Network-Aware Overlays with Network Coordinates

Peter Pietzuch, Jonathan Ledlie, Michael Mitzenmacher, Margo Seltzer
Harvard University, Cambridge, MA, USA
hourglass@eecs.harvard.edu

Abstract

Network coordinates, which embed network distance measurements in a coordinate system, were introduced as a method for determining the proximity of nodes for routing table updates in overlay networks. Their power has far broader reach: due to their low overhead and automatic adaptation to changes in the network, network coordinates provide a new paradigm for managing dynamic overlay networks. We compare network coordinates to other proposals for network-aware overlays and show how they permit the lucid expression of a range of distributed systems problems in well-understood geometric terms.

1. Introduction

Overlay networks, initially designed for simple routing and storage, are increasingly used for network-sensitive applications such as distributed web caching, content dissemination, and stream processing. Application performance depends on the relation of the logical overlay topology to the current physical network topology, and the two have become more tightly coupled.

The first large-scale overlays, distributed hash tables (DHTs), were based on a purely logical identifier space, designed for load balancing and routing resilience. Such *network-oblivious* designs led to poor application-perceived performance. Overlays now incorporate full *network-awareness*, where it is a fundamental requirement to understand the physical network topology when constructing the overlay. As a result, network-aware overlays (NAO) are used to build applications that optimize for network metrics such as latency, bandwidth, and packet loss.

Maintaining network-awareness while the underlying network changes is a fundamental challenge for NAOs. Unfortunately, many NAOs have high measurement overhead, claiming it as a requirement for accuracy and adaptability [1, 45]. Also, NAOs are designed to solve a single task such as nearest neighbor search, limiting their generality.

Network coordinates (NC) are a promising new research direction for the decentralized construction of adaptive NAOs. With NCs, each node calculates its position in a virtual coordinate space using a small number of inter-node network measurements. The virtual distance between two coordinates is an estimate of network latency. Nodes continuously adjust coordinates to adapt to changes in the underlying network.

NCs have attractive properties and are a powerful abstraction: they have low run-time overhead; their embedding error is sufficiently low for practical applications; and by adjusting the measurement frequency, the trade-off between overhead and accuracy becomes explicit.

We show that NCs provide a rich assortment of geometric primitives for solving distributed systems problems such as nearest cache selection, content distribution, and resource placement. For example, a web cache can be placed at the centroid of all the client coordinates accessing the cache. The low dimensionality of NCs makes a wide range of algorithms from computational geometry applicable to networking problems, and their geometric interpretation unifies wired and wireless networks, making similar algorithms usable in both domains, simplifying the design of dynamic heterogeneous systems.

The paper is structured as follows: In Section 2, we describe the distinctions between network-oblivious, proximity-aware, and network-aware overlays. We then discuss NCs in Section 3, comparing different approaches for building coordinate spaces. We show the full power of NCs in Section 4, where we analyze how they can be used to solve several fundamental distributed system problems scalably and elegantly. In Section 5, we conclude.

2. Taxonomy of Overlays

In an overlay network, nodes create a limited number of connections to neighbors to form a topology. These connections are updated as the underlying network changes. Figure 1 shows different approaches to neighbor selection, ranging from *network-oblivious* to *network-aware*.

Network-Oblivious Overlays. *Network-oblivious* overlays create links to neighbors based on identifiers in a logical

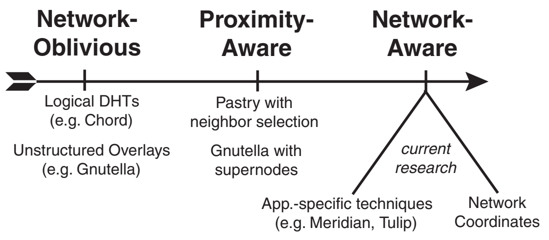
Network-Oblivious
Proximity-Aware
Network-Aware
Logical DHTs (e.g. Chord)
Unstructured Overlays (e.g. Gnutella)
Pastry with neighbor selection
Gnutella with supernodes
current research
App.-specific techniques (e.g. Meridian, Tulip)
Network Coordinates

Figure 1. Spectrum of research into overlays.

space. Structured DHTs, such as *Chord* [43], and *CAN* [34] are examples of network-oblivious overlays. In a structured DHT, each node creates network connections to its immediate neighbors in the logical identifier space and to a small subset of distant nodes to reduce the number of overlay routing hops [43]. However, a short distance in the identifier space may translate to a large distance in the underlying physical network, making routing inefficient.

DHTs are network-oblivious by design because their goal is a uniform data and load distribution. Because nodes pick logical identifiers at random, data items can be replicated on neighbors in the identifier space, causing replicas to be hosted at distant sites with independent failure distributions. However, many applications need short network paths in addition to good load and data distribution.

Proximity-Aware Overlays. DHTs have some freedom when selecting nodes for routing tables. Proximity-aware DHTs [48] such as *Pastry* [38] and *Tapestry* [47] exploit this observation by preferring to include physically close nodes in routing tables. For example, Pastry updates its routing table when it discovers a new node that shares the same identifier prefix but has lower latency than an existing entry. An extension to Chord [43] updates its finger tables in the background, replacing distant entries with closer ones. A *proximity-aware* overlay is an overlay that does not rely on locality for correctness but uses it to improve network efficiency. The disadvantage of proximity-aware overlays is that routing is still based on a logical identifier space. Proximity-aware decisions are made only when there is a choice between nodes. The dynamism of the network should determine the rate of routing table updates to maintain proximity-awareness but the rate often cannot be controlled independently from the DHT routing workload. This more recent work on proximity-aware overlays preserves some degree of randomized load balancing while increasing routing efficiency. Fundamentally, however, the techniques function *within* the bounds of network-oblivious overlays.

Network-Aware Overlays. Current research on NAOs departs from a logical identifier space, creating an overlay topology that is based purely on physical node distance [26, 44]. A *network-aware* overlay exploits locality for its underlying routing strategy. The *locality principle*, which states that network traffic with only local relevance should stay local [11], is the fundamental element of network-awareness. Local network traffic experiences better quality of service characteristics, such as latency, bandwidth, and reliability, due to the smaller number of routing hops involved. Network-awareness also enables intelligent wide-area choices, such as formation of a content distribution tree or selection of a good node on which to host a distributed join. These applications benefit from a network-aware approach.

There are two approaches for NAOs to adapt to network dynamism: *reactive* overlays initiate network measurements to determine node distances when routing a message, and *proactive* overlays periodically perform measurements in the background to maintain up-to-date routing state. Reactive overlays always use fresh measurements, resulting in a large overhead when overlay usage is high, and adaptivity based upon load instead of network dynamism. Proactive overlays decouple measurement overhead from overlay usage but can suffer from stale information.

Meridian [45] is an example of a reactive NAO. Each Meridian node keeps track of a fixed set of neighbors and organizes them into exponentially increasing rings according to network distance. To find the nearest neighbor to a non-Meridian node (the target), an initiating node measures its latency to the target. The initiating node then requests all neighbors in the ring corresponding to the target's latency and half of the nodes in the adjacent rings to probe the target. The initiating node selects the neighbor reporting the shortest distance to the target, sends the message to that neighbor, and the process repeats with the neighbor as the initiator. No latency errors are introduced because routing is based only on current measurements. However, each query has a large measurement overhead and the results may be inaccurate when a node's ring membership changes.

Tulip [1] exemplifies a network-aware, proactive, two-hop routing overlay. Each node selects a random color and maintains a *color list* with all nodes of its color and a *vicinity list*, containing one node of every other color. A node routes a message by sending it to the node in the vicinity list of the target's color. That node routes the message directly to the target using its color list. Tulip uses gossip and direct node-to-node measurements to ensure that a node is close to the nodes in its vicinity list. These measurements result in Tulip's maintenance overhead.

In contrast to Tulip's direct measurements, *Mithos* [44] is a reactive NAO that uses NCs for routing. Each node calculates a static NC and maintains links to its immediate neighbors in every direction. Messages are routed greedily towards a target. Since routing tables do not contain long-distance links, hop count is high unless a space with high dimensionality (and overhead) is used. Also, the coordinates are not updated dynamically to adapt to latency changes.

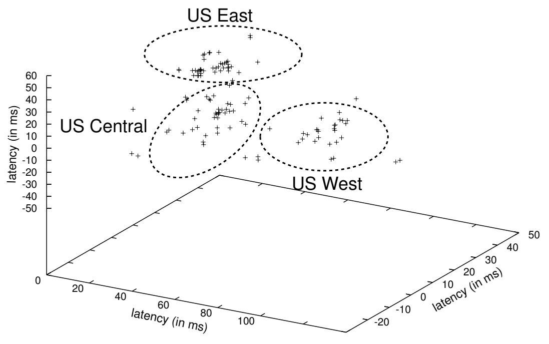

Figure 2. A latency space on PlanetLab

3. Network Coordinates

NCs, which embed inter-node latency in a low-dimensional geometric space [19, 28], provide an alternative approach to constructing NAOs. As network conditions change, each node maintains its location in the coordinate space keeping the distance in virtual space an estimate of the latency. Nodes can route messages in the coordinate space, mapping between coordinates and physical nodes.

In Figure 3 we used *Vivaldi* [9] to construct a 3-dimensional latency space with NCs of 115 North American PlanetLab (PL) [33] nodes. The three clusters of nodes that become apparent correspond to three geographic regions, as annotated. Continuous coordinate spaces encoding network latencies provide a number of advantages:

Overhead. They reduce measurement overhead to a linear or near-linear number of node pairs because non-existent measurements are approximated, making them feasable for large-scale networks. This approximation works well, because nodes that share the same local-area network often exhibit similar communication characteristics.

Dynamism. The coordinate space adapts to dynamic network changes as overlay nodes update their coordinates iteratively [22]. The maintenance overhead can be chosen to reflect the amount of dynamism in the network. Its proactive nature means there is no measurement overhead when messages are routed using the space. The space can also be maintained in a distributed fashion [42].

Algorithms. NCs give distributed routing problems a geometric meaning. As a result, many well-understood primitives from computational geometry become applicable to distributed routing and location problems. In Section 4, we describe general algorithms that solve common problems.

Domains. NCs unify wired and wireless network domains. Since wireless networks have a geographic communication model, nodes can only communicate directly with nodes in their vicinity. NCs approximate this by bringing locality and direction to the wired network world. Algorithms developed for wireless networks [2, 35] can be used in wired networks with NCs.

The main drawback of NCs is the embedding error that arises when Internet latencies violate the triangle inequality [25, 49]. In practice, however, we have shown that a low dimensional space can predict latencies with an accuracy that is sufficient for many applications [31].

3.1. Algorithms

Several algorithms for calculating NCs using measured latencies exist. There are two classes of algorithms: *landmark-based* schemes, in which overlay nodes use a fixed number of landmark nodes to calculate their coordinates, and *simulation-based* schemes, which are decentralized and calculate coordinates by modelling nodes as entities in a physical system.

Landmark-based. In *GNP* [28], nodes contact multiple landmark nodes to calculate their coordinates. The drawbacks of this approach are that the accuracy of the coordinates depends on the choice of landmark nodes and landmark nodes may become a bottleneck. *Lighthouses* [30] addresses this by supporting multiple independent sets of landmarks with their own coordinate systems. These local coordinates are mapped into a global coordinate system. *PIC* [8] does not use explicit landmarks, incorporating any node's measurements using a simplex optimization algorithm to obtain an up-to-date coordinate.

Simulation-based. *Vivaldi* [10] and *Big Bang Simulation* [39] determine coordinates using spring-relaxation and force-field simulation, respectively. In both, nodes attract and repel each other according to network distance measurements. The low-energy state of the physical system corresponds to the coordinates with minimum error. Vivaldi is the most widely-used, because of its clean decentralized implementation. *Azureus*, a BitTorrent client [6], and *SBON* [32], a distributed streaming query optimizer, use Vivaldi to create long-running NCs.

3.2. Accuracy and Overhead

Two fallacies slowed the acceptance of NCs: they have too low accuracy and too high measurement overhead.

The relative error, the baseline metric for accuracy, is the ratio of the absolute prediction error to the actual network latency. Coordinates using Vivaldi exhibit median relative errors between 5–10% using 3–5 dimensions on wide area networks [10]. However, the error's effect on application-specific operations is a more significant measure. For example, can NCs be used to reliably find a node's nearest neighbor? To answer this question, we gathered all-pairs ping measurements between 226 PL nodes and assigned the nodes coordinates with Vivaldi. We use *Nearest Neighbor Loss* (NNL) to capture the application-observed latency

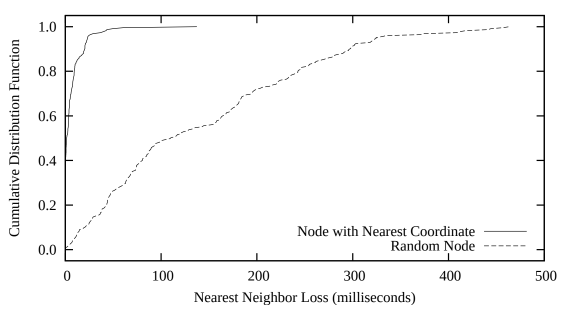

Figure 3. Nearest Neighbor Accuracy of NCs.

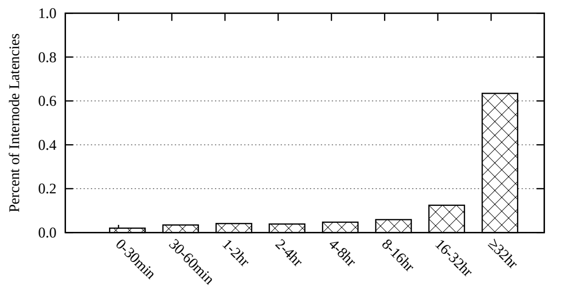

Figure 4. Frequency of PL latency change.

penalty for using a node that is not the true nearest neighbor. We define this loss as the difference between the latency to the node thought to be the nearest neighbor and the latency to the true one. In Figure 3, we show NNL for nodes if they use the node with the nearest coordinate as a proxy for the true nearest neighbor. We also show the baseline penalty if they had assumed a random node was their nearest neighbor. Not only are NCs far more capable of predicting "closeness" than choosing randomly, but also the absolute penalty is low: 35% of the time the NCs have no loss – they accurately find the nearest neighbor – and 96% of the time NCs predict within a $25ms$ penalty. Because the absolute error is small, in the vast majority of cases, the nearest coordinate is a sufficient substitute for the true nearest neighbor. In fact, NCs supply an answer that is "good enough" for a wide range of problems.

The overhead required to maintain an accurate network coordinates depends on the rate of node churn and internode latency, in particular. Adapting to churn is a fairly well-understood problem [36, 23], but adapting to internode latency change in the context of NCs is not. If the observed latency between nodes does not change over time, a static all-pairs matrix with one-off measurements could eliminate the need for repeated measurements. If link latencies change very frequently, maintaining accurate NCs would require overheads that dominate using a reactive query technique. In Figure 4, we show the frequency with which internode latencies change, where a change is defined as $\pm 10\%$

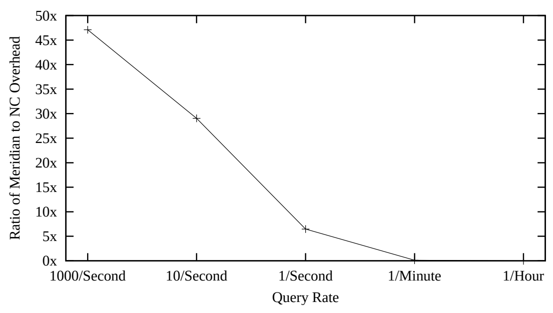

Figure 5. NAO Overhead.

from the current latency (after eliminating outliers according to [31]). The data portrays that some links are very constant, but approx. 30% change significantly in observed latency with 5.4% changing more than once per hour: continuous adaptation is required, even in this wired, well-provisioned network. In practice, we found that a maintenance rate of only one measurement per minute per node is sufficient to keep coordinates reasonably accurate on PL.

There is a trade-off between reactive, accurate, but expensive queries, and slightly less accurate but cheap queries that can be performed with NCs. When queries are infrequent or when network latencies are highly dynamic, building and maintaining coordinates is at best inefficient and at worst inaccurate. To discern the trade-off point in overhead between NCs (Vivaldi) and a reactive, active measurement mechanism (Meridian), we estimate a node's overhead over a range of query frequencies. We use the same parameters as in the evaluation of Meridian (2048 nodes, 16 nodes per ring) [45] and assume that NCs use CAN to find the nearest neighbor. A CAN lookup takes $(d/4)n^{1/d}$ hops. In Figure 5, we show the ratio of the number of messages a node needs to route using NCs and using Meridian with varying query rates. These estimates include the maintenance overhead for $3d$ NCs.

The results show that when a system rarely performs network-aware tasks, maintaining NCs is not worthwhile. However, when the system-wide query frequency is greater than once per minute (*i.e.*, each node is doing a lookup once every 34 hours on average), it is cheaper to maintain NCs than to perform reactive network measurements for each query. Assuming that the network is moderately, not exceedingly, dynamic, applications that perform network-aware tasks at this frequency or higher and that can accept reasonable, but not perfect, accuracy should use NCs instead of a heavyweight, reactive mechanism.

3.3. Discussion

Latency is the primary network metric that has been embedded in coordinate spaces. There are at least two approaches to including other network characteristics, such as

Dist. Sys. Problem	Geometric Algorithm
Message routing	Geographic routing [3, 14, 17], Small world routing [18]
Content distribution	Minimum spanning tree [16], Cluster analysis [7]
Resource location	Nearest neighbor query [37], Geographic routing [3, 14, 17]
Resource placement	Centroid calculation, Facility location [40], Steiner trees [15], Spring relaxation [32]
Resource replication	Furthest neighbor query [13], Reverse nearest neighbor query [20], Geographic quorums [12]
Network analysis	Cluster analysis [7], Principal component analysis [21]
Unknown problem	Travelling salesman problem [4] Minimum length matching [13]

Table 1. Mapping to geometric solutions.

bandwidth and jitter. We can make them additional dimensions in existing latency space. We demonstrated that per-node characteristics, such as load, can be included to form a *cost space* that allows hot-spot detection [41]. Second, Oppenheimer *et al.* and Lee *et al.* have investigated the inverse correlation between latency and bandwidth [24, 29]. The correlation Oppenheimer found implies that network-aware decisions made in the latency space may result in good bandwidth characteristics.

Load balancing was an initial motivation for the network-oblivious design of storage-focused DHTs. NCs enable several new techniques for load balancing and replication. First, node load can be expressed as a dimension in the coordinate space. An over-loaded node will eventually "move away" in the coordinate space, reducing its participation in routing and queries. Second, diffusion techniques avoid congested regions in the coordinate space. Finally, we can do better than random replication by placing replicas with furthest neighbor queries.

4. Using Network Coordinates

NCs solve many distributed systems problems using primitives from computational geometry. This provides a framework in which to handle these problems in a clean and uniform way by considering their geometric meaning. Table 1 shows a number of distributed systems problems and their mapping to geometric techniques. Note that there are geometric solutions that do not correspond immediately to networking problems. These may become meaningful in the future, enabling applications not possible today.

Since low-dimensional NCs obtain good results in terms of physical network metrics, this makes many geometric algorithms appealing because their running time is often a function of the dimensionality of the coordinate space. In addition, many NP-hard geometric optimization problems have approximate solutions sufficient for most practical applications [5] and NCs enable distributed applications to take advantage of these algorithms. We now describe the problems and the proposed algorithms in more detail. We use N to refer to the number of overlay nodes and D to the network diameter.

Message Routing. The routing problem is to send a message from a source node to a single recipient node via hops in the overlay network. An efficient route minimizes the *hop count* and *delay stretch*, which is the ratio of the experienced delay to the direct physical communication delay.

Intuitively, an algorithm that uses NCs can take advantage of the "sense of direction" of the space to achieve efficient routing with a small number of neighbor links. Most algorithms, including the ones mentioned below, create routing tables that connect nodes to their neighbors and then use greedy routing with each hop to reduce the metric distance to the recipient. Care must be taken that a greedy routing approach does not fail due to local minima.

The simplest approach constructs a θ*-graph spanner* [17, 44] linking nodes to their closest neighbors in every angle θ, but this results in a linear number of hops. An improvement is to add *long-distance links* that reduce the hop count to $O(\log D)$ [14]. This results in a technique similar to small-world routing [18], which reduces hop count by creating a network with a small diameter. The *compact routing* algorithm in a coordinate space improves routing efficiency further [3]. This algorithm has a logarithmic hop count, a $(1 + \epsilon)$ delay stretch, and requires only a constant number of neighbors.

Content Distribution. In content distribution, a source node disseminates messages to a set of recipient nodes. The standard approach builds a multicast tree optimizing hop count and delay stretch.

In a geometric space, an approximate minimum spanning tree [16] algorithm selects overlay nodes for content dissemination in $O(N)$ running time. An alternative approach identifies clusters of nodes in the coordinate space and builds a hierarchical dissemination tree using information that respects the structure of the coordinate space. This approach resembles previous work on the construction of hierarchies on top of logical rings in network-oblivious overlays [46]. There are also several algorithms for clustering in geometric spaces [7] that are applicable.

Resource Location. In the resource location problem, an overlay node wants to find the nearest overlay node to a given target location in the network. This problem arises in many applications such as web cache selection, database replication, and multi-player game server selection.

The geometric interpretation of this problem is a *nearest neighbor search* in a coordinate space. The database community developed centralized algorithms for finding nearest neighbors in multi-dimensional spaces to answer spatial queries [37]. The routing algorithms from Section 4 lend themselves to distributed implementation solutions to this problem. Each routing hop attempts to bring the message closer to the target location. Therefore, a message that is sent to a target node will often be routed via the nearest neighbor to the target in the overlay because of the greediness of the routing strategy.

Resource Placement. The resource placement problem is finding a node that can host a resource, while minimizing the network distance to clients.

There are several geometric algorithms for resource placement. A simple geometric solution places the resource at the *centroid* of the clients in the coordinate space. This minimizes the network distance for all surrounding clients. For more general geometric layouts, this problem corresponds to finding the Steiner tree [15]. Approximation algorithms, for example based on *spring relaxation* [32], which minimizes the potential energy of a network of springs in the coordinate space, provide a solution to the resource placement problem.

An algorithm to solve the *facility location problem* [27] attempts to find optimal locations for facilities, such as warehouses, given a set of clients, such as stores, and their locations. There are approximation algorithms [40] that minimize the incurred costs in terms of distance from stores to warehouses. A variation of this is the *k-median problem* [40] where a fixed number of k facilities must be placed. These algorithms can be used to compute optimal placement locations for resources in the coordinate space. The computed placement locations for resources are then mapped back (using a nearest neighbor search) to existing overlay nodes that are closest to these locations.

Resource Replication. The goal of resource replication is to add a new replica to an existing set of replicas in the overlay. This can be regarded as a specialized version of resource placement problem. A good replica placement ensures that failures of replicas are independently distributed, while placing replicas close to clients.

Algorithms for *furthest neighbor search* [13] ensure that replicas are distant, and hence more likely to experience independent faults in the network. A *reverse nearest neighbor query* [20] returns all nodes that would benefit from the placement of a replica at a given location by finding the nodes whose nearest neighbor the new replica would be come. This allows a system to decide on the most beneficial location for new replicas. A distributed system with replicas can implement shared memory using *GeoQuorums* [12], borrowing a technique from wireless routing. GeoQuorums use locks on geographic regions to form a quorum that ensures consistency.

Network Analysis. Network analysis is a broad area. Here an overlay network can discovery the topology or monitor the state of the underlying physical network. Such a system can exploit the latency information captured by the NCs to infer network properties.

Many statistical methods exist to analyze and infer properties of a geometric space. *Cluster analysis* [7] can be used to discover congestion hot spots in the network by looking at overlay nodes that are close to each other and receive much of the traffic. In recent work *principal component analysis* has been used to discover structure of measured network flows [21]. Similar techniques could be used with NCs after mapping measured flow information into the coordinate space.

Other Algorithms. We have so far listed only those geometric algorithms that have an obvious potential application in overlay networks. However, there exist many other geometric algorithms that may not currently have an apparent network interpretation, but may prove applicable in the future. For example, there are many fast approximation algorithms for the *traveling salesman problem* [4] or *minimum length matching* [13]. Using the coordinate paradigm allows a large class of algorithmic primitives to be immediately available for future distributed systems problems.

5. Conclusion

NCs offer such powerful potential for overlay networks that we are confident they will become a fundamental paradigm for dynamic overlay management. By providing an underlying geometric framework, NCs allow application of a rich, unified set of algorithmic tools and techniques to a variety of network problems. While developing coordinates with near-perfect accuracy is a long-term challenge, current approaches are already sufficiently accurate for most applications and allow trade-offs between accuracy and measurement overhead for dynamic network-aware overlays.

References

[1] I. Abraham, A. Badola, D. Bickson, D. Malkhi, S. Maloo, and S. Ron. Practical Locality-Awareness for Large Scale Information Sharing. In *Proc. of IPTPS'05*, Feb. 2005.

[2] I. Abraham, D. Dolev, and D. Malkhi. LLS: a Locality Aware Location Service for Mobile Ad Hoc Networks. In *Proc. of DIAL-M-POMC'04*, Philadelphia, PA, Oct. 2004.

[3] I. Abraham and D. Malkhi. Compact Routing on Euclidian Metrics. In *Proc. of PODC'04*, July 2004.

[4] S. Arora. Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems. *Journal of the ACM*, 45(5), Sept. 1998.

[5] S. Arora. Approximation Schemes for NP-hard Geometric Optimization Problems: A Survey. *Math Prog.*, Aug. 2003.

[6] Azureus BitTorrent Client. `http://azureus.sourceforce.net/`.

[7] Y. Bartal, M. Charikar, and D. Raz. Approximating Min-Sum k-Clustering in Metric Spaces. In *STOC'01*, July 2001.

[8] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. In *Proc. of ICDCS'04*, Tokyo, Japan, Mar. 2004.

[9] R. Cox, F. Dabek, F. Kaashoek, et al. Practical, Distributed Network Coordinates. In *Proc. of HotNets'03*, Nov. 2003.

[10] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proc. of SIGCOMM'04*, Aug. 2004.

[11] P. J. Denning. The Locality Principle. *Communications of the ACM*, 48(7), July 2005.

[12] S. Dolev, S. Gilbert, N. A. Lynch, A. A. Shvartsman, and J. Welch. GeoQuorums: Implementing Atomic Memory in Mobile Ad Hoc Networks. In *Proc. of DISC'05*, Sept. 2005.

[13] A. Goel, P. Indyk, and K. R. Varadarajan. Reductions among High Dim. Proximity Prob. In *SODA'01*, Jan. 2001.

[14] Y. Hassin and D. Peleg. Sparse Communication Networks and Efficient Routing in the Plane. In *Proc. of PODC'00*, Portland, Oregon, 2000.

[15] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*. North-Holland, 1992.

[16] P. Indyk. Sublinear Time Algorithms for Metric Space Problems. In *Proc. of STOC'99*, Atlanta, GA, May 1999.

[17] J. M. Keil and C. A. Gutwin. Classes of Graphs Which Approximate the Complete Euclidean Graph. *Discrete & Computational Geometry*, 7(1), 1992.

[18] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proc. of STOC'00*, May 2000.

[19] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and Embedding Using Small Sets of Beacons. In *Proc. of FOCS'04*, Rome, Italy, 2004.

[20] F. Korn and S. Muthukrishnan. Influence Sets based on Reverse Nearest Neighbor Queries. In *SIGMOD*, May 2000.

[21] A. Lakhina, K. Papagiannaki, M. Crovella, et al. Structural Analysis of Network Traffic Flows. *SIGMETRICS Perform. Eval. Rev.*, 32(1), June 2004.

[22] J. Ledlie, P. Pietzuch, and M. Seltzer. Stable and Accurate Network Coordinates. In *Proc. of ICDCS'06*, Lisbon, Portugal, July 2006.

[23] J. Ledlie and M. Seltzer. Distributed, Secure Load Balancing with Skew, Heterogeneity, and Churn. In *INFOCOM*, Miami, FL, Mar. 2005.

[24] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and F. Rodrigo. Measuring bandwidth between planetlab nodes. In *PAM*, Boston, MA, Mar. 2005.

[25] E. K. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft. On the Accuracy of Embeddings for Internet Coordinate Systems. In *IMC*, Berkeley, CA, Oct. 2005.

[26] D. Malkhi. Locality-Aware Network Solutions. Technical Report TR 2004-6, School of CS and Engineering, The Hebrew University, June 2004.

[27] P. B. Mirchandani, , and R. L. Francis, editors. *Discrete Location Theory*. Wiley-Interscience, 1999.

[28] T. S. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proc. of INFOCOM'02*, June 2002.

[29] D. Oppenheimer, D. A. Patterson, and A. Vahdat. A Case for Informed Service Placement on PlanetLab. Technical Report 04-025, PlanetLab, 2004.

[30] M. Pias, J. Crowcroft, S. Wilbur, et al. Lighthouses for Scalable Distributed Location. In *Proc. of IPTPS'03*, Berkeley, CA, Feb. 2003.

[31] P. Pietzuch, J. Ledlie, and M. Seltzer. Supporting Network Coordinates on PlanetLab. In *Proc. of WORLDS'05*, San Francisco, CA, Dec. 2005.

[32] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer. Network-Aware Operator Placement for Stream-Processing Systems. In *Proc. of ICDE'06*, Atlanta, GA, Apr. 2006.

[33] PLC. Planetlab. http://www.planet-lab.org, 2002.

[34] S. Ratnasamy, P. Francis, M. Handley, et al. A Scalable Content-Addressable Network. In *Proc. of SIGCOMM'01*, San Diega, CA, 2001.

[35] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: A Geographic Hash Table for Data-Centric Storage. In *Proc. of WSNA'02*, Sept. 2002.

[36] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling Churn in a DHT. In *USENIX '04*, Boston, MA, June 2004.

[37] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest Neighbor Queries. In *Proc. of SIGMOD'95*, May 1995.

[38] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of Middleware'01*, Nov. 2001.

[39] Y. Shavitt and T. Tankel. Big-Bang Simulation for Embedding Network Distances in Euclidean Space. In *Proc. of INFOCOM'03*, San Francisco, CA, Mar. 2003.

[40] D. B. Shmoys, E. Tardos, and K. Aardal. Approx. Algorithms for Facility Location Problems. In *Proc. of STOC'97*, El Paso, TX, 1997.

[41] J. Shneidman, P. Pietzuch, M. Welsh, M. Seltzer, and M. Roussopoulos. A Cost-Space Approach to Distributed Query Optimization in Stream Based Overlays. In *Proc. of NetDB'05*, Tokyo, Japan, Apr. 2005.

[42] A. Slivkins. Distributed Approaches to Triangulation and Embedding. In *Proc. of SODA'05*, Jan. 2005.

[43] I. Stoica, R. Morris, D. Karger, et al. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM'01*, San Diego, CA, Aug. 2001.

[44] M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. *SIGCOMM Rev.*, 33(1), 2003.

[45] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *Proc. of SIGCOMM'05*, Aug. 2005.

[46] Z. Zhang, S.-M. Shi, and J. Zhu. SOMO: Self-Organized Metadata Overlay for Resource Management in P2P DHT. In *Proc. of IPTPS'03*, Berkeley, CA, Feb. 2003.

[47] B. Y. Zhao, L. Huang, J. Stribling, et al. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Comm.*, 22(1), Jan. 2004.

[48] B. Y. Zhao, A. Joseph, and J. Kubiatowicz. Locality-Aware Mechanisms for Large-Scale Networks. In *Proc. of FuDiCo'02*, 2002.

[49] H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin. Internet Routing Policies and Round-Trip-Times. In *Proc. of PAM'05*, Mar. 2005.