Proxy Network Coordinates

Jonathan Ledlie
Nokia Research
jonathan.ledlie@nokia.com

Margo Seltzer
Harvard University
margo@eecs.harvard.edu

Peter Pietzuch
Imperial College London
prp@doc.ic.ac.uk

January 2008

Abstract

Network coordinates can be used in large-scale overlay applications to reduce the cost of latency estimation. Previous proposals assumed that all nodes for which latencies were to be estimated actively participated in measurement and computation of network coordinates. In this paper, we introduce *proxy network coordinates*, a method that enables an overlay network to calculate network coordinates for external nodes without their direct involvement. We describe an algorithm for maintaining proxy network coordinates and show that their accuracy and stability properties are comparable to directly-maintained network coordinates.

1 Introduction

Network coordinates are a powerful, new tool for efficient latency estimation in networks with millions of nodes [3, 10]. By embedding a small set of latency measurements between Internet nodes into a metric space, they enable overlay applications to estimate missing measurements and reason geometrically about distributed systems problems such as routing and service discovery [6, 8, 13].

Previous work on network coordinates has assumed that the nodes to which coordinates were assigned are members of an overlay network. That is, they are hosts that run specialized application-level software (such as the *Pyxida* network coordinate implementation [15]) enabling them to cooperate in ways beyond what the operating system and standard network protocols provide. In this paper, we describe *proxy network coordinates*, which provide the same power of estimation in large networks, but free of the requirement that *all* nodes run the software for maintaining network coordinates. Instead, some overlay nodes maintain proxy coordinates for other Internet hosts without their involvement by managing latency measurements and recomputing coordinates.

This paper describes an algorithm for creating and maintaining proxy network coordinates and examines their convergence and accuracy empirically. We show that, in an on-going coordinate system (1) proxy network coordinates converge *faster* than standard coordinates undergoing system-wide start-up and (2) accuracy is comparable to standard ones. For an overview on network coordinates, the reader should refer to Chapter 2 of Ledlie's thesis [6].

The rest of the paper is organized as follows. In Section 2, we describe two application scenarios that benefit from proxy network coordinates. We introduce our algorithm for maintaining proxy coordinates in Section 3. In Section 4, we evaluate the performance of proxy coordinates on PlanetLab. Section 5 describes related work and Section 6 concludes.

2 Application Scenarios

The reader can imagine numerous contexts where legacy, non-participant, or, equally, *oblivious* hosts should be added to a set of network coordinates. In this section, we describe two examples of large-scale overlay network applications that benefit from using proxy network coordinates.

The first example from our work on a *Stream-Based Overlay Network (SBON)* [14] is incorporating data sources that are not running the SBON software. An SBON provides an Internet-scale stream-processing service that allows users to execute continuous streaming queries across a large number of data sources. It intelligently places in-network services "close" to where data is produced, much like selection operators are pushed down the operator stack in standard database optimization. For example, legacy web sites, collections of sensors, and other stream-based web services and feeds may be operated by third-party service providers that are not participating in, nor even aware of, the overlay. Nevertheless, the optimizer must take the network locality of these data sources into account, for example, in order to push query operators closer to data sources. For the SBON to place services based on oblivious hosts, it must compute network coordinates for them. By generating proxy network coordinates for these sites, we can extend the SBON's network-aware query optimization to include these oblivious hosts.

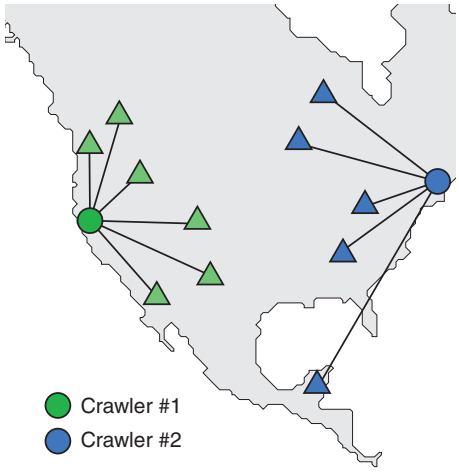

Figure 1: **Locality-aware Crawler Mapping.** This figures illustrates how web servers hosting RSS feeds are assigned to near-by crawlers. The assignment can be done by maintaining proxy network coordinates for the web servers without their direct involvement.

A second application scenario is a global system for content-based filtering of RSS feeds, such as our *Cobra* system [16]. Such a system must delegate hosts to crawl a large number of web servers, fetching updated articles from RSS feeds. A question in this setting is how to map RSS feeds to web crawlers. Figure 1 shows how two RSS feeds can be assigned to two crawlers based on *locality*, *i.e.,* the latency distance between the crawler and the web server hosting the RSS feed. This assignment has the advantage of providing low update latency, high bandwidth, and good crawling reliability. Using network coordinates for latency estimation reduces the cost of latency measurements. However, for this application, we cannot assume that web servers hosting RSS feeds are part of our system and thus calculate their own network coordinates. Instead, the crawler nodes can maintain proxy network coordinates on behalf of the web servers and use these coordinates to map web servers to crawlers.

3 Algorithm

In our approach for constructing a proxy network coordinate for an oblivious node, a host within the overlay network initiates a series of latency measurements from different viewpoints in the overlay and combines these measurements into the oblivious node's coordinate. We identify the overlay network host as the *proxy host*, and the oblivious node as the *target*. This network coordinate for the target will be refined over time as additional measurements are taken. The proxy host is then responsible for reporting the network coordinate for the corresponding target node.

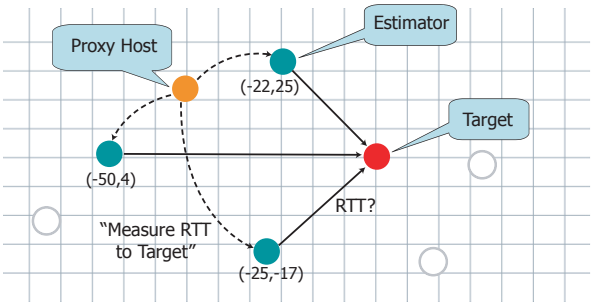

Figure 2: **Computing a Proxy Coordinate.** The proxy host contacts $log(n)$ estimators. Estimators measure their round trip time to the target node. They return this measurement and their current coordinate to the proxy host. The proxy host then computes a coordinate for the target node. This process continues over time, keeping the target's coordinate accurate as network conditions change

Proxy coordinates are computed as follows. Periodically, the proxy host asks other nodes in the overlay network to compute their latency to the target node. We call these other nodes *estimators*. Upon receiving such a request, the estimator measures the round-trip latency to the target and returns this information, along with its own network coordinate and its confidence in its coordinate, to the proxy. The proxy collects this information and uses it to compute the network coordinate for the target. Figure 2 illustrates this process.

With a new latency observation l_{ij}, the estimator j's coordinate $\overrightarrow{x_j}$, and j's confidence w_j, the proxy coordinate host is able to update the coordinate using the standard Vivaldi algorithm [3]. The update first determines the error of the observation,

$$\epsilon = \frac{|\ \|\overrightarrow{x_i} - \overrightarrow{x_j}\| - l_{ij}\ |}{l_{ij}} \quad (1)$$

where $\overrightarrow{x_i}$ is the proxy coordinate. Next, the update finds how much to weigh the observation based on the confidence of the proxy coordinate and of the estimator. Confidence ranges from $(0 \ldots 1)$ and summarizes the accuracy of a coordinate over time using an exponentially-weighted moving average. For example, if the proxy coordinate has low confidence, as it does when it is first created, and the update uses an estimator with high confidence, the measurement will exert a strong "pull" on the proxy coordinate. Finally, the proxy node updates the proxy coordinate's confidence w_i and its position:

$$\overrightarrow{x_i} = \overrightarrow{x_i} + \delta \times (\|\overrightarrow{x_i} - \overrightarrow{x_j}\| - l_{ij}) \times u(\overrightarrow{x_i} - \overrightarrow{x_j}) \quad (2)$$

where δ is the "pull" of the sample and u is the unit vector function.

The main distinction between a standard coordinate and a proxy coordinate is that the measurement occurs only in one direction: from estimators to targets. Of course, this makes the overly-simplistic assumption that links are symmetric; however, we find that in practice, proxy coordinate accuracy is almost as high as that of standard coordinates.

Within the overlay network, application-level UDP ping packets are used to measure internode latencies. This means that coordinates can often be computed even when hosts are situated behind firewalls that block ICMP pings. Of course, such application-level measurements cannot be used with oblivious hosts. Instead, proxy coordinate calculation uses two methods to obtain latency measurements: (1) *Scriptroute*, Spring's tool for remote ping measurement [19] and (2) *King*, Gummadi's *et al.* DNS-based latency measurement tool [4]. Both of these tools run on the proxy host. Scriptroute has the advantage that it is more accurate because it measures directly to the endpoint but has the disadvantage that its pings are blocked by many ISPs. King is less accurate because it measures distances between local DNS servers, not the endpoints themselves, but works even when pings are blocked. For our evaluations, we rely on the King method to generate latency measurements because it scales well with a larger set of hosts.

4 Evaluation

In this section we describe our evaluation of proxy network coordinates. We report our experience in maintaining a large number of proxy coordinates for Internet web servers and analyze the convergence and accuracy of proxy coordinates for PlanetLab hosts. Our results indicate that proxy coordinates only incur a small accuracy penalty compared to regular network coordinates and have favorable convergence properties.

4.1 Proxy Coordinates for Web Servers

As described in Section 2, overlay applications often need to estimate latencies to Internet web servers to make locality-aware decisions. Proxy coordinates are important for such applications because the involved overlay nodes cannot expect any cooperation from the probed web servers.

In our first experiment we analyze the computation of proxy coordinates for a large number of web servers on the Internet. For this, we deployed our overlay middleware on 200 PlanetLab nodes. We then used the implementation to calculate proxy coordinates for 1591 web servers hosting RSS feeds. We plot the proxy coordinates in a three-dimensional metric space after convergence in Figure 3. As one would expect, the plots shows that, at a coarse granularity, there are clusters with web logs from Asia, North America, and Europe. This is because connections of latencies between web servers on the same continent tend to be lower than connections spanning continents, which have to use high-latency, long-distance links. A locality-aware overlay application accessing these web servers could use such a clustering to ensure that web servers are only accessed from overlay nodes belonging to the same cluster. This would keep access latencies low and is also likely to improve reliability due to the short connections paths.

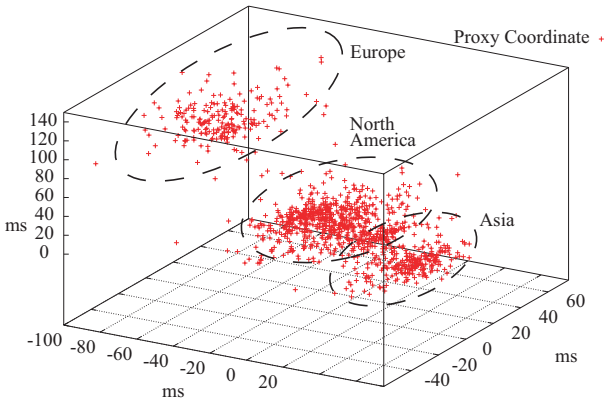

Figure 3: **Proxy Coordinates for Web Servers.** This plot shows the proxy coordinates of 1591 web servers hosting weblogs on the Internet. The clustering of coordinates according to geographic location is clearly visible.

4.2 Convergence and Accuracy

When generating proxy coordinates for oblivious hosts, we are concerned with two metrics: *convergence time* and *accuracy*. Convergence time is the time until a stable estimate of a node's "true" network coordinate can be determined. Ideally, convergence time should be low (on the order of minutes). Accuracy is a measure of ability of the network coordinate to predict latencies to other network nodes. In contrast to previous work [17], our goal is to obtain accurate and stable network coordinates over long period of time instead of rapid, instantaneous measurements with potentially low accuracy.

As an examination of how both of these properties hold, we show the convergence of a single coordinate and overall accuracy in Figure 4. To test convergence, we created a proxy coordinate for an overlay node (*i.e.,* a node capable of calculating its own coordinate) and measured the coordinate's accuracy at one minute intervals. We used 190 PlanetLab nodes and started the proxy coordinate after the overlay coordinates had stabilized. In Figure 4, the data shows that not only does the proxy coordinate stabilize quickly (under a few minutes), but it does so in less time than the standard coordinate. This is

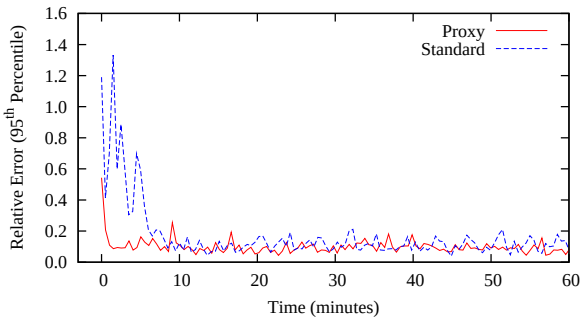

Figure 4: **Proxy Coordinate Convergence Time.** When an proxy coordinate is instantiated in an existing coordinate system, it quickly converges to a low error position. The data show that the convergence time for a proxy coordinate added to an existing coordinate system is negligible compared to the initial convergence time of a non-proxy coordinate for the same node. The figure shows the convergence time of two coordinates, one proxy and one non-proxy, for a PlanetLab node hosted at the University of Michigan.

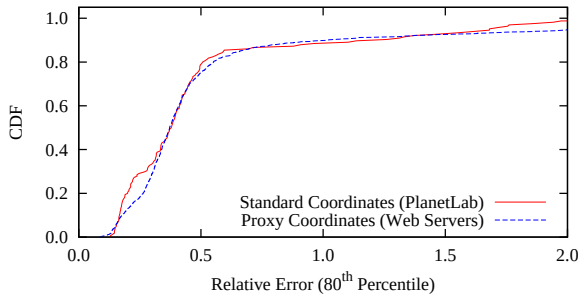

Figure 5: **Proxy Coordinate Accuracy matches Standard Coordinate Accuracy.** The data show the overall accuracy distribution for proxy coordinates for 1591 web servers and standard coordinates for 166 PlanetLab nodes. Each data point is the 80^{th} percentile of the accuracy distribution for that coordinate.

because the overlay nodes already had high confidence in their own coordinates when the proxy coordinate was created, which leads to quick convergence of new coordinates. This quick convergence would still be the case when new coordinates were created as part of an existing large system.

Figure 5 shows overall coordinate accuracy of the 166 PlanetLab nodes, which we use as the overlay, and the same 1591 web servers from Section 4.1. We plot the cumulative distribution of the 80^{th} percentile of the relative error of all links. The data indicates that the latency embedding provides a good estimate ($< 50\%$ relative error) for most links ($> 80\%$). The data also show that the accuracy of the proxy coordinates is as high as standard coordinates. This makes sense because the network round trip time from one node to another will be the same regardless of which host initiates the measurement (exclusive of end-point load and events). Thus, the same data is being used for coordinate computation even though, with proxy coordinates, only one side is doing the measuring. The experiment illustrates that proxy coordinates provide a quick and accurate method to include non-overlay data producers into network-aware optimizations.

5 Related Work

Shanahan and Freedman examine the efficacy of network embeddings for finding nearby servers for unmodified clients [17]. They envision a scenario where an *oblivious* client attempts to find the closest of many service replicas, which are part of an overlay: the replicas have network coordinates at the time of the query and the client does not. They assume the client knows one member of the overlay, the *ingress* node; this node attempts to determine a coordinate for the client and find it a nearby replica. There are several interesting aspects to Shanahan and Freedman's approach. Because they aim to generate a proxy coordinate extremely quickly, they do not add the oblivious node to a continuous measurement process. Instead, nodes are measured *once* from a variety of viewpoints. As we have shown in previous work, this scarcity of measurements can lead to anomalous measurements being included in the computed coordinate, resulting in artificially high error [9]. Further producing low quality coordinates, they choose to use coordinates of only two dimensions, which can experience higher error than those with more dimensions [7]. Their core result is that selecting a *direct* neighbor of the ingress node with the minimum round trip time is strictly superior (in terms of latency) to selecting its neighbor with the coordinate closest to the oblivious node.

Since Ng and Zhang provided the first examination of how to embed inter-node latencies in a metric space [10], a series of different approaches have emerged. In their initial work, called Global Network Positioning, a coordinate space was built in two stages: first, a collection of well-known *landmarks* placed themselves in a vector space through all-pairs ping measurements; second, each joining node measured its distance to all of the landmarks and picked a coordinate that minimized the error to all of them. This approach does not allow for a smooth evolution of the space over time, nor is it decentralized. However, it did establish that, even with the error induced by triangle inequality violations, a high-quality space was

possible. Lighthouses [12] Mithos [20], and NPS [11] extended the landmark approach by using multiple local coordinate systems, by building the space through preferring to measure nearby neighbors, and through a hierarchical architecture, respectively. More recently, Costa *et al.* developed PIC, another landmark scheme, which runs a Simplex solver on each node to minimize error [1]. PIC readjusts coordinates through periodically re-running this solver process and includes a test to defend its coordinate system against malicious participants. Cox *et al.* initially proposed Vivaldi [2] and Dabek *et al.* later improved its accuracy in two-dimensions with *height*, which was intended to explicitly capture the latency to a high speed link [3]. Shavitt and Tankel's Big-Bang Simulations is an embedding technique similar to Vivaldi, although it models a potential force field instead of a mass-spring system [18]. Kleinberg has developed a theoretical grounding for network embeddings, analyzing how to embed coordinates with arbitrarily low errors [5].

6 Conclusions

We described proxy network coordinates, a mechanism for overlay network nodes to maintain network coordinates for external nodes, such as clients and web servers, without their direct involvement. To calculate proxy coordinates, latency measurements to target nodes are assigned to different overlay nodes acting as latency estimators. Proxy hosts then combine the results and compute the proxy coordinates. Our evaluation shows that this approach leads to proxy coordinates having the same level of accuracy as standard ones and that proxy coordinates have good convergence properties.

We believe that proxy network coordinates will increase the applicability of network coordinates to new application domains. As future work, we are investigating a distributed content distribution system that uses proxy coordinates to map clients to content servers in a locality-aware fashion.

References

[1] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. In *Proc. of International Conference on Distributed Computing Systems*, Tokyo, Japan, March 2004.

[2] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Practical distributed network coordinates. In *Proc. of Workshop on Hot Topics in Networks*, Cambridge, MA, November 2003.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proc. of SIGCOMM*, Portland, OR, Aug. 2004.

[4] K. Gummadi, S. Saroiu, and S. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proc. of Internet Measurement Workshop*, Marseille, France, Nov. 2002.

[5] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *45th Symposium on Foundations of Computer Science*, Rome, Italy, October 2004.

[6] J. Ledlie. *A Locality-Aware Approach to Distributed Systems*. PhD thesis, Harvard School of Engineering and Applied Sciences, Cambridge, MA, 2007.

[7] J. Ledlie, P. Gardner, and M. Seltzer. Network Coordinates in the Wild. In *Proc. of the Symposium on Networked Systems Design and Implementation*, Boston, MA, Apr. 2007.

[8] J. Ledlie, P. Pietzuch, M. Mitzenmacher, and M. Seltzer. Wired Geometric Routing. In *Proc. of International Workshop on Peer-to-Peer Systems*, Bellevue, WA, Feb. 2007.

[9] J. Ledlie, P. Pietzuch, and M. Seltzer. Stable and Accurate Network Coordinates. In *Proc. of International Conference on Distributed Computing Systems*, Lisbon, Portugal, July 2006.

[10] E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proc. of INFOCOM*, New York, NY, June 2002.

[11] E. Ng and H. Zhang. A Network Positioning System for the Internet. In *Proc. of USENIX Annual Technical Conference*, Boston, MA, June 2004.

[12] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. Lighthouses for Scalable Distributed Location. In *Proc. of International Workshop on Peer-to-Peer Systems*, Berkeley, CA, February 2003.

[13] P. Pietzuch, J. Ledlie, M. Mitzenmacher, and M. Seltzer. Network-Aware Overlays with Network Coordinates. In *Proc. of International Workshop on Dynamic Distributed Systems*, Lisbon, Portugal, July 2006.

[14] P. Pietzuch, J. Ledlie, J. Shneidman, M. Welsh, M. Seltzer, and M. Roussopoulos. Network-Aware Operator Placement for Stream-Processing Systems. In *Proc. of International Conference on Data Engineering*, Atlanta, GA, April 2006.

[15] Pyxida: An Open Source Network Coordinate Library and Application. `http://pyxida.sourceforge.net/`.

[16] I. Rose, R. Murty, P. Pietzuch, J. Ledlie, M. Roussopoulos, and M. Welsh. Cobra: Content-based Filtering and Aggregation of Blogs and RSS Feeds. In *Proc. of the Symposium on Networked Systems Design and Implementation*, Boston, MA, Apr. 2007.

[17] K. Shanahan and M. Freedman. Locality Prediction for Oblivious Clients. In *Proc. of International Workshop on Peer-to-Peer Systems*, Ithaca, NY, February 2005.

[18] Y. Shavitt and T. Tankel. Big-Bang Simulation for embedding network distances in Euclidean space. In *Proc. of INFOCOM*, San Francisco, CA, June 2003.

[19] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A Public Internet Measurement Facility. In *Proc. of the Symposium on Internet Technolgies and Systems*, Seattle, WA, Mar. 2003.

[20] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. In *HotNets-I*, Princeton, NJ, Oct. 2002.