates iteratively; at each coordinate update, PCoord uses the "optimal" solution found in the previous iteration as its initial guess for the minimization procedure in the next iteration. In some ways, each coordinate update in PCoord is a way to perform the "restart". Unlike the restart in the Simplex algorithm, however, each PCoord coordinate update attempts to optimize the loss function using a different batch of samples each time.

4.2 Vivaldi

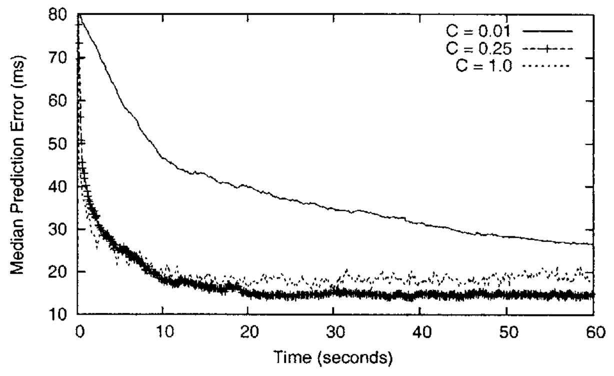
Vivaldi does not specify any peer discovery mechanism. The Vivaldi work [10] uses various peer configurations for evaluations. We describe three of them here as a basis for comparison with PCoord. The first two schemes assume that a node has a fixed set of logical neighbors throughout the simulation period. Each time a node selects a peer reference node from its fixed logical neighbor set to update its own coordinates. The third scheme, Random Global, assumes that each node knows the identities of all other nodes in the system; at each coordinate update, a node randomly selects a peer from the global population for coordinate update.

A more precise definition for each of these three schemes is as follows:

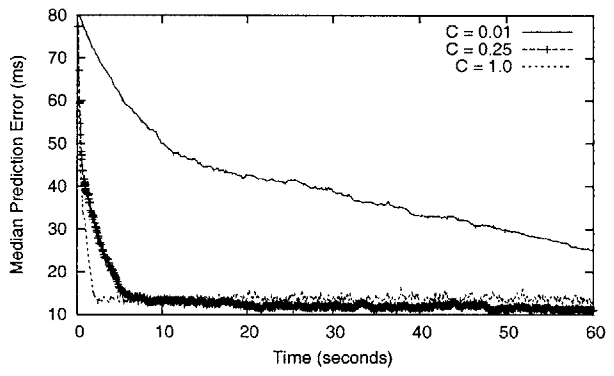
- Random-K Neighbors: each node is initialized with K logical neighbors randomly selected from the global peer population. A peer randomly selects from its K logical neighbors to perform coordinate updates each time.
- Half-Near-K Neighbors: each node is initialized with K logical neighbors, half of which are the node's actual nearest neighbors and the other half are randomly selected from the global peer population. A peer randomly selects from among its K logical neighbors to perform coordinate updates.
- Random Global: there is no notion of logical neighbors in this sampling setting; a peer randomly selects from the global pool of peers each time it updates its coordinates. Each node is assumed to have access to a list of all other nodes in the system.

In this section, we present Vivaldi's performance generated using simulation code in the p2psim [34]. All Vivaldi simulations presented in this work use the adaptive time step mechanism described in [10]. The adaptive time step in Vivaldi is used to adjust how much a node should move its own coordinates in response to a sampled RTT and the corresponding peer's coordinates. A constant C_c ($0 \leq C_c \leq 1$) is used to control how much a node reacts to each new sample.

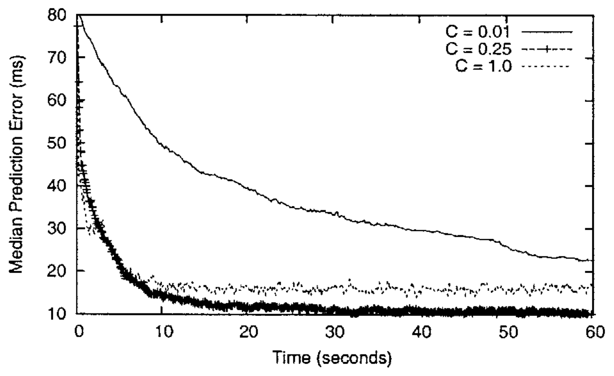
Figure 4-3 presents the effect of time step constant C_c on the rate of convergence under the three different configurations described above. The number of logical neighbors is sixteen for the Random-K and Half-Near-K configurations. The plot shows the median prediction error as a function of time. We note that a C_c value of 0.25 yields both quick error reduction and low oscillation for all three different sampling strategies. When $C_c = 1$, the system converges faster in some cases but to a higher median error than when it is 0.25. This is consistent with the results reported in [10].



(a) Random-K Neighbors



(b) Half-Near-K Neighbors



(c) Random Global

Figure 4-3: Vivaldi timestep effect. PlanetLab, $N = 127$, $K = 16$, $D = 3$.

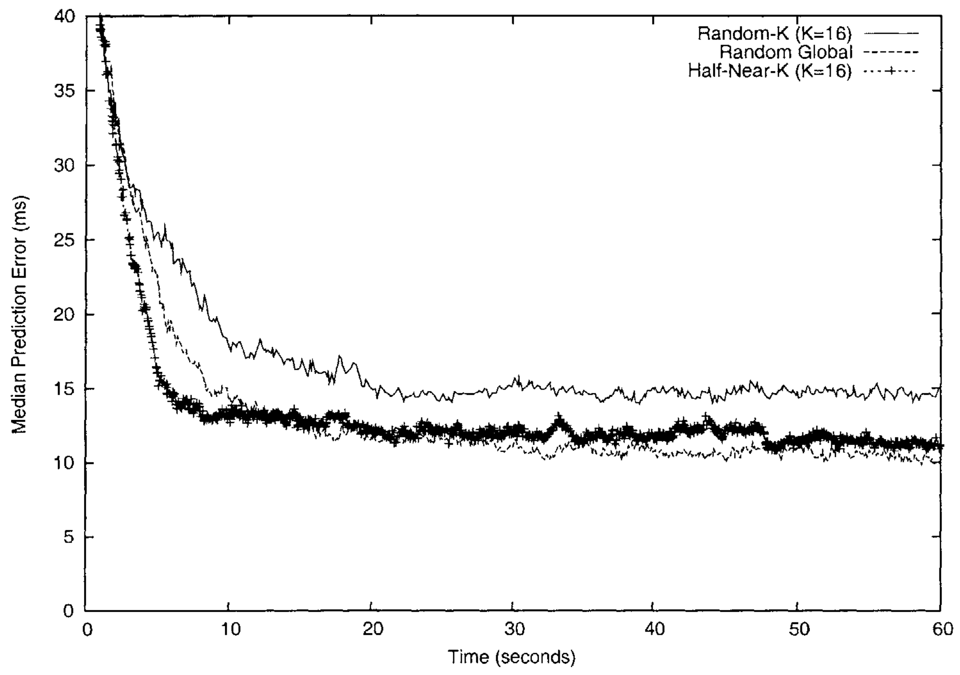


Figure 4-4: Vivaldi, $C_c = 0.25$. PlanetLab, $N = 127$, $D = 3$. Comparing three different configurations.

Figure 4-4 compares the convergence behavior of all three different configurations when the time step C_c is fixed at 0.25. The plot shows the median prediction error as a function of time. The Half-Near-K-Neighbors and Random Global sampling perform better than sampling from a fixed set of randomly selected neighbors. Additionally, when sampling from a fixed set of logical neighbors, the prediction accuracy of Vivaldi can be improved by including nearby nodes into the logical neighbor set.

Figure 4-5 shows the Vivaldi convergence for King data set using the three different configurations mentioned above, where the number of logical neighbors is 64 for the Random-K and Half-Near-K neighbors configurations. The Half-Near-K configuration converges faster to low prediction error in comparison to the other two configurations. For the rest of this thesis, we use the Half-Near-K neighbor configuration with adaptive timestep constant $C_c = 0.25$ for all Vivaldi simulations, unless otherwise noted.

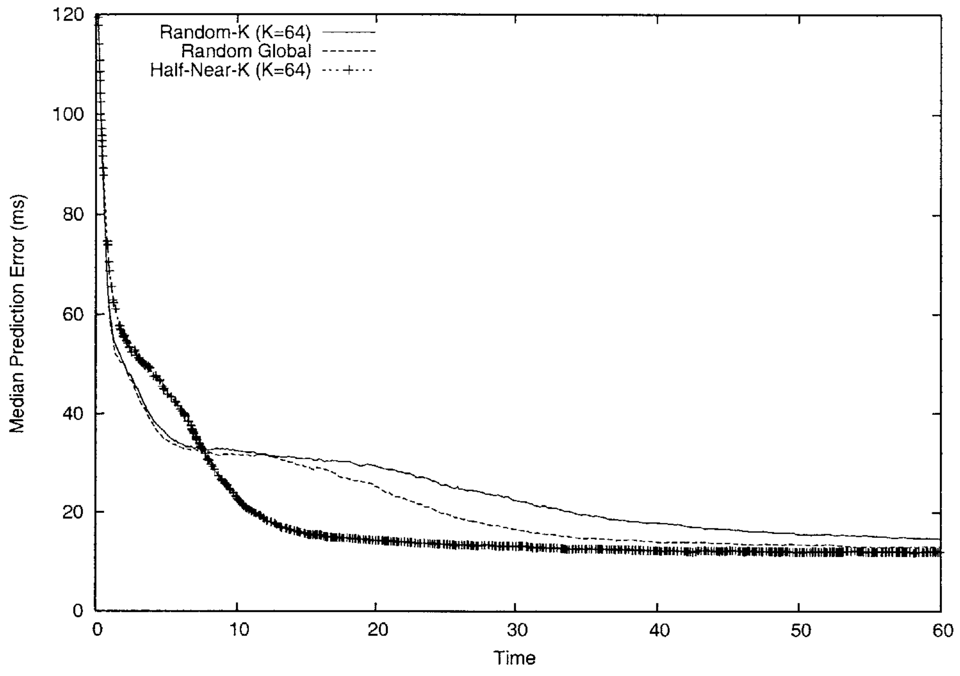


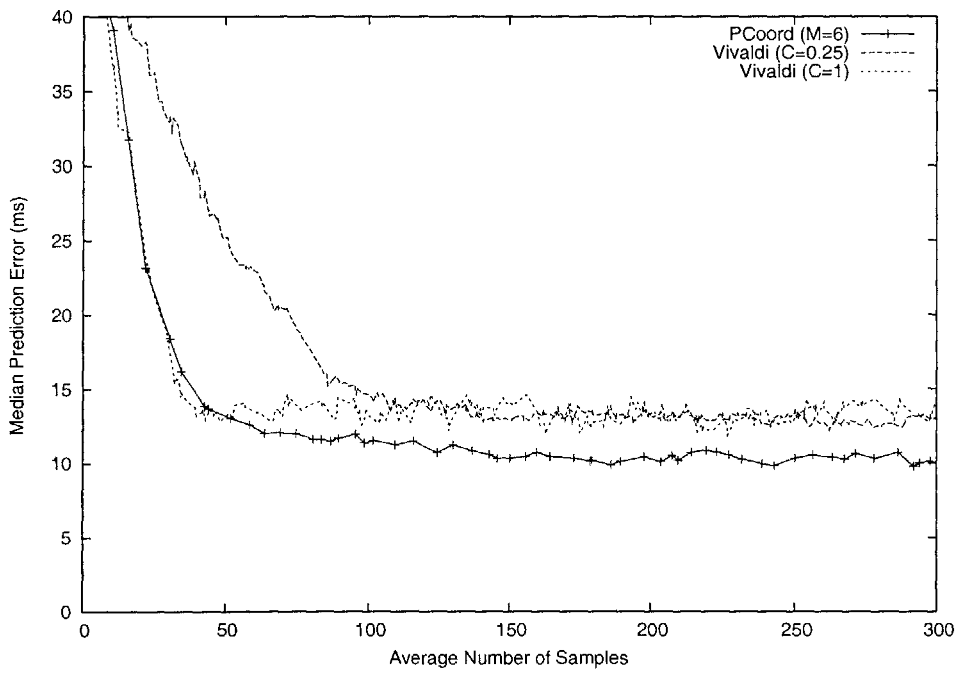
Figure 4-5: Convergence behavior of Vivaldi, King, $N = 1740$, $D = 5$.

4.3 PCoord Results

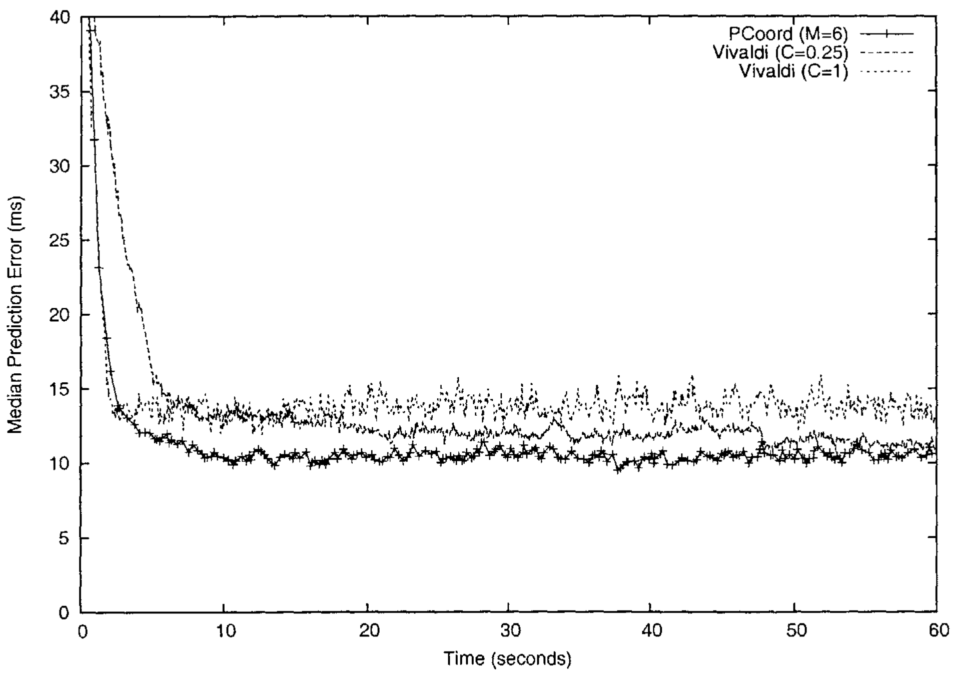
In this section, we present the PCoord performance using the PlanetLab and the King data set.

4.3.1 PCoord PlanetLab Results

In this section, we compare PCoord with Vivaldi using the PlanetLab data set. Figure 4-6 compare PCoord and Vivaldi convergence behavior in terms of average number of samples and time. For PCoord, the sample batch size M is six. We use a conservative estimate of the Simplex Downhill processing cost of 10 ms, which is the median processing cost for 5-dimensional coordinates with batch size ten. It is conservative in the sense that the processing cost in general increases as the dimensionality and number of reference points increase. The Vivaldi results presented here use Half-Near-K neighbors configuration with $K = 16$, and $C_c = 1$ and 0.25. We note that PCoord converges faster to a lower error both in terms of time and number of samples used.



(a) Compare PCoord and Vivaldi in terms of average number of samples



(b) Compare PCoord and Vivaldi in terms of time

Figure 4-6: Compare PCoord and Vivaldi in terms of time and number of samples, PlanetLab, $N = 127$. PCoord uses six reference points per coordinate update. Vivaldi uses Half-Near-K neighbors configuration with $K = 16$, $D = 3$.

4.3.2 PCoord Convergence Behavior Using King Data Set

In this section, we examine PCoord's convergence behavior with various sample batch size using the King data set. Figures 4-7, 4-8, and 4-9 plot the convergence of PCoord as a function of number of samples used, time, and number of coordinate updates. The numbers of reference points used at each coordinate update are 10, 20 and 30. The convergence time is 5 to 10 seconds in all three cases. Our observations are as follow.

- PCoord converges to a low median prediction error (approximately 12 ms) in about 100 to 120 samples, or equivalently, 10 to 12 coordinate updates when 10 reference points are used for each coordinate update.

- For PCoord, including larger numbers of reference points at each update step yields slightly slower convergence in terms of number of samples used, but faster convergence in terms of time and number of iterations of coordinates updates.

Overall, using 10 reference points at each update step yields quick convergence to low error and achieves good tradeoff between communication and computation overhead.

Variations Among Different Simulation Runs

In PCoord simulations, a node's logical neighbors are randomly drawn from the global population in each simulation run. In order to test how PCoord's performance vary under different initial logical neighbor assignments, we run PCoord with five different logical neighbor configurations using the King data set. Each node is assigned 10 logical neighbors randomly drawn from the 1740-nodes global population. In Table 4.2, we present the mean and standard deviations of medium system errors across the five different simulation runs at fixed time intervals. As a reference, we also show the average number of samples used by each node at each sampled time interval. We note that the largest standard deviation at the first second is less than 5 ms with an average of about 45 ms. After the first second, the standard deviation is about

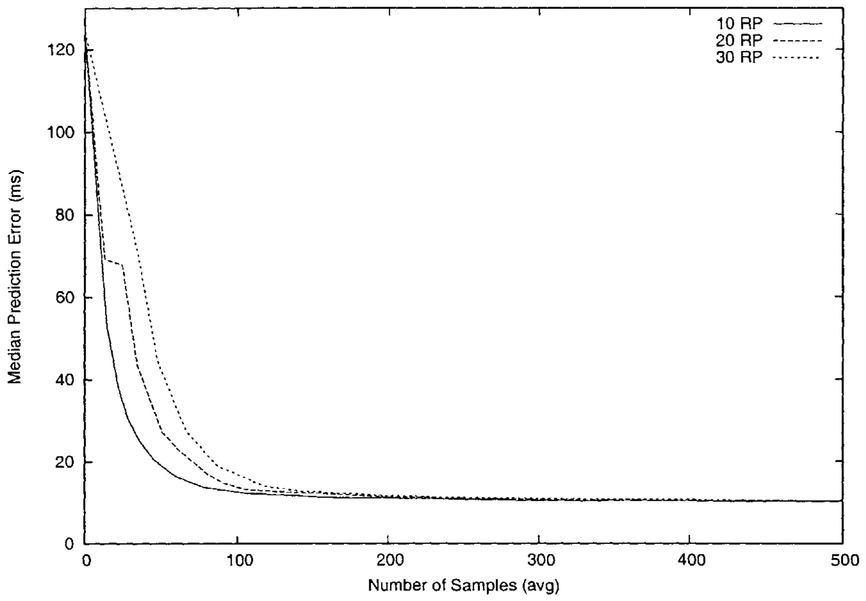


Figure 4-7: PCoord convergence as a function of number of samples when 10, 20 and 30 reference points are used at each coordinate update. King, $N = 1740$, $D = 5$.

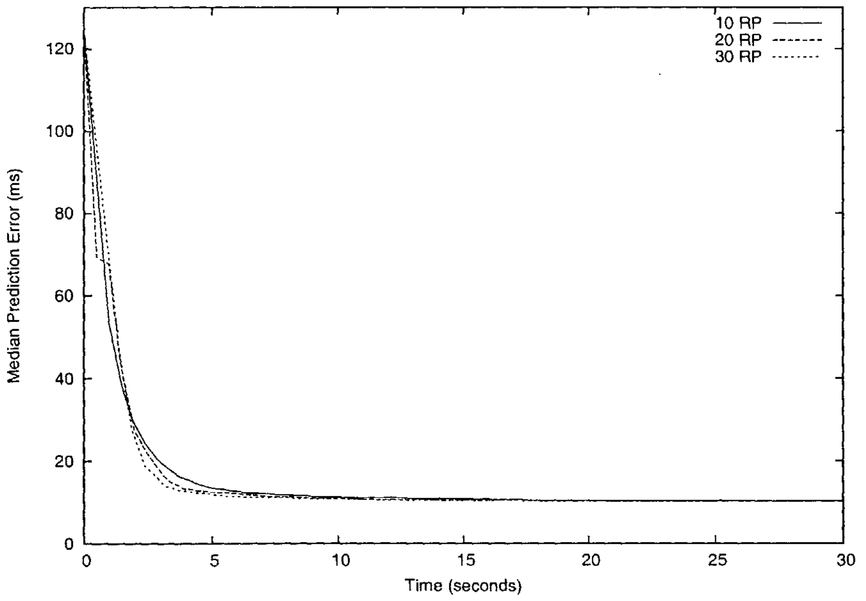


Figure 4-8: PCoord Convergence as a function of time when 10, 20 and 30 reference points are used at each coordinate update. King, $N = 1740$, $D = 5$.

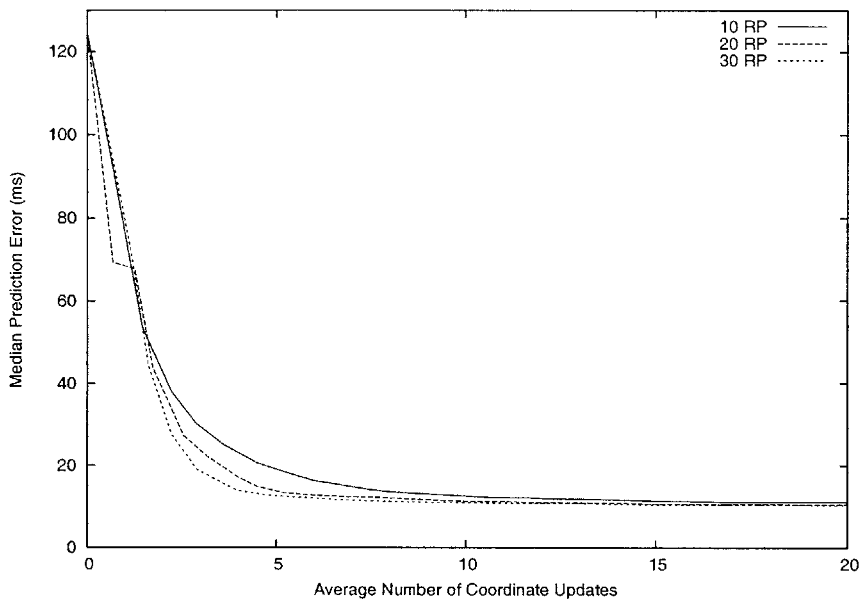


Figure 4-9: PCoord convergence as a function of number of coordinate updates when 10, 20 and 30 reference points are used at each coordinate update. King, $N = 1740$, $D = 5$.

Time (Seconds)	1	2	3	4	5	10	20	30
Std PE (ms)	4.77	1.29	2.21	2.09	1.57	0.26	0.08	0.07
Mean PE (ms)	44.57	27.06	20.20	16.83	14.37	11.53	10.46	10.39
Mean Samples	18.26	35.73	52.13	67.57	90.81	164.72	319.74	471.96

Table 4.2: PCoord mean and standard deviation of system prediction error (in ms). Statistics from five different logical neighbor configurations, King, M=10.

1 - 2 ms until after 10 seconds, the standard deviation drops to below 1 ms when the mean error stays at the 10 ms range. This suggests that the variability of the prediction error is rather small across different simulation runs under different initial logical neighbor configurations.

4.4 Comparison of PCoord, Vivaldi and FixedLM

In this section, we compare the performance of PCoord with Vivaldi and the FixedLM scheme. For the FixedLM scheme, we randomly select 10, 20 and 30 nodes as landmarks using the King data set, with twenty different randomly generated landmark configurations. The results with the lowest median prediction error are reported. Using 10 and 20 landmarks, the lowest median prediction errors are 12.16 and 11.44 ms respectively. Using 30 fixed landmarks, the lowest median prediction error is approximately 11 ms; the lowest 95th percentile prediction error is 32 ms.

4.4.1 Relative Error Distribution

Figure 4-10 shows the cumulative distribution of relative error of PCoord, Vivaldi, and the FixedLM scheme. With the FixedLM scheme, each host uses 30 fixed landmarks; the results shown in plot have the lowest median error of all 20 different randomly-generated landmark configurations.

In Plot 4-10a, PCoord's relative error distribution are based on coordinates generated after five seconds of simulated time, with each host using on average 100 samples in total. Plot 4-10b and 4-10c show PCoord results generated after 10 and 20 seconds of simulated time, with each host using on average 165 and 320 samples respectively. Vivaldi results shown use about the same number of samples per host after 10, 15, and 30 seconds of simulated time. Vivaldi uses Half-Near-K neighbors configuration, where $K = 64$ and C_c=0.25.

Note that the number of samples reported in the PCoord and Vivaldi are non-unique samples; i.e., if a node uses the coordinates and distances sampled from the same node twice at different iterations, it is counted as two distinct samples.

We note that PCoord's prediction accuracy can in fact come fairly close to that of the FixedLM scheme after 100 samples. The FixedLM scheme's median and 75th percentile relative error are approximately 10% and 22% respectively. After 100 samples, PCoord's median relative error is approximately 12% which is only 2% more than that of the FixedLM scheme with 30 landmarks. The 75th percentile relative error is approximate 33% after 100 samples. At 165 samples, PCoord can do as well as FixedLM with 30 landmarks. Beyond 250 samples, PCoord performs slightly better than the FixedLM scheme in terms of relative error distributions.

Vivaldi ($C_c = 0.25$) takes about 320 samples to achieve 12% median relative error. After 100 samples, Vivaldi's median relative error is 24%, which is about twice as much as that of PCoord's using the same number of samples. After 165 samples, the median relative error is approximately 16%.

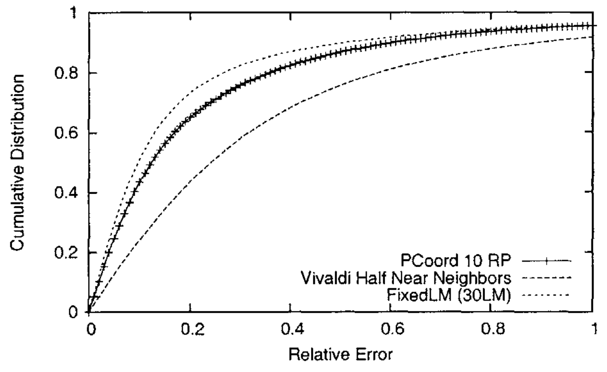
In summary, our results indicate that PCoord can achieve comparable performance as the FixedLM scheme when "sufficient" number of samples are used. For PCoord, it takes approximately 100 - 165 samples to achieve comparable accuracy as the FixedLM scheme. For Vivaldi, it takes over 300 samples to achieve comparable performance as the FixedLM scheme.

4.4.2 Convergence Behavior

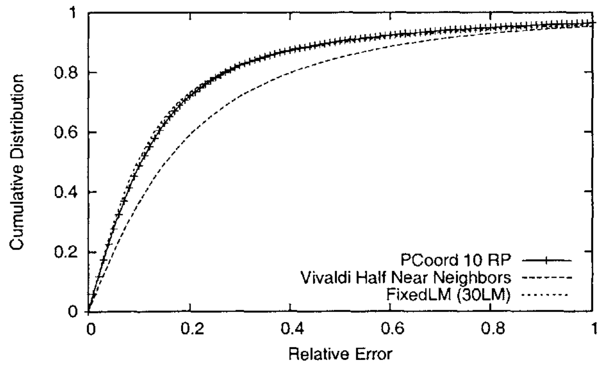
In this section, we compare PCoord and Vivaldi in terms of time and number of samples required for convergence. We note that the Vivaldi simulation updates coordinates using one sample at a time. PCoord has faster convergence in terms of time (or number of seconds) due to the fact that we use asynchronous communication to sample a batch of distances at a time. Vivaldi could potentially speed up the convergence time by sampling nodes' distances and coordinates in a batch mode. However, it is unclear how that might impact its prediction accuracy. We show PCoord's convergence in terms of time to illustrate that PCoord can converge within a reasonable time frame despite the additional computational overhead from the Simplex algorithm. For comparison purposes between PCoord and Vivaldi, we focus on the number of samples required to converge to a low prediction error.

Figures 4-11, 4-12, and 4-13 compare the convergence behavior of PCoord and Vivaldi in terms of 5th, 50th, and 95th percentile system error as a function of time and number of samples used. Our results indicate that PCoord converges faster than Vivaldi across all three error measurements. Using 10 reference points, the median prediction error of PCoord decreases to 13 ms range with each host using on average 80 to 100 sample, and 12 ms range using 100 to 120 samples. It takes Vivaldi twice as many samples on average to reach the same level of median error.

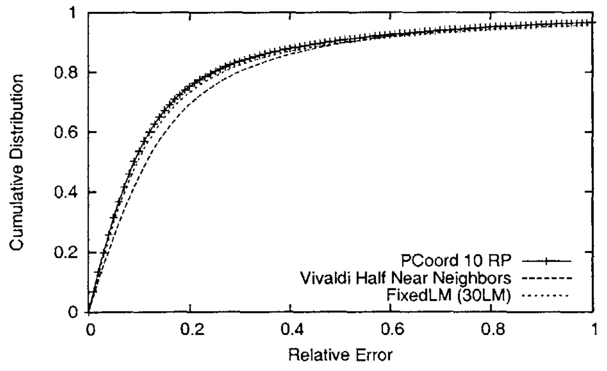
We also observe that, in comparison to the median error, the 95th percentile error takes longer to converge for both PCoord and Vivaldi. The lowest 95th percentile prediction error is 32 ms for the FixedLM scheme using 30 fixed landmarks. For



(a) After 100 Samples

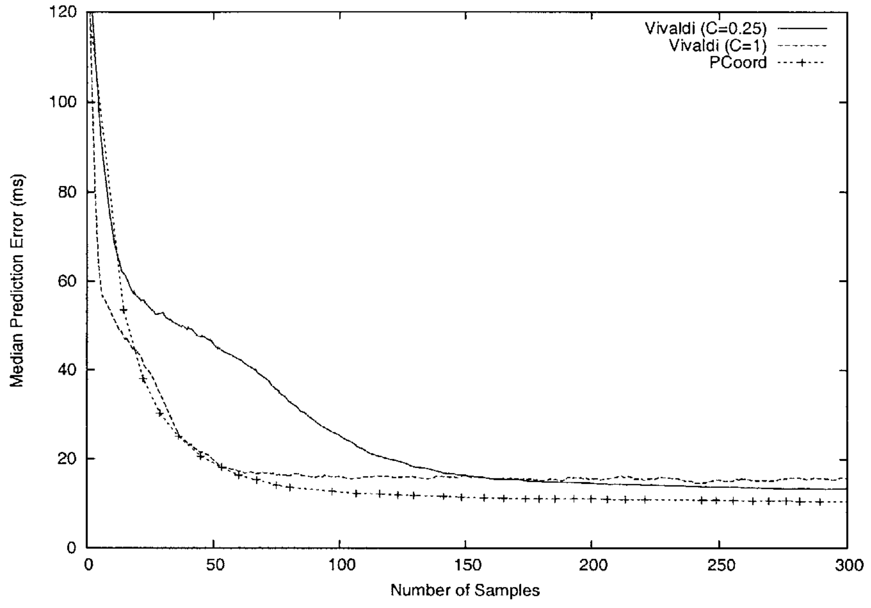


(b) After 165 Samples

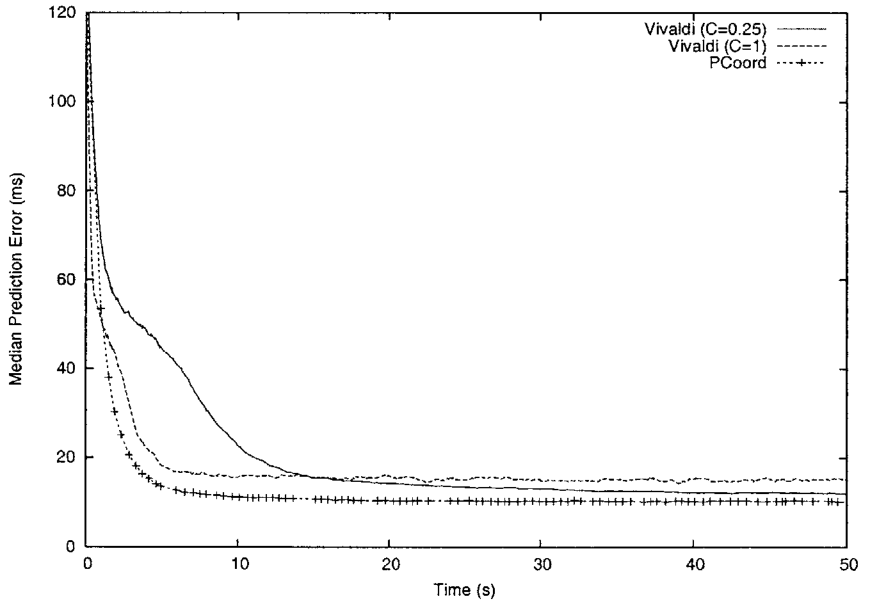


(c) After 320 Samples

Figure 4-10: Compare PCoord, Vivaldi and FixedLM in terms of relative error, King, $N = 1740$, $D = 5$, $M = 10$.

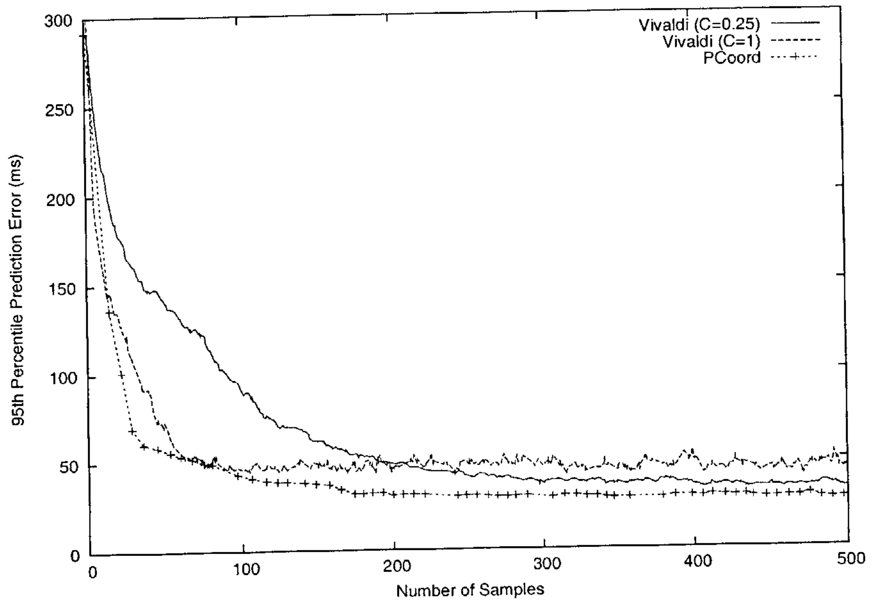


(a) Median error in terms of number of samples

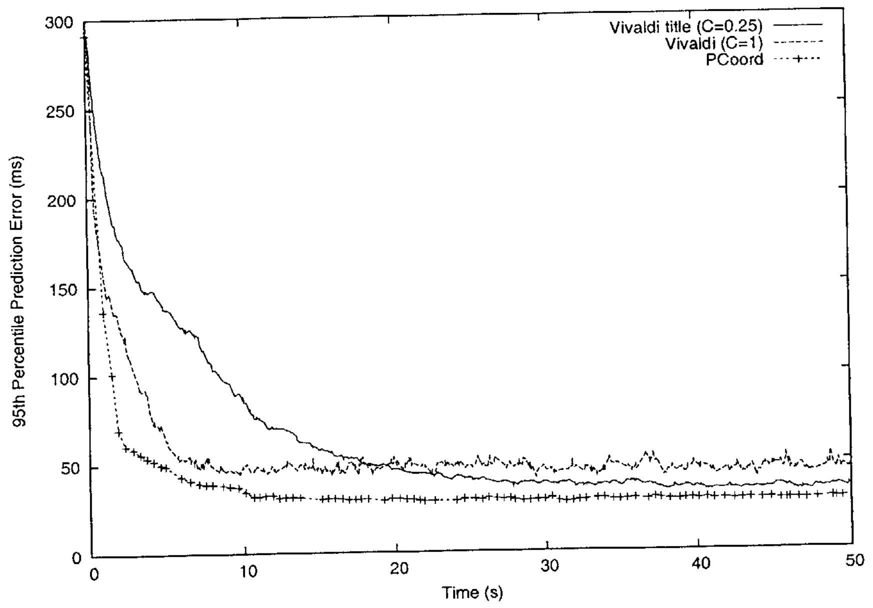


(b) Median Error in terms of time

Figure 4-11: Convergence of PCoord (10 reference points) and Vivaldi (C_c=0.25 and 1, Half-Near-K neighbors, K = 64) in terms of median error. King data.

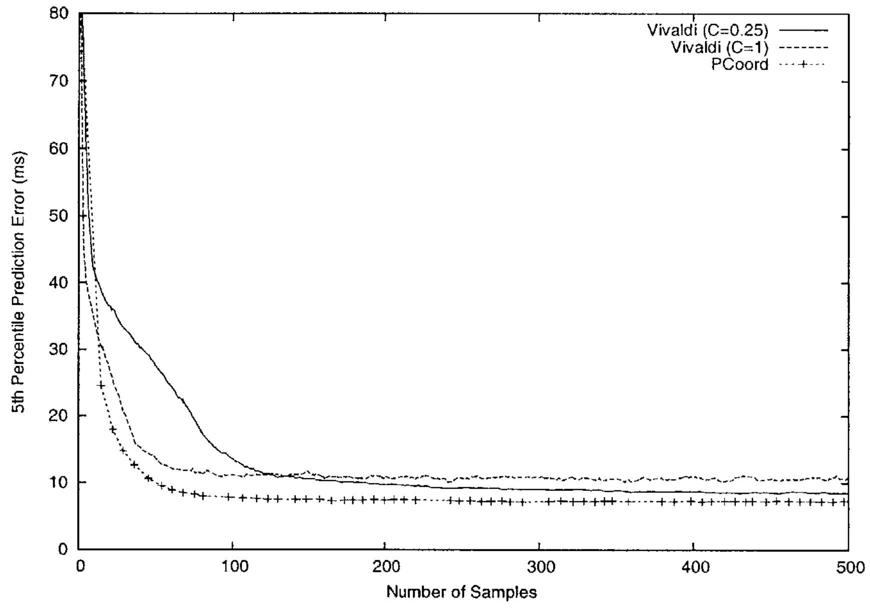


(a) 95th percentile error in terms of Sample

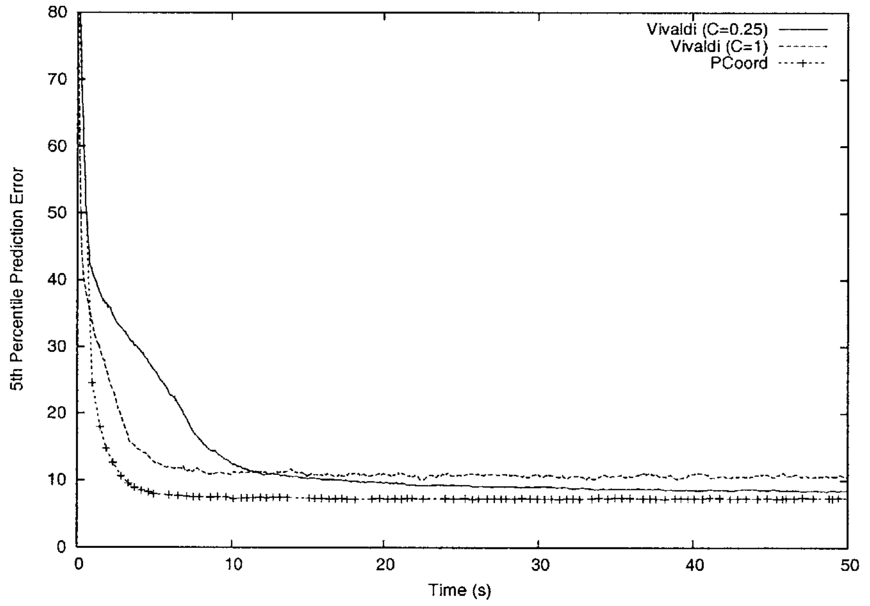


(b) 95th percentile error in terms of time

Figure 4-12: Convergence of PCoord (10 reference points) and Vivaldi in terms of 95th percentile error. King data.



(a) 5th percentile error in terms of Sample



(b) 5th percentile error in terms of time

Figure 4-13: Convergence of PCoord (10 reference points) and Vivaldi in terms of 5th percentile error. King data.

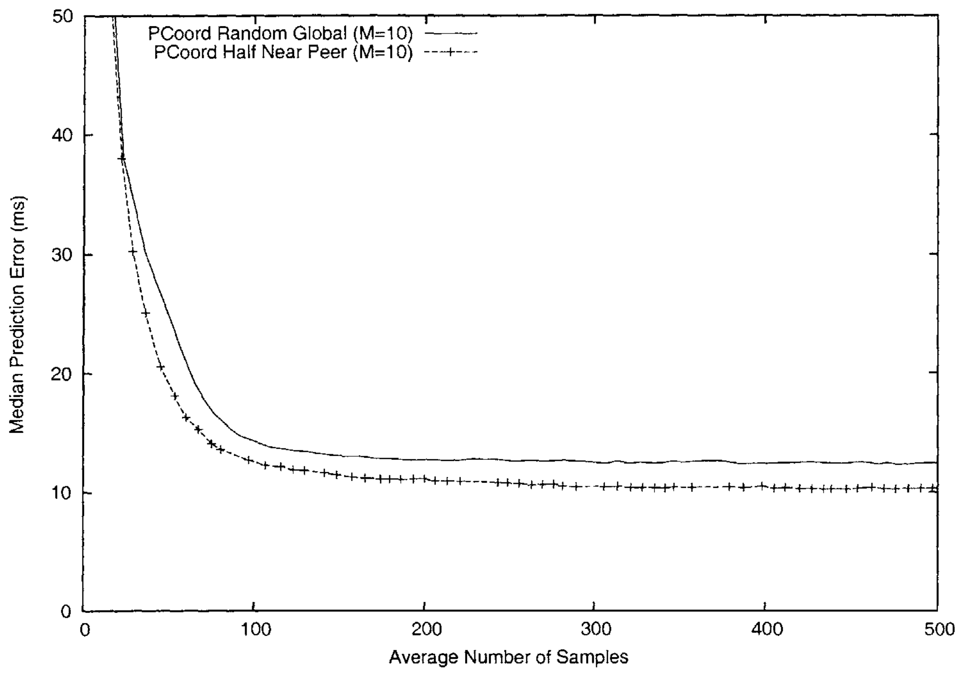


Figure 4-14: PCoord: compare near peer based sampling (default) vs. random global sampling. King, $N = 1740$.

PCoord, it takes approximately 180 to 200 samples to converge to 32 ms for 95th percentile error, and it takes Vivaldi approximately 500 samples to converge to the same 95th percentile error range.

4.5 Effects of Including Nearby Neighbors in Reference Set

In the previous sections, PCoord uses an active peer discovery mechanism that allows peers to discover each other by exchanging messages. Each node includes the nearest peers discovered so far in its reference set for coordinate update. In this section, we ask the question whether and by how much including nearby neighbors in the reference set help to improve prediction accuracy?

To answer the above question, we run PCoord using 10 reference points in two different sampling strategies: (1) the PCoord default strategy, i.e., the near-peer based PCoord sampling strategy described earlier which includes both nearby and random

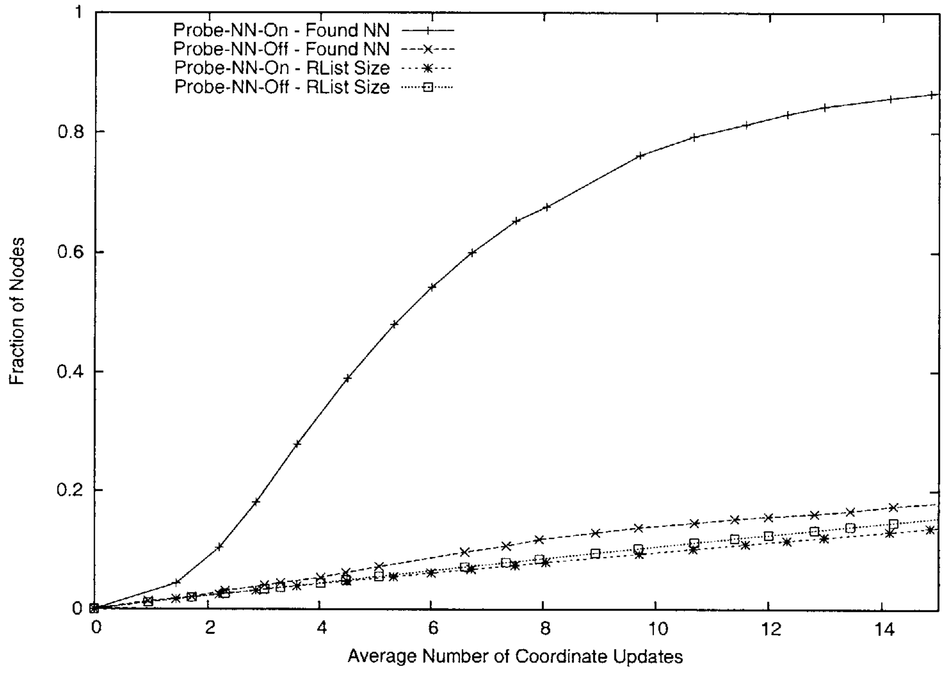


Figure 4-15: PCoord King data set. N=1740.

peers from a node's peer list, and (2) the Random Global strategy which assumes that each node has access to a list of all other nodes in the system; at each update round, a node randomly sample M nodes from the global peer list for coordinate update.

Figure 4-14 plots the median error of PCoord when using the default near-peer based sampling vs. the Random Global strategy. The number of reference points M is 10 for both cases. For the default PCoord strategy, $numNN = 5$. Our results indicate that after approximately 100 samples, the median errors of both options drop to the 13 ms range, but the near-peer based sampling strategy can converge further to lower median error: its median error is in the 10 ms range after 200 samples; whereas the median error of the Global Random strategy stays above 12 ms. This suggests that including nearby neighbors in the reference set improves the distance prediction accuracy.

4.6 Effectiveness of PCoord Near Neighbor Search

Next, we turn to the question of how effective PCoord is in discovering nearby peers. To answer this question, we run PCoord using 10 reference points in two different modes: with the probe nearest neighbor option turned on (probe-NN-on) and off (probe-NN-off). The probe-NN-on option is the default PCoord algorithm. In the probe-NN-off option, each node randomly selects 10 peers from its peer list as its reference points for coordinate update, and it does not probe its triangulated list for nearest neighbors.

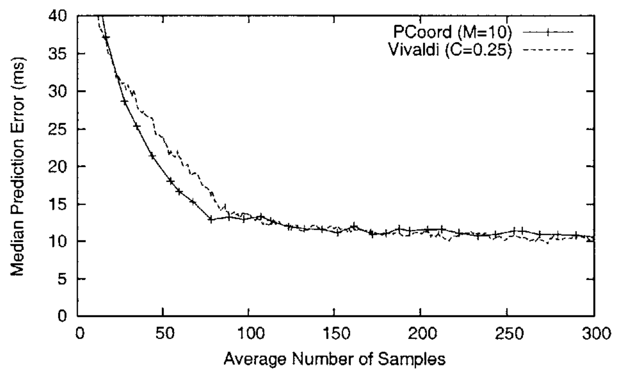
Figure 4-15 shows the fraction of peers that found its actual nearest neighbor, the average fraction of nodes probed by each node and the average RTT peer list as a function of average number of coordinate updates performed. Our results indicate that the triangulated distance based nearest neighbor search scheme is very effective in discovering nearby peers: using the probe-NN-on option, approximately 80% of the hosts found their nearest peers by probing less than 10% of the peer population; with the probe-NN-off option, less than 15% of the hosts found their nearest neighbors with similar probing cost.

4.7 Compare PCoord and Vivaldi Using Random Peer Sampling

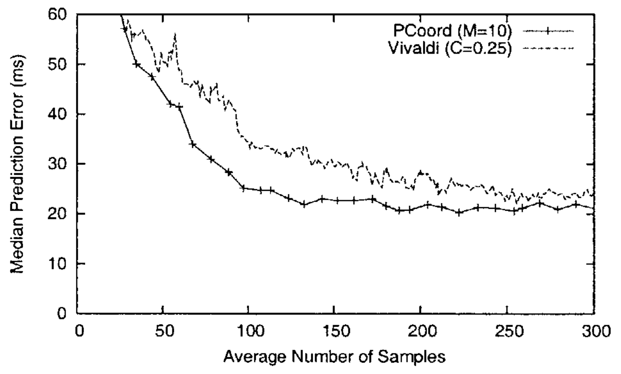
Our results so far indicate that PCoord outperforms Vivaldi in that it can converge to lower prediction error than Vivaldi using fewer number of samples. In this section, we ask how much of PCoord's performance advantage over Vivaldi is due to its peer discovery and sampling strategy.

To answer this question, we compare PCoord and Vivaldi using the same sampling strategy. We turn off PCoord's near-peer based sampling strategy and compare PCoord and Vivaldi when both schemes use the Random Global sampling strategy.

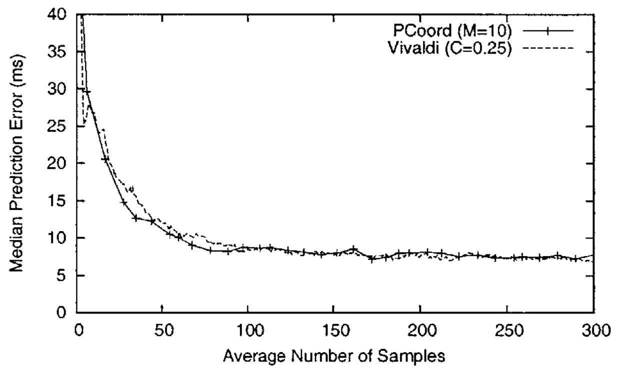
Figure 4-16 plots the median, 95th and 5th percentile error of PCoord and Vivaldi using the PlanetLab data set with the Random Global sampling strategy. In figure 4-



(a) Median Error

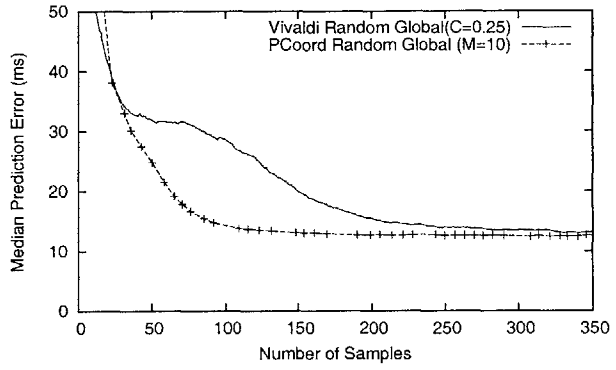


(b) 95th Percentile Error

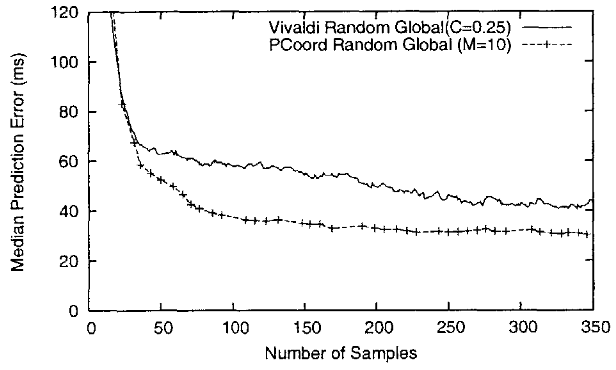


(c) 5th Percentile Error

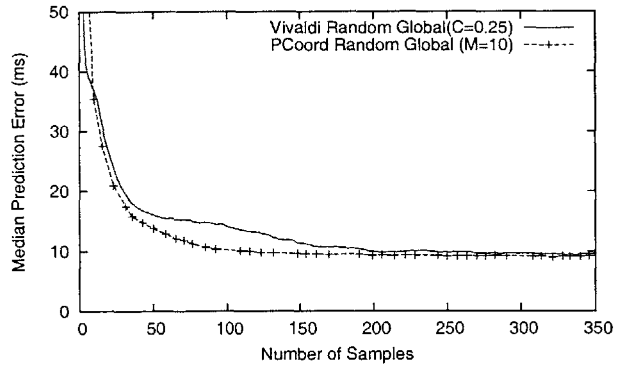
Figure 4-16: Compare PCoord and Vivaldi when both use Random Global sampling strategy, PlanetLab, N=127, D = 5, M = 10.



(a) Median error



(b) 95th Percentile Error



(c) 5th Percentile Error

Figure 4-17: Convergence of PCoord (10 reference points) and Vivaldi (C_c=0.25) using Random-Global peer sampling. King data.

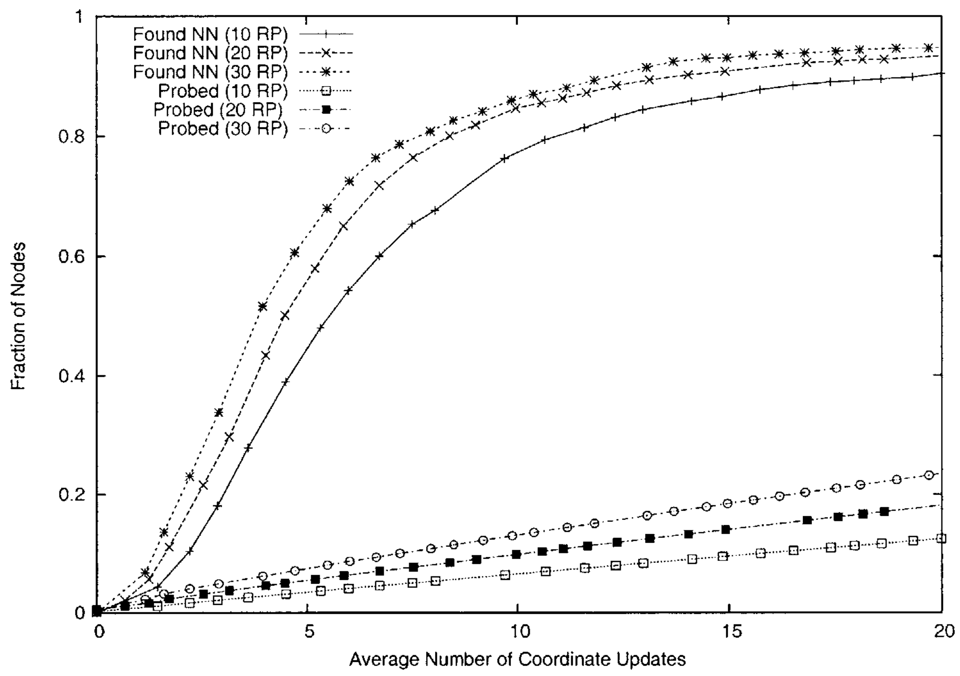


Figure 4-18: King data set. $N = 1740$.

17, we perform the same comparison between PCoord and Vivaldi using the King data set with the Random Global sampling strategy.

Our results indicate that PCoord is able to converge faster than Vivaldi when both schemes use the Global-Random sampling strategy. Using the PlanetLab data, PCoord has lower median prediction error when the number of samples used is 150 or less. Using the King data set, Vivaldi's 95th percentile error drops to 35 ms after approximately 100 samples. After the same number of samples, PCoord's 95th percentile prediction error is 25 ms, which is 40% less than that of Vivaldi's.

4.8 Discussion of PCoord Communication Cost

In this section, we examine the storage and communication cost of PCoord. Figure 4-18 illustrates the effectiveness of PCoord peer discovery mechanism when 10, 20 and 30 reference points are used and the associated communication cost measured as fraction of nodes each host probed as the coordinate update process progresses. The plot shows the fraction of nodes that found their nearest neighbors and the

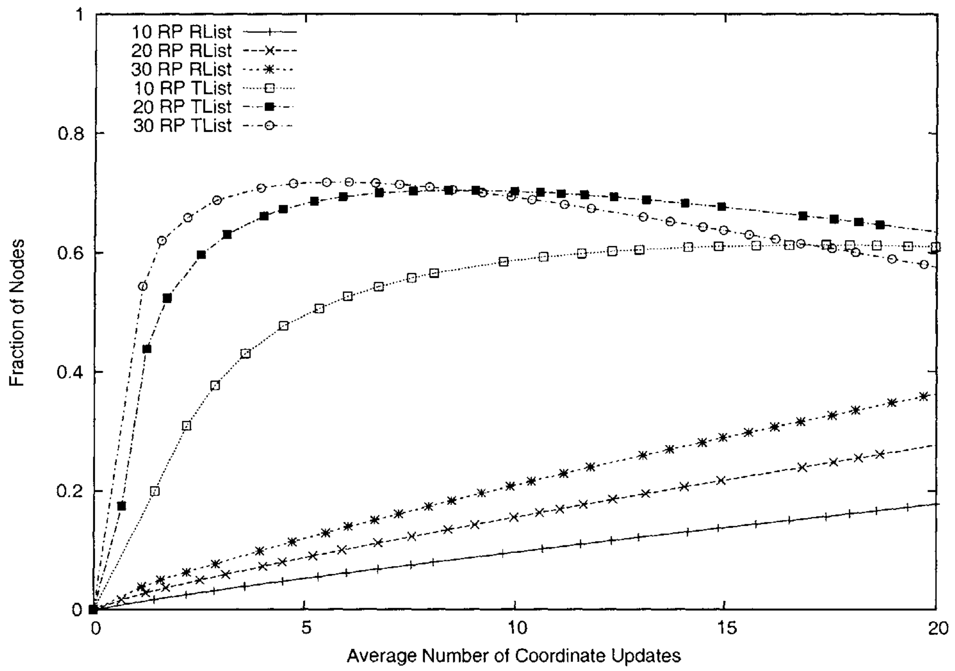


Figure 4-19: King data set. $N = 1740$.

fraction of nodes probed as a function of the average number of coordinate updates with different number of reference points. We note that, using 10 reference points, after five coordinate update iterations, 50% of the nodes found their actual nearest neighbor by probing only 3.8% of the global peer population; after 10 coordinate update iterations, over 80% of the hosts found their nearest neighbors by probing 7% of the total peer population.

Figure 4-19 shows the average R list and T list size as the coordinate update progresses when 10, 20 and 30 reference points are used. The number of RTT entries that each host exchanges with each of its reference points is limited to 30 entries in all three cases. As expected, the higher the number of reference points used at each iteration, the faster a host discovers other peers in the system. Recall that PCoord converges in about 5 to 10 coordinate updates when 20 and 30 reference points are used, and 10 to 15 coordinate updates when 10 reference points are used. The R list represents a list of peers that a host has communicated with directly. The T list represents a list of peers that a host has not communicated with directly, but has triangulated information for. We note that, after 5 coordinate updates, on average, each peer has directly communicated with approximately 5%, 9%, 13% of the peer population, and obtained triangulated information for 49%, 69%, and 72% of the peers when 10, 20 and 30 reference points are used respectively.

4.9 Performance of Newly Joined Nodes

In the previous sections, we evaluated PCoord's convergence behavior when all nodes join at approximately the same time. Our results suggest that, in a 1740 nodes peer-to-peer system, PCoord can converge within 10 seconds to low median prediction error, where each node performs less than ten coordinates updates using 10 reference points per update.

In practice, however, it is not likely that thousands of nodes all start at the same time. It is more likely that nodes join incrementally as the system evolves. Intuitively, a node joining a system that has already converged should take much less time and number of samples to converge to coordinates with low prediction error.

In this section, we evaluate the performance of PCoord for newly joined nodes. We divide the 1740 nodes in the King data set in half: 870 nodes join at approximately the same time in the beginning; the other 870 nodes start joining the system after the initially joined nodes' coordinates have converged; the new nodes join incrementally, one at a time. We call this joining scenario, Half N Incremental Join, in contrast to the joining scenario in previous sections, which we call All N Simultaneous Join, where all N nodes join in the beginning at the same time.

We simulated the incremental join case for 225 seconds in total, where 870 nodes start at approximately the same time in the beginning, and the other 870 nodes start joining 20 seconds afterwards, in a 200 ms interval. In both the simultaneous and incremental join cases, each host samples RTT and coordinates from ten peers at each coordinate update. The dimensionality of the Euclidean mapping is five. Each of the newly joined node is initialized with ten logical neighbors, randomly drawn from nodes that are in the system at the time. The new node samples RTT and coordinates from its randomly assigned logical neighbors for its first coordinate update, and runs the

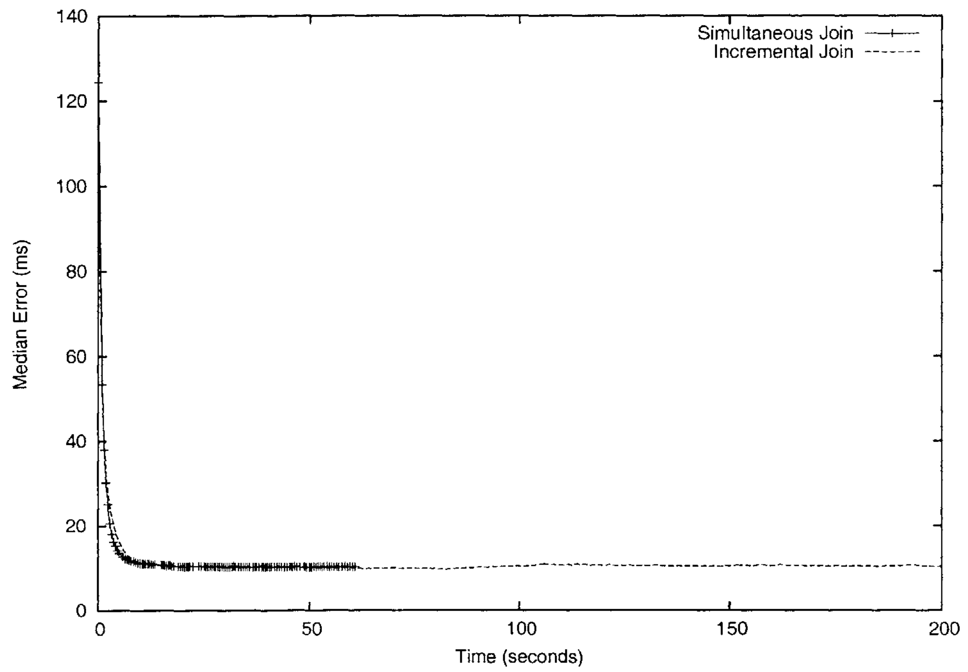


Figure 4-20: Compare convergence behavior of PCoord in two join scenarios: (1) All N Simultaneous Join: all nodes join at the beginning of system start time, and (2) Half N Incremental Join: when half the nodes join all at once in the beginning and the other half starting to join 20 seconds afterwards one at a time, in a 200 ms interval. King data set. $N = 1740$.

PCoord peer information exchange protocol.

Figure 4-20 compares the convergence of system-wide median error of the two joining scenarios: All N Simultaneous Join vs. Half N Incremental Join. The x-axis is the absolute time, i.e., time since the beginning of system start time. The y-axis is the median error across all nodes in the system at the time. For the simultaneous join case, we only show the error for the initial 60 seconds. We note that the incremental join, starting at time 20 (seconds), does not change the system-wide median error significantly, suggesting that the newly joined nodes introduce very little disturbances to the existing nodes.

Figures 4-21 and 4-22 evaluate the error convergence of the newly joined nodes as a function of time in system and number of coordinate updates. We also show the system-wide statistics in the same plot as a comparison. The system-wide statistics is taken as a snapshot of the system error at some instance in time, i.e., the system

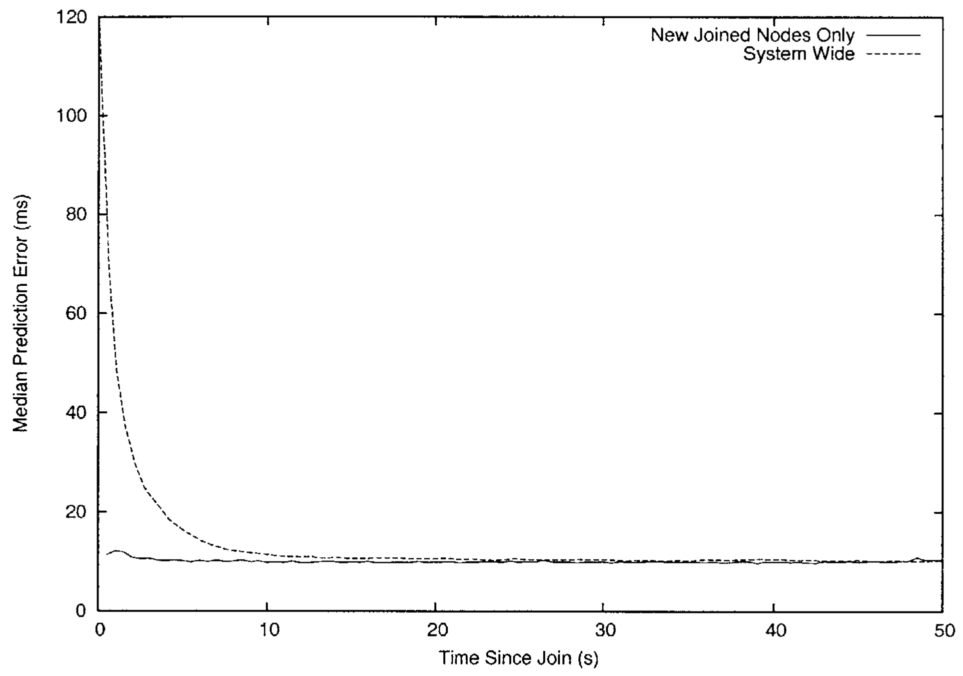


Figure 4-21: Convergence behavior of the Half N Incremental Join scenario. Plot shows the convergence of the median node error as a function of time since join. The curve for the "System Wide" represents the system median error as a function of time, where the system consists of all nodes currently in the system at that instance of time. The "New Joined Nodes Only" curve is the median error of all newly joined nodes as a function of time since their respective join time; the x-axis in this case, represents the relative time with respective to each node's own join time. King data set. $N = 1740$.

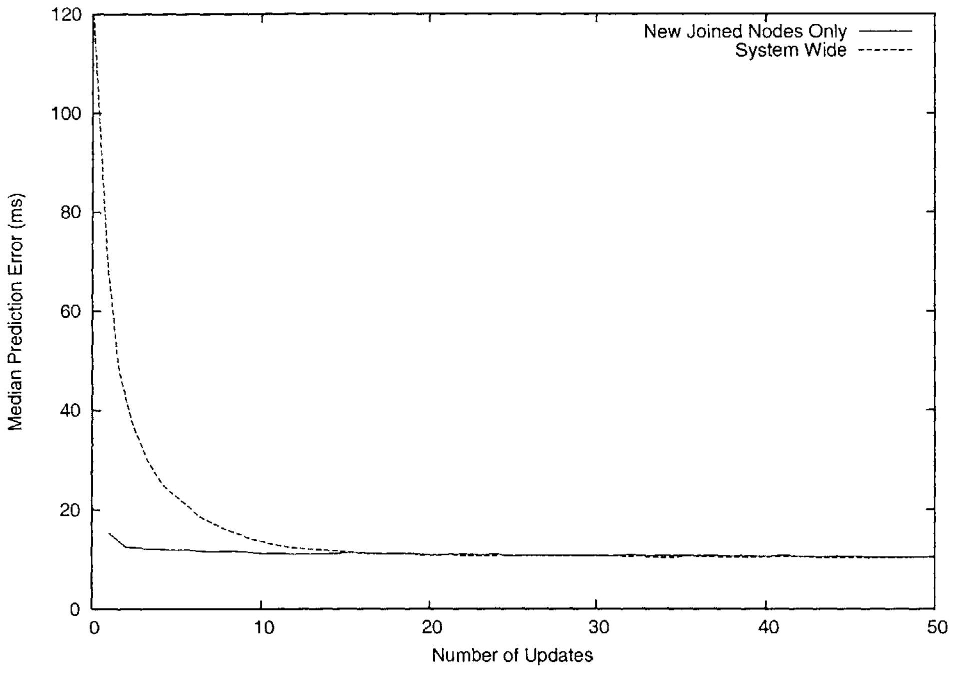


Figure 4-22: Convergence behavior of the Half N Incremental Join scenario. Plot shows the convergence of the median node error as a function of number of updates performed by each node since join. The curve for the "System Wide" represents the system median error, where the system consists of all nodes currently in the system, and the number of updates is the average number of updates done by all nodes in the system at that instance of time. The "New Joined Nodes Only" curve is the median error of all newly joined nodes as a function of number of coordinates updates performed. King data set. $N = 1740$.

median error is the median of the median errors of all nodes that are in the system at the time of the snapshot.

In Figure 4-21, the "New Joined Nodes Only" curve is the median error of all newly joined nodes as a function of time since their respective join time; the x-axis in this case, represents the relative time with respective to each node's own join time. The plot confirms our intuition that newly joined nodes take much less time to converge to a reasonably low median error than in the simultaneous join case: the median error of the newly joined nodes (the median of all new node's median error after t seconds since its initial join time) is 12.13 ms after merely one second.

In Figure 4-22, the "New Joined Nodes Only" curve is the median error of all 870 newly joined nodes as a function of number of coordinates update performed by each newly joined node so far. Our results indicate that after one coordinate update (using the randomly assigned ten logical neighbors), the median error of the newly joined nodes is 15.3 ms; after the second update, the median error drops to 12.56 ms. Within five updates, the median system error is 11.88 ms. As a reference, the 870 nodes that joined in the beginning have performed on average about 30 coordinate updates by the time the first new node joins at time 20 (seconds).

4.10 PCoord Performance under Dynamic Join and Leave

In the previous section, we examined PCoord's prediction accuracy when new nodes join the system in an incremental fashion. 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?

We examine PCoord's performance under churn using the King data set. Under the dynamic join and leave mode, each node alternately leaves and re-joins the system.

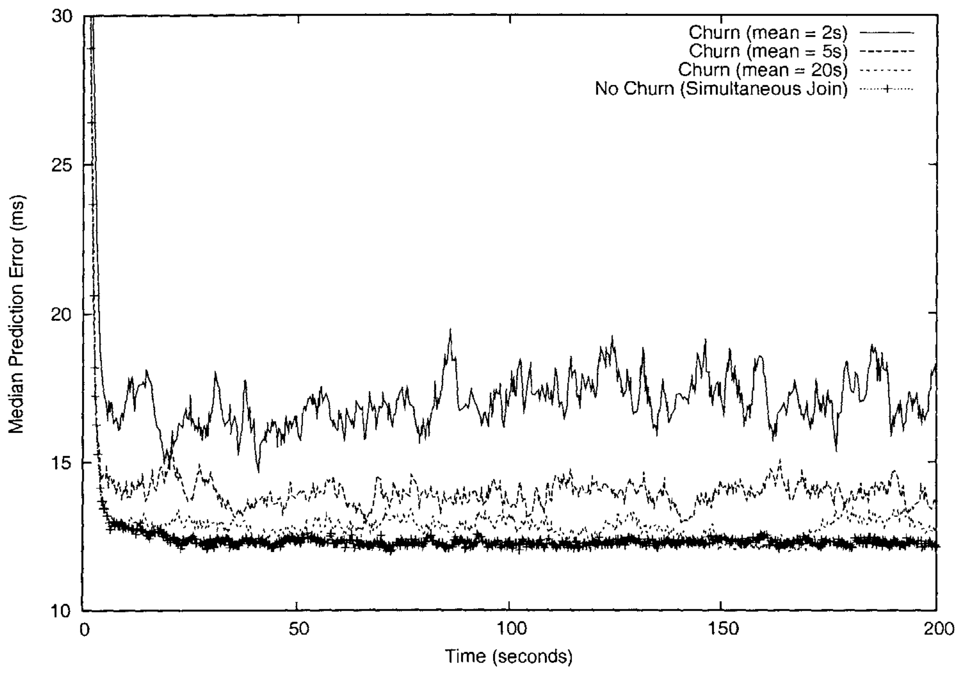


Figure 4-23: PCoord performance under dynamic join and leave. King, N = 1740, M = 10, and D = 5.

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 [45]. 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. Each node uses random peer sampling with a default sample batch size of ten samples per coordinate update.

Figure 4-23 plots the median prediction accuracy of PCoord as a function of time when t is 2, 5 and 20 seconds. As a comparison, we also plot prediction accuracy of PCoord when there is no churn, i.e., when all nodes join simultaneously in the beginning and none subsequently leave. Figure 4-24 plots the average median prediction accuracy (averaged over time) as a function of the mean host session life time, t. The average median prediction accuracy represents the steady-state prediction error of

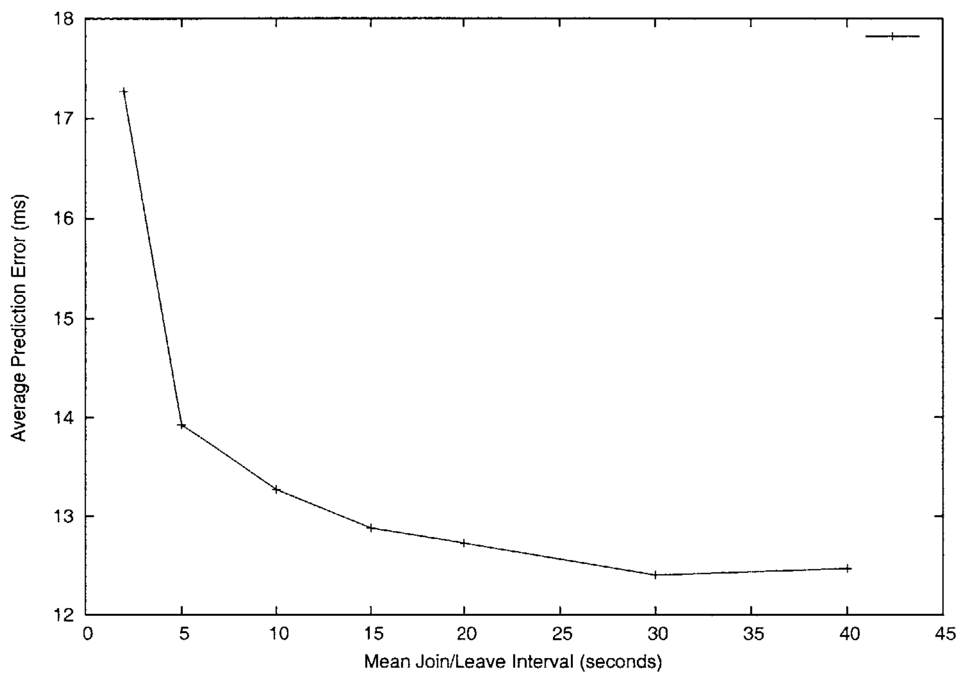


Figure 4-24: PCoord prediction accuracy as a function of mean join/leave intervals. King, $N = 1740$, $M = 10$, and $D = 5$.

the system averaged over time using statistics gathered after 60 seconds of simulated time.

Figure 4-23 shows that when the system has no churn, the steady-state system prediction error is in the 12 ms range. As expected, the system prediction error increases under high churn: Figure 4-24 shows that when the mean join/leave time interval is 2 seconds, the prediction error is approximately 40% higher than when there is no churn. However, the performance degrades very modestly when the mean join/leave interval is 5 seconds or longer: the average median prediction error of PCoord with a t of 5 seconds is approximately 13.9 ms, which is only 2 ms worse than the no churn case. When join/leave interval t is 10 seconds or greater, the median prediction error stays in the range of 12 ms, indicating that the churn has very little effect on the prediction accuracy of PCoord.

In conclusion, PCoord can achieve low prediction error even under high churn. In this section, we have shown that dynamic join and leave have minimal effects on PCoord's prediction accuracy when the dynamic join/leave time interval (or host mean session life time) is 10 seconds or longer. This suggests that PCoord can do well under the dynamic membership changes of existing peer-to-peer systems, which have a median session duration on the order of 60 minutes [45].

4.11 Effects of Different PCoord Mechanisms

In this section, we examine the effects of different PCoord mechanisms under churn. In particular, we would like to answer the following questions. How much better is PCoord in comparison to the Simple algorithm? How much added benefit does damping provide? Is the damping mechanism alone sufficient to yield good prediction accuracy?

To answer the above questions, we have simulated PCoord's performance with different combinations of the mechanisms:

- Simple: this is the Simple algorithm without weighted loss, resistance, or damping.
- WLoss + Resistance: this is a version of PCoord that implements weighted loss function and the resistance mechanism. Damping is turned off in this version.
- Damp: this version only implements the damping mechanism without the weighted loss and resistance mechanisms.
- PCoord (WLoss + Resistance + Damp): this is the default PCoord algorithm with all three mechanisms turned on.

We examine performance of the above PCoord options using the King data set with a dynamic join/leave interval of 20 seconds. Each node uses random peer sampling with a default sample batch size of ten samples per coordinate update. The results are presented in Figure 4-25. We observe that the default PCoord mechanism has the best prediction accuracy, with median system prediction error in the 12 ms range. The prediction error of the Simple algorithm is approximately 60% higher than that of PCoord.

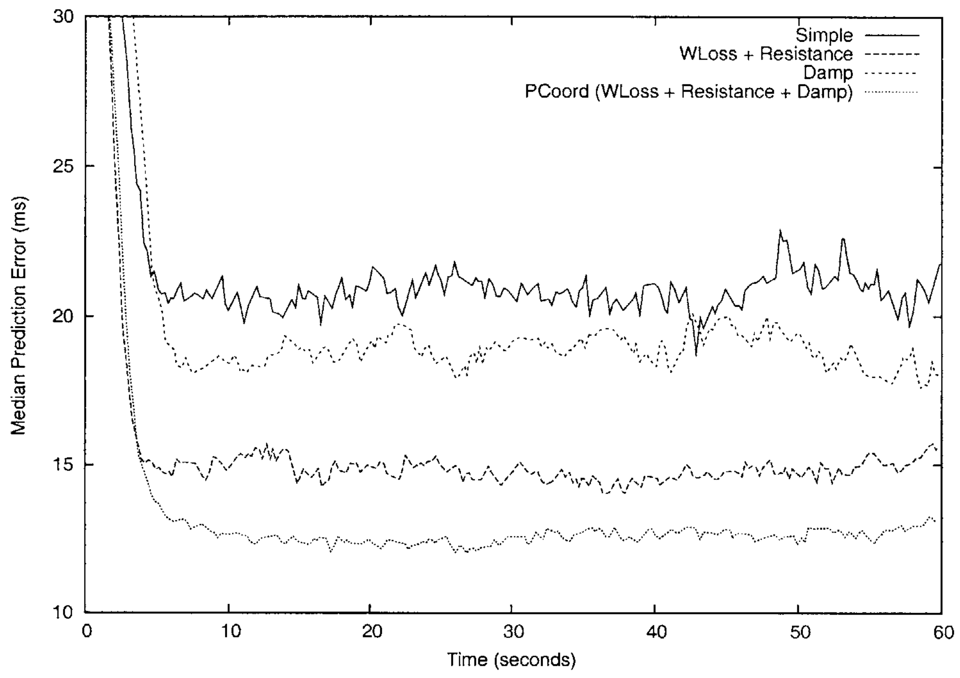


Figure 4-25: PCoord performance under dynamic join and leave. Join/Leave mean interval is 20 seconds. King data set. $N = 1740$, $M = 10$, $D = 5$.

Damping alone decreases the prediction error the Simple algorithm by about 10 to 15%. A combination of weighted loss and resistance out-performs the Simple algorithm by almost 30%. Adding damping further decreases the prediction error by 20%.

4.12 Robustness of PCoord against Faulty Information

In this section, we examine PCoord's robustness against faulty information. We are interested in understanding how PCoord behaves under increasing amount of faulty or corrupted information. Such studies will provide insights in designing further security mechanisms that guard against faulty information.

We study the effects of faults by randomly selecting some fraction of nodes as "faulty" nodes that provide incorrect information to the other nodes. We then measure the prediction accuracy among non-faulty nodes as a function of the fraction of faulty nodes in the system. We model two types of faulty information: faulty information due to (1) buggy implementations and (2) malicious attempts to confuse other nodes.

- Buggy implementations. A buggy implementation of PCoord may cause a host to report a variety of corrupted information, including its current coordinate values, its confidence index on the accuracy of its current coordinates, and its peer list information. In this study, we focus on buggy implementations that cause a host to provide faulty coordinate values when queried by other nodes. We model a buggy node as follows. Instead of computing its coordinates at each iteration, a buggy node draws its coordinates randomly from some fixed range at each coordinate update step. We assume that the confidence index reported by each node is bug-free: i.e., it maintains its weighted moving average of prediction error correctly using the randomly generated coordinates. As a result, buggy nodes will generally report low confidence on their coordinates.

- Malicious nodes. A malicious node may intentionally provide faulty information to confuse or mislead other nodes. In this study, we model a rather naive form of malicious attack: a malicious node reports a randomly generated coordinates to other nodes and claims that the coordinates have low prediction error (i.e., high confidence index). This will cause the other nodes to put high weights on the faulty coordinates. In reality, there may be different levels of malicious attacks; a malicious node may intentionally pick an arbitrary sequence of coordinates to report to other nodes to cause even greater system instability than randomly generated coordinates. The effects of more sophisticated malicious attacks are left for future work.

We vary the fraction of faulty nodes in the system from 0 up to 80%. Figures 4-26 and 4-27 present PCoord's prediction accuracy as a function of the fraction of buggy and malicious nodes respectively. Each node uses random peer sampling with a default sample batch size of ten samples per coordinate update. The average median

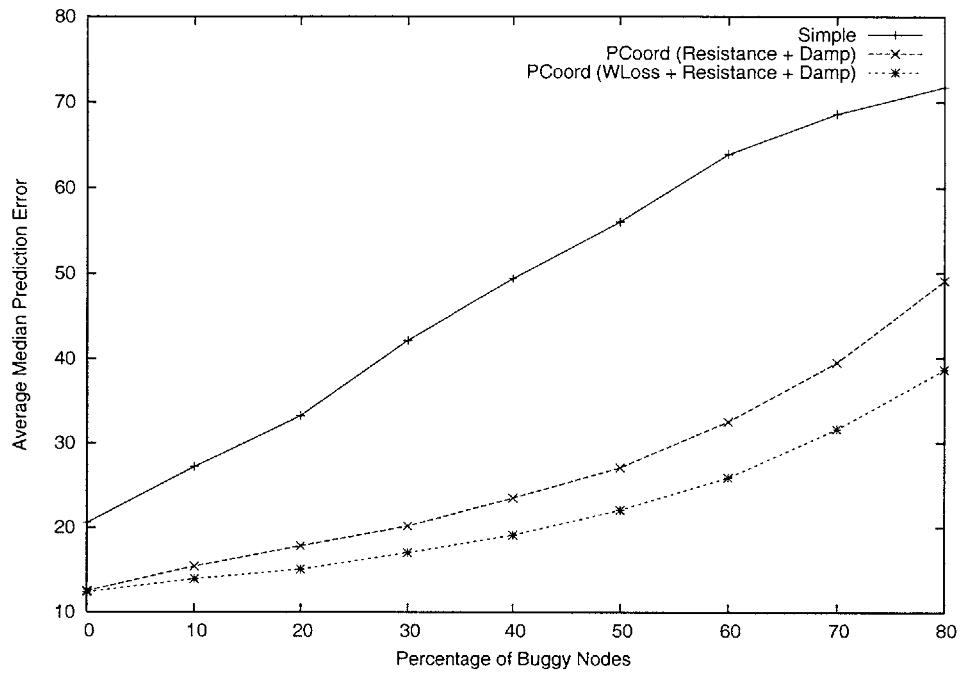


Figure 4-26: Compare different PCoord's mechanisms under various fractions of buggy nodes. King data set. $N = 1740$, $M = 10$, $D = 5$.

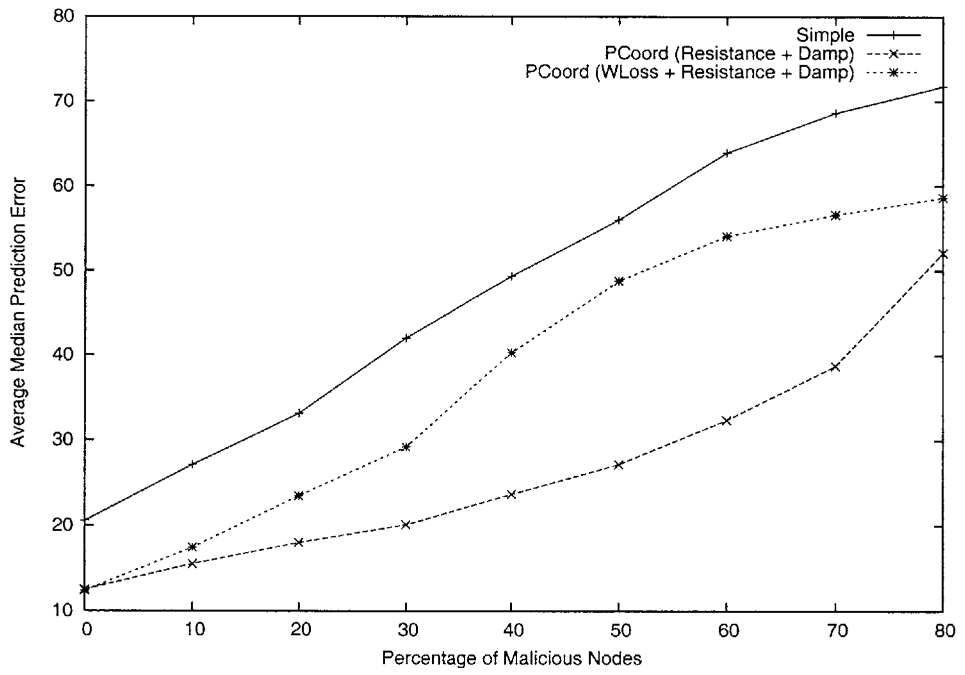


Figure 4-27: Compare different PCoord's mechanisms under various fractions of malicious nodes. King data set. $N = 1740$, $M = 10$, $D = 5$.

prediction accuracy represents the steady-state prediction error of the system averaged over time using statistics gathered after 10 seconds of simulated time, with a total simulated time of 20 seconds.

In order to understand how different PCoord mechanisms are affected by faulty information, we present PCoord's performance with different combinations of the mechanisms:

- Simple: this is the Simple algorithm without weighted loss, resistance, or damping.
- Resistance + Damp: this is a version of PCoord that implements the resistance and damping mechanisms.
- PCoord (WLoss + Resistance + Damp): this is the default PCoord algorithm with all three mechanisms turned on.

Since the Simple algorithm does not associate weights with samples, Simple's performance is the same under buggy and malicious modes. The same is true for PCoord when the weighted loss function is turned off.

We first examine PCoord's performance under the buggy mode. We observe that the Simple algorithm's prediction error rises rapidly as the percentage of faulty nodes increases. When 30% of the nodes are faulty, Simple's prediction error doubles from 20 to 40 ms. In contrast, it takes 60% faulty nodes for PCoord's prediction error to double. When 30% of the nodes are faulty, PCoord's prediction error increases by less than 40%. Both versions of PCoord (with or without the weighted loss function) are significantly more robust than the Simple algorithm in the face of high percentage of buggy nodes.

Under the malicious model, faulty nodes always report zero prediction error (and thus a perfect confidence index) on its randomly generated coordinates. This causes the non-faulty nodes to place a high weight on faulty coordinates. Our results indicate that when 10% of the population are malicious, PCoord with weighted loss only does slightly worse than when weighted loss is turned off. However, the prediction accuracy of PCoord with the weighted loss turned on can degrade quickly under heavy malicious attacks. In particular, when the fraction of malicious nodes are up to 50% of the total population, PCoord does more than 40% better by turning off the weighted loss mechanism. In general, the combination of damping and resistance provides a fairly robust mechanism in coping with both buggy and malicious nodes. This suggests that, in a real-world deployment, a good engineering choice may be to turn off the weighted loss function if a large number of malicious nodes is expected.

4.13 Conclusions

In this section, we have evaluated PCoord using simulations, and compared its performance with that of Vivaldi and the FixedLM scheme. Our primary focus is to evaluate the number of samples it takes for the system to converge to a low prediction error. In particular, we have examined PCoord's convergence behavior in several different scenarios: (1) the simultaneous-join scenario involving a 1740-node network with all nodes joining the coordinate system at approximately the same time, (2) the incremental join scenario, in which we evaluate the number of samples required for a newly joined node to converge to a low prediction error when the rest of the system has already converged, and (3) the high churn scenario, in which the system experiences continuous membership changes with dynamic node join and leave.

We summarize our findings for the simultaneous join scenario as follow.

- Under the simultaneous join scenario in the King data set, PCoord can achieve comparable performance as the FixedLM scheme after each host updates its coordinates using approximately 100 to 120 samples. In particular, it takes PCoord 100 to 120 samples, or approximately 10 to 12 coordinate updates, for the system median prediction error to decrease to the 12 ms range, which is competitive with the FixedLM's median prediction error at 12.16 ms when using 10 landmarks.

- Under the simultaneous join scenario, we have compared PCoord with Vivaldi using various sampling strategies. In general, Vivaldi takes more samples to converge. It takes Vivaldi over 300 samples to reach the 12 ms range using both the Half-Near-K neighbors and Random-Global configurations. After 100 samples, Vivaldi's median prediction errors are 24.6 ms and 15.8 ms respectively when setting C_c to be 0.25 and 1.0 respectively in the Half-Near-K neighbor configuration.

- It takes more samples for both PCoord and Vivaldi to reach the same 95th percentile prediction error range comparable to that of the FixedLM scheme. The 95th percentile prediction error of the FixedLM scheme is 32 ms. For PCoord, it takes approximately 180 to 200 samples to converge to 32 ms for 95th percentile prediction error, and it takes Vivaldi approximately 500 samples to converge to the same 95th percentile error range.

We have also demonstrated in this chapter that, it takes significantly fewer samples for a newly joined node to converge to low prediction error when the rest of the system has already converged. In particular, with the King data set, we observe that, 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). In comparison, the median prediction error of the FixedLM scheme using fixed 20 landmarks is 11.4 ms.

Further, we have demonstrated that PCoord is robust under high churn. In particular, we have examined PCoord's prediction accuracy when nodes join and leave continuously with exponentially distributed mean host session life times of 2 to 40 seconds. We have shown that dynamic join and leave have minimal effects on PCoord's prediction accuracy when the host's mean session life time is 10 seconds or longer. This suggests that PCoord can 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 60 minutes [45].

Finally, we have examined PCoord's robustness against faulty information. We model two types of faults: buggy implementations that report randomly-generated coordinates, and malicious nodes which report random coordinates with misleadingly high confidence index on those coordinates. Our results suggest that PCoord cope with buggy implementations effectively. We have also observed that the weighted loss function is helpful in guarding against buggy implementations; however, its performance is very sensitive to false information due to malicious nodes. In general, the combination of damping and resistance provides a fairly robust mechanism in coping with both buggy and malicious nodes. Our results suggest that, in a real-world deployment, a good engineering choice may be to turn off the weighted loss function if a large number of malicious nodes is expected.

In conclusion, PCoord is able to achieve prediction accuracy comparable to the FixedLM scheme while providing better scalability, flexibility and fault tolerance. PCoord does not rely on fixed landmark nodes, and allows hosts to sample *any* other peers in the system to construct their coordinates. Under a scenario where nodes join incrementally, PCoord is able to achieve competitive prediction accuracy as the FixedLM scheme using the same number of samples. Under a more challenging scenario when all nodes join simultaneously, we have demonstrated that using a 1740-node real Internet measurements PCoord can achieve the same level of accuracy as the FixedLM scheme with approximately 100 - 165 samples per host, which is half as many samples as what Vivaldi would require to converge to the same level of prediction accuracy.

Chapter 5

Effect of Triangle Inequality Violations

In the previous chapter, we have demonstrated that PCoord can achieve low prediction error. In this chapter, we ask the following questions.

- What are the sources of error in a decentralized coordinate system such as PCoord?

- How does the prediction error of PCoord vary as a function of the actual path length? Does PCoord tend to over- or under-predict the path length?

- How does triangle inequality violation affect prediction accuracy of decentralized coordinate systems such as PCoord?

The rest of this chapter is organized as follows. We first present a taxonomy of the sources of error in a decentralized coordinate system. We then study the error characteristics of PCoord and Vivaldi as a function of the path lengths. Finally, we examine prediction accuracy of PCoord and Vivaldi as a function of the degree of triangle inequality violations in the data set.

5.1 Sources of Error

In order to facilitate our discussions of where errors come from, we break down the types of errors in a decentralized coordinate system into the following categories.

- Structural error: this type of error is due to the mismatch between the host space (which we would like to model) and the geometric space used to model it. In the Internet distance prediction context, if we use distances to all pairs of nodes to build our geometric model, the structural error is the difference between the actual and the geometric distances using an optimal embedding with global knowledge. Example factors that determine the structural error of a geometric model include the distance function, dimensionalities of the geometric space, and properties of the host space.

- Prediction or Sampling error: this is due to the fact that we do not have perfect global distance information. In particular, each node uses a small set of sampled distances to embed itself in the geometric space.

- Algorithmic error: The algorithmic error is due to limitations of the algorithms we use to derive the embedding. For example, the algorithmic error in PCoord could be due to the parameter settings in the Simplex Downhill algorithm used for embedding. Example parameters include the tolerance value and maximum number of function evaluations used to define the stopping criteria in the Simplex Downhill minimization procedure.

- Distributed error: Another source of error is due to the fact that the embedding is done in a decentralized fashion; each node computes its own coordinates that minimize the error relative to other nodes' coordinates. Since all nodes update their own coordinates in a parallel, independent fashion, the resulting coordinates will likely have a higher prediction error than if the coordinates were generated in a centralized fashion.

- Other error: examples in this category include errors due to noise, latency measurements, and faulty information from buggy implementations or malicious attacks.

In Chapter 4, we focused on the sampling error, and examined how the prediction error of PCoord and Vivaldi decreases as the number of samples increases. In this chapter, we put more focus on the structural error. In particular, we examine how violations of the triangle inequality property affect the overall prediction accuracy of PCoord and Vivaldi.

5.2 Error Statistics by RTT Groups

In this section, we investigate the performance properties of both Vivaldi and PCoord by classifying the evaluated paths into groups of 25 ms each (i.e., groups of [0,25), [25,50), [50,75) ms, etc.).

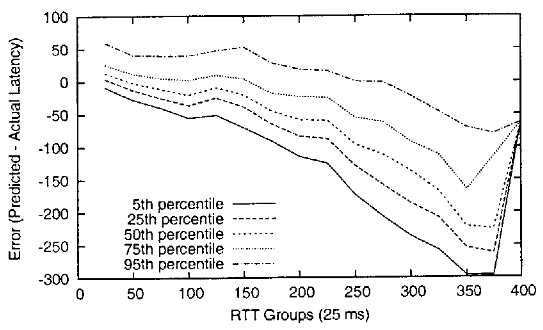
5.2.1 Statistics by RTT Groups on PlanetLab Data Set

Figures 5-1 and 5-2 show the error statistics of PCoord and Vivaldi respectively with the PlanetLab data set. We observe that for both PCoord and Vivaldi, the errors are "symmetrical" for smaller RTT groups (below 150 ms) in that the distribution of errors due to over- and under-predictions are approximately the same. However, for larger RTT groups, both schemes tend to under-predict long distances.

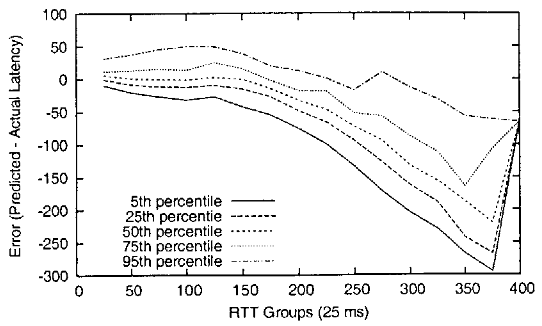
We have also experimented with embedding the PlanetLab data set using a 5-dimensional coordinate space with $M = 10$ (results not shown). Our results suggest that while increasing the dimensionality and the batch size M improve the prediction accuracy, they do not qualitatively change the error characteristics among these RTT groups.

5.2.2 Statistics by RTT Groups on King Data Set

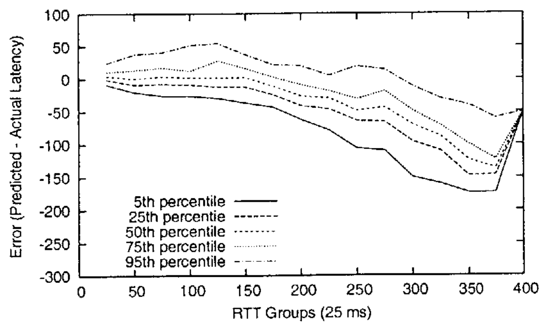
Next, we show the error statistics by RTT groups of both schemes using the King data set. Figure 5-3 shows the histogram of King distances in 25 ms groups. Figure 5-4



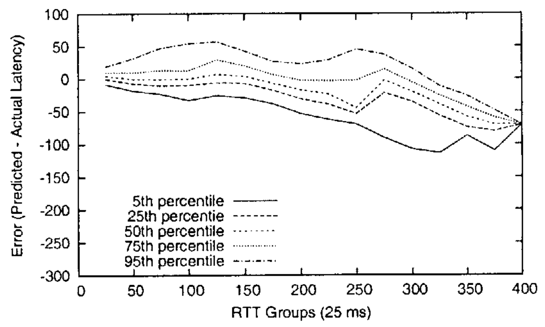
(a) One second



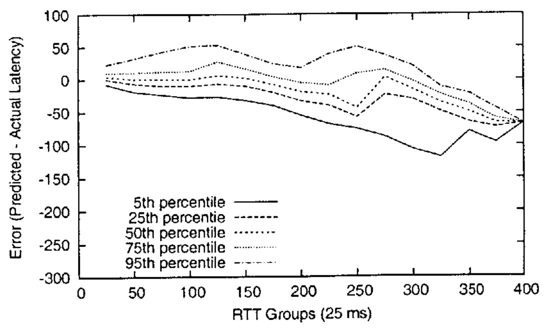
(b) Three seconds



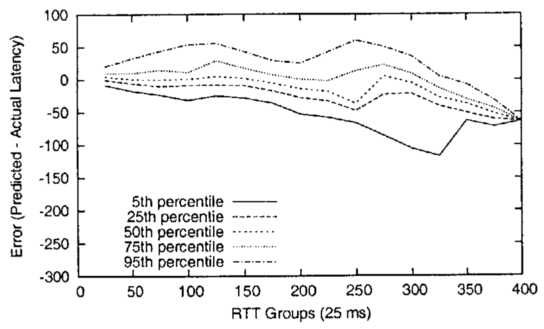
(c) Five seconds



(d) Ten seconds

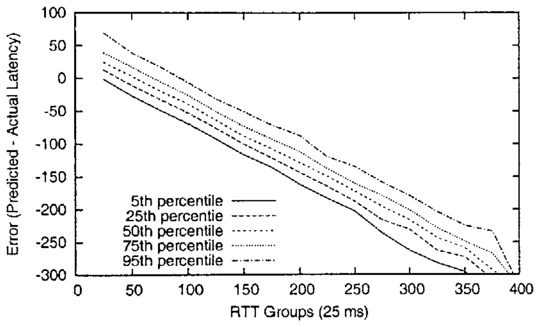


(e) Fifteen seconds

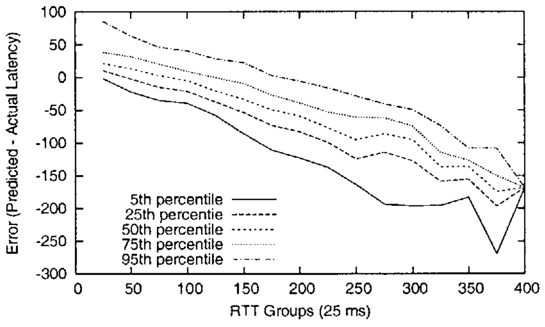


(f) Twenty Seconds

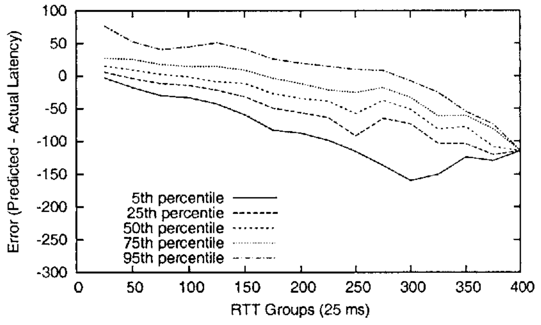
Figure 5-1: PCoord, error statistics by RTT groups. PlanetLab, $N = 127$, $D = 3$, $M = 6$.



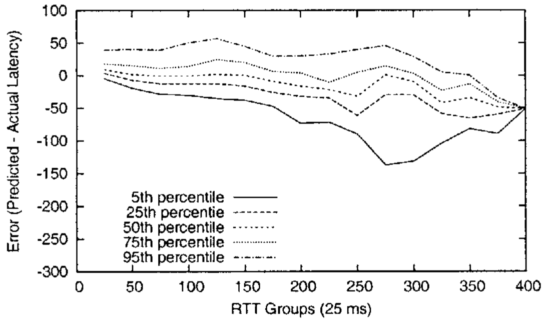
(a) One second



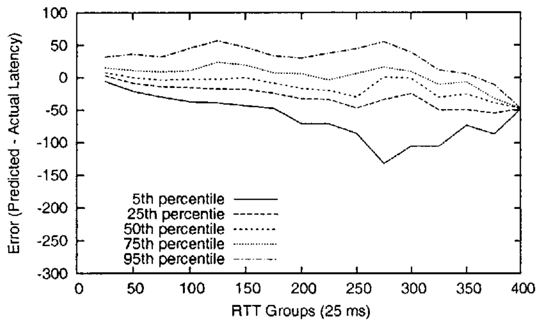
(b) Three seconds



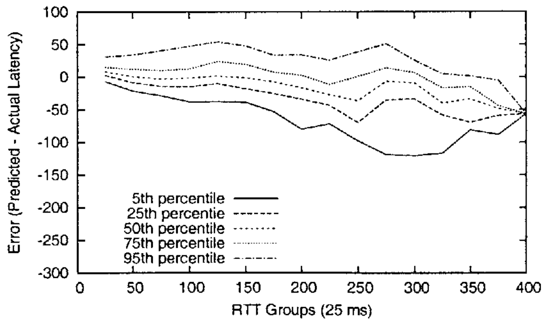
(c) Five seconds



(d) Ten seconds



(e) Fifteen seconds



(f) Twenty seconds

Figure 5-2: Vivaldi, error statistics by RTT groups. Half-Near-K neighbors configuration ($K = 6$), timestep constant $C_c = 0.25$. PlanetLab, $N = 127$, $D = 3$.

shows the RTT error statistics before any coordinate updates are performed. Since all nodes initialize their coordinates to the origin, the predicted distance among all nodes is zero initially. As a result, the under-prediction of distance between a pair of nodes equals the RTT exactly. This applies to both PCoord and Vivaldi, as both schemes initialize nodes' coordinates to the origin in the beginning.

Figure 5-5 and 5-6 show the error statistics of PCoord when 10 and 30 reference points are used respectively with the King data set. Figure 5-7 shows the Vivaldi error statistics with the King data set. We make the following observations.

- Consistent with our observations in the PlanetLab data set, both PCoord and Vivaldi have large under-prediction errors for some fraction of large RTT paths in the King data set.

- The worst under-prediction errors for PCoord (10 reference points) appear in the 525 ms and 800 ms groups, with median prediction errors of -143.3 and -89.3 ms respectively (see ten seconds scenario in Figure 5-5). For Vivaldi, the worst median errors after 20 seconds of running Vivaldi also appear in the 525 ms and 800 ms groups with median error of -152 and -182 ms respectively. We will seek possible explanation of this phenomena in a later section.

- Further, increasing the number of samples in both PCoord and Vivaldi does not seem to rectify the under-prediction problem. PCoord's error distribution among different RTT bins does not change significantly after 5 seconds of simulated time. We thus speculate that the large under-prediction errors for these long latency paths are mostly due to structural error, for example, from triangle inequality violations. We will explore this issue further in Section 5.3.

- Finally, increasing the batch size M in PCoord from 10 to 30 (see Figure 5-6) does not significantly change the error characteristics among these RTT bins. In particular, the 525ms and 800 ms groups remain to be the most problematic RTT groups when 30 reference points are used.

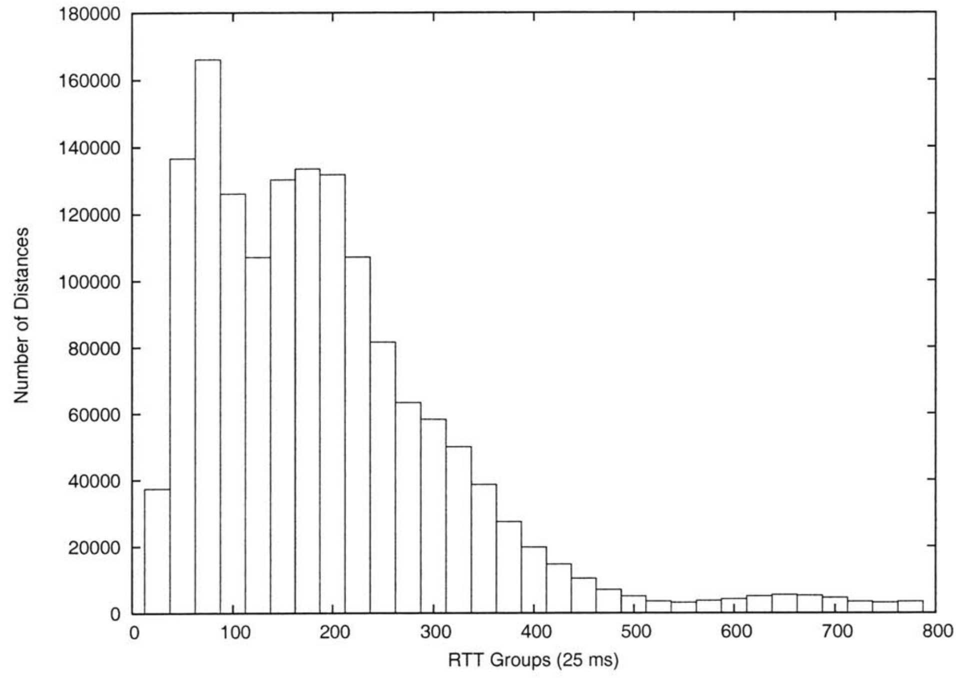


Figure 5-3: King data set. RTT group histogram. $N = 1740$.

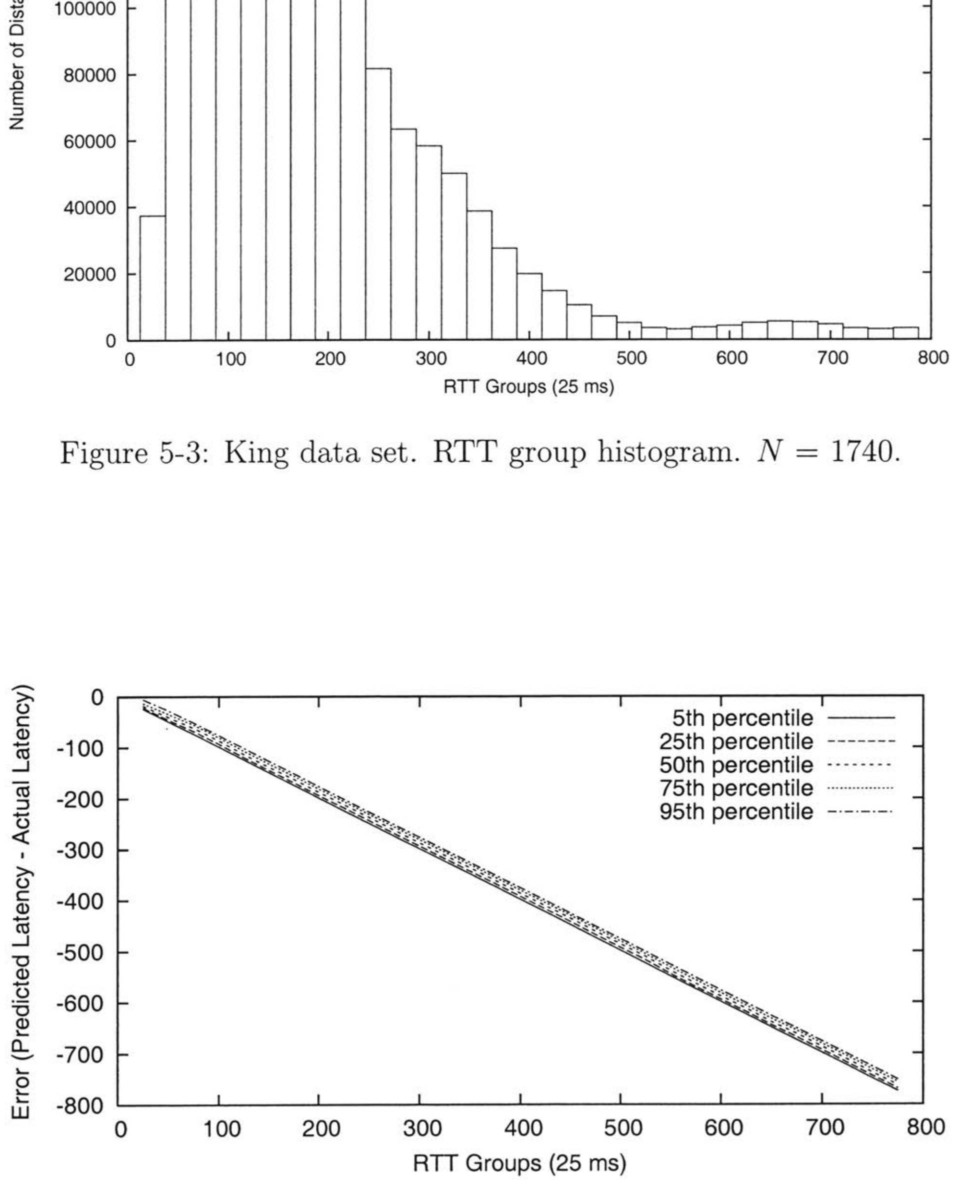
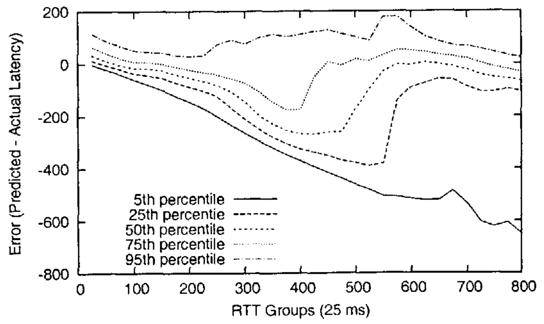
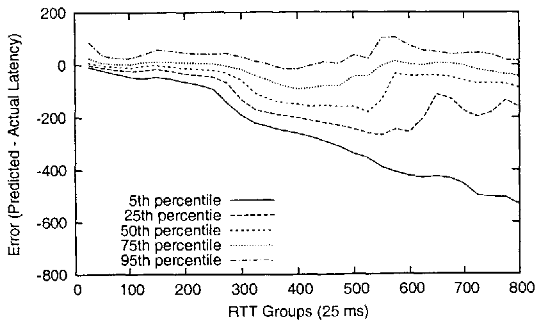


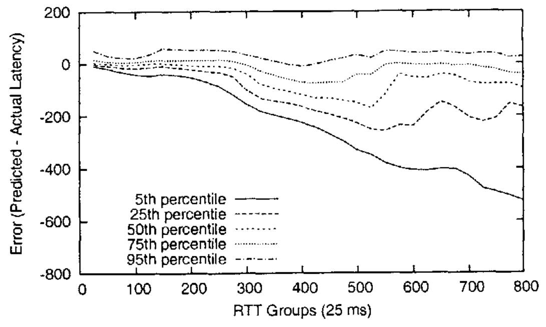
Figure 5-4: King RTT error statistics by RTT groups before any coordinate updates are performed by PCoord or Vivaldi, both of which initialize coordinates to the origin.



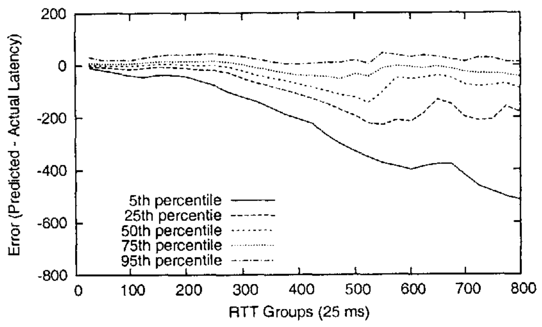
(a) One second



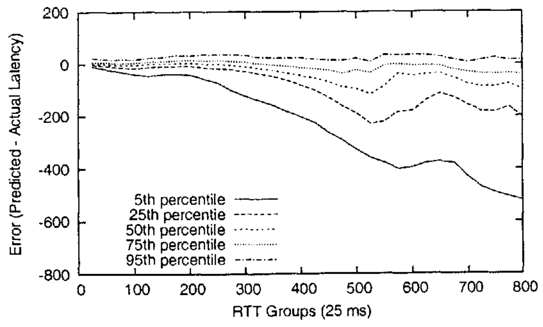
(b) Three seconds



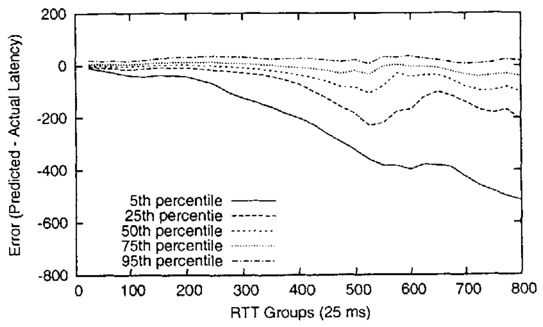
(c) Five seconds



(d) Ten seconds

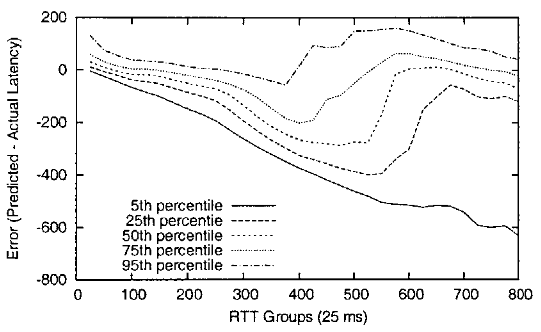


(e) Fifteen seconds

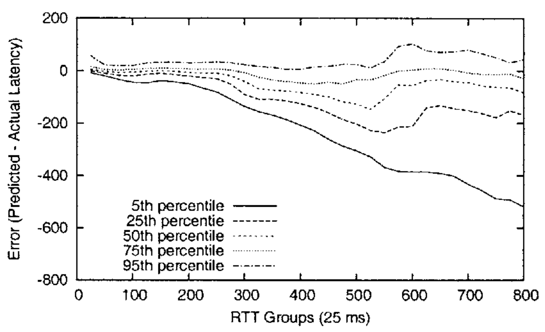


(f) Twenty seconds

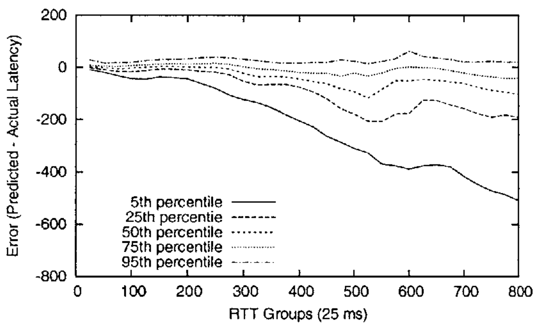
Figure 5-5: PCoord Error statistics by RTT groups, 10 reference points. King, $N = 1740$, $D = 5$.



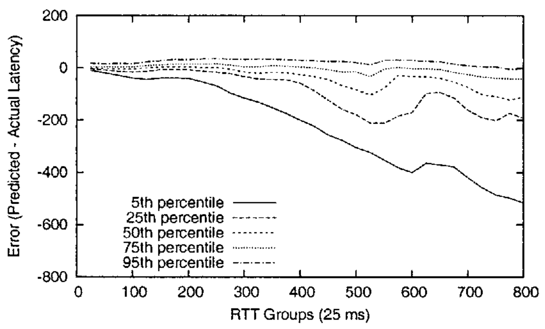
(a) One second



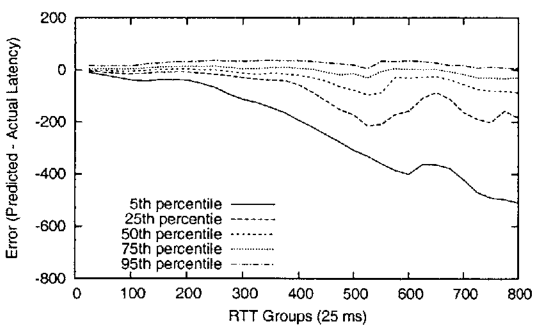
(b) Three seconds



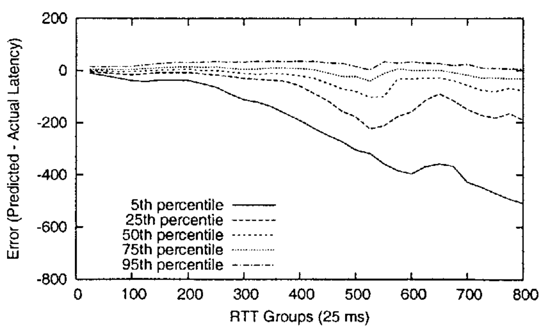
(c) Five seconds



(d) Ten seconds

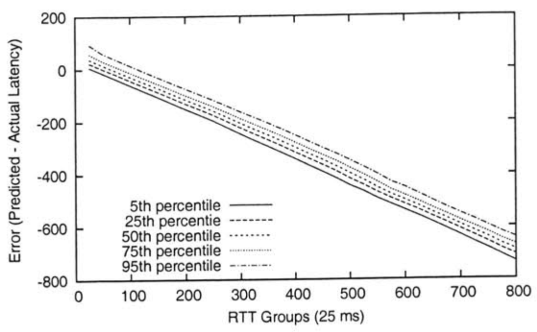


(e) Fifteen seconds

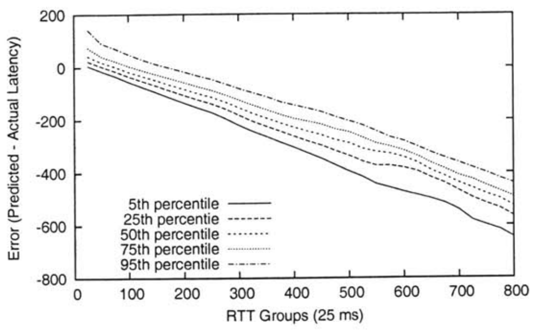


(f) Twenty seconds

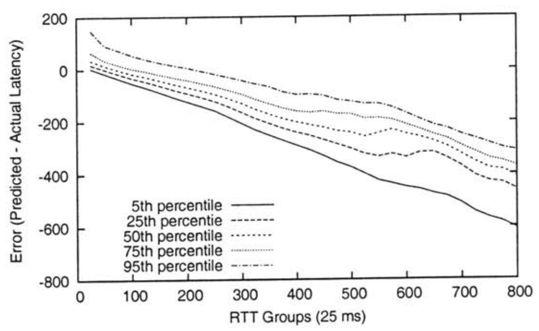
Figure 5-6: PCoord Error statistics by RTT groups, 30 reference points. King, $N = 1740$, $D = 5$.



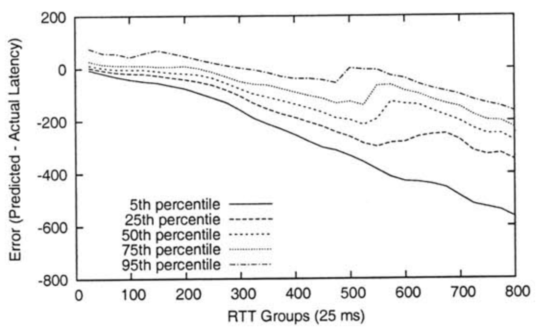
(a) One second



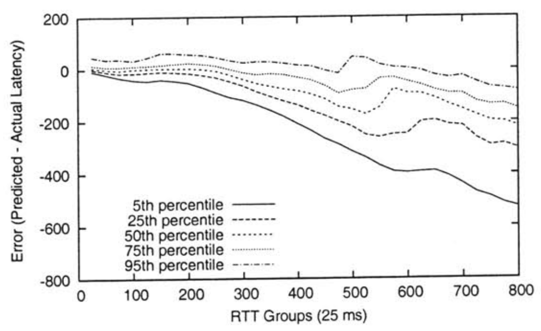
(b) Three seconds



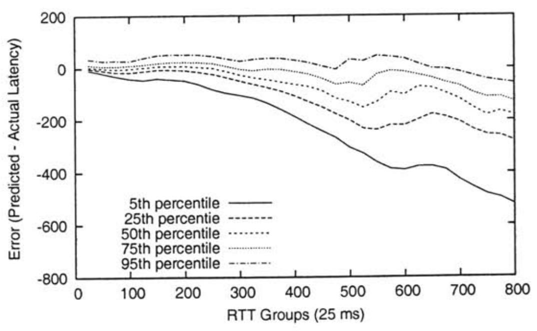
(c) Five seconds



(d) Ten seconds



(e) Fifteen seconds



(f) Twenty seconds

Figure 5-7: Vivaldi Error statistics by RTT groups, Half Near-K neighbors configuration ($K = 64$). King, $N = 1740$, $D = 5$.

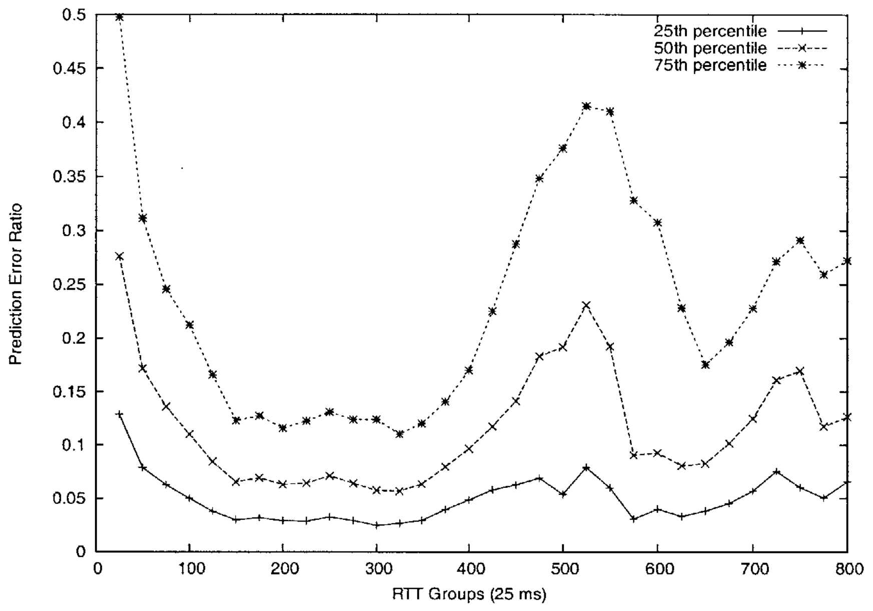


Figure 5-8: PCoord, Error statistics in terms of absolute prediction error ratio (PER) by RTT groups after 20 seconds of running PCoord. King, N = 1740, D = 5, M = 10.

While we present errors in terms of the difference between the actual and the predicted distances, the relative errors in each RTT groups can be easily inferred as the ratio between the errors shown and the RTT bounds for that RTT group. In Figure 5-8, we present the error statistics in terms of absolute prediction error ratio (PER) by RTT groups after 20 seconds of running PCoord when M is ten. Short RTTs tend to have slightly higher error ratio due to the smaller denominator in the ratio. The error ratios of paths in the 525 ms or longer RTT groups are higher than the other RTT groups. This is consistent with results in previous figures which involve error statistics by RTT groups in terms of directional prediction error.

5.3 Triangle Inequality

Previously, we have seen that both PCoord and Vivaldi have high prediction error for distances with larger RTTs. In particular, PCoord and Vivaldi tend to under-predict large RTTs, and increasing the number of samples used for prediction does not seem to rectify this under-prediction problem. We speculate that the high prediction error in the RTT groups with long latencies is due to structural error, for example from triangle inequality violation. To validate this intuition, we examine the degree of triangle inequality violation for both the King and the PlanetLab data set, and its implications for the prediction accuracy of decentralized network coordinate systems.

Similar to the Vivaldi work [10], we declare that the path between a node pair i and j violates the triangle inequality if it violates the constraint $RTT(i,k) + RTT(j,k) > RTT(i,j) - \epsilon$, where k is another node in the data set $(k \neq i, j)$, and ϵ is a constant used to avoid counting marginal violations due to measurement errors. The Vivaldi work [10] reported that 4.5% of the triples in the King data set violate triangle inequality when ϵ is 5 ms. However, it remains unclear how violation of the triangle inequality impacts the accuracy of the geometric mapping. Here, instead of reporting triangle inequality violations across all triples, we examine the triangle inequality on a per path basis to examine how the degree of violation impacts the accuracy of geometric mapping.

We use the *violation ratio* and *violation amount* as two representative indicators of the degree to which a path violates triangle inequality. The path triangle inequality violation ratio and amount measure how frequently and by how much a path between two nodes violate the triangle inequality relative to another node in the data set. We define the two metrics more precisely below.

Path Triangle Inequality Violation Ratio

We define the path triangle inequality violation ratio as follows: for each pair of nodes i and j in the data set, we check how often the RTT between i and j violates the triangle inequality when using another node k $(k \neq i, j)$ to form a triangle. More specifically, for each path formed by a node pair i and j, we count the number of times the path violates the constraint $RTT(i,k) + RTT(j,k) > RTT(i,j) - \epsilon$, where k is another node in the data set and $(k \neq i, j)$. The path violation ratio of the path formed by i and j is the number of violations divided by the total number of triples formed by i, j, and a third node in the data set.

Path Triangle Inequality Violation Amount

The path triangle inequality violation amount measures by how much a path between two nodes violates the triangle inequality relative to another node in the data set. More specifically, the violation amount of a node pair i and j relative to a third node k is $|RTT(i,k)+RTT(j,k)-RTT(i,j)|$ when the constraint $RTT(i,k)+RTT(j,k) > RTT(i,j) - \epsilon$ is violated, and is zero otherwise.

5.3.1 Violation Ratio of King and PlanetLab Data Set

Figure 5-9 plots the path triangle inequality violation ratio for both the King and PlanetLab data set when ϵ = 5ms. Our data suggests that approximately 65% of the paths in the King data set has a zero violation ratio when $\epsilon = 5ms$, indicating that most of the paths satisfy the triangle inequality at all times with respect to triangles formed by using all other nodes in the data set. We also note that 95% of the King paths have path violation ratio of 0.15 or less.

For the PlanetLab data set, only 40% of the paths have zero violation ratio. Approximately 95% of the paths have violation ratio of 0.41 or less. For both King and PlanetLab data set, there are approximately 2-3% of the paths that violate the triangle inequality over half the time – for these paths, more than half the times, an indirect path going through another intermediate node in the system would have provided lower RTT than the direct path.

5.3.2 Violation in Different RTT Groups

In Section 5.2, we observed that both Vivaldi and PCoord heavily under-predict distances in the 525 ms RTT group (i.e., RTTs between 500 to 525 ms). In this section, we examine the relationship between the degree of violation of a path and its length. Figures 5-10 and 5-11 plot the summary statistics of path violation ratio in different RTT groups in the King and PlanetLab data respectively. We divide the paths into groups of 25 ms each, and plot the 5th, 25th, 50th, 75th, and 95th percentile path violation ratio in each RTT group. We note that in general the paths

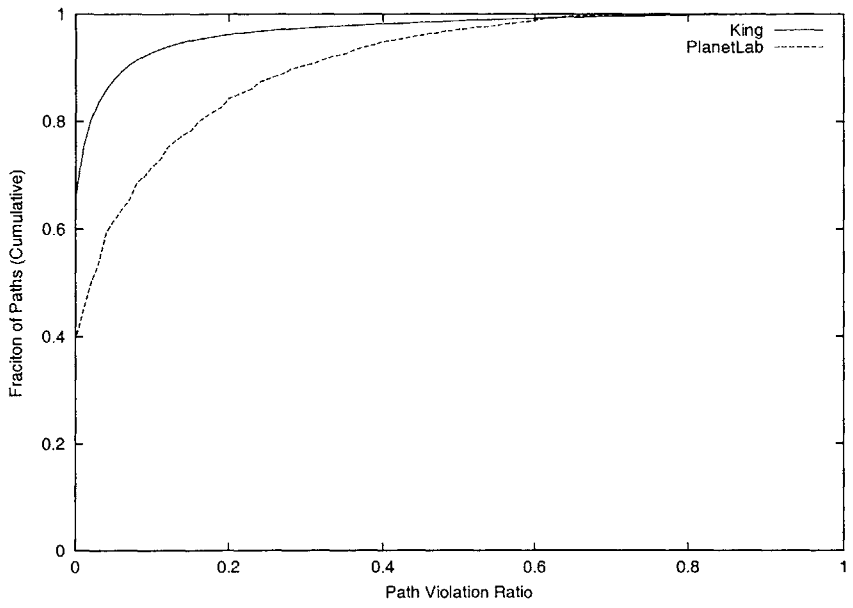


Figure 5-9: Cumulative distribution of path violation ratio of King and PlanetLab data.

with small RTTs have low violation ratio. The median violation ratio is close to 0 for paths that are less than 300 ms. Median violation ratio is in fact rather low across all RTT groups, except for the paths between 450 to 600 ms range. In particular, the 525 ms RTT group has the highest path violation in comparison to all other RTT groups – more than half the paths in that group have violation ratio greater than 0.2. This suggests that the high prediction error of both PCoord and Vivaldi in the 525 ms RTT group are mostly due to triangle inequality violations. We also observe that the 95th percentile path violation increases rapidly as RTTs increase. For the PlanetLab data, the paths with the highest violation fraction fall in the 250 ms and 375 ms RTT groups.

Figure 5-12 plots the violation amount as a function of the path's RTT. For each path, we define the (conditional) median path violation amount of path (i, j) to be the median of all non-zero violation amounts of (i, j) in all the triples formed by the path. A path in a triple that does not violate triangle inequality has zero violation amount, which is not included in the statistics, so that we get a better understanding of the distribution of violation amount when a violation does occur. We observe a very similar pattern for the distribution of the violation amount across the RTT

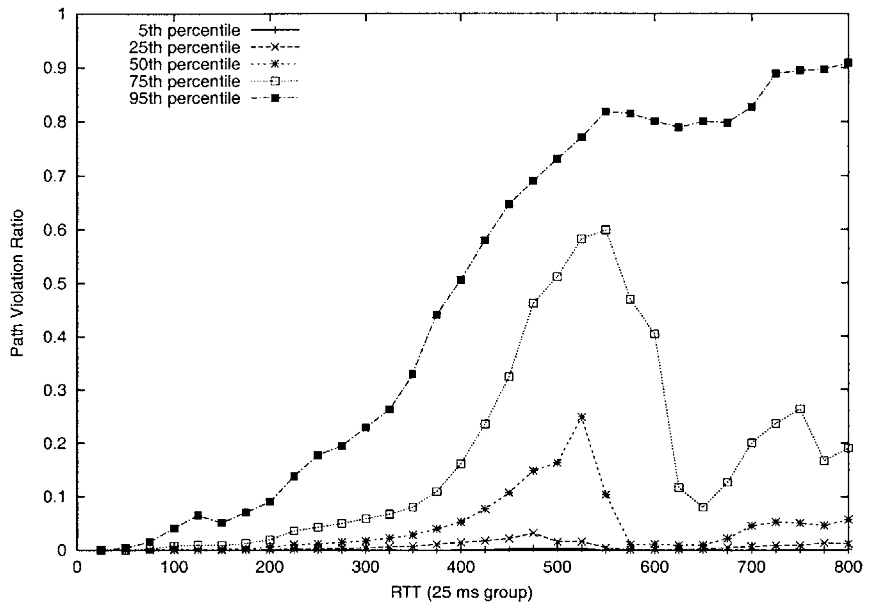


Figure 5-10: Summary statistics of path violation ratio in different RTT groups (25ms group). King data, $N = 1740$.

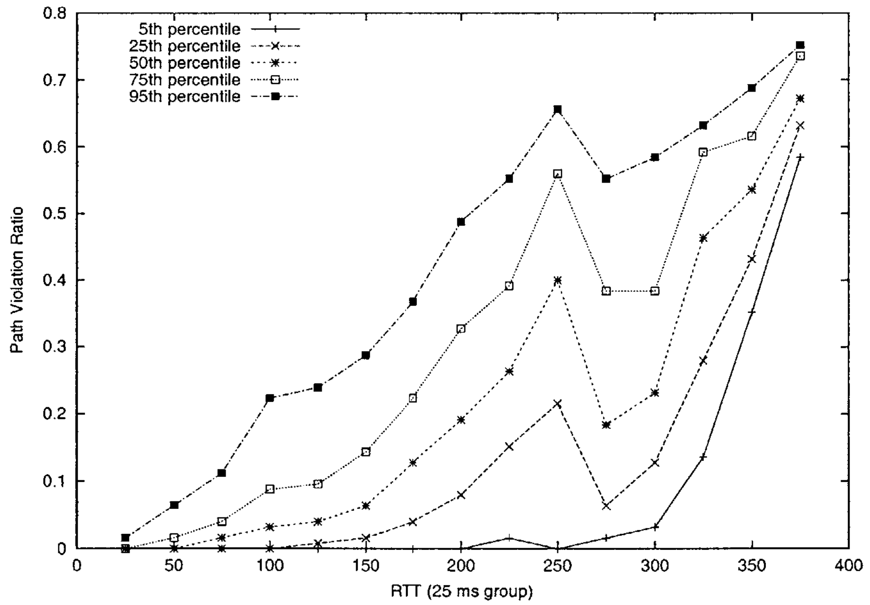


Figure 5-11: Summary statistics of path violation ratio in different RTT groups (25ms group). PlanetLab data, $N = 127$.

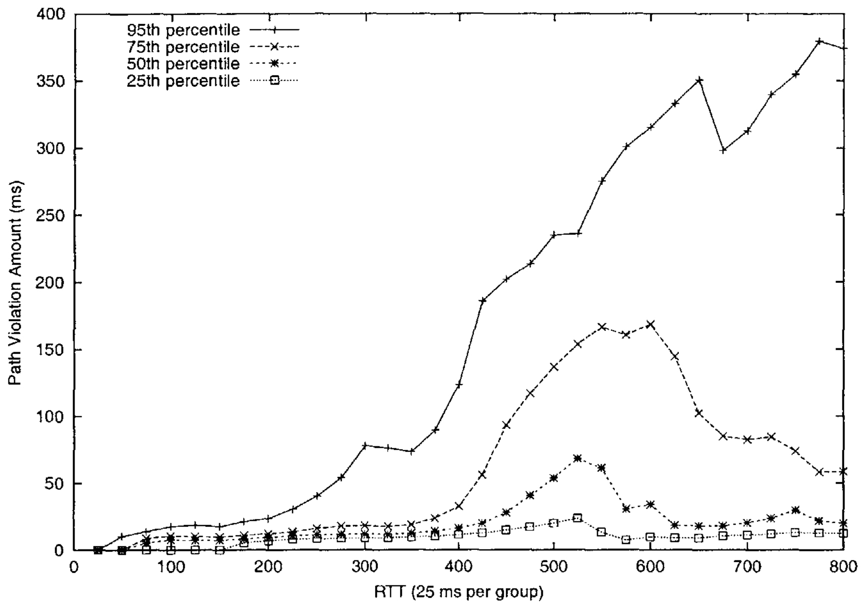


Figure 5-12: Summary statistics of path median violation amount in different RTT groups (25ms group). King, $N = 1740$.

groups as the violation ratio distribution. Median violation amount is in fact rather modest (less than 30 ms) across all RTT groups, except for the paths between 450 to 600 ms range. Further, by comparing Figures 5-12 and 5-5, we note that the under-predictions by PCoord in each RTT category is proportional to the violation amount in the corresponding RTT group.

5.3.3 RTT and Violation Amount in Different Violation Ratio Groups

Violation ratio only captures one aspect of the degree of violation of a path. Intuitively, paths with high violation ratios can still have accurate Euclidean mappings, if most of the violations are small. Similarly, paths with high violation amounts may not impact prediction accuracy too much, if such violation happens very rarely. Therefore, the accuracy of the Euclidean mapping is likely affected by the combined effect of violation ratio and amount. In this section, we classify the paths in the King data set based on their violation ratio into groups of 0.01 each (i.e., groups of [0,0.01), [0.01,0.02), [0.02,0.03), ..., etc), and present the RTT distribution and

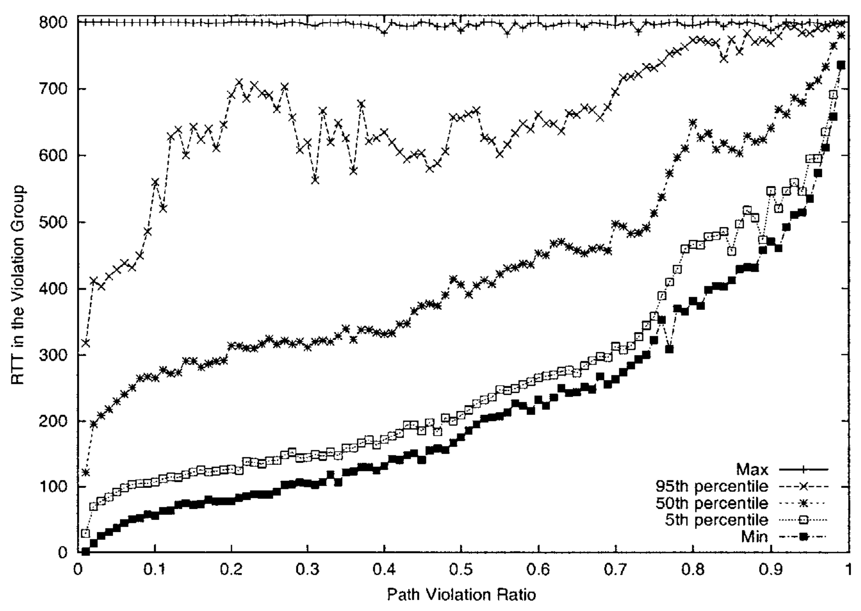


Figure 5-13: Summary statistics of RTTs of paths in each triangle inequality violation ratio group (violation ratio of 0.01 per group). King, $N = 1740$.

violation amount in each group.

Figure 5-13 plots the minimum, maximum, median, 5th and 95th percentile RTT of paths in each violation group. In general, we observe that short RTTs tend to have low path violation ratios, and long RTTs tend to have high violation ratios. We observe that almost all paths with zero or small violation ratio (less than 0.01) are less than 300 ms. We also note that the minimum RTT of paths with violation ratio greater than 0.8 is 300 ms. This means that high violation paths are strictly for paths with large RTTs. This should not come as a surprise, since the longer the path is, the more probable it is for an indirect path through a third node on the Internet shortcut the direct path.

Figure 5-14 plots the violation amount as a function of the path's violation ratio. Again, the violation ratio is divided into groups of 0.01 each. We then plot the 95th and 50th percentile path median violation amount in each violation group. We note that for paths with violation ratio of 0.4 or less, the median violation amount is rather small (less than 43 ms). However, beyond violation of 0.4, the violation amount doubles for every 0.2 increment in the violation ratio: e.g., the median violation amount is 77 ms for the 0.5 violation ratio groups; whereas the violation amount is

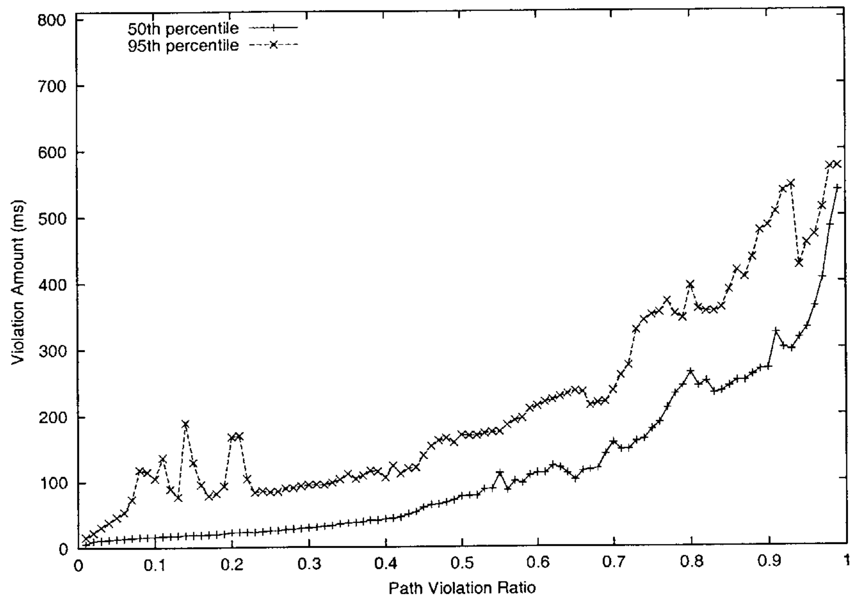


Figure 5-14: Median violation amount of paths in each triangle inequality violation group (violation ratio of 0.01 per group). King data.

more than doubled (158 ms) for paths in the 0.7 violation ratio group.

5.3.4 Prediction Accuracy as a Function of Path Violation Ratio

Next, we examine the effect of triangle inequality violation on PCoord and Vivaldi prediction accuracy. We take the coordinates generated by PCoord and Vivaldi after convergence (after approximately 50 seconds of running the algorithms), and compute the directional prediction error (DPE) and directional prediction error ratio (DPER) for each path. These metrics were defined in Chapter 4. A DPE or DPER of zero indicate perfect prediction. A negative and positive DPE/DPER indicates under-prediction and over-prediction respectively.

Figure 5-15 plots the directional prediction error ratio (DPER) as a function of degree of violation. Figure 5-16 plots the directional prediction error (DPE). We classify all paths based on their triangle inequality violation ratio into groups of 0.01 each, and plot the 5th, 95th percentile, and median DPE/DPER in each violation group. We note that PCoord and Vivaldi have nearly identical error distribution

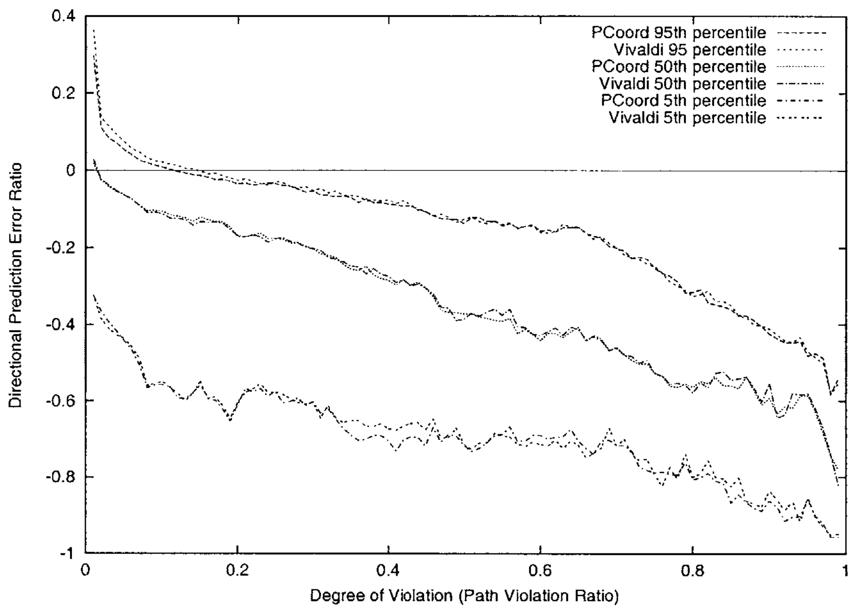


Figure 5-15: Directional prediction error ratio (DPER) as a function of degree of violation (measured as path violation ratio). King, $N = 1740$, $D = 5$, $M = 10$.

in each group, as their 5th, 95th, and median errors nearly overlap. We make the following observations, which apply to both PCoord and Vivaldi.

- When the violation ratio is less than 0.01, both PCoord and Vivaldi have a small positive median DPER near zero for paths in this group, and the errors are evenly distributed among over- and under-predictions for both PCoord and Vivaldi. We recall that about 75% of the paths have violation ratio less than 0.01. This suggests that for a majority of the paths in the King data set, PCoord and Vivaldi have a chance to embed these paths in the Euclidean space with relatively low error.

- As the violation ratio increases, errors due to under-prediction start to dominate. For paths with violation ratio greater than 0.15, the 95th percentile error becomes negative, indicating that almost all errors are due to under-predictions.

- The prediction accuracy of both PCoord and Vivaldi drops significantly as the violation ratio rises.

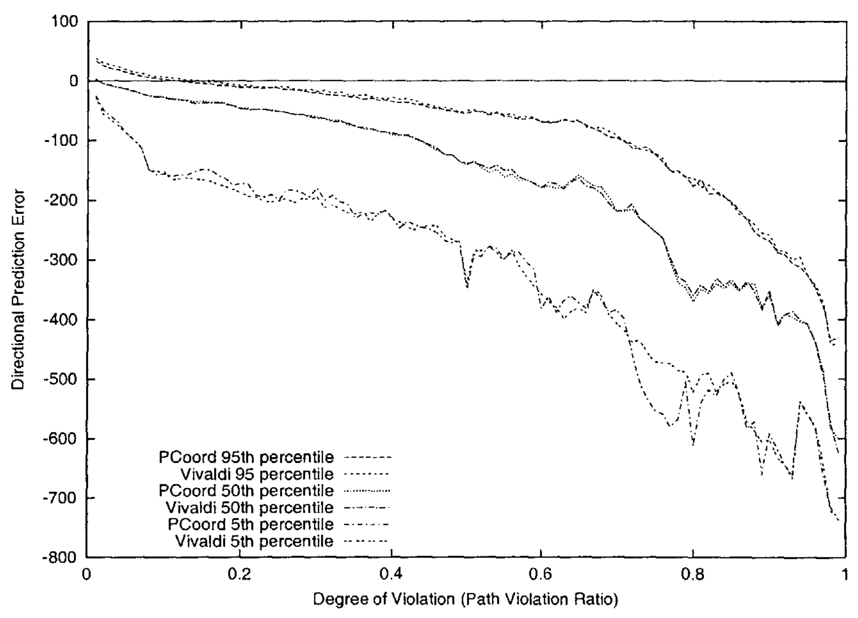


Figure 5-16: Summary statistics of directional prediction error (predicted - actual RTT) as a function of path violation ratio. King, $N = 1740$, $D = 5$, $M = 10$.

5.3.5 Prediction Accuracy as a Function of Mean Path Violation Amount

In this section, we examine PCoord prediction accuracy as a function of a path's mean violation amount. In contrast to the median path violation amount defined in an earlier section, the mean path violation amount we present here includes zero violation amount when computing the mean. We believe that the mean, instead of the conditional mean, of a path's violation amount has more impact on the prediction accuracy of the Euclidean mapping. (We have plotted the prediction accuracy as a function of conditional mean, and the correlation between the two seems to be somewhat more erratic: paths with large conditional mean can have very small prediction error.)

Figure 5-17 plots the the 95th, 50th, and 5th percentile prediction error ratio of PCoord as a function of mean path violation amount. The coordinates are generated after 50 seconds of running the PCoord algorithm, where each node uses ten reference points in each coordinate update. As expected, the prediction error ratio increases as the mean violation amount increases.

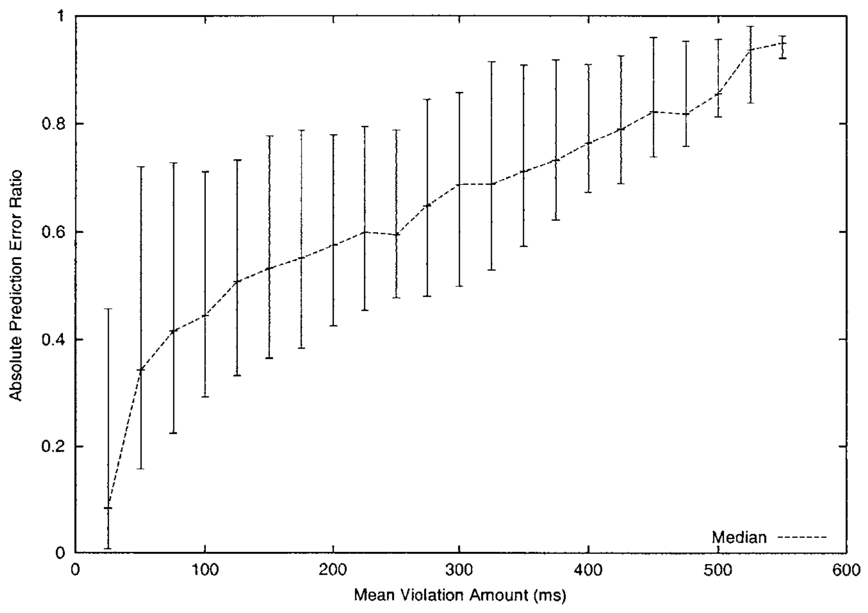


Figure 5-17: Summary statistics (95th, 50th, and 5th percentile) of absolute prediction error ratio (PER) of PCoord as a function of mean path violation amount. Mean path violation amount is classified into 25 ms per group. King, N = 1740, D = 5, M = 10. The PCoord coordinates are generated after 50 seconds of running the algorithm.

5.4 Conclusions

In this chapter, we presented a taxonomy of the sources of prediction errors in a decentralized network coordinate system. We then study the error characteristics of PCoord and Vivaldi as a function of the path lengths. We observed that both PCoord and Vivaldi tend to under-predict distances with large RTTs in the King and PlanetLab data sets.

We examined how the degree of triangle inequality violations in network distances impacts the prediction accuracy of PCoord and Vivaldi. Our results suggest that the under-predictions of large RTTs in the King and PlanetLab data sets are primarily due to violations of the triangle inequality property in the network distances. Our results suggest that increasing the number of samples in both PCoord and Vivaldi does not seem to rectify the under-prediction problem of these paths. We argue that these mis-predictions are mostly due to structural errors stemming from inherent mismatches between the host space (which we would like to model) and the geometric space used to model it. As a result, increasing the number of samples is not likely to improve the under-predictions of these large RTTs.

Chapter 6

Exploring the PCoord Framework

In previous chapters, we have evaluated PCoord using real, large-scale Internet latency data, which in general does not satisfy the triangle inequality property. In this chapter, we ask whether PCoord can do significantly better when the underlying distance measurements among nodes have no triangle inequality violations or when the underlying distances are in fact Euclidean distances. We examine PCoord's convergence behavior and error characteristics for latencies with little or no triangle inequality violations. We focus on the following questions:

- How low is the system error and how many samples does it take for PCoord to converge when the actual network distances have no triangle inequality violations?
- If the underlying metric space is Euclidean, can PCoord recover the topology? If so, how many samples does it take for PCoord to converge?

Another related issue of interest is how the dimensionality of the geometric space affects the structural error (i.e., the error due to the mismatch between the geometric space and the general metric space modeled) for topologies with varying degrees of triangle inequality violations. Does increased dimensionality improve prediction accuracy when the degree of triangle violation is high?

To answer the above questions, we use the RON2 data set collected as part of the RON (Resilient Overlay Network) project [43, 2] to test the performance of PCoord.

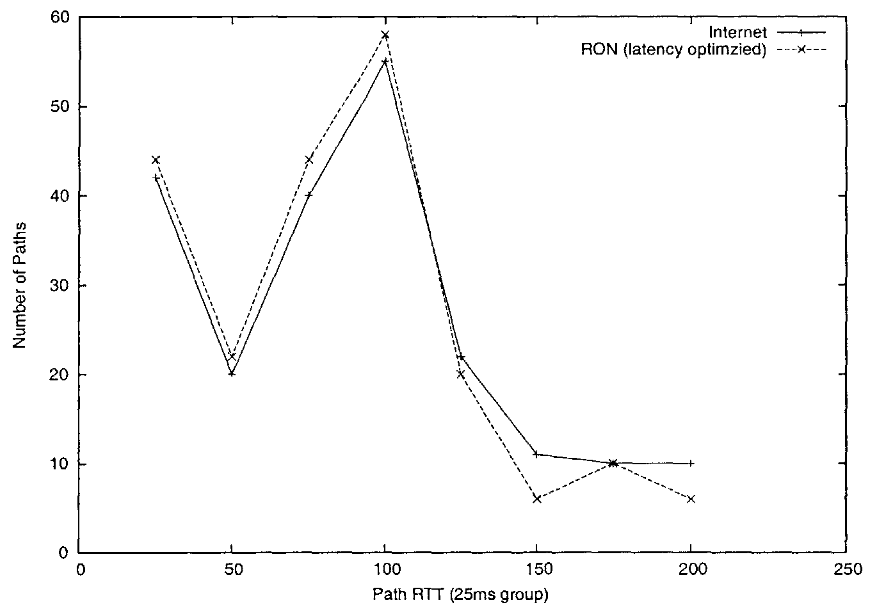


Figure 6-1: RTT bin size distribution for RON2 Internet and latency optimized data.

RON2 data set measures the RTTs among 15 Internet hosts between May 5th to May 11th in 2001. A total of over 3 million latency samples were gathered. For each pair of nodes, we use the minimum latency value in the sampling period as its inter-node distance. RON2 collects latency data using two different routing schemes: the default Internet routes and the latency-optimized paths that use RON nodes in the overlay to forward packets. Since the latency optimized RON paths have noticeably less triangle inequality violations than the default Internet paths, examining PCoord's performance using the two data sets will give us a good idea on how the degree of triangle inequality violations impacts performance.

The median RTTs of the Internet and latency-optimized RON2 data are 77.25 ms and 71.69 ms respectively. Figure 6-1 plots the latency distribution of the two data sets. We note that the two data sets have nearly identical RTT distributions – the latency-optimized data has slightly more shorter RTTs than the Internet data.

We construct four different 15x15 latency matrices, three of which are based on the RON measurements:

- Internet RON: about 40% of the paths violate triangle inequality. The maximum violation amount is 93.30 ms.

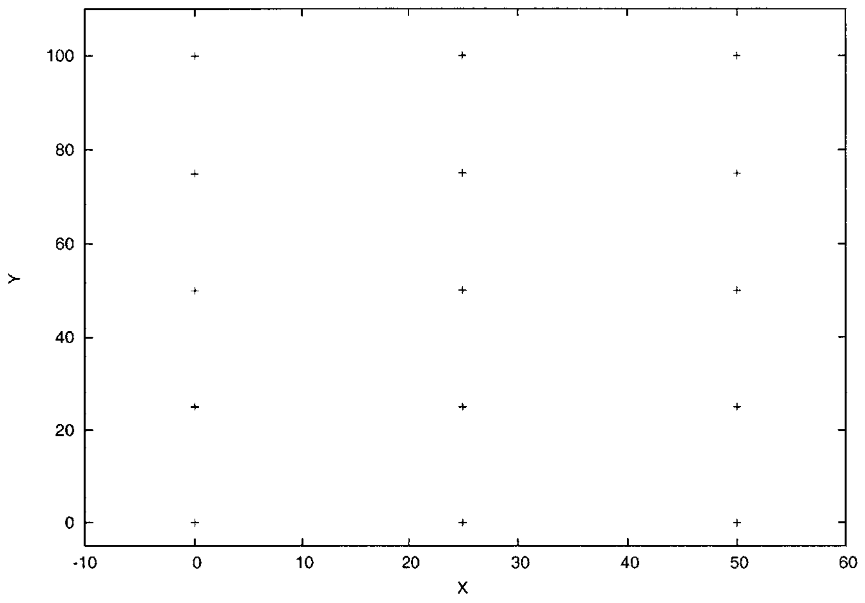


Figure 6-2: A 15 node topology with 3x5 grid on a 2-D Euclidean plane.

- Latency Optimized RON: about 27.6% of the paths violate triangle inequality. The maximum violation amount is 9.68 ms.
- Perfect RON: we add 12 ms to each path in the latency-optimized RON2 data to generate an artificial latency matrix with no triangle inequality violations.
- 15-node Grid: we generate a 15-node topology with a 3 by 5 grid on a 2-D plane, where the inter-node latencies are simply their Euclidean distances on the plane.

6.1 PCoord Convergence Using RON Data Set

Figure 6-3 shows the convergence of PCoord using the three RON-based data sets. To eliminate the effect of sampling error, each node uses latencies to all other 14 nodes to construct coordinates at each coordinate iteration. The lowest error after convergence is primarily from the structural error. We note that the number of samples required for convergence is approximately the same for all three data sets. As expected, the latency-optimized and perfect RON data sets can achieve lower system prediction error than the Internet RON data set.

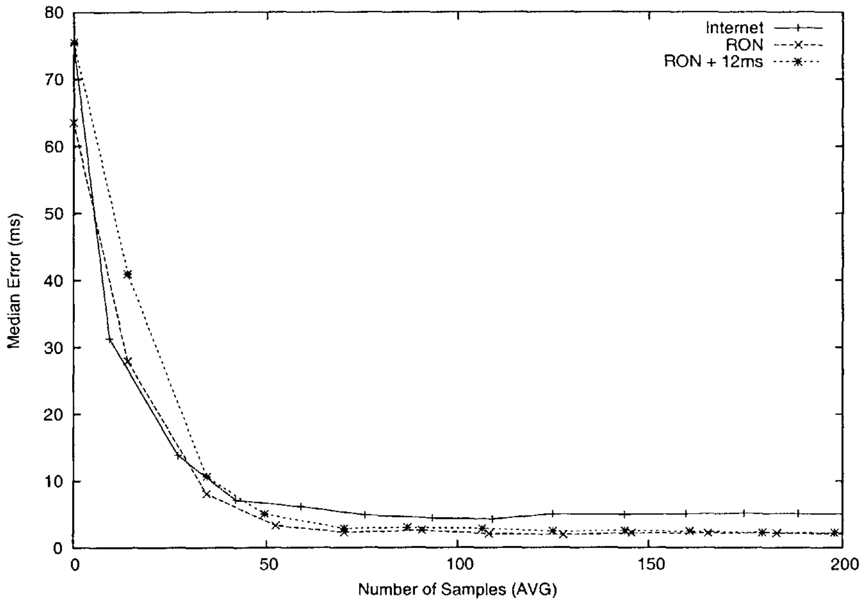


Figure 6-3: PCoord error convergence in terms of number of samples for three different RON2 data sets: Internet paths, RON2 latency optimized paths, and the RON2 latency optimized paths RTT + 12 ms to remove all triangle inequality violation. N = 15, M = 14 in all three data sets.

Figure 6-4 presents the prediction error ratio distribution of the three data sets after 20 seconds of running PCoord. We note that PCoord is able to predict the latency-optimized and perfect RON data sets significantly better than the Internet RON data set.

6.1.1 PCoord Convergence with Euclidean Distances

So far we have demonstrated that PCoord can converge to a low prediction error using real Internet measurements with varying degrees of triangle inequality violation. In this section, we present PCoord's convergence behavior when the underlying distances are in fact Euclidean distances using the 3x5 grid topology we described earlier.

Figure 6-5 shows the convergence of PCoord when the node latencies in fact form a Euclidean metric space. We present Vivaldi's performance as a comparison. We note that both PCoord and Vivaldi are able to reconstruct the grid topology. However, the convergence of the 15-node grid requires approximately 60-70 samples, which is no better than the Internet RON measurements involving the same number of nodes.

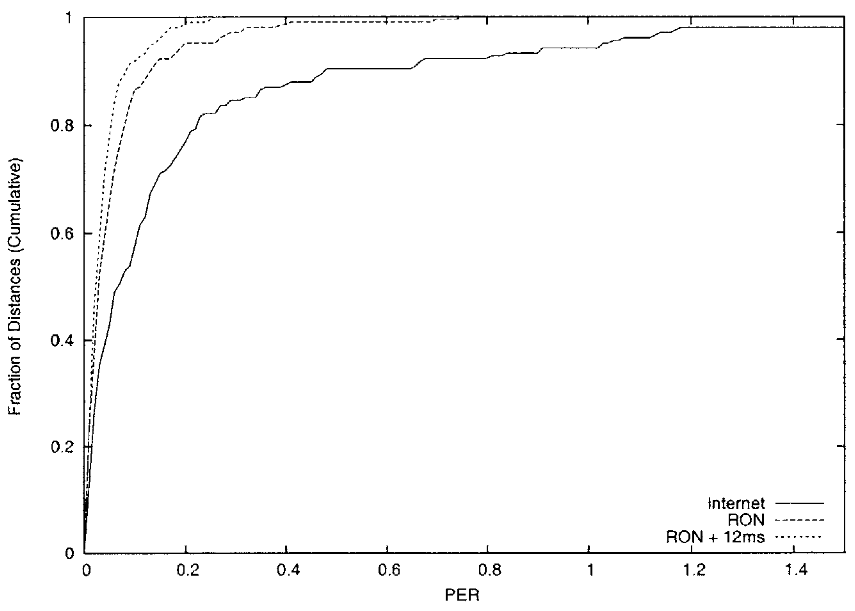


Figure 6-4: PCoord absolute prediction error ratio (PER) distribution for three different RON2 data sets, $N = 15$, $M = 14$, $D = 5$.

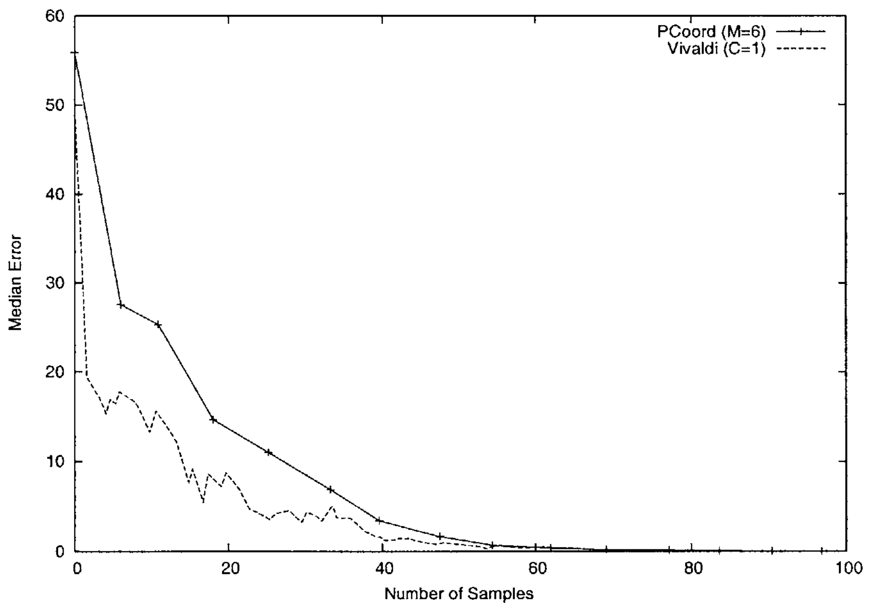


Figure 6-5: PCoord error convergence when latencies are Euclidean distances among 15 (3x5) nodes from a 3x5 grid, $N = 15$, $D = 2$.

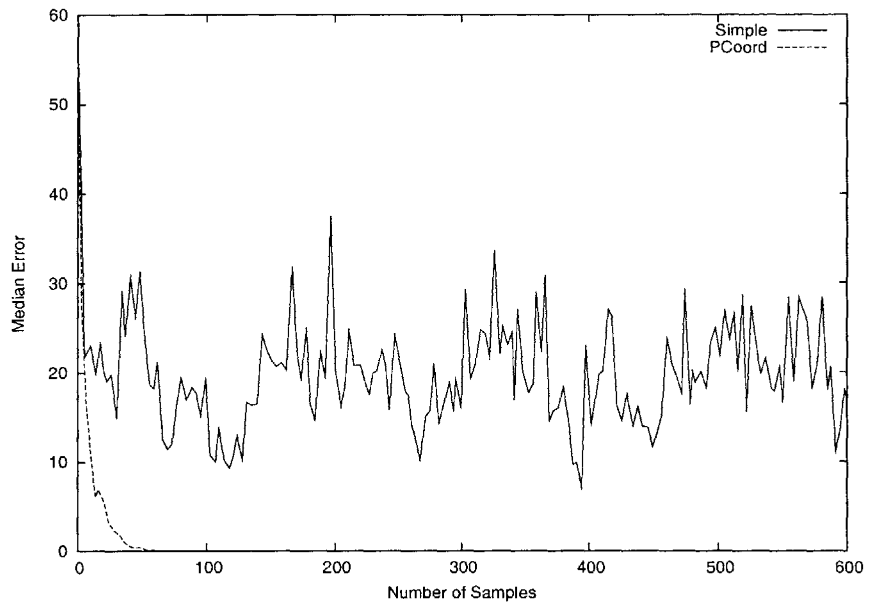


Figure 6-6: PCoord error convergence in the grid topology. The Simple algorithm does not converge. M=3, $D = 2$.

Next, we compare PCoord's performance with the Simple algorithm and show that the "resistance" factor added in the weighted loss function is in fact critical for convergence in some cases. Figure 6-6 shows the median prediction error of PCoord and the Simple algorithm. Both algorithms use three reference points for each coordinate update. For the PCoord algorithm, we have turned off the damping effect using the fit error, so that the only difference between PCoord and simple is the weighted loss function with the added resistance term. Figure 6-6 shows that PCoord converges in 60-70 samples; however, the Simple algorithm, without the resistance factor, tends to oscillate and never converges for the entire simulation period with each node doing the updates using over 1000 samples. Increasing the batch size M for the simple algorithm solves the convergence problem. However, in general, the system tends to oscillate for the Simple algorithm when the number of reference points used per batch is small relative to the dimensionality of the geometric space.

6.2 PCoord Prediction Error by RTT Groups

In a previous chapter, we presented PCoord's error statistics across different RTT groups. We observed large under-prediction errors for larger RTTs, and presented evidence that these under-predictions are due to triangle inequality violations in the data set, instead of a general performance characteristics of the PCoord algorithm. In order to validate this claim, we present the error statistics of PCoord using the three RON-based data set in figure 6-7.

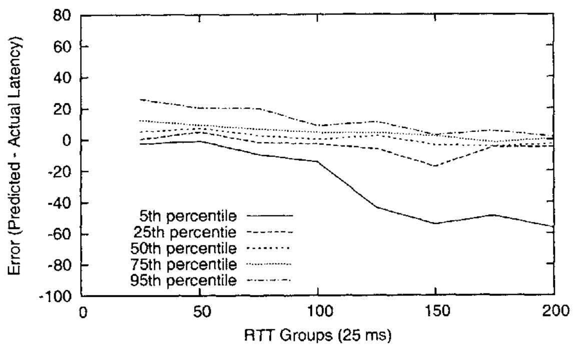
For the Internet RON2 data set, we again observe large under-prediction error for large RTTs due to triangle inequality violation. There is also over-prediction of shorter RTTs. By reducing the degree of triangle inequality violation, both under- and over-predictions are greatly attenuated in the latency-optimized case. The perfect RON2 data set incurs very small prediction error, symmetrical around zero, across all RTT groups.

6.3 Dimensionality of Euclidean Approximation

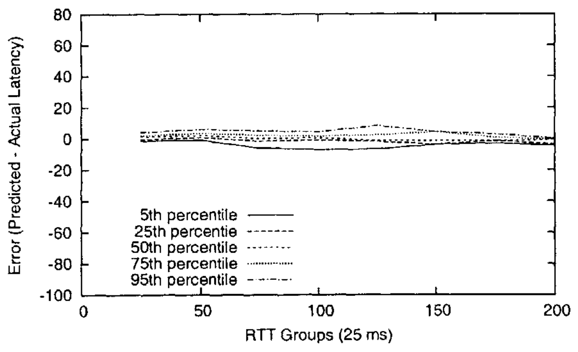
In this section, we examine the effect of dimensionality of Euclidean approximation on the prediction accuracy of the PCoord scheme.

6.3.1 Effect of Dimensionality on King Data Set

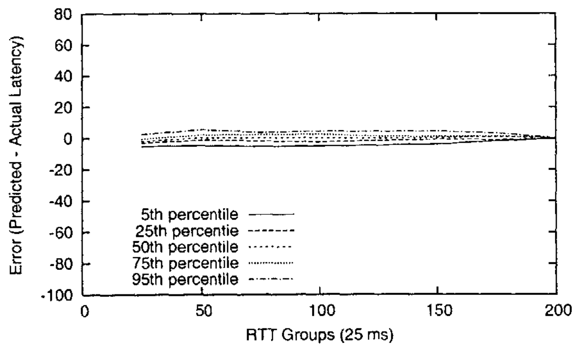
Figure 6-8 plots the error convergence for the PCoord scheme using varying dimensionalities. The number of reference points is fixed at ten. Our results indicate that significant performance improvement can be observed as the dimensionality increases from one to five. The incremental performance improvement beyond five dimensions, however, is very small.



(a) Internet RON2



(b) Latency-Optimized RON2



(c) Perfect RON2

Figure 6-7: PCoord error statistics by RTT groups in three different RON2-based data sets, $N = 15$, $D = 5$, $M = 14$ in all three cases.

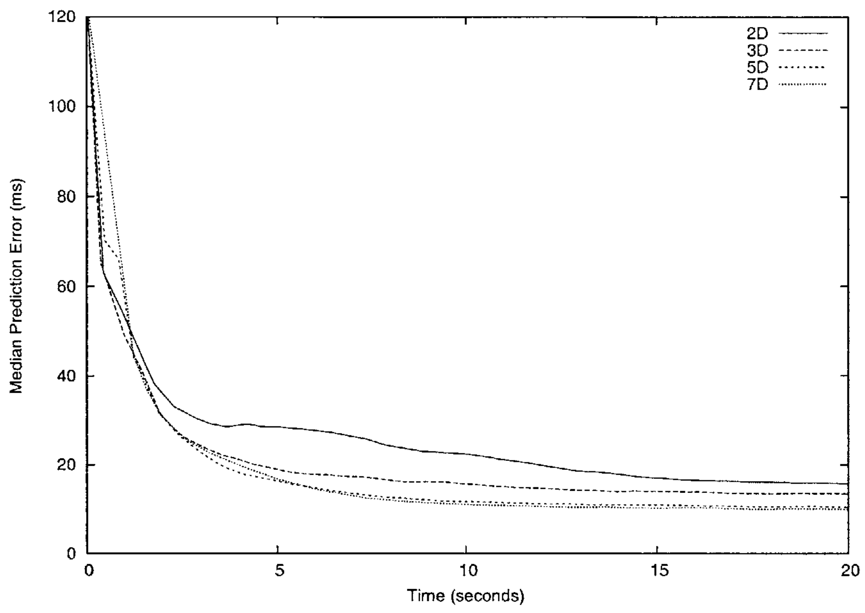


Figure 6-8: The effect of dimensionality. Plot shows PCoord median prediction error with various dimensionalities when 10 reference points are used at each coordinate update. King, N = 1740.

6.3.2 Effect of Dimensionality on Paths with High Violation Ratio

Next, we examine the effect of dimensionality of the Euclidean mapping on PCoord's prediction accuracy of paths with high triangle inequality violations. Figure 6-9 shows PCoord's distribution of prediction error ratio for paths with low violation ratio (less than or equal to 0.3) vs. paths with high violation ratio (more than 0.3). We show the cumulative distribution of prediction error ratio for these two groups for both 2 and 5 dimensional PCoord coordinates. We make the following two observations.

- The prediction accuracy of paths in the low violation group is significantly better than those in the high violation group. Using 5 dimensional coordinates, more than 83% of the paths in the low violation group have prediction error less than 0.25; whereas, in the high violation group, only 30% of the paths have prediction error less than 0.25.

- The 80th percentile prediction error for the 5 and 2 dimensional coordinates in the low violation group are 0.22 and 0.32 respectively. So, increasing the dimen-

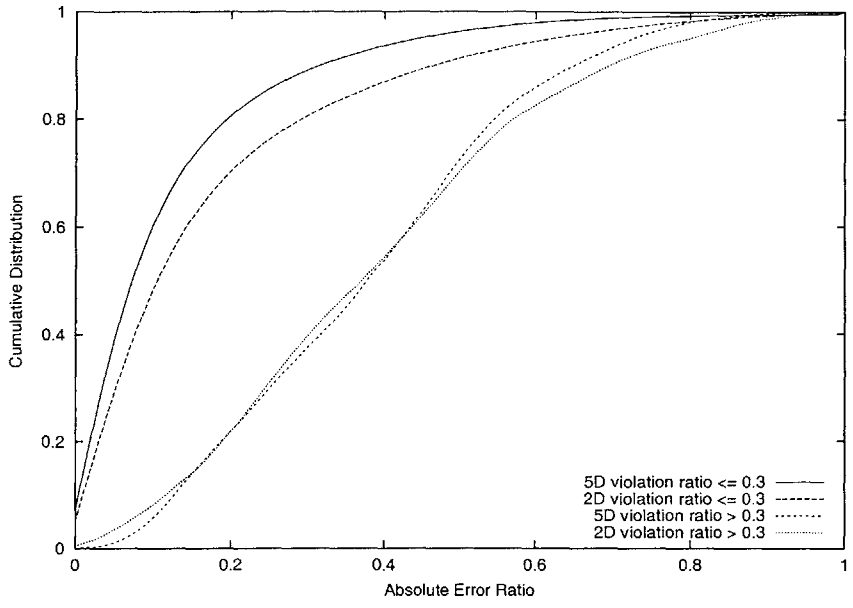


Figure 6-9: PCoord distribution of prediction error ratio for paths with low violation ratio (less than or equal to 0.3) vs. paths with high violation ratio (more than 0.3). Plot shows the distribution of errors for 5 and 2 dimensional coordinates. King, $N = 1740$, $M = 10$.

sionality from 2 to 5 helps improve the prediction accuracy by more than 30% for paths in the low violation group. However, there is no obvious improvement in prediction accuracy in using higher dimensional coordinates when the paths have triangle inequality violation ratio of 0.3 or higher.

Upon closer examination, the 5-dimensional embedding improves the absolute prediction error over the 2-dimensional embedding by about the same amount across all violation groups. We recall that the median RTTs in paths with low violation ratios are below 300 ms in the King data set. The larger improvement in terms of *relative* errors in the smaller violation group is probably due to the smaller RTTs used as the denominator in computing relative error.

6.3.3 Effect of Dimensionality on RON2 Data Set

In this section, we examine the effect of dimensionality on structural error of PCoord when the underlying distances have varying degrees of triangle inequality. We are interested in understanding how increased dimensionality helps in improving prediction accuracy. In particular, we ask whether increased dimensionality helps improve prediction accuracy when the degree of triangle violation is high? Again, to eliminate the effect of sampling error, each node uses latencies to all other 14 nodes to construct coordinates at each coordinate iteration.

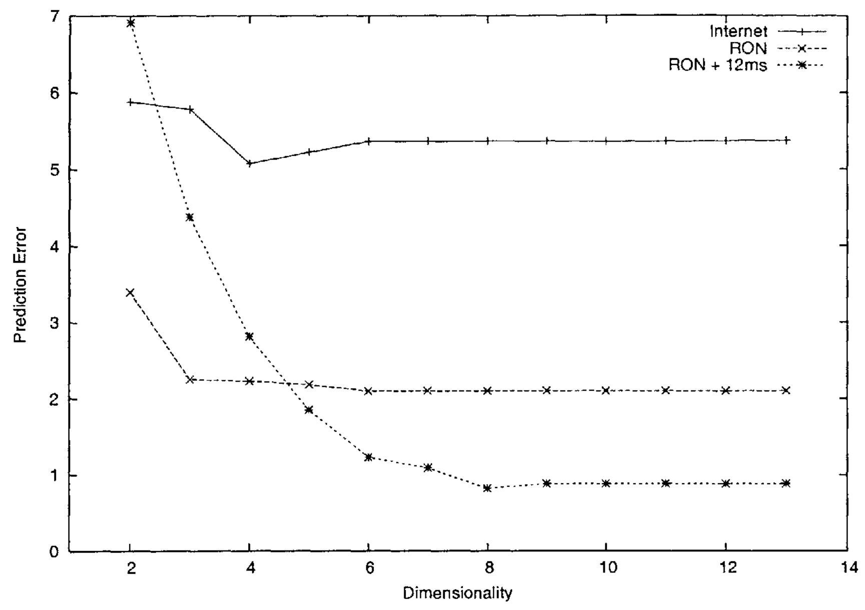
Figure 6-10 shows the PCoord's prediction error as a function of dimensionality of the geometric space under three different RON-based topologies. We note that for topologies with some level of triangle inequality violations, increasing the dimensionality beyond 3 or 4 does not help to reduce the prediction error. Interestingly, we note that the latency-optimized and perfect RON data sets respond to the effect of dimensionality rather differently, even though the only difference between the two is 12 ms for each pair of distance. For the perfect RON data set without any triangle inequality violation, the effect of dimensionality does not saturate until it increases beyond 8 dimensions.

6.4 Conclusions

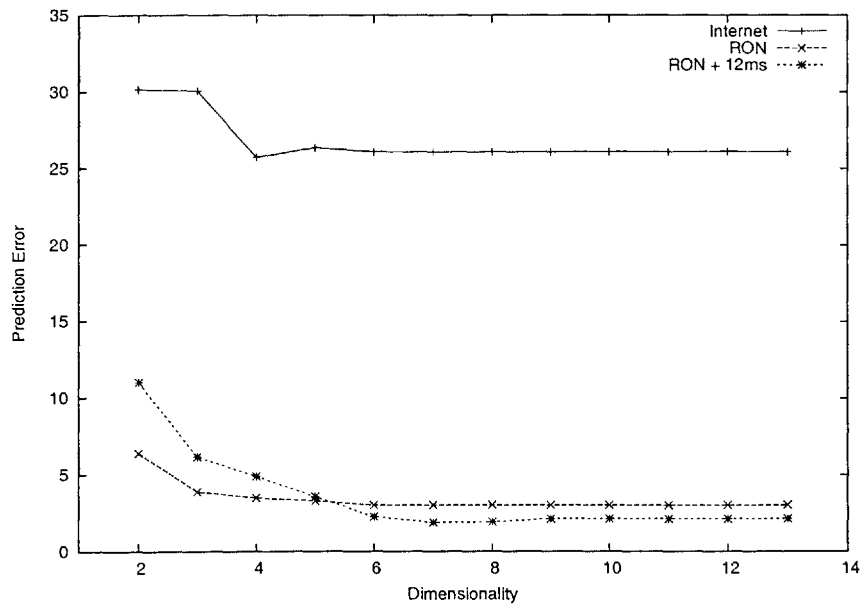
In this chapter, we have examined PCoord's prediction accuracy and error characteristics under several network topologies with varying degrees of triangle inequality violations. We have shown that by reducing the degree of triangle inequality violations in the data set, both under- and over-predictions are greatly attenuated.

Using an artificial grid topology, we have demonstrated that PCoord is able to reconstruct distances from a 2-D Euclidean space. Additionally, we showed that the "resistance" factor in the weighted loss function in the PCoord algorithm is in fact critical for convergence in some cases. One somewhat surprising result is that the number of samples required to reconstruct a 15-node grid in a 2-D plane is about the same as those required for convergence using less "well-behaved" real Internet measurements from network of the same size.

Finally, our results suggest that increasing the dimensionality of the geometric space from two to five does not significantly improve the relative error of paths with high triangle inequality violation ratio in the King data set. However, relative error



(a) Medium Prediction Error



(b) 95th Percentile Prediction Error

Figure 6-10: PCoord prediction error as a function of dimensionality of the geometric space under three different RON-based topologies. $N = 15$, $M = 14$.

distribution for the low violation ratio paths improve substantially with increased dimensionality.

Chapter 7

Conclusions and Future Work

7.1 PCoord Conclusions

In this thesis, we have designed and evaluated a fully-decentralized coordinate system called PCoord. Through extensive simulations using both real network measurements and simulated topologies, we compared the performance of PCoord with another decentralized network coordinate system Vivaldi, and the original GNP scheme using fixed landmarks. Our simulation results indicate that PCoord can achieve competitive prediction accuracy in comparison to the FixedLM scheme without relying on a fixed set of landmark nodes. Our results also suggest that, though PCoord incurs a higher computation overhead in comparison to Vivaldi, it can converge to a lower prediction error using fewer samples of round-trip network times than Vivaldi.

PCoord also fills in one of the missing pieces not addressed in Vivaldi – i.e., how a peer discovers and samples other peers. While Vivaldi's simulations in [10] simply assume that nodes will have access to a list of its nearest peers, we provide an efficient peer discovery mechanism using triangulated distances. We believe PCoord provides a competitive alternative to Vivaldi as a decentralized coordinate system due to the following novel features:

- A weighted loss function to distinguish between nodes with high and low errors and a "resistance" factor in the loss function that helps to stabilize the convergence and avoid oscillation.

- A threshold-based mechanism to dampen the amount a node moves toward new coordinates based on the confidence on current batch of samples.
- A message exchange protocol that enables fast discovery of nearby peers using triangulated distances.

We have examined PCoord's convergence behavior using simulations based on measurements among 1740 Internet hosts in three different scenarios: (1) the simultaneous join scenario with all nodes joining the coordinate system at approximately the same time, (2) the incremental join scenario, in which we evaluate the number of samples required for a newly joined node to converge to a low prediction error when the rest of the system has already converged, and (3) the high churn scenario, in which the system experiences continuous membership changes with dynamic node join and leave.

Our results indicate that, under the simultaneous join scenario, PCoord can converge to a lower median system prediction error with less number of samples than Vivaldi. In particular, after each host updates its coordinates using 100 - 120 samples, PCoord's median system prediction error reduces to the 12 ms range, which is comparable to FixedLM scheme's performance using 10 fixed landmarks. It takes Vivaldi twice as many samples to achieve the same level of accuracy under the simultaneous join scenario.

We have also demonstrated that it takes a small number of samples for a newly joined node in PCoord to converge to a low prediction error when the rest of the system has already converged. In particular, in simulations based on measurements among 1740 nodes, we observe that 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).

Our simulation results suggest that PCoord is robust under high churn when the system experiences continuous membership changes, and can effectively guard against faulty coordinate information due to buggy implementations of the algorithm. We have also examined PCoord's performance under malicious attacks. Our results suggest that the weighted loss function's performance can be sensitive to false information due to malicious nodes. In a real-world deployment, a good engineering choice may be to turn off the weighted loss function if a large number of malicious nodes is expected.

7.2 PALM Conclusions

Another contribution of this thesis is the design and evaluation of a proof-of-concept coordinate system named PALM. Our work (see Appendix A) demonstrates the feasibility of a decentralized approach in building coordinates systems by using distance measurements to *any* subset of hosts. Through simulation-based evaluations, we show that the PALM based approaches have rather different performance characteristics than the fixed landmarks based approach, such as GNP. We believe that many of our findings with respect to the performance characteristics of PALM provide valuable insights for designers of decentralized network coordinate systems or peer-to-peer location systems in general. Some of our findings are:

- Overall, PALM can achieve competitive prediction accuracy in comparison to that of the FixedLM scheme for a significant fraction of the distances. When a random peer sampling strategy is used, PALM achieves comparable prediction accuracy as the FixedLM scheme when a reasonably large number of reference points is used. When the number of reference points used is small, PALM achieves performance comparable to that of the FixedLM scheme by using a sampling strategy that exploits the topological information in the coordinates of existing nodes.

- The PALM based approaches have rather different performance characteristics than the fixed landmarks based approach. The PALM-based approaches tend to over-predict small RTTs. The FixedLM scheme, in contrast, tends to under-predict large RTTs. The performance implication is that PALM coordinates do not provide quite enough resolution needed for nearest peer selection; however, PALM coordinates have comparable performance to FixedLM in clustering peer nodes based on their proximity relationships.

- The performance of the FixedLM approach can be very sensitive to the landmark placement. PALM nodes have the advantage of being able to exploit the coordinates information of the existing nodes and select well-distributed sets of peers as their reference point set. Another important observation is that the performance of PALM seems to be unaffected by the bootstrap nodes locations. Unlike the GNP landmarks whose placement greatly impacts the system performance, bootstrap nodes that are clustered in network do not perform worse than a well-distributed set of bootstrap nodes.

7.3 Future Work

There are several issues that we did not address in this thesis work. Some of these issues include:

- How does PCoord respond to changing network conditions? Although we have demonstrated that PCoord adapts to dynamic node join, we did not explore its performance under dynamic Internet route changes.

- Can PCoord converge with less number of samples? What are the theoretical bounds on the number of samples it takes for PCoord to converge? In this thesis, we demonstrated that PCoord can converge to low prediction error by sampling a mixture of near and far peers. One interesting question is whether there exists a peer sampling strategy that can significantly decrease the number of samples it takes for PCoord to converge to a low prediction error.

- Are there better geometric models for Internet distances? In this thesis, we have chosen the Euclidean space to model Internet distances. One future direction is to explore alternative models to better capture Internet distances.

Finally, as part of our future work, we would like to explore whether it is possible to extend the PCoord framework to model "distance" measurements other than Internet latencies. For example, the distance measurement can be a metric that measures the similarity between the content of web documents. One interesting direction is to extend the PCoord framework to implement a decentralized semantic overlay to support distributed searching of data objects.

Appendix A

PALM

The landmark-based architecture has been commonly adopted in the networking community as a mechanism to measure and characterize a host's location on the Internet [32, 35, 41, 42, 37, 15]. In most existing landmark based approaches, end hosts use the distance measurements to a common, fixed set of hosts to derive location estimations on the Internet. The Global Network Positioning (GNP) system [32], for example, uses a host's distance measurements to a fixed set of infrastructure nodes to compute absolute coordinates to characterize the host's location on the Internet.

However, using a fixed set of landmarks presents a potential performance bottleneck. More importantly, the accuracy of the fixed landmark schemes often depends highly on the strategic placement of the landmarks. Although the developers of GNP reported good prediction accuracy with a careful selection of landmarks when hosts are globally distributed, in practice, it will be difficult to pre-determine the strategic placement of landmarks without some prior knowledge of the topological distribution of the participating hosts.

In this study, we investigate the performance of a coordinate-based scheme, PALM, which uses peers as landmarks. We extend the absolute coordinates framework from GNP and apply it in a decentralized, peer-to-peer environment. More specifically, instead of using a fixed set of nodes as landmarks, any peer node which has already derived its coordinates can be selected by another peer node to function as a landmark. We call such a peer-to-peer based approach in topology discovery PALM (Peers as

Landmarks).

The focus of this study is to evaluate the performance characteristics of such a decentralized coordinates-based approach under several factors, including the dimensionality of the coordinate space, peer distance distribution, and the number of peer-to-peer distance measurements used. We evaluate two PALM-based schemes: RandPalm and Island. In RandPalm, a peer node randomly selects from existing peer nodes as its landmarks. In Island (Intelligent Selection of Landmarks), each peer node selects its landmarks by exploiting the topological information derived based on existing peer nodes' coordinates values.

Through simulations using both real network measurements and simulated topologies, we compare the performance of RandPalm and Island with the original GNP scheme (referred to as the FixedLM scheme from now on). The rest of this Appendix is organized as follows. We first briefly describe the FixedLM and the PALM approach. We then evaluate the PALM approach through simulations using both real network measurements and simulated topologies. We compare the performance of RandPalm and Island with that of the FixedLM scheme in terms of errors in network distance prediction and their effectiveness in selecting nearest peer nodes.

A.1 The PALM Approach

Before we describe the PALM approach, we first briefly introduce the GNP[32] framework as background information.

A.1.1 GNP

In GNP, the Internet is modeled as a D-dimensional geometric space. End hosts maintain absolute coordinates in this geometric space to characterize their locations on the Internet. Network distances are predicted by evaluating a distance function over hosts' coordinates. A small distributed set of hosts known as landmarks provide a set of reference coordinates. Hosts measure their latencies to a fixed set of landmark nodes in order to compute their coordinates. While the absolute coordinates provide a scalable mechanism to exchange location information in a peer-to-peer environment, the GNP scheme presented so far used distance measurements to a fixed set of landmarks to build the geometric model.

A.1.2 PALM

In PALM, there is no specially designated landmark nodes; any peer node can potentially be selected as a landmark by another node. As part of the bootstrap process, we assume that an arbitrary set of initial peer nodes function as bootstrap landmarks to provide reference coordinates to other nodes. The PALM bootstrap nodes compute their coordinates the same way that the GNP landmarks compute their coordinates. The main difference between PALM and GNP lies in how regular hosts compute their coordinates. In the rest of this section, we first define notations. We then present the PALM scheme in two parts: bootstrap operation and regular host operation.

Let:

B = Set of bootstrap nodes

M = Number of reference points for each coordinate update

Y = Set of peers selected as reference points

c_i = Coordinates of host i

d_{ij} = Measured RTT between hosts i and j

Bootstrap Operation

The bootstrap nodes measure the inter-node round-trip ping times to produce a $|B|$x$|B|$ distance matrix, where $|B|$ is the number of bootstrap nodes. A set of coordinates are computed for the $|B|$ bootstrap nodes to minimize the overall error between the measured distances and the computed distances as follows. Let c_i and c_j be the coordinates of node i and j respectively, where $i, j \in B$. The coordinates of the bootstrap nodes can be computed by minimizing the following loss function:

$$\sum_{i,j \in B} f(d_{ij}, \|c_i - c_j\|)$$

where $f(\bullet)$ denotes an error measurement function, and $\|\|$ denotes the norm. The GNP work used several error functions, including the squared-error function:

$$f(d_{ij}, \|c_i - c_j\|) = (d_{ij} - \|c_i - c_j\|)^2$$

Another error function used in [32] is the relative squared-error:

$$f(d_{ij}, \|c_i - c_j\|) = (\frac{d_{ij} - \|c_i - c_j\|}{d_{ij}})^2$$

In comparison to the squared loss function, the *relative* squared error function puts more "weight" on shorter RTTs. In this study, we use the relative squared error function when evaluating PALM.

The global minimization problem can be approximately solved using many generic multi-dimensional minimization algorithms, such as Simplex Downhill method, which we use in this thesis.

Once the bootstrap nodes have been mapped, their coordinates along with the description of the geometric space and possibly the distance function used can be made available for other peer nodes to compute their own coordinates. A peer node is said to have been **mapped** once it has computed its absolute coordinates.

Regular Host Operation

In PALM, any peer node which has already derived its absolute coordinates can be selected by another peer node to serve as one of its landmarks. In order for a host i to compute its coordinates, it selects any M existing mapped peer nodes to function as its landmarks ($D + 1 <= M <= |B|$). Let Y be the set of peers selected by host i. Using the coordinates of those M peer nodes and the distance between i and each of the M selected peer nodes, host i can compute its coordinates c_i to minimize the overall error between the measured and the computed distances to the selected M

peers. More specifically, host i computes its coordinates by finding c_i that minimizes the following loss function:

$$\sum_{j \in Y} f(d_{ij}, \|c_i - c_j\|)$$

Again, we use the relative error function described above as our error measurement function.

We previously proposed RandPalm and Island in [24], independent of PIC [8]. Although one of the PIC strategies (namely, the random strategy) has the same basic algorithm as RandPalm, PIC [8] did not explore the behavior of the random strategy in depth. Further, it did not address the issue of selecting well-distributed peers as reference points. In this work, we examine the performance of RandPalm as a function of the number of reference points used, and explore the effect of bootstrap nodes placement.

A.2 Comparing RandPalm with the Fixed Landmark Scheme

We evaluate the PALM approach through simulations using both real network measurements and simulated topologies. We compare the performance of PALM with the FixedLM scheme in terms of errors in network distance prediction and their effectiveness in selecting nearest peer nodes.

A.2.1 Performance Metrics

Pairwise Distance Prediction

As in GNP [32], we use the absolute relative error (RE) as our performance metric. For each pair of nodes, their absolute relative error is defined as $\frac{|E-A|}{min(E,A)}$, where E is the predicted Euclidean distance, and A is the actual measured RTT (round trip time) between the two nodes. The directional relative error (DRE) is $\frac{E-A}{min(E,A)}$.

Additionally, we define directional prediction error (DPE) as $E - A$, and prediction error (PE) as $|E - A|$.

Nearest Neighbor Selection

We use the stretch metric to evaluate the the effectiveness of the proposed schemes in selecting nearest neighbor. Let $RTT(i, j)$ denote the mesaured round-trip latency between node i and j. Given a Euclidean mapping of a set of peer nodes, the stretch of a node i is computed as $\frac{RTT(i,p)}{RTT(i,q)}$, where p is the node with minimal Euclidean distance to i, and q is the closest neighbor to i according to the actual latency measurement.

A.2.2 Network Topologies

We evaluate our scheme using both real network measurements and simulated topologies:

- The Active Measurement Project (AMP) at the National Laboratory for Applied Network Research (NLANR) collects network measurements between over 100 active monitors distributed over the Internet [13]. We use the RTT measurements between 110 of such monitors on July 16, 2002 for our experiments. The RTTs are the round trip ping time between each pair of hosts measured at a frequency of once every minute over a 24 hour period (i.e., total of 1440 round trip times reported between each pair of hosts).

- The GT-ITM Internet Topology Generator is used to generate transit stub topologies of a 10,000 node network. We then randomly select 3492 out of the 10,000 nodes as peer nodes of our test overlay network. The latency between two nodes is defined as the sum of the links on the shortest path between the two nodes using the weighted graph generated by ITM.

The GNP paper evaluated their scheme using distance measured between 19 landmark nodes and 869 hosts. However, since no inter-hosts distances between the 869 hosts are available, we used other network measurements and simulated topologies to test our approach.

For the RandPalm and Island schemes, ten experiments with different selection of bootstrap nodes are performed for each topology. For each experiment, the same set of hosts that serve as bootstrap nodes in the RandPalm and Island are also used as the fixed landmarks in the FixedLM scheme. Unless otherwise noted, the FixedLM landmark nodes and the PALM bootstrap nodes are randomly selected hosts from the peer population. In a later section, we examine the effect of biased selection of landmarks on PALM's performance. The default dimension of the geometric space, D, used is five, unless otherwise noted.

A.2.3 Effects of Number of Landmarks

In this section, we compare the distance prediction performance of the FixedLM scheme with that of the RandPalm scheme when different number of landmarks are used.

In Figures A-1 and A-2, we compare the cumulative distribution of the absolute relative error of FixedLM scheme vs. the RandPalm scheme when different numbers of landmarks are used. Figure A-1 shows the results from the AMP measurements for 6, 10 and 15 landmarks. Figure A-2 compares the relative error distribition of the two schemes using the GT-ITM topology when 10, 20 and 30 landmarks are used respectively. In both schemes, the performance improves as the number of landmarks increases.

We note that the performance of the two schemes are very similar. In both schemes, the performance monotonically improves as the number of landmarks increases. The performance of the 20 landmarks case is much better than that of the 10 landmarks under both schemes. Further, the gap between the distributed landmark selection scheme and the fixed landmark selection scheme is even smaller when the number of landmarks is increased to 20.

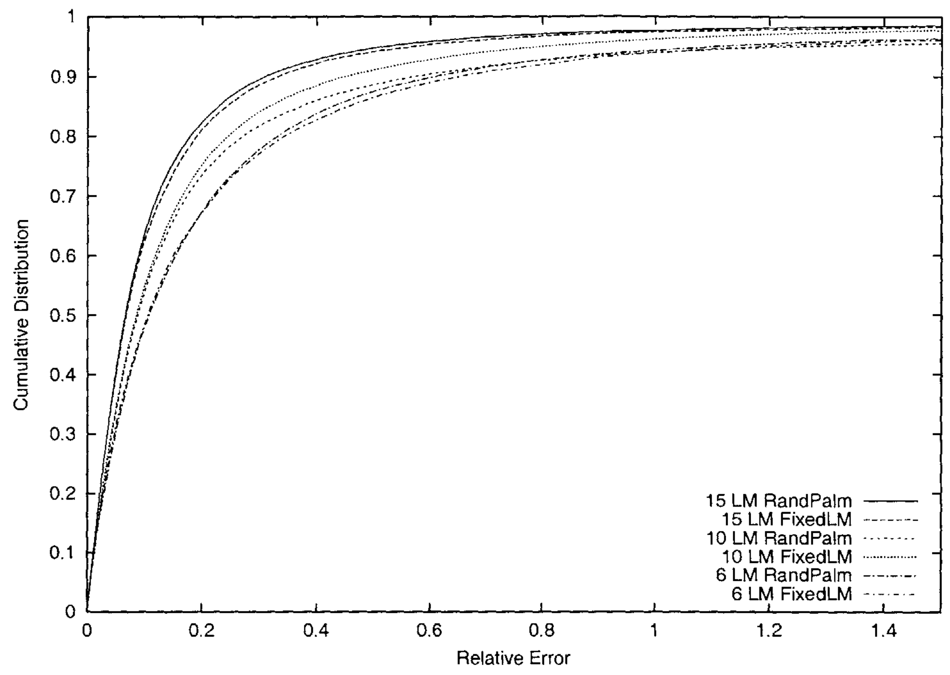


Figure A-1: AMP results. Cumulative distribution of relative error, FixedLM vs. RandPalm. $N = 110$, 5-Dimensions.

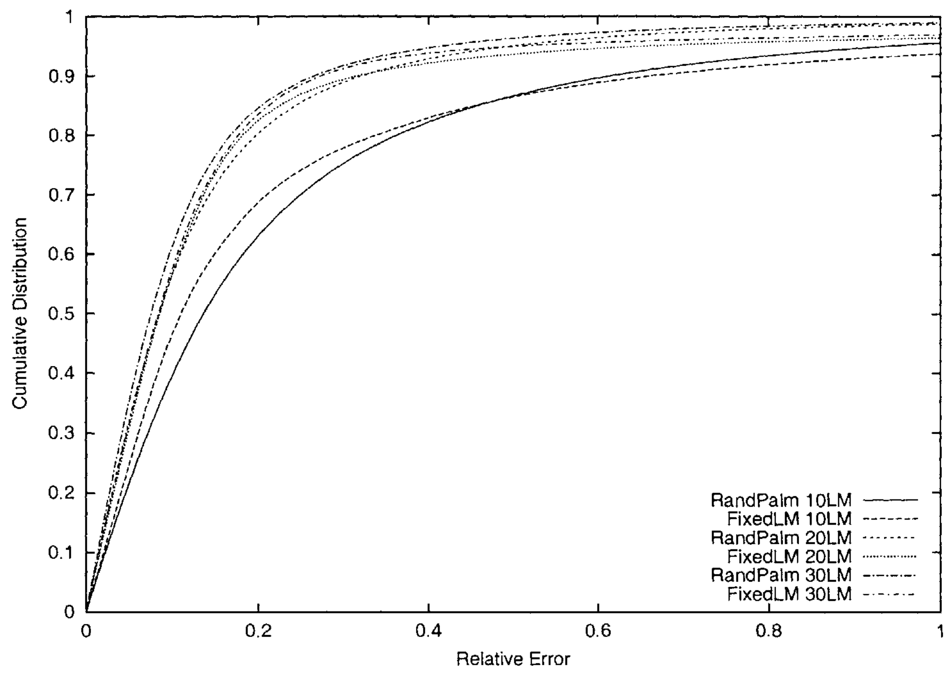


Figure A-2: GT-ITM results. Cumulative distribution of relative error, FixedLM vs. RandPalm. $N = 3492$, 5-Dimensions.

A.2.4 Comparison of FixedLM and RandPalm with Different Number of Landmarks

To better understand the performance characteristics of the RandPalm vs. the FixedLM scheme, we plot the summary statistics that describe the distance prediction error of both schemes as a function of the number of landmarks used. Figures A-3 and A-4 plot the median, 5th, 25th, 75th, and 95th percentile relative error (RE) and directional relative error (DRE) respectively of both schemes as a function of the number of landmarks.

We note that a zero value in RE and DRE indicates a perfect prediction in the network distance. RE expresses the prediction error as an absolute value, and therefore is always positive. A positive DRE value indicates an over prediction in network distance, while a negative DRE value indicates an underestimation of actual network distance.

We note that RandPalm performs worse than FixedLM when the number of landmarks is low. In particular, when six and ten landmarks are used, RandPalm has a tendency to over predict network distances between hosts, as can be observed from the large positive 95th percentile DRE value in Figure A-3. The FixedLM scheme, on the other hand, has a tendency to under-predict inter-hosts distances when the number of landmarks is low. This can be observed from the large negative 5th percentile DRE values in Figure A-4.

We note that for both schemes, the RE and DRE values improve monotonically with increasing the number of landmarks. For RandPalm, the performance improvement is especially significant when the number of landmarks is increased from 6 to 15. The performance of both schemes tends to flatten beyond 25 landmarks.

An important observation is that the performance of RandPalm eventually catches up to that of the FixedLM scheme when increasingly large numbers of landmarks are used. We also observe that the 5th percentile DRE value of the FixedLM scheme is consistently lower than that of the RandPalm scheme across all landmark values, indicating a large under-prediction problem in the FixedLM scheme. This is consistent

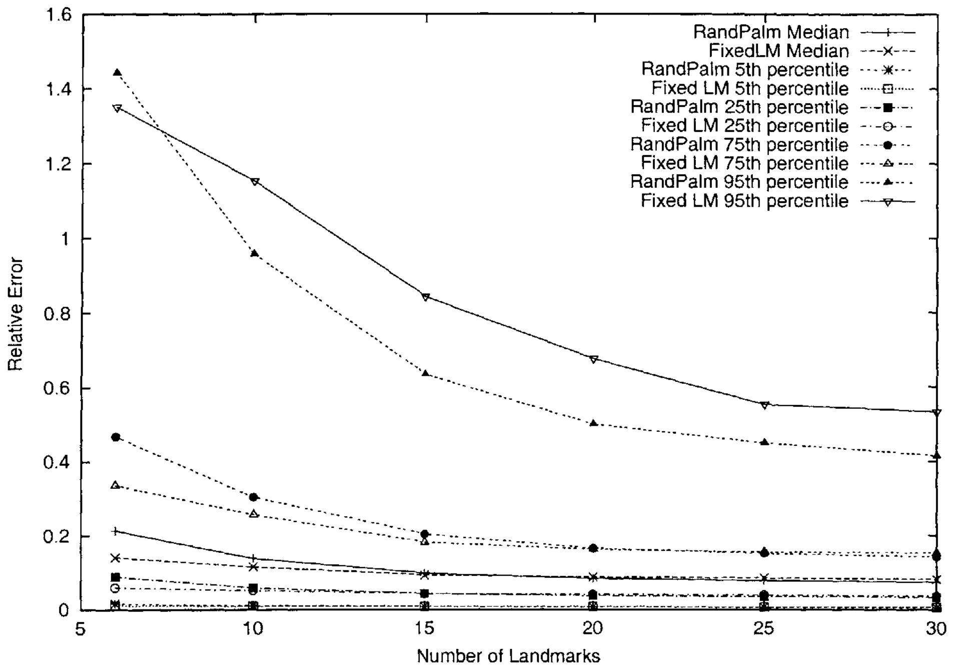


Figure A-3: Relative Error. Comparing FixedLM and RandPalm schemes with summary statistics of relative error: GT-ITM,$N = 3492$. Dimensionality is 5. Number of landmarks: 6, 10, 15, 20, 25, 30.

with the original GNP results in [32], which reported a large under-prediction error using their data set when predicting large RTT measurements.

A.2.5 Comparison of FixedLM and RandPalm by RTT Groups

From the previous section, we observe that the FixedLM scheme tends to under-predict while the RandPalm scheme tends to over-predict. To understand the sources of these under- and over-predictions, we further investigate the performance properties of both schemes by classifying the evaluated paths into groups of 50 ms each.

Figure A-5 shows the RTT group size distribution of our GT-ITM topology. We show the summary statistics of the directional prediction error, defined as (predicted RTT - actual RTT), for each RTT group. Figures A-6 and A-7 show the median, mean, 5th, 25th, 75th and 95th percentile prediction error of each RTT group using FixedLM and RandPalm respectively. Ten landmarks are used for both figures. Figures A-8 and A-9 show the same statistics when 20 landmarks are used.

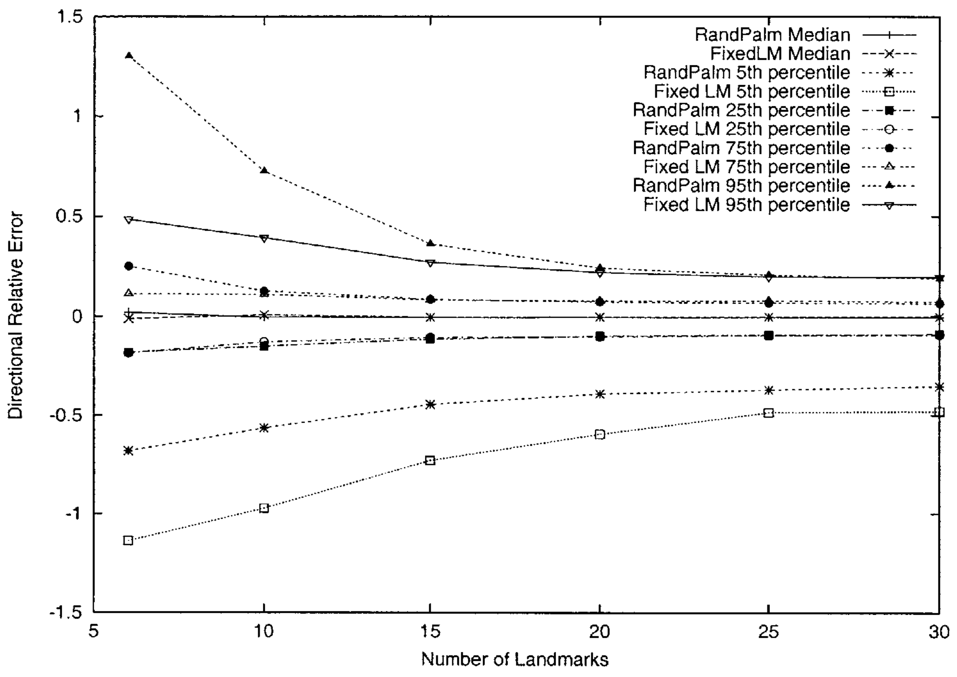


Figure A-4: Directional Relative Error. Comparing FixedLM and RandPalm schemes with summary statistics of directional relative error: GT-ITM, $N = 3492$. Dimensionality is 5. Number of landmarks: 6, 10, 15, 20, 25, 30.

For the ten landmark case, Figure A-6 shows that the FixedLM scheme is very accurate in predicting distances less than 50 ms, but tends to under-predict distances that are beyond 250ms.

Figure A-7 shows that the RandPalm scheme has the most trouble in predicting short distances when ten landmarks are used. The 95th and 75th percentile prediction errors are as high as 694 and 385 ms respectively, showing a gross over-estimation of distances less than 50 ms. The RandPalm scheme also tends to under-estimate distances over 700 ms, although the extent of the under-estimation is not nearly as severe as the over-estimation for the 50 ms group case.

Increasing the number of landmarks to 20 helps both schemes in narrowing down the extent of their prediction errors across all RTT groups. We note that increasing the number of landmarks from 10 (see Figure A-7) to 20 (see Figure A-9) significantly reduces the extent of over-prediction in the short RTT groups for RandPalm: there is a 40-50% reduction in the 95th and 75th percentile prediction errors in the 50 ms RTT group. Though the RandPalm prediction errors in the 20 landmark scenario are

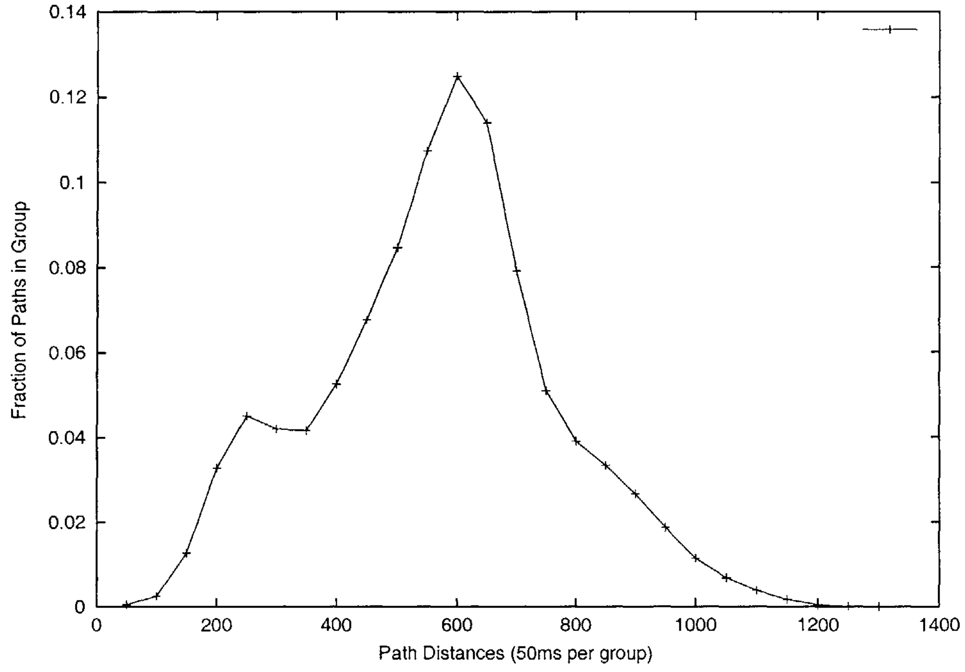


Figure A-5: Bin size distribution by RTT groups: GT-ITM, $N = 3492$

distributed fairly evenly between over- and under-predictions, the mis-predictions in the 50 ms RTT group are still dominated by over-predictions.

A.2.6 Dimensionality

In this section, we examine the effect of dimensionality on the prediction accuracy of the RandPalm scheme. Figures A-10 and A-11 plot the relative error distribution for the FixedLM and RandPalm schemes respectively using varying number of dimensionalities. The number of landmarks is fixed at ten in both schemes. Due to space constraints, we only show the AMP results. The GT-ITM results are qualitatively similar.

From both figures, significant performance improvement can be observed as the dimensionality increase from one to five in both schemes. The incremental performance improvement beyond five dimension, however, is very small.

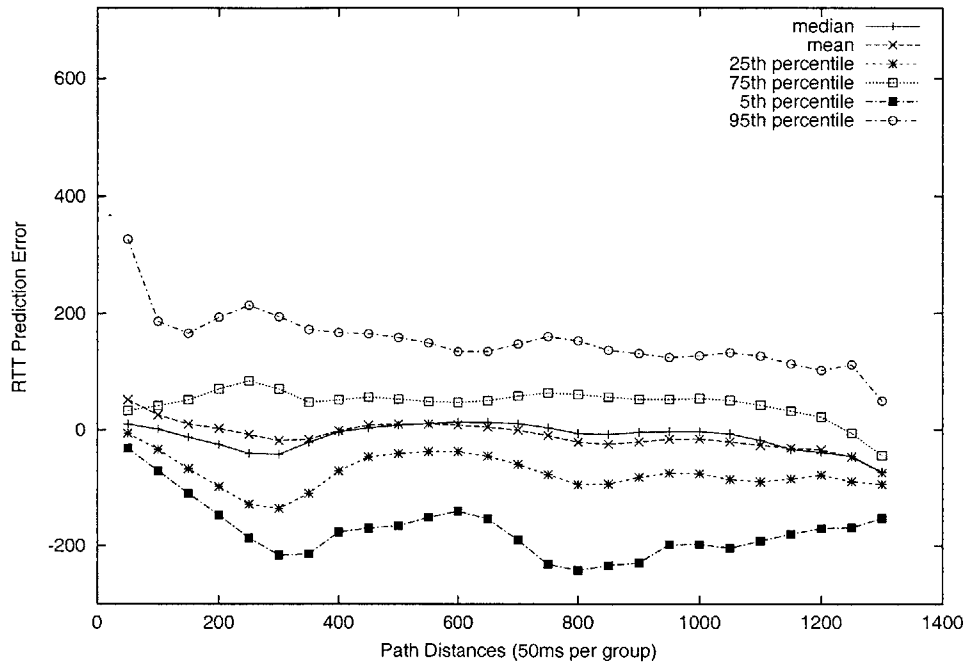


Figure A-6: FixedLM performance by RTT groups using 10 landmarks. GT-ITM, $N = 3492$, 5 Dimensions.

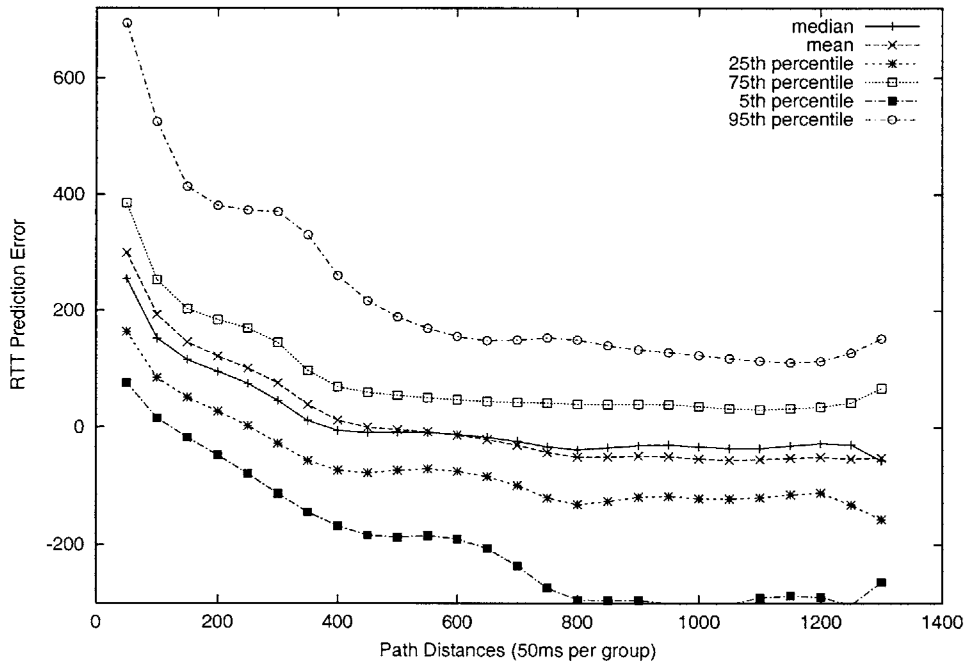


Figure A-7: RandPalm performance by RTT groups using 10 landmarks. GT-ITM, $N = 3492$, 5 Dimensions.

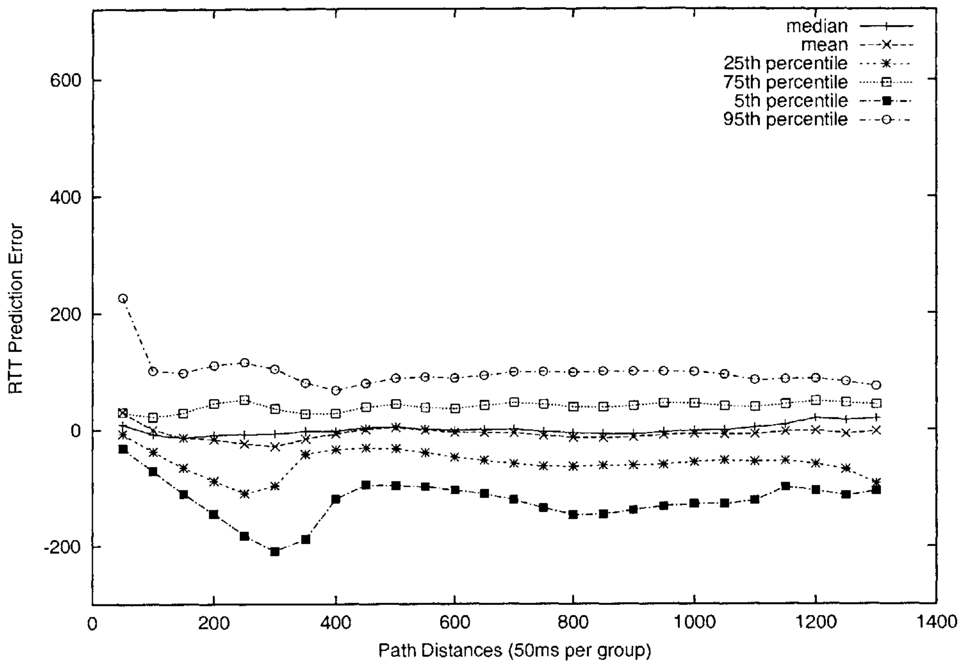


Figure A-8: FixedLM performance by RTT groups using 20 landmarks. GT-ITM, $N = 3492$, 5 Dimensions.

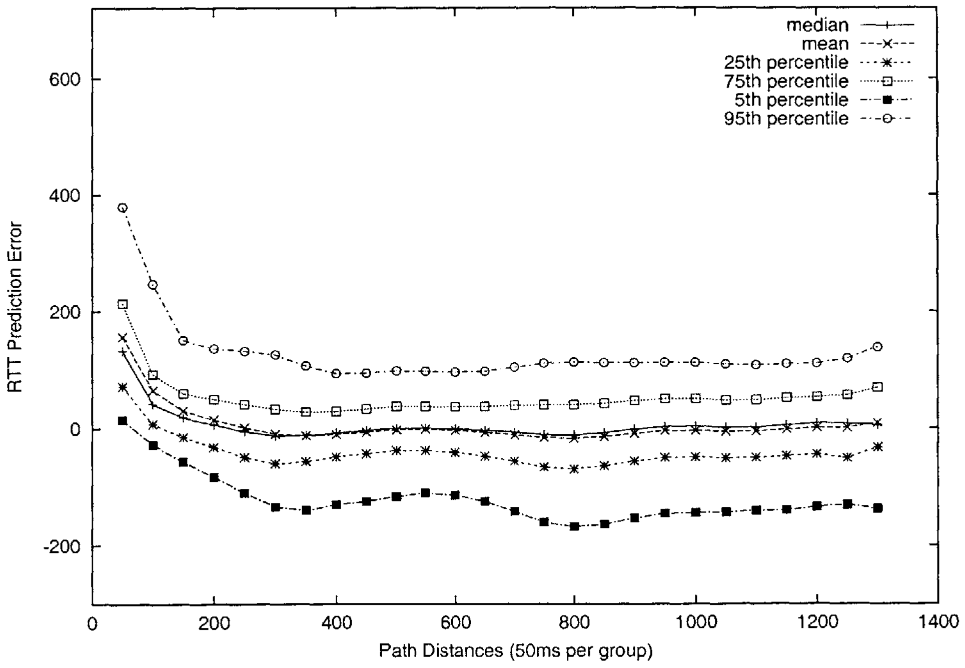


Figure A-9: RandPalm performance by RTT groups using 20 landmarks. GTITM, $N = 3492$, 5 Dimensions.

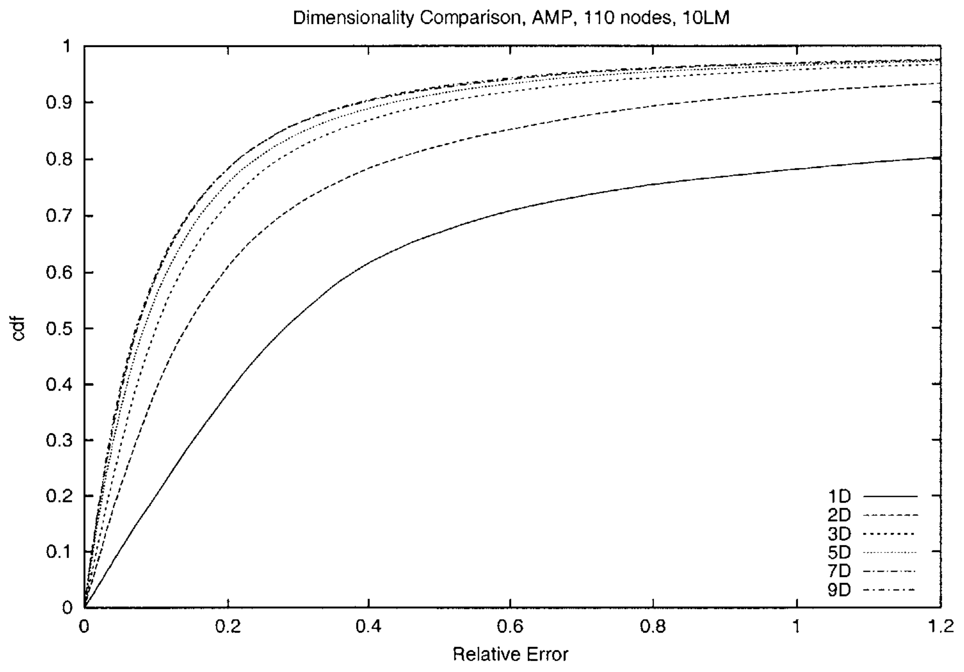


Figure A-10: FixedLM. Effect of dimensionality on performance. AMP, $N = 110$, 10 Landmarks.

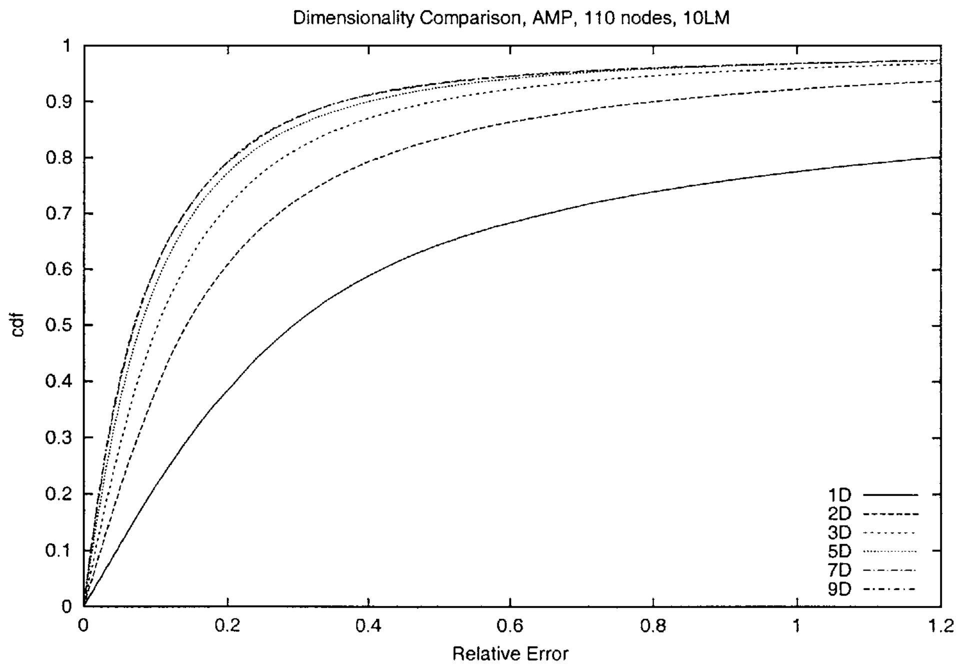


Figure A-11: RandPalm. Effect of dimensionality on performance. AMP, $N = 110$, 10 Landmarks.

A.3 Robustness in Landmark Placement

The results we have presented so far randomly select from a global pool of peer nodes to function as landmarks. In the FixedLM scheme, this randomly selected set of peer nodes are used by all other peer nodes to construct their solution coordinates. In the RandPalm scheme, this randomly selected set of peer nodes function as the bootstrap nodes that provide a set of reference coordinates to other peer nodes.

In this section, we compare the performance of RandPalm with the FixedLM scheme when the landmark placement is not well distributed. We generate ten different sets of badly placed landmarks, which tend to be clustered in network topology, and compare the performance of FixedLM and RandPalm. Figure A-12 and A-13 plot the cumulative relative error distribution with poorly placed landmarks in AMP and GT-ITM respectively.

We use the following procedure to generate ten different sets of badly placed bootstrap nodes, which tend to be clustered in network topology. First, a hierarchical clustering algorithm is used to cluster peer nodes into G clusters based on their actual RTT measurements. Let G' be the number of clusters with no less than $|B|$ nodes in them. We then randomly pick a cluster from the G' clusters. Finally, randomly pick $|B|$ nodes from the above cluster. Ten different sets of clustered landmark selections are generated for each topology, and the cumulative results are presented here.

Figures A-14 and A-15 show the summary statistics of the FixedLM scheme when a clustered landmark set is used. Comparing the summary statistics in Figure A-6 using randomly selected landmarks, we note that the FixedLM scheme has the tendency to grossly underestimate RTT groups larger than 50ms when clustered landmarks are used. A sharp dip of the 5th percentile DRE value around the 200 ms RTT group in Figure A-14 is caused by under-predicting some 200 ms paths by almost 100%. This causes the DRE value to dip dramatically around the 200ms RTT group, because the DRE metric divides the prediction error by the minimum of the measured and the computed RTT values. Similar levels of mis-predictions at higher RTT groups do not have as low of a DRE value, because of the larger denominators.

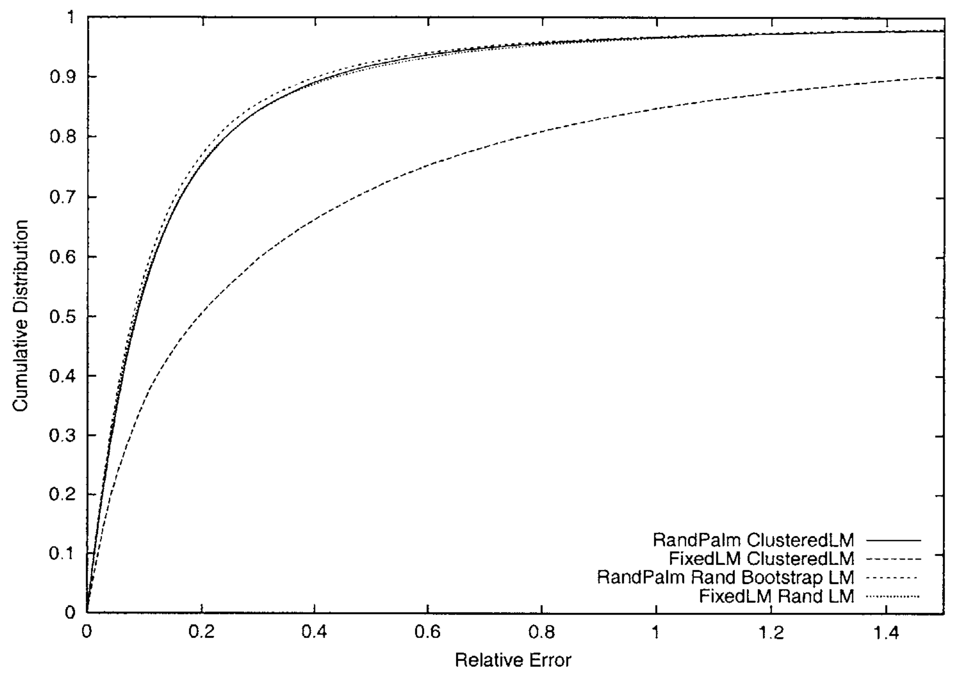


Figure A-12: Effect of clustered landmark placement on relative error of distance prediction. AMP, $N = 110$, 10 Landmarks, $G = 36$.

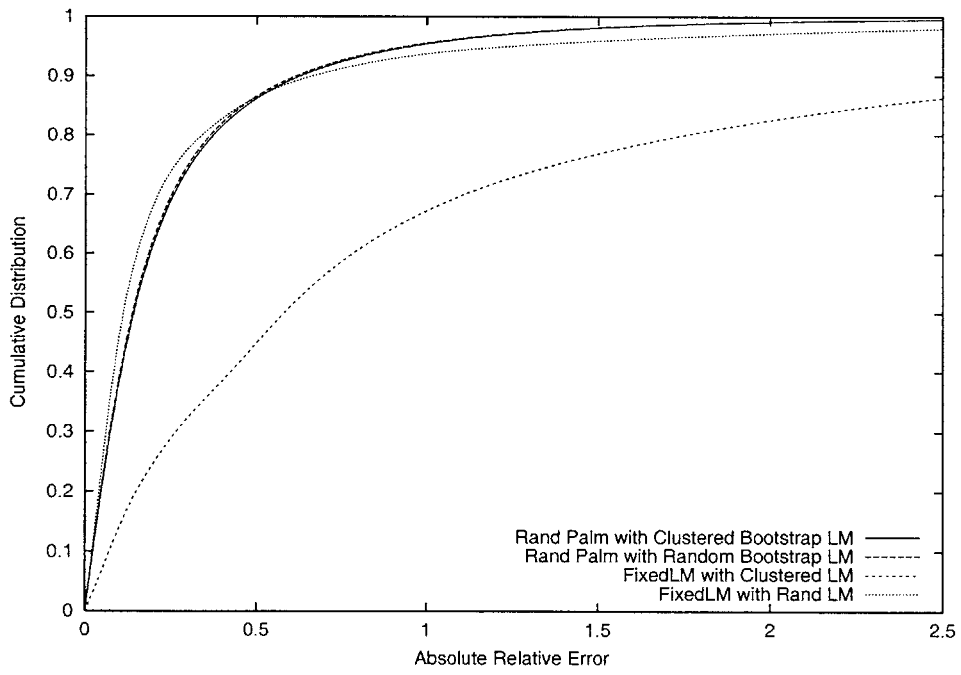


Figure A-13: Effect of clustered landmark placement. GT-ITM, $N = 3492$, 10 Landmarks. $G = 30$.

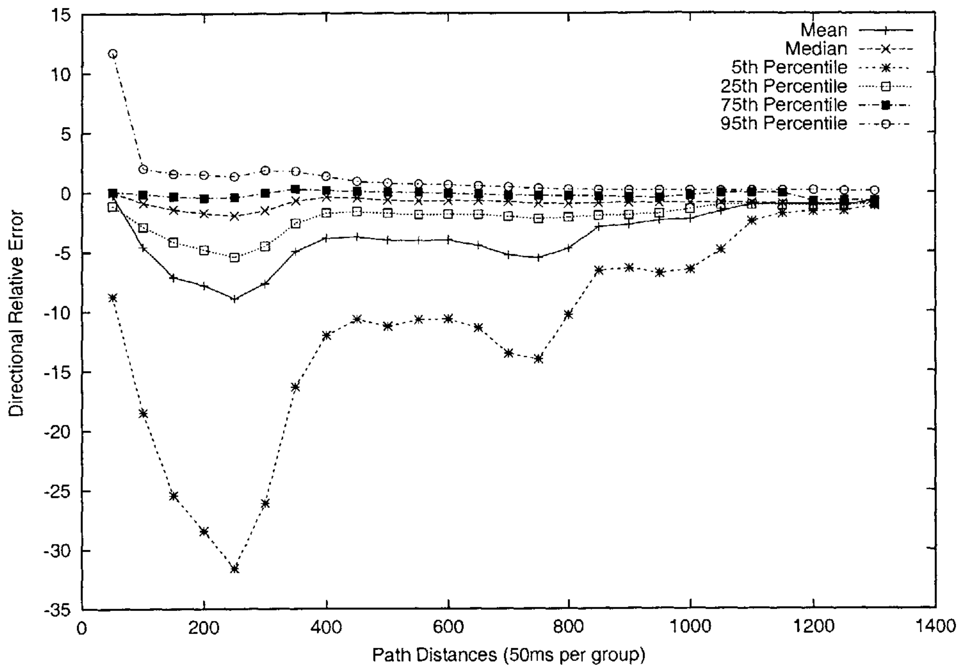


Figure A-14: Summary statistics of directional relative error for the FixedLM scheme under clustered landmark placement. $N = 3492$, 10 Landmarks.

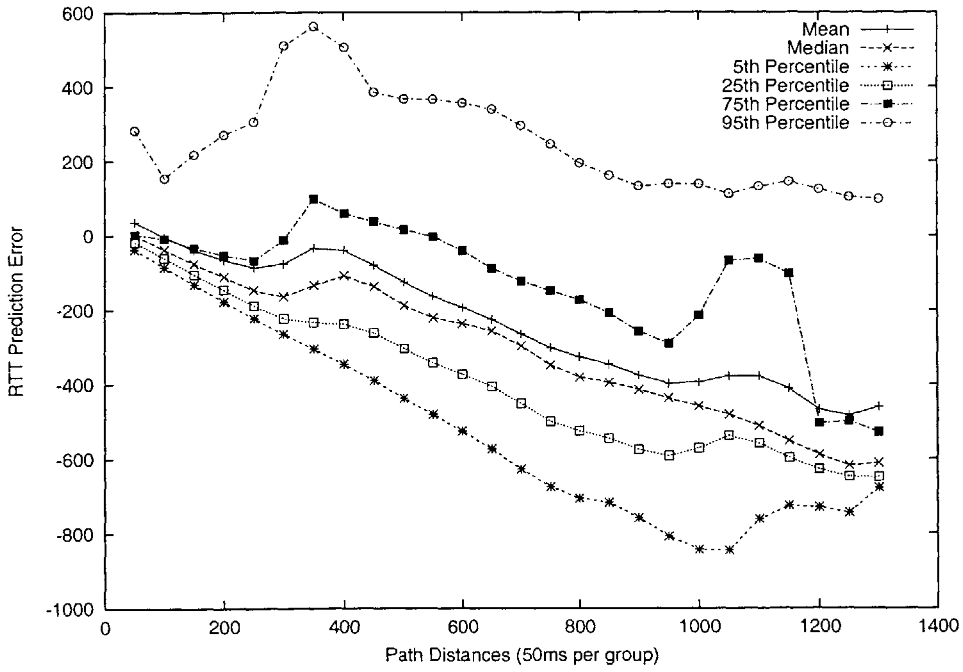


Figure A-15: Summary statistics of RTT prediction error for the FixedLM scheme under clustered landmark placement. $N = 3492$, 10 Landmarks.

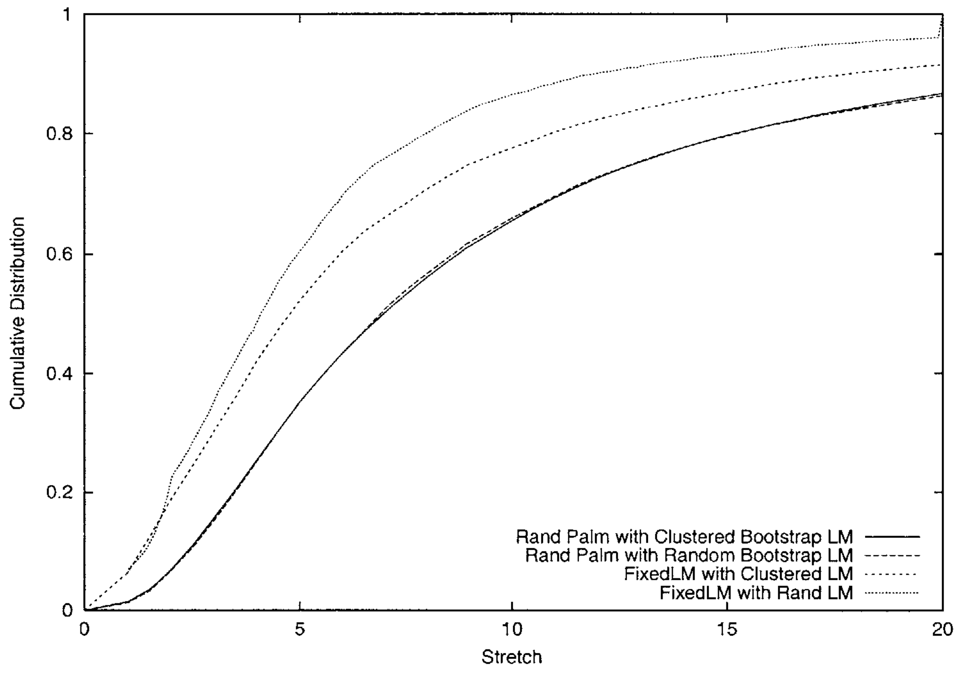


Figure A-16: Effect of bad landmark placement on nearest neighbor selection. GT-ITM, $N = 3492$, 10 Landmarks.

Figure A-16 show the effect of clustered landmark placement on nearest neighbor selection performance for both FixedLM and RandPalm. Although the overall prediction accuracy of the FixedLM scheme suffers when landmarks are poorly placed, we note interestingly that its nearest neighbor selection performance does not seem to be affected as much. Even with bad landmark placement, the FixedLM scheme still outperforms the RandPalm scheme in nearest neighbor prediction.

A.4 Intelligent Landmark Selection Using PALM Maps

In the previous section, we presented some interesting performance properties of RandPalm. As the number of landmarks increases, the overall distance prediction performance of RandPalm converges to that of the FixedLM case. However, unlike the FixedLM scheme, it is very robust against suboptimal bootstrap nodes placement.

In this section, we describe an approach called Island (Intelligent Selection of

Landmarks using PALM Maps) to improve on the performance of the RandPalm scheme. Our goal is to achieve network distance prediction accuracy of the FixedLM scheme with fewer landmark nodes while preserving the robustness of the RandPalm scheme. The idea of Island is to have each peer node intelligently select its landmarks by exploiting the topological information contained in the PALM map. The PALM map contains the IP addresses of existing peer nodes, and their coordinates values in the geometric space. We assume that each existing peer node in the system has access to a copy of the current PALM map. Note that Island does not require each peer node to have a global PALM map that contains all of the peer nodes in the system. A partial PALM map is sufficient, provided that it contains a reasonably well-represented set of peer nodes in terms of network topology. The dissemination of the PALM map is beyond the scope of this work, and will be left as future work.

In Island, each peer node uses the following heuristic to select landmarks.

- Upon joining, a peer node i contacts any existing peer node j in the system to obtain a copy of the existing PALM map. The map contains the IP addresses of existing peer nodes known to node j, and their coordinate values.

- The existing peer nodes are classified into clusters based on their coordinates in the geometric space. The results presented in this section use the Euclidean distance between nodes' position in the geometric space to cluster the existing peer nodes. We will experiment with other distance functions in future work.

- Node i then randomly picks M clusters from the clusters formed above, and then randomly picks a node in each cluster as its landmarks. By picking each landmark node from a different cluster, we attempt to achieve a well-dispersed landmark set, and avoid the degenerate case where all landmarks are from the same network region.

The clustering can be done offline by existing peer nodes in the system, so that a newly joined peer node can quickly select a set of landmark nodes based on the clustered PALM map.

We have examined the performance of the Island scheme with simulation using both the GT-ITM and AMP topology. We present only the GT-ITM results here, since the AMP results are similar. We compare the performance of Island, RandPalm and the FixedLM schemes under the following scenarios.

- Random bootstrap landmarks. The $|B|$ bootstrap nodes are randomly selected from the N nodes.
- Clustered bootstrap landmarks. The bootstrap landmarks are all from the same cluster.
- Dispersed bootstrap landmarks. The boostrap landmarks are from different clusters. The performance of this scenario is not shown as it does not differ significantly from the random bootstrap landmarks case for all three schemes.

For each scenario, $|B|$ nodes are selected to function as the bootstrap landmarks. In the FixedLM case, these $|B|$ nodes are the landmark nodes that are used by all peer nodes to generate their coordinates. In the RandPalm and Island case, these $|B|$ nodes function as the bootstrap nodes. The difference between the RandPalm scheme and Island is that, in RandPalm, peer nodes randomly select M nodes from the PALM map; whereas Island selects the M nodes by exploiting the cluster information in the PALM map. For simplicity, the simulations in this section set M equal to $|B|$.

Figure A-17 compares Island with RandPalm and FixedLM schemes using the GT-ITM topology. The performance of Island is better than the RandPalm and FixedLM schemes when ten landmarks are used by all schemes. Further, we note that the performance of Island using 10 landmarks is comparable to the performance of the FixedLM scheme when 15 landmarks are used. Finally, when the bootstrap landmarks are clustered (Figure A-19) both Island and RandPalm greatly outperforms the FixedLM scheme.

Figure A-18 shows the summary statistics of the Island scheme under random bootstrap node placement. Note that the performance of the Island scheme is much better than the RandPalm summary statistics presented in Figure A-7.

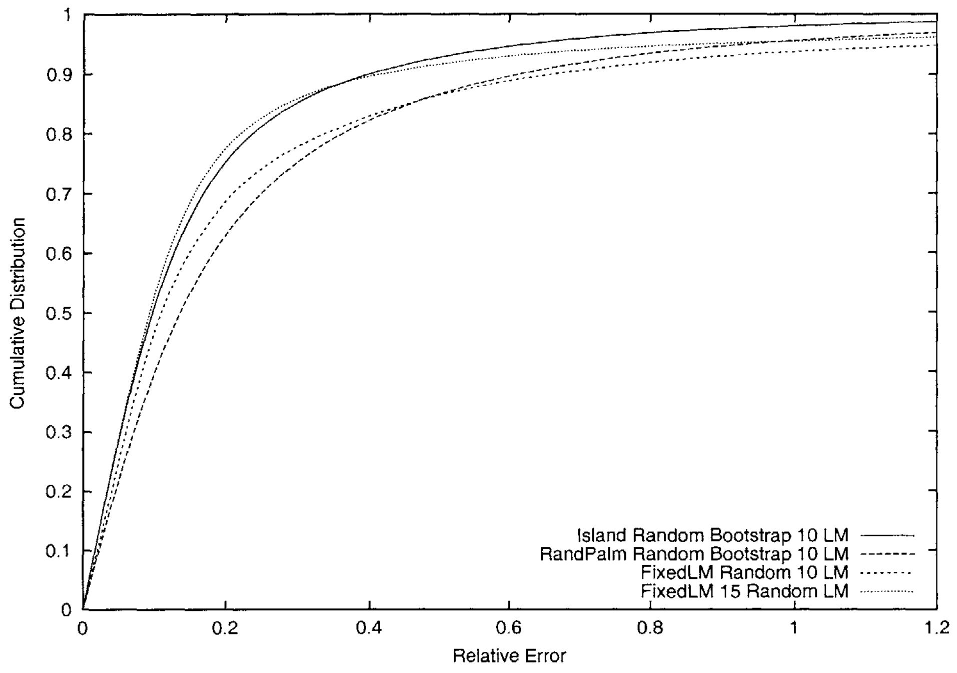


Figure A-17: Island relative error distribution using randomly selected bootstrap landmarks. GT-ITM, $N = 3492$, 10 landmarks.

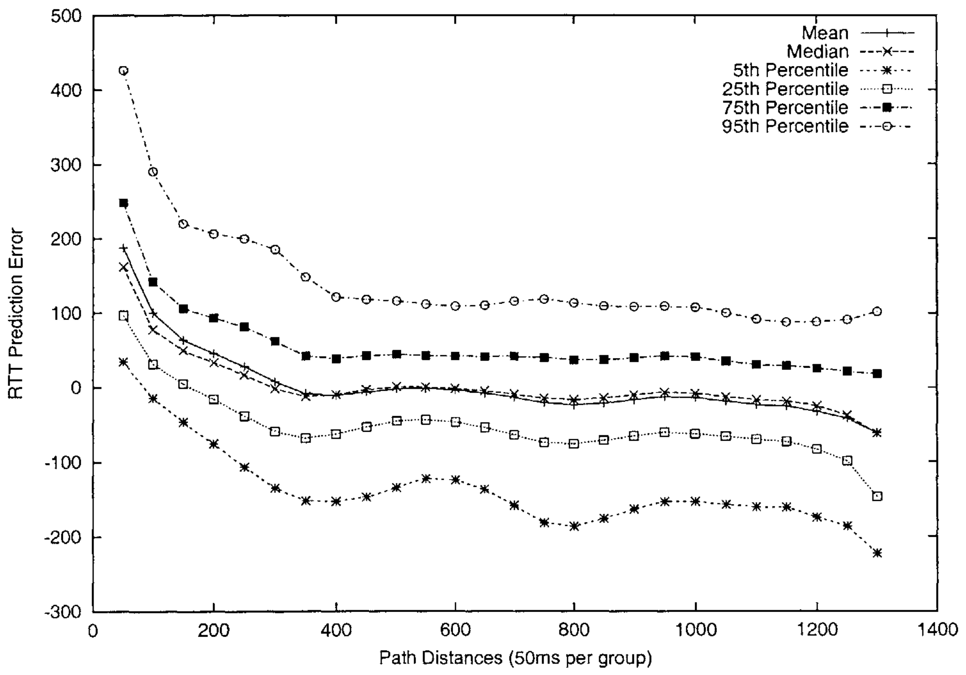


Figure A-18: Summary statistics of RTT prediction error for the Island scheme under random bootstrap landmark placement. GT-ITM, $N = 3492$, 10 Landmarks.

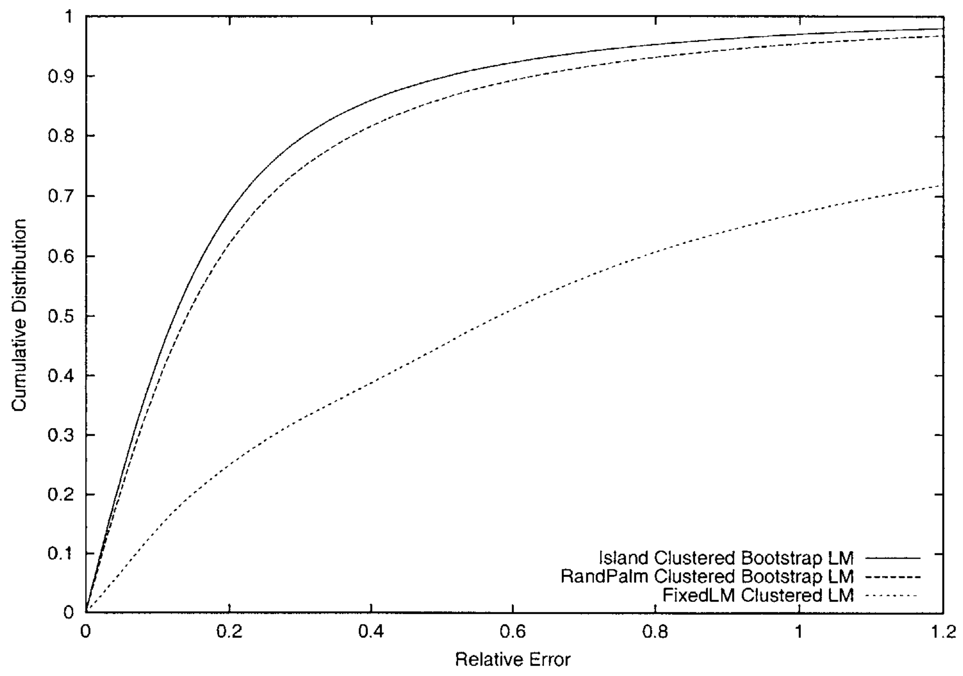


Figure A-19: Island relative error distribution using clustered bootstrap landmark placement. GT-ITM, $N = 3492$, 10 Landmarks.

Figure A-20 shows the summary statistics of the Island scheme when a clustered landmark set is used (compare with Figure A-15).

A.5 Nearest Peer Node Selection and Proximity Clustering

The ability to select the nearest node from a set of peer nodes is important to many applications, including nearest server/proxy selection, proximity routing in peer-to-peer networks and neighbor selection in overlay network construction. We use the stretch metric as our performance metric.

Figure A-21 shows the cumulative distribution of the stretch. We note that both RandPalm and Island perform worse than the FixedLM scheme in the nearest neighbor selection. This result should not come as a surprise. As discussed in the earlier section, the RandPalm scheme tends to grossly over-estimate RTT distances that are between 0 and 50ms, which negatively affects the nearest node selection performance

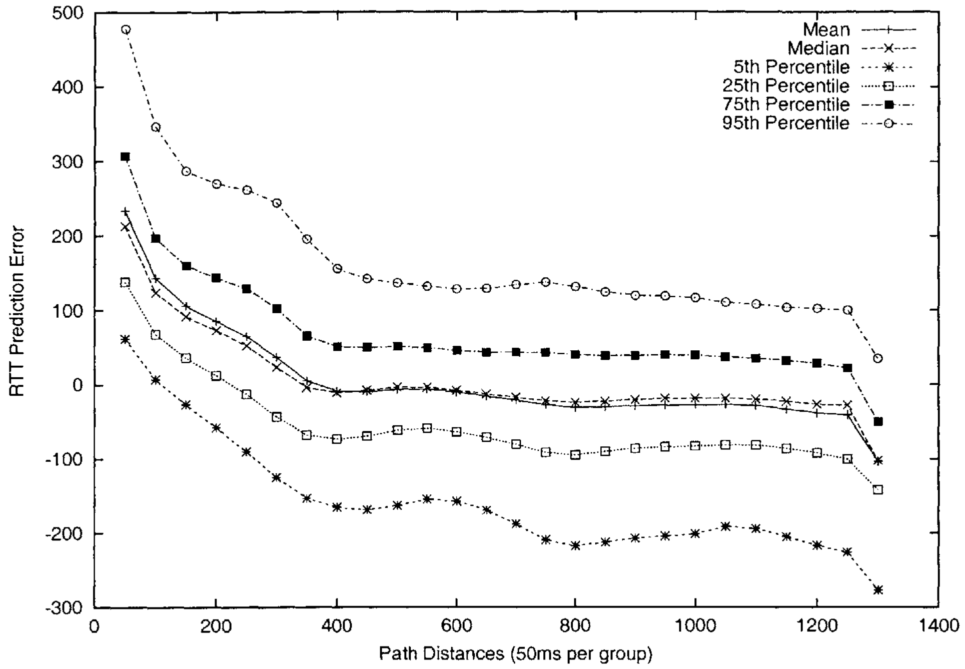


Figure A-20: Summary statistics of RTT prediction error for the Island scheme under clustered bootstrap landmark placement. GT-ITM, $N = 3492$, 10 Landmarks.

of the RandPalm scheme.

In order to further understand how well each scheme captures the network proximity relationships among hosts, we apply the KMeans clustering algorithm [20] on the coordinates generated by each scheme. The clustering criterion is the inter-host Euclidean distances defined by the coordinates. We then compute the weighted intra-cluster RTT averages for each clustering assignment, where the weight is the number of peers in each cluster, and the averages are computed using actual RTTs among hosts assigned to the same cluster. Figure A-22 shows the results of clustering performance when 10 and 30 landmarks are used. The RandPalm performs significantly worse than the other schemes when the number of landmarks used is small. We note that Island with the same number of landmarks yields cluster averages that are approximately 25% less than that of the RandPalm. When the number of landmarks is increased to thirty both RandPalm and Island yield cluster averages that are close to those of the FixedLM schemes.

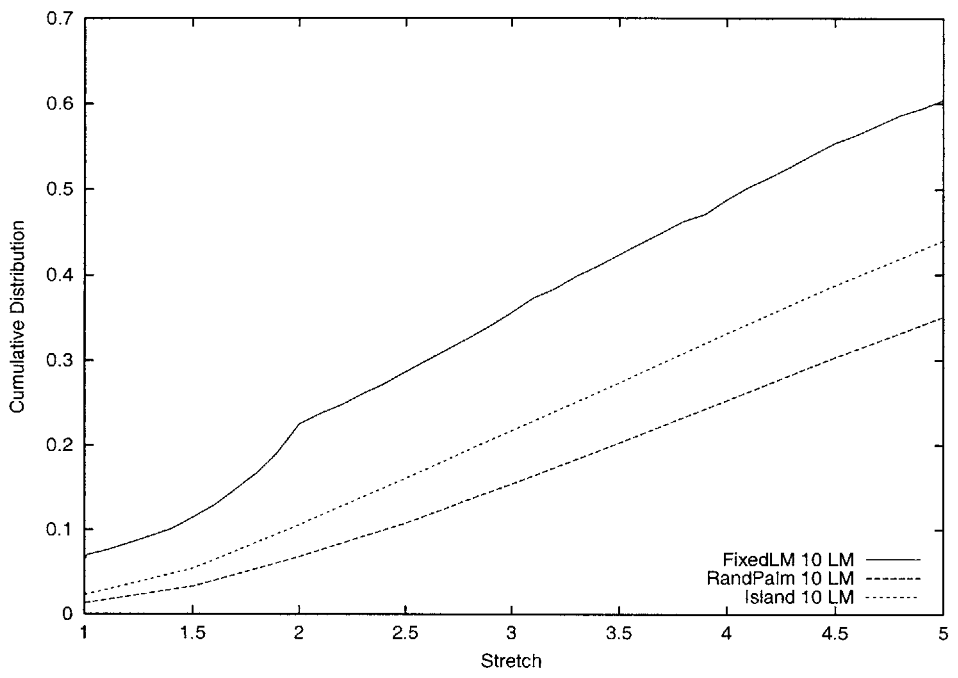


Figure A-21: Performance of selecting nearest peer node. GT-ITM, N = 3492.

A.6 Conclusion

In this Appendix, we examined the performance characteristics of a peer-to-peer approach in network topology modeling and distance prediction, named PALM. Similar to GNP[32], PALM models the Internet as a geometric space. End hosts compute their absolute coordinates to characterize their network locations based on distance measurements to a set of landmarks. In contrast to the GNP approach, which used a fixed set of landmarks, the goal of PALM is to allow peer nodes to construct their coordinates by using distance measurements to *any* participating peer nodes. We present two PALM-based schemes: RandPalm and Island. In RandPalm, a peer node randomly selects from existing peer nodes as its landmarks. In Island, each peer node intelligently selects its landmarks by exploiting the topological information contained in the PALM map (which contains coordinates of the existing peer nodes).

Through simulations using both real network measurements and simulated topologies, we compare the performance of RandPalm and Island with the original GNP scheme using fixed landmarks. We conclude with the following observations.

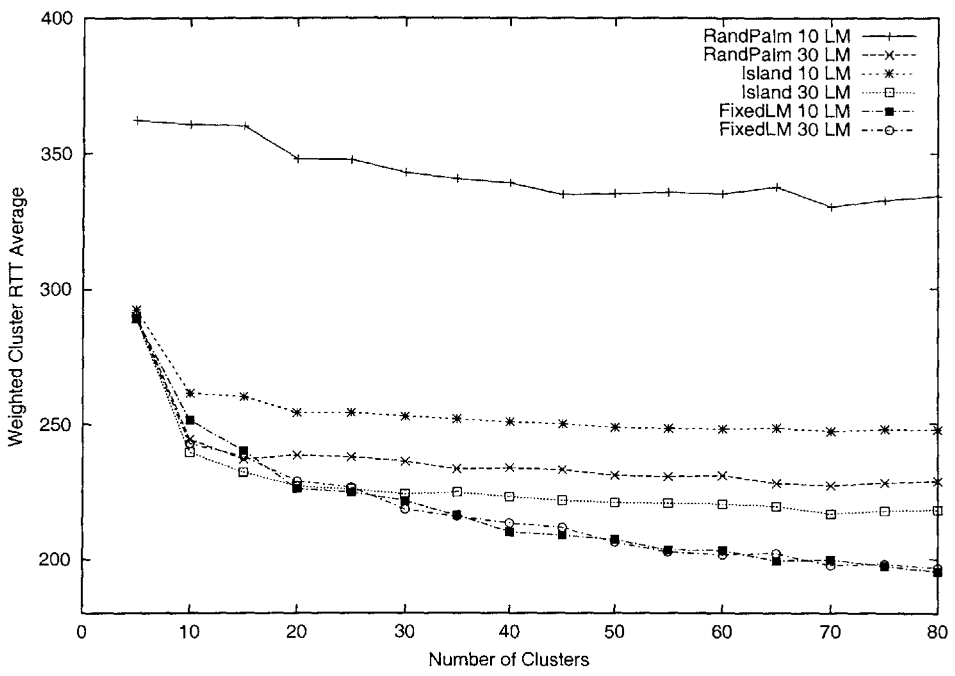


Figure A-22: Weighted Intra-Cluster RTT. Randomly selected bootstrap nodes. GT-ITM, N = 3492, 10 landmarks.

- Overall, PALM can achieve competitive prediction accuracy comparable to that of the FixedLM scheme for a significant fraction of the distances. When a random peer sampling strategy is used, PALM achieves comparable prediction accuracy to the FixedLM scheme when a reasonably large number of reference points, for example 30, is used. When the number of reference points used is small, PALM achieves performance comparable to that of the FixedLM scheme by using a sampling strategy that exploits the topological information in the coordinates of existing nodes.

- The PALM based approaches have rather different performance characteristics than the fixed landmarks based approach. The PALM-based approaches tend to over-predict small RTTs. The FixedLM scheme, in contrast, tends to under-predict large RTTs. The performance implication is that PALM coordinates do not provide quite enough resolution needed for nearest peer selection; however, PALM coordinates have comparable performance to FixedLM in clustering peer nodes based on their proximity relationships.

- The performance of the FixedLM approach can be very sensitive to the landmark placement. In contrast, PALM nodes have the advantage of being able to exploit the information about the coordinates of the existing nodes and select well-distributed sets of peers as their reference point set. Another important observation is that the performance of the PALM approaches are robust even in the face of suboptimal placement of the bootstrap nodes. Unlike the GNP landmarks whose placement greatly impacts the system performance, bootstrap nodes that are clustered in network do not perform worse than a well-distributed set of bootstrap nodes.

Besides the above observations, some interesting insights about the FixedLM scheme have also been presented in this Appendix. Our results show that although the overall distance prediction performance of the FixedLM scheme can suffer substantially when landmarks are misplaced, FixedLM is, however, very robust in predicting short network distances across all landmark configurations that we have tried.

In summary, our results indicate that RandPalm and Island yield good pair-wise distance prediction accuracy for a substantial fraction of distances. However, they are of limited value in predicting short distances. Additionally, there are several important issues that were not addressed. For example, we did not specify how bootstrap nodes are selected, and how new nodes find the coordinates of existing mapped nodes. Another important issue we did not address is whether and by how much error accumulate for later joining nodes. In a dynamic peer-to-peer environment, where both the membership and the underlying network conditions can change on a rather short time scale, there needs to be a mechanism for continuous re-calibration. Finally, there is a need to evaluate the scheme using larger-scale real Internet distance measurements.

Bibliography

[1] Joe Albowicz, Alvin Chen, and Lixia Zhang. Recursive position estimation in sensor networks. In *Proceedings of the Ninth International Conference on Network Protocols*. IEEE Computer Society, November 2001.

[2] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symp. on Operating Systems Principles (SOSP)*, Banff, Canada, October 2001.

[3] Mihai Badoiu. Approximation algorithm for embedding metrics into a two-dimensional space. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, Maryland, 2003.

[4] Suman Banerjee, Bobby Bhattacharjee, and Christopher Kommareddy. Scalable application layer multicast. In *Proceedings of ACM SIGCOMM'02*, Pittsburgh, Pennsylvania, August 2002.

[5] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28–34, October 2000. Special Issue on Smart Spaces and Environments.

[6] Yan Chen, Khian Hao Lim, Randy H. Katz, and Chris Overton. On the stability of network distance estimation. In *ACM SIGMETRICS Performance Evaluation Review (PER)*, September 2002.

[7] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics*, June 2000.

[8] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter 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.

[9] Russ Cox, Frank Dabek, Frans Kaashoek, Jinyang Li, and Robert Morris. Practical, distributed network coordinates. In *Proceedings of HotNets-II*, November 2003.

[10] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of SIGCOMM'04*, August 2004.

[11] Lance Doherty, Kristofer Pister, and Laurent El Ghaoui. Convex position estimation in wireless sensor networks. In *Proceedings of IEEE INFOCOM*, pages 1655–1663. IEEE Computer Society, April 2001.

[12] 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.

[13] Todd Hansen, Jose Otero, Tony Mcgregor, and Hans-Werner Braun. Active measurement data analysis techniques. http://amp.nlanr.net/, 2002.

[14] Jeffrey Hightower and Gaetano Borriello. A survey and taxonomy of location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.

[15] S.M. Hotz. *Routing Information Organization to Support Scalable Interdomain Routing with Heterogeneous Path Requirements.* PhD thesis, University of Southern California, 1994.

[16] Bradley Huffaker, Marina Fomenkov, Daniel J. Plummer, David Moore, and K Claffy. Distance metrics in the Internet. In *Proceedings of IEEE International Telecommunications Symposium*, 2002.

[17] Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01)*, 2001.

[18] Piotr Indyk and Jiri Matousek. Low distortion embeddings of finite metric spaces. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry (2nd edition)*. CRC Press LLC, to appear.

[19] Jon Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, pages 444–453, October 2004.

[20] MatLab statistics toolbox. http://www.mathworks.com/.

[21] R. Krauthgamer and J. R. Lee. The intrinsic dimensionality of graphs. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 438–447, jun 2003.

[22] R. Krauthgamer, N. Linial, and A. Magen. Metric embeddings–beyond one-dimensional distortion. *Discrete Computational Geometry*, 31(3):339–356, 2004.

[23] Liwei Lehman and Steven 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.

[24] Liwei Lehman and Steven Lerman. Predicting Internet network distances using peer-to-peer measurements. Internal technical report, appeared in Annual Singapore-MIT Alliance Symposium, January 2004.

[25] Jorg Liebeherr, Michael Nahas, and Weisheng Si. Application-layer multicasting with delaunay triangulation overlays. *IEEE Journal on Selected Areas in Communications*, 20(8):1472–1488, October 2002.

[26] Hyuk Lim, Jennifer Hou, and Chong-Ho Choi. Constructing Internet coordinate system based on delay measurement. In *Proceedings of Internet Measurement Conference(IMC'03)*, October 2003.

[27] Nathan Linial. Finite metric spaces - combinatorics, geometry and algorithms. In *Proceedings of International Congress on Mathematicians*, pages 573–586, Beijing, August 2002.

[28] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 557–591, 1994.

[29] Nathan Linial and Avner Magen. Least-distortion euclidean embeddings of graphs: Products of cycles and expanders. *Journal of Combinatorial Theory*, 79(157-171), 2000.

[30] T. S. Eugene Ng, Yang hua Chu, Sanjay G. Rao, Kunwadee Sripanidkulchai, and Hui Zhang. Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems. In *Proceedings of IEEE INFOCOM*, 2003.

[31] T.E. 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.

[32] T.S. Eugene Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of INFOCOM*, 2002.

[33] Katia Obraczka and Fabio Silva. Network latency metrics for server proximity. In *Proceedings of IEEE Globecom*, 2000.

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

[35] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA, August 2001.

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

[37] Marcelo Pias, John Crowcroft, Steven Wilbur, Tim Harris, and Saleem Bhatti. Lighthouses for scalable distributed location. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, February 2003.

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

[39] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C.* Cambridge Press, 1988.

[40] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking*, August 2000.

[41] Sylvia Ratnasamy, Paul Francis, Mark Handley, and Richard Karp. A scalable content-addressable network. In *Proceedings of SIGCOMM'01*, San Diego, CA, 2001.

[42] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of INFOCOM'02*, New York, 2002.

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

[44] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the International Conference on Distributed Systems Platforms*, November 2001.

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

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

[47] 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.

[48] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of SIGCOMM'01*, 2001.

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

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

[51] 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.

[52] Bernard Wong and Emin Gun Sirer. A lightweight approach to network positioning. Technical Report TR2004-1949, Cornell University, Department of Computer Science, 2004.

[53] Ben Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB/CSD-01-1141, UCB/CSD, April 2001.

4923-62