Resource Addressable Network: An Adaptive Peer-to-Peer Discovery Substrate for Internet-Scale Service Platforms

Balasubramaneyam Maniymaran
Department of Electrical & Computer Engineering,
McGill University, Montreal, QC, Canada
bmaniy@cs.mcgill.ca

Supervisor: Dr. Muthucumaru Maheswaran

Abstract

The primary enabling component for using a *peer-to-peer* (P2P) system as a general service platform is an efficient P2P resource discovery mechanism. The design of a P2P resource discovery mechanism differ from other resource discovery mechanisms because it has to handle the volatile memberships in P2P systems. Controlling the management overhead, implementing fully decentralized mechanisms, and designing against node failures are some of the challenges a P2P discovery service has to handle. I introduce a novel P2P discovery mechanism called *resource addressable network* (RAN) that is designed to address the above issues. The scalability is a primary concern in the design of the RAN and the constituting algorithms are designed to achieve this goal. RAN can discover resources, i.e., the machines comprising the system, based on both their attributes and their locations. It implements number of supplemental schemes to support this two-tier discovery. RAN's location-based discovery provides a $O(1)$ route complexity based on the observation that locationing on the Internet is not worthwhile beyond a given resolution. This report provides detailed analysis of resource discovery concepts, presents the design of RAN, compares existing algorithms, presents new algorithms motivated from the drawbacks in the existing ones, and provides simulation results to evaluate the performance of RAN and its components.

Contents

1 Introduction

Resource discovery is a well researched area in the computer systems engineering. There are many successful discovery mechanisms deployed for different systems, like DNS for the Internet and LDAP [1] for administration of networked systems. The word “resource” is an overloaded term in resource discovery research as its meaning varies based on what the discovery mechanism is designed to find. For example, a resource is a machine in DNS, a document in a file sharing system, and a service in a web service [2] framework. Despite the existence of successful discovery mechanisms, new networked architectures and service platforms always force another look at resource discovery, because the efficiency of the discovery mechanism are controlled by the structure of the system.

One such networked architecture is *peer-to-peer* (P2P) systems that recently received much attention from the research community. At present, P2P systems are mostly used as file sharing applications and the resources in these P2P systems are files found in the peers. These file sharing platforms showed the potential of the P2P architectures to create large scale overlays economically. Also, SETI@home project [3] demonstrated the possibility of harnessing immense computing power in the P2P systems. Following these successes, the P2P systems are now pushed beyond their original implementation as file sharing platforms. P2P *content delivery networks* (CDNs) like Swarmcast [4] have demonstrated that P2P CDNs can deliver content without requiring high capacity content servers. Also Grid community is considering using P2P network in Grid computing [5]. These new directions in P2P technology motivates a design of a common P2P service platform that manages the underlying resources efficiently so that many different P2P applications can be developed on top it. The need for such *underlay* for an overlay is argued in [6]. The "resources" in this underlay are more general; for the rest of this document, the term "resource" denotes a machine defined by its physical and/or functional attributes. Physical attributes can be its CPU type, hard disk capacity or memory size; The functional attribute of a resource denotes a special functionality it provides to the system. The need for an underlay resource discovery mechanism for P2P service platforms prompts another look at the P2P discovery services.

I introduce a P2P resource discovery system called *resource addressable network* (RAN) as a discovery substrate for Internet-scale P2P service platforms. RAN can discover resources along two axes: based on the attributes of the resources and their locations in the Internet. The augmentation of *location-based discovery* to the *attribute-based discovery* is very useful in P2P environment as it can be used to discover a particular type of resource at any location in the Internet. For example, location-based discovery can be used to find capable servers for content delivery closer to the end user concentration; it can be used to co-locate computing servers to create a computing cluster that provides low inter-processor communication delay; or it can be used to strategically place game servers between two gamers. The primary concern in the design of RAN is scalability. It should have low overhead in managing the overlay, in managing the information about participating resources, and in query routing or discovery. RAN achieves this scalability using three design concepts: (a) fully decentralized architecture: the overlay is self-organizing and self-healing such that each resource in the system equally participates in building the overlay and maintaining it against resource failures; (b) adaptive design: the overlay structure and details of the maintained information in the overlay change with the change in the environment, for example, the change in the popularity of an attribute; (c) distributed knowledge: every node maintains only a part of the whole information in the system.

In the next section, I introduce the overall design and concepts of the resource addressable network. A concise description of the related works to my research is given in Section 3. Section 4, 5, and 6 present three major components of the RAN framework. And Section 7 concludes this report presenting the contributions of my work to the resource discovery research.

2 Resource Addressable Network

As shown in Figure 1 RAN operates as middle layer service between the overlay service platform and the constituting physical machines. RAN uses *profile-based discovery* to find a resource based on a given resource description and *location-based discovery* to find a resource at a given location in the Internet. Depending on whether a resource query specifies resource attributes, resource location, or both, RAN uses these discovery mechanisms selectively or combined.

There are many research threads dedicated to document discovery in P2P systems. Generally, they fall into two types: unstructured and structured. *Unstructured document discovery* like the one used by Gnutella [7] uses query flooding to discover a document in the overlay. It needs no special structure in the overlay: a node can have neighborhood relationship with any other node(s) in the system. While it eliminates the

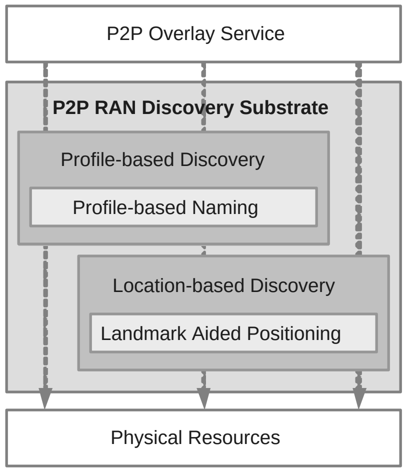

Figure 1: Structure of RAN substrate.

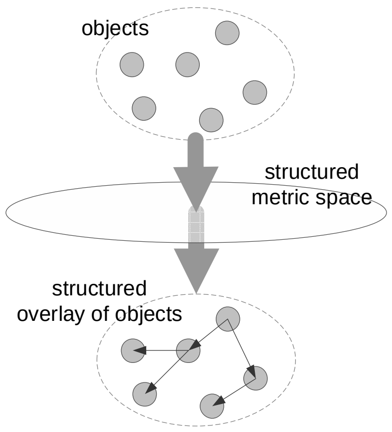

Figure 2: Creation of structured overlay for discovery architectures.

message overhead required to maintain a structured overlay, it suffers from heavy message overhead due to the flooding of discovery messages. On the other hand, *Structured document discovery* (e.g.: [8, 9]) prevents this message flooding by constructing a structured overlay of participating nodes. Generally, the structured overlay of the nodes is created as a *distributed hash table* (DHT). The DHT-based structure assigns a unique fixed width ID for each node. The *n*-ary representation of the ID can be used to assume a hierarchical structure among the IDs and this hierarchical structure can be used to construct a structured overlay of the nodes. The placement of documents in this scheme in not arbitrary as in the unstructured scheme. Documents are assigned with IDs in the same ID space as the participating nodes and they are placed in the nodes whose IDs have the closest match with that of the documents. This controlled placement of documents enables a fast document discovery with much reduced message overhead than the unstructured case.

A structured discovery system consists of two phases as shown in Figure 2. In the fist phase, the objects to be discovered are mapped on to a structured *metric space*. In the next phase, these objects are arranged in a routing overlay using the structure of the metric space. RAN introduces a mechanism that uses this concept of structured discovery. However, the mapping of resources on to a structured metric space in RAN is not as simple as mapping the documents in the DHT-based metric space. Whereas the document discovery mechanisms can construct a metric space with the required uniformity and structure, the metric space in RAN is pre-defined. For example, in location-based resource discovery, the metric space is the topology of the Internet which can not be redefined by the system. Also, the placement of resources in this metric space is not under the control of the system. RAN devises a method to fit a structure on to the unstructured metric space. RAN considers the Internet as a network delay space and uses the *landmark-aided positioning* (LAP) mechanism to assign coordinates to the points in the space. Then the points are processed through a mathematical data structure called *Hilbert curve* to assigned with *location IDs* (LIDs). The LIDs created by the Hilbert indexing is hierarchically structured so that they produce a structured metric space similar to a DHT-based overlay out of the unstructured network delay space.

A similar approach is used for attribute-based discovery in RAN. Here, the metric space is the *description space* consists of all possible attribute-value descriptions of the resources. Similar to the network delay space in location-based discovery, this description space is unstructured. But, when the network delay space is bounded (as the maximum network delay in the Internet is bounded), the description space is not. Therefore, the first step in translating the unstructured description space into a structured metric space is to compress the description space into a bounded space, which I call as the *profile space*. The *profile-based* *naming* scheme is used for this process and the points in the profile space are called *profiles* or *types*. These profiles are processed through the Hilbert indexing procedure to be assigned with *profile IDs* (PIDs). The PIDs of the resource construct the structured metric space for attribute-based discovery. The attribute-based discovery is called as *profile-based discovery* in RAN due to the usage of the profiles.

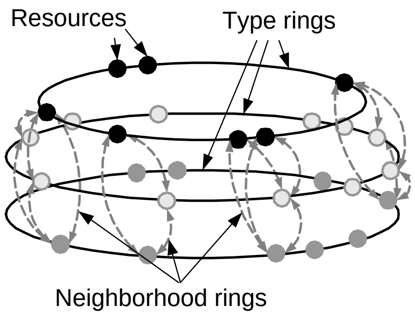

Figure 3: Structure of the RAN overlay.

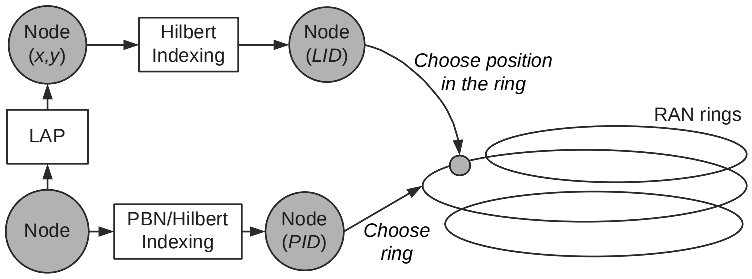

Figure 4: Creation of RAN overlay.

Figure 3 shows the structure of the RAN overlay and Figure 4 shows how it is constructed using the PIDs and LIDs. RAN overlay is composed of many interconnected *type rings* each dedicated to one profile. Positions of the nodes in the rings are determined by their LIDs. The connections between the resources which create this ring overlay structure are maintained by RAN routing tables in each node. The type rings are interconnected through every node in a ring via *neighborhood rings*. There will be at least four routing pointers at every node: two for constructing the type ring and another two for neighborhood ring. Resource discovery in RAN is handled in two phases using this overlay structure: (1) a resource discovery query is routed along a neighborhood ring to the appropriate type ring (one that matches the attribute specification); and (2) then, the query is routed along the chosen type ring to reach the desired location.

Potential Applications of RAN

RAN is designed to operate as the discovery substrate for P2P-based service platforms. Below I illustrate how different applications can benefit from RAN.

- P2P content delivery networks: Content delivery networks are interconnections of content servers in Internet. The content is pushed from a origin to a subset of content servers from which it is served to the end users. A CDN performs better than a traditional client-server setup, because it places the content servers at the edge of the Internet so that the delivery is not affected by the delay in the Internet backbone. It is called *edge delivery*. P2P CDNs differ from the conventional CDNs in that they can not have a centralized information repository about the resources constructing the overlay. Therefore they have to discover resources on-demand. These resources have to have enough capacity to host the content and preferably be at the localities of the end user concentrations to facilitate edge delivery. Capacity for a content delivery can vary depending on the type of the content (text or multimedia) and the *quality of service* (QoS) expected by the consumers. As RAN can discover resources based on both the resources' capacity and their locations, it is an ideal candidate to be the discovery substrate for P2P CDNs.

- *Public computing utilities* (PCUs): PCUs are computing utilities made up of public resources. Similar to the private, single-site computing utilities, PCU provides partitioned computing environments for the computing needs of individuals, but these partitions consist of public resources collected over large geographical span. RAN can help a PCU to discover appropriate resources from the P2P public resource pool for the end user partitions. It can allocate resources which are close to each other in the network to reduce the inter-resource communication in the computing platform, thus increasing the PCU's performance. Such a PCU design using RAN is presented in [10].

- Shortest path routing: The Internet frequently acts as a multimedia pathway; The usage of voice over IP is growing very fast. Generally multimedia traffic is very sensitive to delays. However, due to the policy based routing management at the autonomous system level, Internet packets can fail to take the shortest path between two end systems. Exploiting the RAN overlay which is constructed based on proximity, it is feasible to create an application level routing structure which can provide shortest path routing.

My Research

This report is composed of two primary parts, location-based discovery and profile-based discovery. The location-based discovery implements the network mapping module called landmark-aided positioning and constructs efficient routing mechanisms that uses the produced network map. Similarly, the profile-based discovery implements profile-based naming as the basic module. At this stage of my research, I have completed the design of the location-based discovery mechanism. For the landmark-aided positioning, my proposal includes new algorithms that are more robust and accurate and produce less message overhead than the algorithms used in the existing network positioning research. I support this claim with simulation results. These simulations are constructed using random data and the data collected from the Planetlab network. The design of the RAN location-based overlay construction is complete and the simulation is in progress. I have conducted an experiment that supports my selection of profile-based resource discovery over attribute-based discovery. A popularity based mapping of resource descriptions on to a profile space is proposed and further modification is considered. In addition to the simulation based studies, a prototype implementation on Planetlab network is in progress in parallel based on my design. This report mainly focuses on the implementation of the landmark-aided positioning and the related experiments. The rest of the document describes the rest of the RAN design and the road map for the further research.

3 Related Works

This section provides concise descriptions of the previous research works related to three main components of the RAN design. Section 3.1 provides related works for landmark-aided positioning, Section 3.2 on location-based discovery, and Section 3.3 on attribute-based discovery.

3.1 Network Positioning

Internet distance map service or IDMaps [11] is one of the early network delay prediction research that uses triangulation on the Internet. Triangulation is a method of estimating distance between two points A and B, by measuring distances from them to a third node C. Distance between two nodes in the Internet is measured using ICMP pings. The distance between A and B is upper bounded by the summation of distances between A and C and B and C, and lower bounded by the difference between them. IDMaps assumes that special nodes called *tracers* are installed well distributed in the network to act as the middle points. Despite its seminal work, IDMaps approach provides only the bounds for the delay estimation. There is no best way proposed for how the bounds can be used to predict network delays. Further, the delay estimation using a middle point can be inaccurate, especially in estimating short distances ([12]).

IDMaps established the possibility and the need for network delay prediction. When it is found to be inefficient, *network positioning* mechanisms are proposed as the alternatives. Similar to tracers in IDMaps, network positioning schemes rely on special nodes called *landmarks*. Tracers are considered to be "active" as they have to take distance measurements to other tracers. But, landmarks are mostly "passive", as they just respond to the ping messages from normal nodes. A node in a network positioning scheme uses a set of landmarks as reference points to assign itself a coordinate in a virtual Cartesian space. The Euclidean distance between two nodes in the space is used as the predicted network delay between them. Network positioning schemes can be categorized into *binning schemes* or *beaconing schemes* based on how a node uses landmark distances to position itself.

3.1.1 Binning Schemes

The binning scheme (sometimes called *triangulated scheme*) was proposed by S. M. Hotz [13]. Some works like [14] uses binning approach to construct a topology aware document discovery mechanism. The dimension of the coordinate space in the binning scheme is equal to the number of landmarks used. If the L landmarks are assumed to have an order $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_L\}$ and the ping value from a node i to landmark $\mathcal{L}_j$ is l_{ij}, then the coordinate of the node is simply set as $(l_{i1}, l_{i2}, \ldots, l_{iL})$. While this type of positioning can be directly used to derive the proximity information, the absolute distance between any two nodes has to be computed using triangulation as in the IDMaps scheme. Another sub-class of binning scheme is called *landmark ordering* mechanism [15]. In this scheme, the coordinate of a node is set using the indices of the landmarks in ascending order. The dimension of the Coordinate space is still L, but the range of the axis also is now L. This system can be used only to extract the proximity information, not network delays.

The binning scheme is the simplest network positioning scheme as it requires no sophisticated algorithms to find the coordinates of a node. However, there are a number of issues in this approach: Primarily, the Cartesian space of the binning scheme depends on the identities and even the presumed order of the landmarks. Therefore, when a landmark vanishes or a new one is added, coordinates of all the node changes and, therefore, the whole system has to be reconfigured. Therefore, it is not suitable for a P2P environment where even the landmarks can leave the system at any time. Further, the load on the landmarks is high in a binning system, because all the nodes use the same set of landmarks. While the position of the nodes in binning scheme can be directly used to predict the proximity, the possibility of "false positive," i.e., wrongly identifying nodes as neighbors, is high with fewer landmarks. Therefore, the binned-space distance becomes practically useless and, even to find the proximity, the triangulation has to be used.

Virtual landmarks [16] is binning-based network positioning solution that uses mathematical routines to overcome the issues in standard binning schemes. A node uses a set of landmarks to produce the binned coordinates. Generally, the number of landmarks and hence the dimension of the coordinate space are large to reduce errors. The system then uses the *principal component analysis* (PCA) to map the coordinates onto another Cartesian space of manageable low dimension. The new coordinates can be considered as the binned coordinates created using a set of "virtual landmarks." This scheme further extends this virtual landmarking concept so that different node can contact different set of landmarks. Even though these improvements over the original binning scheme are very interesting, the practicality of this scheme is not very clear. It requires a centralized procedure to find the mapping matrix needed for PCA and a large set of data must be collected to produce a good mapping matrix. Further, when nodes are allowed to use different landmark sets, there should be a common *spanner* set that translates the coordinate spaces produced by different landmark sets.

3.1.2 Beaconing Schemes

In contrast to binning schemes, beaconing schemes try to allocate coordinates for the nodes so that the distance in the target Cartesian space can directly relate to the network delay. In other words, it finds a mapping for the nodes on to the network delay space. If the coordinates of two nodes A and B are $\vec{x}_A$ and $\vec{x}_B$ in a beaconing scheme, the network delay between them is predicted as $d_{AB} = |\vec{x}_A - \vec{x}_B|$.

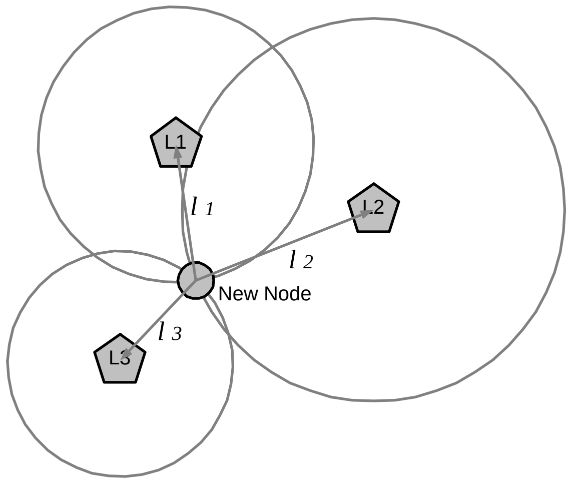

Figure 5: Network positioning in perfect conditions.

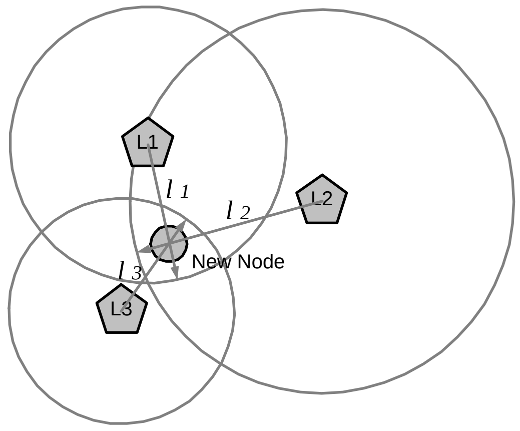

Figure 6: Network positioning in real conditions.

Positioning of a node using beaconing is similar to a positioning a GPS enabled device using the satellite beacons. Under perfect conditions, where the pings travel in straight line, the network positioning becomes a geometric problem as illustrated in Figure 5. In this case, network positioning can be solved using geometric equations for the intersection point of L circles, where L is the number of landmarks used. But, in practice, pings never travel in straight line in wired network. Therefore, the problem becomes similar to the one illustrated in Figure 6. In this case, the solution is found using optimization algorithms which minimizes the error in the distance prediction, i.e. $\{|d_i - l_i|\}, i \in [1, L]$, where d_i and l_i are the predicted and measured (ping) distances to the landmark i respectively.

In the beaconing schemes, the landmarks have to be positioned correctly at the beginning. The landmarks have no reference points to orient themselves, therefore each landmark acts as a landmark for others. The optimization process finds the landmark coordinates by minimizing the total $L \times L$ prediction errors.

Beaconing schemes have several advantages over the binning schemes despite the complex positioning algorithms needed: (a) landmarks can be added or removed in beaconing schemes without disturbing the basis of the coordinate space; (b) the dimension of the Cartesian space is not dependent on the number of landmarks; (c) as the coordinate space is global, each node can use different set of landmarks; (d) the calculated coordinates can be directly used to predict delay or proximity without any additional calculation required in triangulation. In the long run, it compensates the cost of the complex positioning algorithms.

Global network positioning (GNP) [17] is the pioneering work on beaconing-based network positioning systems. GNP uses Simplex downhill optimization to calculate the positions of the nodes. Positioning the landmarks is performed in a centralized manner where a representative node collects the ping data and calculates the coordinates of the landmarks. Node placement is distributed where each node runs Simplex to calculate its coordinate. GNP is shown to perform better than binning schemes and IDMaps in terms of distance prediction. This research work establishes many fundamental concepts of network positioning:

- Validity of network positioning: The triangular inequality property of a Cartesian space may not be applicable for the real network. Even though it is the fundamental source of inaccuracy in the network positioning scheme, the GNP research shows that only a little percentage of the node pairs does not obey triangular inequality and, hence, a network positioning scheme is generally applicable.

- Landmark selection: It concludes that choosing a well distributed set of landmarks is better than randomly selecting them.

- Number of landmarks: It shows that the accuracy increases with number of landmarks used, but improvement in accuracy diminishes with the increasing number of landmarks.

- Cardinality of the Cartesian space: A similar result as of the increasing number of landmarks is observed. A coordinate space with more than four dimension does not show much improvement.

The design of GNP has two issues: centralized implementation of landmark positioning algorithm and fixed set of landmarks for all the nodes.

Network positioning system (NPS) [18] extends the GNP design by allowing any node to be a landmark for another, in addition to the fixed set of landmarks. Node *B* is said to be *depends* on node *A*, if node *B* selects node *A* as one of its landmarks. NPS restricts the landmark selection such that the dependency relation creates a hierarchical structure among the nodes. This prevents nodes forming independent clusters. In NPS, the Simplex algorithm is decentralized among all the landmark nodes. NPS design also concentrates on the practical issues such as damping the noise in the ping values, periodic updates of the coordinates, maintaining the stability of the coordinates against the topology change, preventing landmarks get congested, and handling malicious reference points. While NPS design enables load distribution among all the nodes, the prediction error is amplified down the hierarchy.

Vivaldi [19] is inspired by GNP, but models the system differently. It models the system as a spring system where each node is connected to others by springs. The natural length of a spring connecting two nodes is taken as the measured ping value between them. Hence, minimizing the error in the predicted distances is the same as minimizing the deformation in the connecting springs. Any node can be a landmark to others in Vivaldi. Vivaldi is a completely decentralized mechanism and shows comparable performance with GNP. However, Vivaldi's convergence rate depends on the parameters of the algorithm and introducing a new set of nodes into an already converged system disturbs the whole system. Improved algorithms are presented in [20] to address these problems. However, these improvements requires every node capable of performing *remote procedure calls* on other nodes. Another modification introduced in [20] is the improved distance metric. It augment each point in the Cartesian space with a height vector. While the Cartesian distance represents the long-hop distance, the height vector represents the *access-link delays*. This provides a better fit in the Cartesian space.

Big-bang simulation (BBS) [12] is similar to Vivaldi in terms of the system modeling, but it models the nodes as molecules in a free space. These molecules are assumed to have mass, thus creating a potential force field. The movements of the molecules (nodes) under this force field are modeled using molecular dynamics. Since the molecules are assumed to have mass, the energy in the system consists of both potential and kinetic energy. BBS operation has four phases, each phase consists of many iterations. BBS algorithm is more complex than Vivaldi due to the complex calculation of energy function at each phase. Also, it has a n^2 communication complexity at each iteration creating high system overhead. Further, similar to Vivaldi, the whole system is disturbed when a new node is added.

Practical internet coordinates (PIC) [21] is another network positioning scheme inspired by GNP and conceptually very close to NPS. One of the PIC's major contributions is proposing and testing three landmark selection algorithms, one selecting random nodes as landmarks, the next selecting neighborhood nodes, and the other a hybrid selecting some landmarks randomly and some from neighborhood. PIC shows that the hybrid selection provides an overall performance as good as GNP and a better performance when considering distance prediction in the neighborhood. PIC provides an random walk based approach to find the neighbors. Further, it provides a mechanism to detect malicious nodes so that they can be avoided from being used as landmarks.

Research work	**Basic scheme**	**Landmarks**	**Coordinate space**	**Algorithm(s)**
IDMaps	triangulated distance	no landmarks, used *tracers*	N/A	N/A
Virtual landmarks	binning	fixed set of landmarks; need *spanners* to use multiple landmark set	global binned space; dimension is reduced using mathematical transforms	Principal component analysis
GNP	beaconing	fixed set of landmarks	single global space	centralized Simplex
NPS	beaconing	fixed set of landmarks + other nodes are arranged as hierarchical landmarks; anybody can be a landmark	single global space	distributed Simplex
Vivaldi	beaconing	anybody can be a landmark	global coordinate space	Spring
BBS	beaconing	everybody is a landmark to everybody	single global space	molecular dynamics
PIC	beaconing	anybody can be a landmark	single global space	Simplex; special landmark selection algorithms
Lighthouse	beaconing	anybody can be a landmark	one global space + multiple (no bound) local spaces	vector projection, transition vector
PCoord	beaconing	any node can be a landmark	single global space	special landmark selection and neighbor selection algorithms
LAP (the proposed work, Section 4)	beaconing	any *stable* node can be a landmark	single global space	SpringEq, Spring, CLAP

Table 1: Comparison of the network positioning research. Major feature of each scheme is highlighted.

Lighthouse [22] is another scheme that allows any node to be a landmark. This scheme does not use any optimization algorithm. When a node chooses a set of landmarks, the landmarks form the basis for a new local Cartesian space and the node is positioned in that local space. The positioning procedure is purely algebraic as the distance to the landmarks which form the basis of the space is known. However, if the distance between two nodes from two different local spaces is to be predicted, the coordinates of the nodes have to be translated into a global space. A *transition vector* is used to for the translation. This matrix is calculated by the initial nodes and conveyed to the subsequent nodes. Lighthouse scheme shows a comparative performance as GNP and its algebraic algorithm looks more attractive than the optimization based approaches. However there are number of points that are not clear: how the algebraic solution produces a solution under imperfect conditions that necessitates an optimization approach in other schemes (Figure 6) or how well the transition matrix can be transfered across the system (the results given in the paper are not for a system implementing multiple local coordinate spaces).

PCoord [23] implements different landmark selection algorithms. The governing argument in these algorithms is that the positioning mechanism performs better when the landmarks are well dispersed in the network. Its best landmark selection algorithm uses gossip protocol to get informed about other nodes so that it can properly select the landmarks. PCoord uses simulation to show that accuracy of the coordinate system is improved when the landmarks are well dispersed. PCoord algorithms achieve GNP-level accuracy either with high storage cost (for large amount of route pointers) or increased communication overhead.

A comprehensive comparison of the works in network positioning research is given in Table 1. The prominent feature that discriminates a work from the others is highlighted by shaded cells. The last line of the table presents the network positioning scheme I developed in this research called landmark-aided positioning (LAP). LAP is beaconing based network positioning scheme. Even though LAP identifies a specific set of nodes as landmarks like GNP, the membership in the landmark set is variable. This way, LAP is able to control the error propagation, at the same time eliminating the bottleneck problem at the landmarks. The Vivaldi's Spring algorithm is used for positioning normal nodes and a new algorithm called *SpringEq* is developed for positioning the landmarks. SpringEq provides low message complexity than the Simplex and Spring algorithm. LAP's design is further improved using node clustering technique called *clustered landmark-aided positioning* (CLAP). CLAP is a network positioning scheme susceptible to the non-random errors in ping values introduced by network anomalies. The details of LAP and CLAP is given in Section 4.

For the rest of this paper, the phrase network positioning scheme denotes a beaconing-based positioning scheme unless otherwise mentioned.

3.2 Location-based Resource Discovery

This section describes a few research works related to location-based resource discovery. Most of them are from document discovery research. They becomes important to my research for two reasons: (a) the location-based discovery in RAN uses the same concept of structured document discovery (Section 2); and (b) there are many location-based document discovery schemes available than location-based resource discovery schemes.

SkipNet [24] is a resource discovery mechanism using the *skip list* data structure to construct a locality aware DHT-based overlay network. The locality of the nodes are determined by their DNS names. Each node in the system have two IDs: a *name ID* which is basically its DNS name; and a *numeric ID* which is a unique binary number and can be the hash of the DNS name. Nodes are are arranged both in a ring ordered by their name IDs and in a hierarchical structure based on their numeric IDs. SkipNet uses the name ID space to implement *constrained load balancing* which can distribute contents among a number of servers in a particular locality. As the DNS names does not have a balanced hierarchical structure, routing with name ID does not provide a uniform complexity; it differs between different DNS domains.

Distributed hash table (DHT) is the data structure used in many structured discovery services as described in Section 2. While a centralized hashing provides a $O(1)$ search complexity, DHT-based architectures can provide $O(\log N)$ complexity provided that the keys have a hierarchical structure, where N is the number of objects in the system. The increase in search complexity is justified by the highly scalable management structure. DHT-based schemes suits well for P2P, Internet-scale architectures.

Pastry [9] is one of the pioneering DHT-based document discovery mechanisms. Pastry differs from other DHT-based schemes in that it can find the closest replica. Each node in Pastry maintains a neighborhood list. When a node joins Pastry, it uses the neighborhood lists from the existing nodes to build its routing table that consists of entries pointing to the possibly closest suitable nodes. However, constructing such a Pastry routing table with pointers showing proper proximity properties requires a large number of extra messages exchanged between nodes.

Content addressable network (CAN) [25] is another DHT-based document discovery mechanism. Each node in CAN is randomly assigned a random DHT-key and a point in an imaginary Cartesian space. Each node in this CAN space owns a partition and nodes closer in DHT-key space are located in the adjacent partitions in the CAN space. Routing in CAN is done in the CAN space via routing tables that point to the nodes in the adjacent partitions. Even though the original CAN design does not provide locality-aware routing, it is later extended to include a binning scheme so that the binned coordinate space becomes the CAN space [15]. But, the non-uniform binned space created hot-spots in the CAN routing. The next version of CAN, called *eCAN* [26] introduced another augmenting overlay called *expressway*. Expressway is a multi layered overlay created by aggregating multiple CAN zones at exponentially increasing expanse. eCAN uses a GNP-like approach to handle locality information. However, instead of constructing the eCAN overlay based on locality, the locality information are stored in the overlay in the form of document. Hence, if a closest replica is to be found, a node should first map its coordinates into eCAN space, find the node at that eCAN position (which need not be in the locality), and get the locality information from that node. The work given in [14] further refined the expressway so that now it is built based on proximity. However, the procedure for extracting the locality information is not changed.

Location-based discovery in RAN has novel characteristics than the existing schemes. In a P2P system, even after a pointer to a resource is found, the scheme can fail to find the matching resource due to the volatile memberships. Location-based resource discovery in RAN avoids this problem as it is designed to find a resource not the pointer to the resource. The RAN overlay is constructed based on proximity and therefore avoids triangular routing problem experienced in CAN. Further, unlike SkipNet which can extract locality information only in a intra-domain level, RAN can extract locality information at any resolution as it used the Internet delay space as the metric space (provided by the LAP). RAN's locality-based discovery structure is described in Section 5.

3.3 Attribute-based Resource Discovery

Attribute-based resource discovery mechanisms discover resources that matches the given attribute–value tuples specification. This section summarizes a few research works on attribute-based resource discovery.

The *lightweight directory access protocol* (LDAP) [1] is one of the early works in naming for resource discovery systems. It is a client–server based directory service where the name entries are distributed among hierarchically organized name servers. The name entries are made up of a set of attribute–value tuples that can be used to name entities ranging from people to computers. The read-optimized nature of the LDAP directories restricts its applicability in a highly dynamic wide-area system.

The *meta-directory services* (MDS-1) [27] and the successive *monitoring and discovery services* (MDS-2) [28] are the resource discovery services of Globus toolkit v.1.x and v.2 respectively [29]. MDS-1 extends LDAP so that multiple information sources can update a particular entry in the system. MDS-2 addresses some of the performance bottlenecks associated with MDS-1 that were inherited from LDAP. In MDS-

2, flexibility in discovery is achieved using *aggregate directory services* which can produce customized resource views/collections from the information. The idea of having aggregate services, that are designed to handle popular query types, supports the concept of popularity-based profiling used in my research. MDS-2 implements mechanisms to increase its responsiveness [30] and to keep the information up-to-date [28].

The *relational grid information services* (RGIS) [31, 32] is an implementation of a Grid information service using relational database technology, where resource advertisements are considered as record insertions and resource requests as database queries. The focus of this research is to develop efficient strategies to use relational database technologies towards achieving a high performance naming system.

Due to the success of the structured document discovery mechanism, many research proposed to use a structured document discovery for resource discovery with little modifications. The descriptions of the resources are processed into documents to represent the resources. These documents are placed in the resources the same way as in a structured document discovery scheme. If the document matching the desired description is found, it will point to the resource which actually has that description. Hence, the resource discovery is translated into a document discovery. The primary drawback in this approach is the indirection: it is the pointers that are discovered, but the presence of the pointers need not necessarily imply that the resources they point to are present as well. In addition, even though the system can find the closest pointer, the resource pointed to need not be in the close vicinity. Further, only exact matches can be found in this type of schemes, not closely matching resources (an AMD machine for a query for a Pentium). *Intentional naming system* (INS) [33] and the successor *twine* [34] are examples of such document discovery-based resource discovery mechanisms.

The major issue in attribute-based discovery mechanism is the huge attribute–value name space. Disseminating this name space and maintaining consistency produce heavy message overhead in the system, at the same time, requiring complex matching algorithms. The profile-based naming used in RAN addresses this problem by compressing the attribute–value name space. Only the the popular combinations attribute–value tuples are explicitly recognized by the RAN and others are annexed with the popular ones. Even though this approach sacrifices the flexibility of discovering “any” resource, this approach is justified by the reduced complexity and maintenance cost in the system. The details of profile-based discovery and its concept is given in Section 6.

4 Landmark-Aided Positioning

Landmark aided positioning (LAP) is a network positioning mechanism used in RAN that combines the benefits of GNP [17] and Vivaldi [19]. As part of LAP, I introduce a new algorithm for landmark positioning that provides better results than the algorithms used in GNP and Vivaldi. LAP is further extended as *clustered LAP* (CLAP) that is less susceptible to the non-random errors in the network measurement process.

In LAP, a set of participating nodes function as landmarks and act as reference points to others. Instead of having a fixed set of landmarks as in GNP, LAP allows any *stable* node to become a landmark and allows any unstable landmarks to leave the set. A node who is in a network segment less affected by network errors and stays alive longer compared to others is considered to be stable. I assume that the landmark set has proper security/trust mechanism to prevent a malicious node to become a landmark.

Operation of LAP consists of two phases: *landmark positioning* and *node positioning*. In the landmark positioning phase, landmarks run a positioning algorithm to fix their coordinates. Since there are no reference points for the landmarks themselves, each landmark acts as a landmark for others. The landmark positioning algorithm iteratively refines the location of each landmark until the total error in the predicted distances among all landmark pairs is minimized. At the optimal condition, the coordinates of the landmarks are fixed and form the basis of the Cartesian space for the network positioning.

In the node positioning phase, the non-landmark nodes are placed in the Cartesian space whose basis is fixed by the landmark set in the first phase. Each node chooses a subset of landmarks as reference points. At least $(d+1)$ landmarks should be picked by each node, where d is the dimension of the Cartesian space. LAP uses a 2-D Cartesian space ($d = 2$). The node positioning algorithm sets the position of a node to a coordinate that minimize the error in distance estimation.

A comparison of LAP with the existing network positioning schemes is given in Table 1. Below I enumerate the advantages of LAP over the existing schemes.

a) *Binning schemes*: The drawbacks of the binning schemes are described in Section 3.1.1. They are generally sensitive to landmark selection. It is an undesirable attribute for a P2P environment where all the nodes are autonomous and allowed to leave the system anytime. As LAP is a beaconing scheme it does not have these drawbacks.

b) *Hierarchical landmark organization*: This scheme allows any node be a landmark to another node (thus, avoiding the landmark bottleneck problem), while preventing the nodes from forming independent clusters and non-confirming coordinate spaces. However, it suffers from error amplification down the hierarchy (Section 3.1.2). As the LAP structure is always two-tiered, it alleviates the error propagation and independent cluster creation. Further, LAP's expandable landmark set provides a *self-scalable* framework. When the number of nodes in the P2P environment increases, there is a high possibility for more nodes joining the landmark set and thus distributing the loads in landmarks.

c) *Any node as landmark*: Anticipating continuously changing network conditions, each node has to undergo the node positioning phase periodically. To reduce the message overhead due to these periodic processes, each node remembers its landmark set, so that it need not go through a landmark search procedure every time it requires to run the node positioning algorithm. If a node is allowed to choose any node as one of its landmarks, in a P2P environment, there is a possibility for the landmark leaving the system between two node positioning phases. As LAP allows only the long-living nodes to become landmarks, the landmark set each node remembers rarely changes over the time reducing the need for searching new landmarks for each node positioning phase.

d) *Fully dynamic schemes such as Big-Bang simulation*: In addition to the issues regarding this scheme mentioned in Section 3.1.2, fully dynamic schemes are much costlier in terms of message overhead. LAP implements a semi-dynamic scheme where the landmarks set is fixed at an instance, but allowed to expand when the need arises.

LAP does not directly handle some of the issues the other network positioning schemes address. However, they are handled by other components of the RAN:

a) *Location of the landmarks*: Number of network positioning schemes like PCoord [23] implement specific landmark selection mechanisms to choose a well distributed set of landmarks. But, as LAP is a part of the location-based discovery mechanism, it need not implement any such selection mechanism. It can simply use the location-based discovery service to choose landmarks wherever it wants.

b) *Landmark discovery*: In LAP, each node is given a list of landmark at the initialization phase. If a node looses one of its landmarks, it can use the profile-based discovery (Section 6) to find a new landmark.

c) *Dimension of the Cartesian space*: Previous research have established that the accuracy of the system improves with increasing cardinality of the Cartesian space. LAP agrees with this, but assumes a 2-D space for the present design for the simplicity of the algorithms used in location-based discovery. Extending the design with higher dimension is left as future work.

In the next section, the performance metrics used in the evaluation of the LAP schemes are described. Sections 4.2 and 4.3 provide an analysis of the algorithms used in node and landmarking positioning phases and present the comparison results. Also a new algorithm for landmark positioning is proposed in Section 4.3. An improved LAP design is presented in Section 4.4 that is designed to damn the impact of non-random errors in distance measurements.

4.1 Performance Metrics

The purpose of a network positioning scheme is to assign a coordinate $\vec{x} \in \Re^d$ to a node in a d-dimensional space. The performance of such a scheme depends on how well the coordinates of the nodes represents the network delay space. The performance evaluations of the network positioning algorithms in this report use a number of performance metrics, some used by other previous research, some introduced as new.

Cumulative distribution of relative distance error

The metric *relative distance error* is defined in the report as:

$$\text{relative distance error} = \frac{|\text{ping distance} - \text{predicted distance}|}{\text{ping distance}}$$

where the *ping distance* is the measured network delay between two nodes using ping and the *predicated distance* is the euclidean distance between the nodes in the target Cartesian space. The cumulative distribution of this metric is used by almost all network positioning schemes to analyze the accuracy of the algorithms used. Generally this distribution has a heavy tail. However, the plot is usually truncated at a position just past the knee of the curve so that the region around the knee is emphasized.

Distance correlation

Comparing the network positioning schemes using the relative distance error metric has two issues: (a) relative distance error distribution has to be plotted for each experiment because producing a cumulative distribution plot averaged over many trials is difficult; (b) as mentioned earlier, the heavy tail is generally truncated hiding the maximum error. These issues make the relative distance error metric not a suitable candidate to be used as a system-wide performance evaluation metric. 90-percentile relative distance error could be used as a system measure, but it still hides the system-wide error. I use a new metric called *distance correlation* to compare the system-wide performances. Distance correlation is the correlation between the $N \times N$ ping distance and the predicted distance matrices. Correlation values are always between 0 and 1, where 0 means absolutely no correlation and 1 means a perfect match.

Cumulative distribution of rank error

As described in Section 3.1.2, it is impossible to find perfect coordinates that predict the network delay without any error. It implies that distance error metrics are not an absolute measure to evaluate the performance of the network positioning schemes. Generally, network positioning schemes are used to discover resources based on proximity. Therefore, the metric *rank error* is used in addition to the relative distance error to evaluate the performance of the LAP, as proposed in GNP [17]. The rank error is the measure of the effectiveness of the network positioning scheme in discovering nodes based on their proximity. It is measured as follows: for each node, all other nodes are processed into two lists. The nodes are ordered in one list by the ping distances to them and in the other list by the predicted distances. For each destination node, an ordering error is calculated as the absolute difference of its positions (indexes) in these two lists. The rank error at a node is the average of the ordering errors of all the other nodes. In other words, if the ping value-based and predicted value-based orderings of the nodes at the node i are P_i and Q_i respectively, the rank error at node i is defined as:

$$\text{rank error}_i = \frac{1}{N}\sum_{j=1}^{N}\frac{|j - \text{indexOf}(Q_i, P_i[j])|}{N}$$

where $\text{indexOf}(X, y) = k$, if $X[k] = y$; rank error is always less than 1. If the ordering all the nodes using predicted distances matches with the ordering of nodes using ping values, the rank error will be zero.

Occupancy diameter

The purpose of the network positioning mechanism is to discover/bind resources based on the location of the resources. However, with continuously varying ping values, the position of a node $\vec{x}_i$ can never be a constant. If the oscillations in the positions of the nodes are high, it can disturb the existing location-based bindings, increasing the overhead in the service overlay that uses the resource discovery substrate. I define a new performance metric called *occupancy diameter* to measure the performance of a positioning algorithm in terms of this oscillation. The occupancy diameter of a node is measured over a past history as follows: when a node moves from coordinate $\vec{x}_{t-T}$ at time $t-T$ to $\vec{x}_t$ at time t through the coordinates $\vec{x}_{t-T+i}, i \in [1, (T-1)]$, the occupancy diameter of the node at time t over the window T is defined as the diameter of the minimum area circle that covers all the points $\vec{x}_i, i \in [(t-T), t]$. Mathematically:

$$\text{occupancy diameter}_{t,T} = \max_{i,j}(|\vec{x}_i - \vec{x}_j|), \qquad i, j \in [(t-T), t]$$

The smaller the occupancy diameter, the lower the fluctuation in the location and the higher the performance of location-based bindings.

4.2 Node Positioning

Node positioning phase of LAP finds the positions of the nodes in the network delay-based Cartesian space relative to a set of landmarks. Calculating the position of a node is modeled as an optimization problem. If the number of landmarks the node uses is L, the ping value to the landmark i is l_i, and the coordinates of the landmarks are $\{\vec{x}_i\}$, the coordinates of the node $\vec{x}$ is found by solving the optimization problem:

$$\min \sum_{i=1}^{L}(|\vec{x} - \vec{x}_i| - l_i)^2 \qquad \vec{x}, \vec{x}_i \in \Re^d \tag{1}$$

The value $|\vec{x} - \vec{x}_i|$ is the predicted distance between the node and the landmark i. The optimization function minimizes the total square error in distance estimation between the node and the landmarks. From the Figure 6, it can be seen that L has to be greater than or equal to $(d+1)$ to produce a unique solution, where d is the dimension of the Cartesian space.

There are two approaches in the previous research to solve the above optimization problem. The first one is *Simplex downhill method*, one common algorithm used for solving optimization problems. A short version of Simplex pseudo code is given in Algorithm 1.

The second algorithm, which I call as *Spring* algorithm, is proposed by Vivaldi [19]. This algorithm models the system as a spring network, where the nodes are considered weightless and connected to each other with springs. The *natural lengths* of the springs are assumed to be the ping values between the connected nodes. Natural length of a spring is the length of the spring under no external force applied. Placement of nodes at particular positions can cause deformation in the strings which can be written as $||\vec{x}_i - \vec{x}_j| - l_{ij}|$. Hence, minimizing the deformations of the springs in the system achieves the same effect

Algorithm 1 Simplex downhill algorithm.

```
create simplex
repeat
    for all simplex points do
        evaluate objective function
        find/refine downhill direction
    end for
    move down hill
    if could not move then
        shrink simplex size
    end if
until converged
```

as minimizing the objective function Equation 1. The Vivaldi's Spring algorithm adjusts the coordinate of a node iteratively so that the total deformation in the connecting springs is minimized. The compact pseudo code of the algorithm is given in Algorithm 2.

Algorithm 2 Spring algorithm.

```
$\mathrm{limit}, \mathrm{dec}$
$\alpha \leftarrow 1$
repeat
    for all landmarks do
        $\vec{u} \leftarrow$ unit direction towards landmark
        $\delta \leftarrow$ error in distance estimation
        $\vec{x} \leftarrow \vec{x} + \alpha\delta\vec{u}$
    end for
    $\alpha \leftarrow \max((\alpha - \mathrm{dec}), \mathrm{limit})$
until converged
```

The Simplex downhill algorithm uses a geometric construct called simplex which is a polygon of $d+1$ vertexes. Therefore, as the step 4 in Algorithm 1 has a complexity of $O(Ld)$, the complexity of the steps 3–6 becomes $O(Ld^2)$ and the complexity of the simplex algorithm becomes $O(sLd^2)$, where s is the number of iterations taken in the `repeat...until` loop. From Algorithm 2, the complexity of the Spring algorithm can be found to be $O(sLd)$, as each step between 4 and 8 has complexity of $O(d)$. Vivaldi chose Spring over Simplex due to this reduced complexity. However, it is impossible decide the absolute performances of the algorithms, as the value of s is an unrelated parameter between the algorithms. The value of s depends on both the parameters of the algorithms and the data set. For a fixed data set, it can be tuned by changing the parameters of the algorithms and/or the convergence criteria. But for a varying data set, there is no method to select the optimal algorithm parameters. Therefore, I evaluated the performances of these algorithms through experimental studies.

The experimental setup I used is as follows: The coordinates of a node $\vec{x}$ and the coordinates for L landmarks $\{\vec{x}_i\}$ are randomly selected. The ping value between the node and the i^{th} landmark is set as $l_i + l_i p(1-2r)$, where $l_i = |\vec{x} - \vec{x}_i|$ and $p, r \in [0,1]$. The parameter p is called *ping error* simulating the mismatch between the hypothetical straight line distance and the real ping value. The parameter r simulates the constant variation in the ping values in real network which I call as *ping noise*. The value of p is fixed to a value for an experiment while the value of r is let to randomly vary for every ping calculation. Once the system is fixed as above, the Simplex or Spring algorithms are run for different initial conditions, i.e. the presumed position of the node $\vec{x}'$. The algorithms calculates the position of the node as $\vec{x}''$. The performance of the algorithm is evaluated by averaging the values of the metrics over many experiments.

Figures 7 and 8 show the performances of the algorithms with the varying number of landmarks (L) with fixed ping error ($p = 0.3$). Figures 9 and 10 show the performances with the varying ping error with $L = 5$. Figures 7 and 9 shows the performance in terms of the number of iteration s. The larger the number of iterations, higher the calculation complexity and the computation time. The number of iteration determines how fast a node can find its position in the system, thus how fast it can become operational. Therefore the algorithm providing low number of iteration is preferred. The distance error metric used in Figures 8 and 10 is the relative error defined Section 4.1 averaged over the experiments. It reflects the final value of the objective function Equation 1 and shows how successful an algorithm is in predicting distances. All the results presented here shows that the Spring outperforms the Simplex in all the cases.

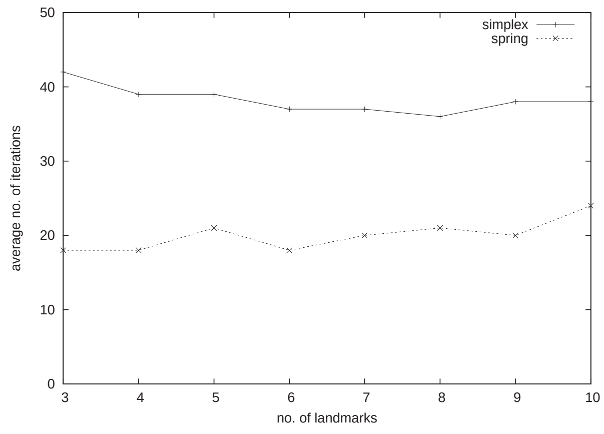

Figure 7: Average number of iterations taken by the algorithms.

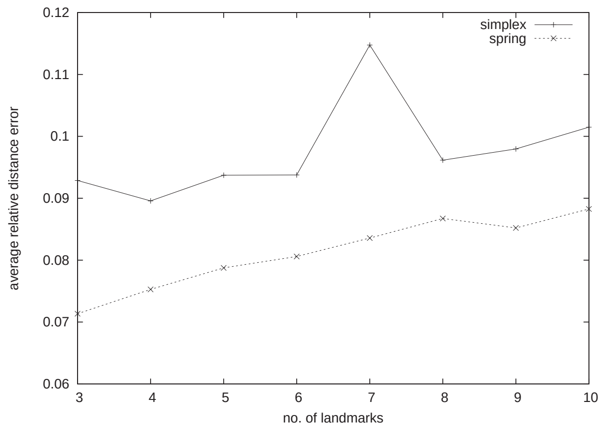

Figure 8: Average error in distance estimation in the algorithms.

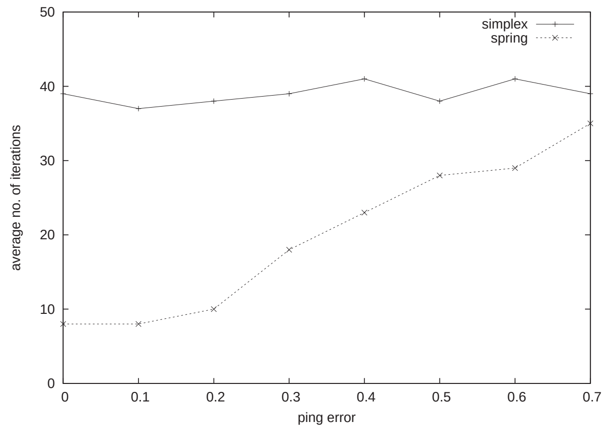

Figure 9: Average number of iterations taken by the algorithms.

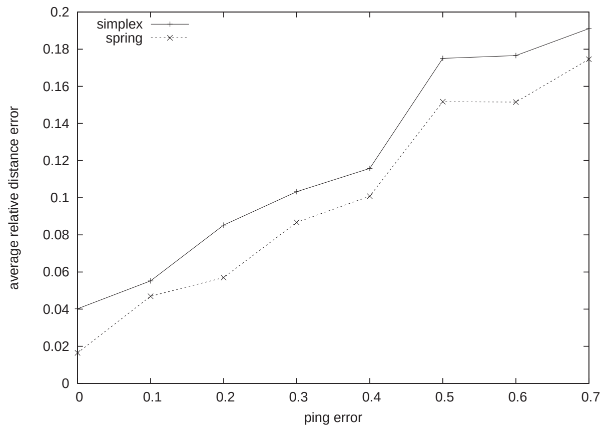

Figure 10: Average error in distance estimation in the algorithms.

4.3 Landmark Positioning

In the landmark positioning phase of LAP, the positions of the landmarks are determined to form the basis of the Cartesian space of the system. The positions of the landmarks are randomly chosen at the beginning of the system operation and the landmark positioning algorithm iteratively refines them so that the total error in distance prediction between all landmark pairs is minimized. For the system of N_l number of landmarks, the algorithm solves the following optimization problem:

$$\min \sum_{i=1}^{N_l} \sum_{j=1}^{N_l} (|\vec{x}_i - \vec{x}_j| - l_{ij})^2 \qquad \vec{x}_i, \vec{x}_j \in \Re^d \tag{2}$$

where l_{ij} is the ping value between landmarks i and j. By comparing the above objective function with the objective function Equation 1 for node positioning phase (Section 4.2), it can be observed that a landmark positioning problem is composed of many node positioning problems.

Based on this observation, two algorithms can be devised for the landmark positioning phase based on the Simplex and Spring algorithms. In these algorithms, while the *outer loop* iterates through all the nodes in the system, the *inner loop* solves node positioning problems for each node using Simplex or Spring algorithm. For each node positioning step, each landmark considers all others as its landmarks. Algorithm 3 shows the pseudo code of landmark positioning.

Algorithm 3 Pseudo code for landmark positioning.

```
1:  repeat
2:    /* outer loop */
3:    for all node $i$ do
4:      /* this for loop is centralized in simulation, but distributed in real scenario */
5:      $x_j \leftarrow$ coordinates of the other landmarks
6:      $l_{ij} \leftarrow$ ping values to the other landmarks
7:      repeat
8:        /* inner loop */
9:        /* running node positioning algorithm to fix $\vec{x}_i$ that minimizes $\sum_j (|\vec{x}_i - \vec{x}_j| - l_{ij})$ */
10:     until node position $\vec{x}_i$ fixed
11:   end for
12: until no node moved
```

Node positioning in LAP is a decentralized process where each node finds their coordinates themselves. Using such algorithm as the basic building block, landmark positioning also becomes decentralized. The number of iteration in the inner loop affects the computational resources in the nodes and decides how fast the system converges. On the other hand, as each node has to contact other nodes to discover their new refined coordinates for every outer loop iteration, the number of iterations in the outer loop determines the message overhead introduced by the landmark positioning phase. If the number of iterations in the outer loop is t, the message overhead will be of $O(tN_l^2)$.

This section evaluates the landmark positioning algorithms for their performance in the outer loop. The values presented here are average values over many runs over many different random network configurations. Random networks are created as explained in Section 4.2. Even though the landmark positioning is decentralized in practice, we ran centralized versions of the algorithms to find the minimum number iterations required in the outer loop. The actual number of iterations in practice can be higher due to lack of synchronization between the inner loops of different landmarks (a landmark might receive repeated queries for its refined coordinates before it finishes its inner loop). In the experiments, for a single network configuration, the nodes begin operation with a random initial coordinates and run the algorithms repeatedly for a certain number of *epochs*. This repetition simulates the periodic rerun of landmark positioning phase in practice.

The Figures 11 – 14 shows the performance of three landmark positioning algorithms: Simplex, Spring, and SpringEq (SpringEq is discussed in Section 4.3.1). The Figures show values of the performance metrics plotted for the first, second, and third epochs. The positions of a landmarks do not change between epochs unless there is a noticeable change in the network conditions. As each calculation phase takes the coordinate values from the previous phase as the initial condition, the number of iterations required to converge to the optimal point reduces for the successive epochs. This characteristic can be seen in the Figures 11 and 13. Figures 11 and 12 evaluates the performance of the algorithms against varying size of the landmark set whereas the Figures 13 and 14 compares the performance with varying p values. Even though Figures 11 and 13 show that Spring outperforms Simplex in terms of number of iterations required in the outer loop, from the Figures 12 and 14, it can be seen that Spring fails to outperforms Simplex in terms of accuracy.

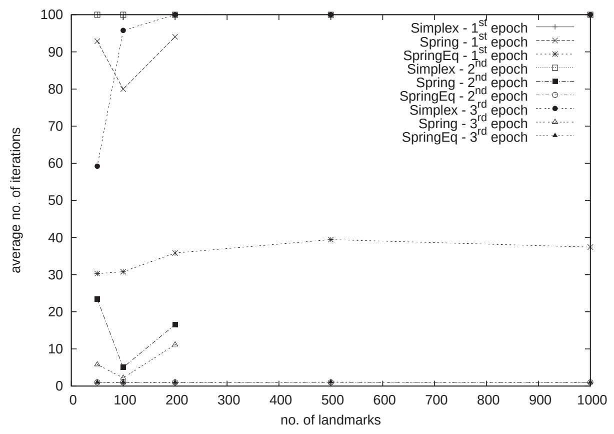

Figure 11: Variation of no. of iterations with no. of landmarks.

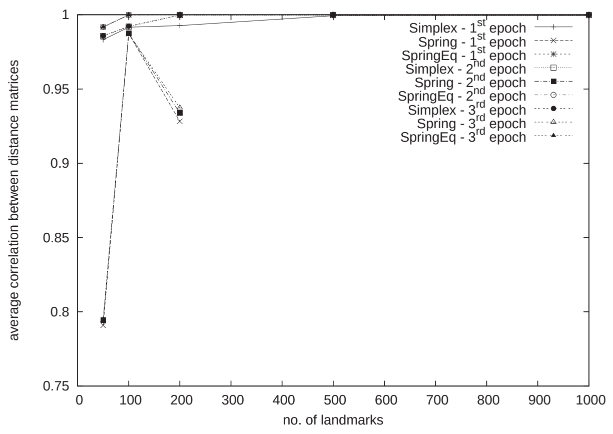

Figure 12: Variation of dist. correlation with no. of landmarks.

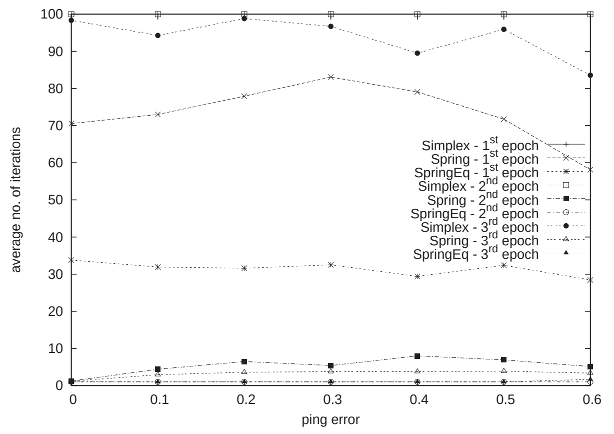

Figure 13: Average number of iterations taken by the algorithms.

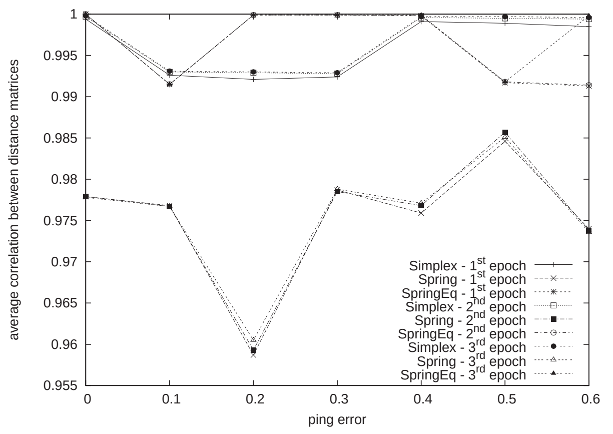

Figure 14: Average correlation between distance matrices.

4.3.1 SpringEq Algorithm

This section introduces a new algorithm called *SpringEq* that addresses the defects observed in Simplex and Spring algorithms when used for landmark positioning. The SpringEq (short for "Spring system in equilibrium") algorithm is inspired by the Vivaldi's spring algorithm and the work [35]. Similar in the Spring algorithm, SpringEq models the network of landmarks as a spring system. However, instead considering the deformation in the springs, SpringEq considers the force exerted on the nodes by the deformations. The optimization of the Equation 2 implies that, at convergence, the energy in the spring system is minimized or, in another words, the spring system comes to an equilibrium. One condition for the equilibrium is that the resultant force applied at each node is zero. This equilibrium condition is the basis of the SpringEq algorithm.

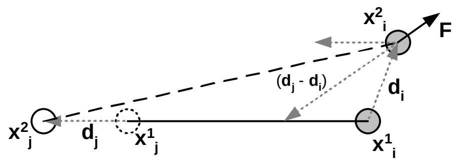

Figure 15: A spring under deformation.

The SpringEq algorithm is formulated as follows: the force $\vec{F}$ exerted by a spring under a linear deformation of $\vec{\delta}$ is given by the Hooke's formula:

$$\vec{F} = k\vec{\delta} \tag{3}$$

where k is the *spring constant*. Consider a spring segment connecting node i and j as shown in Figure 15. Assume that the spring is at rest at the initial condition $(\vec{x}_i^1, \vec{x}_j^1)$ with no force applied. Therefore, the natural length l_{ij_n} of the spring can be written as $|\vec{x}_i^1 - \vec{x}_j^1|$. If the end nodes move by distances $(\vec{\delta}_i, \vec{\delta}_j)$, the force $\vec{F}$ exerted on the spring by the movements is:

$$\vec{F} = k_{ij}(\vec{\delta}_j - \vec{\delta}_i) \tag{4}$$

It is the force applied on the end nodes by Newton's third law. Therefore, the Equation 4 can be used to write the equilibrium condition of the landmark system as:

$$\sum_{j=1}^{N_l-1} \left[k_{ij}(\vec{\delta}_j - \vec{\delta}_i) \right] = 0 \qquad \forall i \tag{5}$$

$\vec{\delta}_j$ can be calculated as $\vec{\delta}_j = (l_{ij} - l_{ij_n})\vec{u}_{ji}$, where $l_{ij} = |\vec{x}_j^2 - \vec{x}_i^1|$ and $\vec{u}_{ji} = (\vec{x}_j^2 - \vec{x}_i^1)/l_{ij}$. With the initial condition set $\{\vec{x}_i^1\}$, solving the linear simultaneous equations given by Equation 5 will give the final positions of the nodes. k_{ij} is taken as unity in LAP.

SpringEq algorithm solves the landmark positioning problem by solving this simultaneous equation using an iterative method. An iterative method is chosen because it can be converted into a distributed solution, where each node i solves the i^{th} iteration. There are many iterative solutions available for solving linear simultaneous equations. SpringEq uses the *successive over relaxation* (SOR) (or *accelerated Gauss-Seidel*) method for its speed and simplicity. Note that, the SpringEq algorithm is not an optimization based algorithm but converging to the same state as other optimization algorithms used for landmark positioning.

The comparison of the performance of the SpringEq algorithm with Simplex and Spring is given in the Figures 11 – 14. The figures shows that the SpringEq outperforms the other algorithms both in terms of number of iterations and distance prediction accuracy.

4.4 Clustered Landmark Aided Positioning

The accuracy of the network positioning schemes depends on the accuracy of the ping values. But, ping values in the Internet are always subjected to variations due to number of factors. Section 4.2 introduced two such factors: ping error (p) and ping noise (r). Ping error is a constant deviation of the measured ping value from the hypothetical straight line distance. As it can not be avoided in a real network, optimization algorithms are used to find the best positions for the nodes despite this error. On the other hand, ping noise is the random variation in the ping values due to the varying path conditions such as varying queuing delay in the routers. This noise can be filtered out by data smoothing.

Another major factor affecting the ping values is the *measurement errors*. Measurement errors are caused by abnormal network conditions. For example, abnormal congestion in network segments can increase the ping measurement; or a malicious node on the path of a ping can return the ping before it reaches the real target and thus showing a low value in the measurement. As these errors are neither constant nor random, their effect can not be removed by neither data smoothing nor optimization algorithms. This section introduces an improved LAP mechanism called *clustered landmark aided positioning* (CLAP) that is capable of producing coordinates with relatively high accuracy despite the measurement errors.

CLAP is an improvement over the simple LAP (SLAP) scheme. In CLAP, the nodes in the system (both landmarks and others) are partitioned into *clusters*. A cluster is a collection of nodes that are located close to each other and have a trusted relationship among them. For example, the nodes within an autonomous system (AS) can trust each other to form a cluster. Each node is assumed to know the other members in the cluster. CLAP mechanism introduces two additional steps in the life cycle of a node: (a) finding the cluster diameter in terms of network delay and (b) CLAP coordinate adjustment. Finding the cluster diameter is an infrequent event, in which the nodes in a cluster ping each other and share the ping information. The maximum ping value between any pair in a cluster is taken as the diameter of the cluster. The CLAP coordinate adjustment works as follows: once the coordinates of the nodes have been found, the nodes exchange their coordinates with others in the same cluster. Each node can find the coordinates of the centroid of the cluster using the shared information. Each node calculates how far away it locates from the centroid and compares its to the previously calculated cluster diameter. If the node found itself in a position that is outside the cluster diameter, it adjusts its coordinates towards the centroid so that it moves within the accepted range. This CLAP adjustment generates additional $O(N_c^2)$ message complexity, where N_c is the average number of nodes in the clusters.

The concept behind CLAP is based on the following assumptions: (a) measurement errors are caused by congestion in the border routers. a cluster is considered erroneous if its border router is congested; (b) an *erroneous cluster* introduces measurement errors only to the traffic crossing them. The traffic constrained within the cluster does not suffer from the congestion; and (c) a node can trust a neighbor better than a remote landmark. As the neighbors are mostly likely to fall under the same administrative domain (LAN, AS domain), it is a practically valid assumption.

This section presents a simulation-based performance comparison of the SLAP and CLAP schemes for varying network sizes and measurement error conditions. I considered a centralized implementation of the schemes for this simulation, i.e. all the nodes resides in a single computer, and the operations of the nodes are serialized. The landmark positioning algorithm is once run at the beginning of a trial to fix the position of the landmarks. For the same network and landmark configurations, each node runs both SLAP and CLAP algorithms repeatedly for a fixed number of *epochs*. The results presented in this section is from the last epoch.

A random network is used to evaluate the schemes for varying network size, while a synthetic network produced from Planetlab ping data is used to evaluate them against the varying measurement error conditions. Random networks are created as explained in Section 4.2. Details of the creation of synthetic network based on Planetlab data is as follows: The synthetic network is created using the data from the all-pair Planetlab pings project [36]. The all-pair pings project runs an automated script in every Planetlab [37] node that periodically pings all other Planetlab nodes and dumps the values in a file. The files from all the nodes are collected at a single point to produce the system-wide ping trace for an epoch. Such trace files for a day was used for my experiment. There were 95 sets of trace files for that day. The trace files were processed to extract only the data comprising *active nodes* (that responded to the pings from other nodes). The process resulted in a $95 \times N \times N$ ping matrix, where $N = 133$ is the number of active nodes. Then the ping values were averaged over the 95 epochs to produce another $N \times N$ matrix which is used to create the synthetic network. During the experiment, a ping between nodes i and j is simulated using the $[i, j]$ element from this matrix as l_{ij} and the ping error parameters p and r as described in Section 4.2. If the path between node i and j is found to be introducing a measurement error, l_{ij} is increased by a factor determined by the *measurement error fraction* $(0 < p_e < 1)$. Whether the path i–j introduces a measurement error or not is determined as follows: each node in the Planetlab has a real IP and hence belongs to a AS domain. I found that the 133 active nodes belongs to a total of 59 AS domains. The data from the *Route Views* project [38] is used to extract the connectivity map of the involved AS domains. A randomly selected number of e_cC clusters are set as erroneous, where e_c is a simulation parameter denoting the fraction of erroneous clusters, and $C = 59$ is the number of total AS domains present in the experiment. If the path i–j, that is determined by the connectivity map, includes any of the erroneous cluster, then the path is considered causing a measurement error and the amount of error in the path is determined by the number of erroneous clusters the path crosses. If the number of erroneous clusters crossed by the path is b, the value l_{ij} is increased by bp_e% (I consider only the additive measurement error at present). Note that if the nodes i and j are in the same cluster, even if the cluster is erroneous, $b = 0$ and, therefore, the path i–j is considered to be free of measurement error. In random networks, the path i–j introduces a measurement error, if either of the nodes i and j is in an erroneous cluster. As the connectivity map is not determined for random network, b is taken as 1 always. p_e is set to be 0.5 in my experiments.

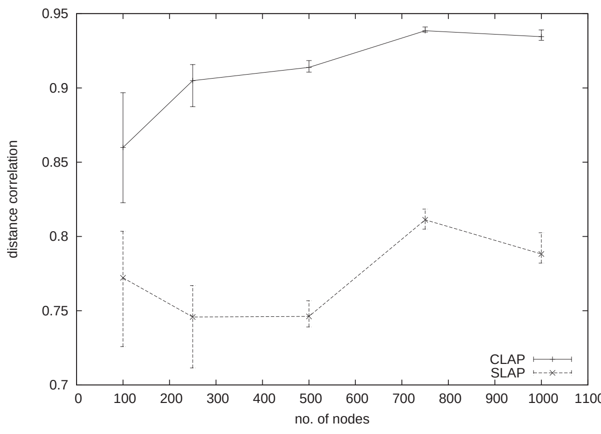

Figure 16: Variation of distance correlation with the size of the network.

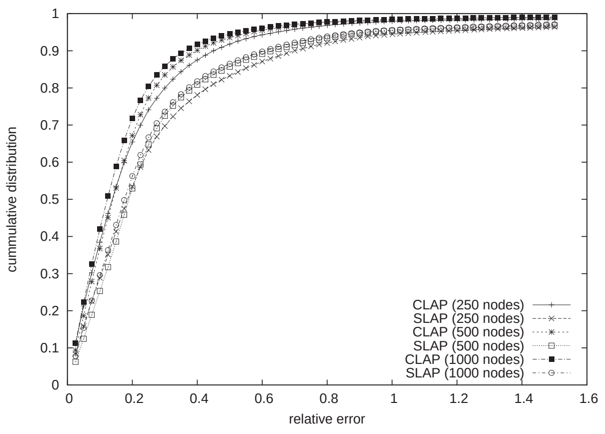

Figure 17: Cumulative distribution of relative error at various sizes of network.

Figures 16 – 19 compare the performance of CLAP with SLAP with varying network size (number of nodes). CLAP performs better than SLAP in all the metrics considered. The CLAP performance increases with the network size, because, with the increasing network size, the number of landmarks in the system also increasing. As each node in a cluster uses different set of landmarks for node positioning, when their coordinates are combined in the CLAP coordinate adjustment procedure, the average measurement error is reduced. Also, as the network size increases, the average number of nodes in the clusters increase, providing better result.

Figures 20 – 23 show the performance comparison for varying amount of error in the network (e_c). Again, the figures show that CLAP provides a better performance than SLAP. In Figure 20, two observations can be made: (1) CLAP definitely performs better than SLAP, because, in addition to the higher average distance correlation, the minimum correlation values of the CLAP are higher than the maximum values of the SLAP most of the time; (2) the CLAP is a more robust scheme than SLAP, because, while the accuracy of

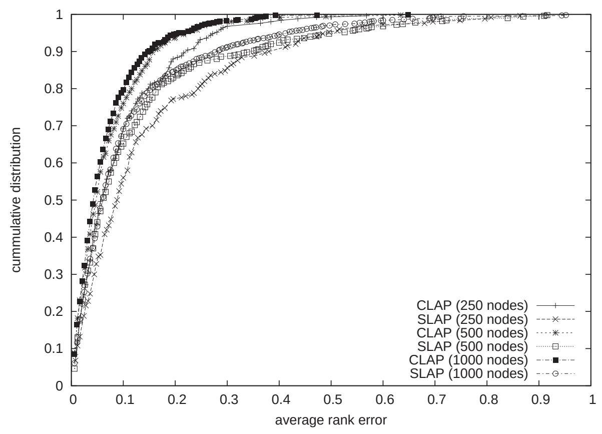

Figure 18: Cumulative distribution of rank error at various sizes of network.

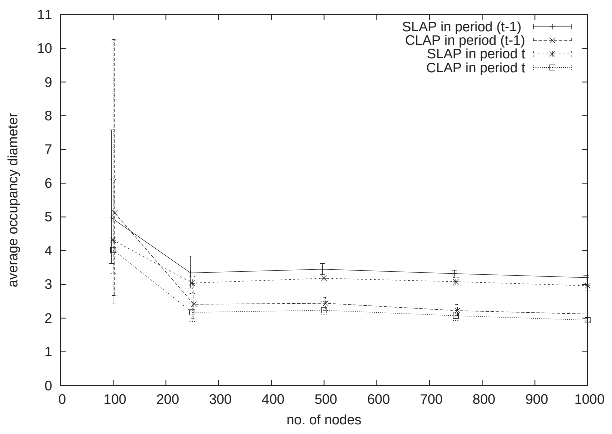

Figure 19: Variation of maximum node movement in the system with network size.

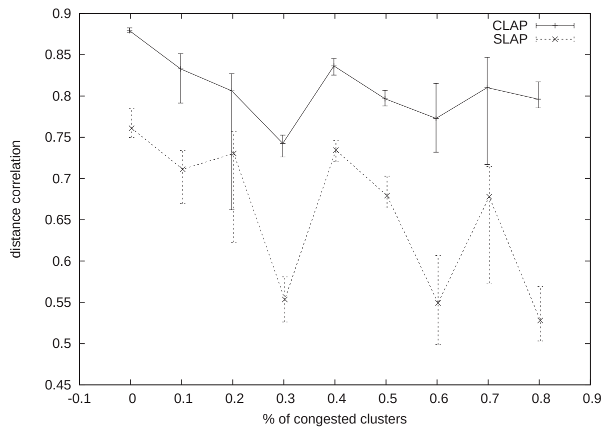

Figure 20: Variation of distance correlation with amount of network error.

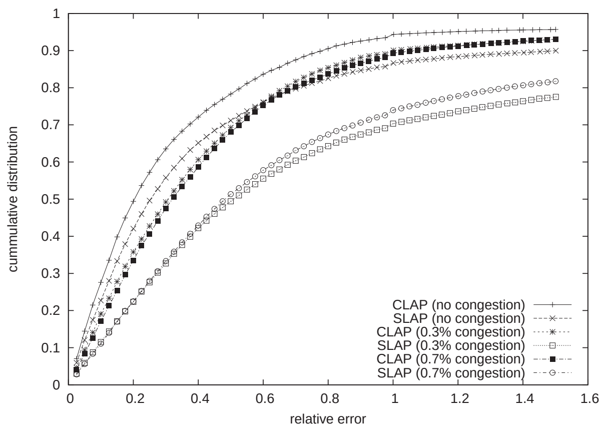

Figure 21: Cumulative distribution of relative error at various amount of network error.

SLAP continuously decreasing with increasing network error, CLAP shows a relatively steady performance.

In the LAP design, landmarks are selected from less congested network segments. According to the design, It is the nodes who recognize themselves as feasible candidates to function as landmarks and join the landmark set. Therefore, there should be a mechanism to detect the congestion in the network segment. I propose that the occupancy diameter metric can be used to detect network congestion. Figure 24 shows the variation of the average occupancy diameter of a LAP system. The system is initially not congested and at epoch 70 one cluster is set to become congested. The average occupancy diameter of the nodes inside the congested cluster is plotted separately from the rest. The figures clearly shows that occupancy diameter inside the erroneous cluster suddenly changes after the cluster has become congested. Therefore, it is clear that a node can detect the network segment it located getting congested by observing its occupancy diameter.

5 Location-based Discovery

Location-based discovery layer of RAN organizes the nodes in a routing overlay (Figure 4) using the coordinates of the nodes produced by LAP. Even though, this overlay structure is similar to other DHT-based

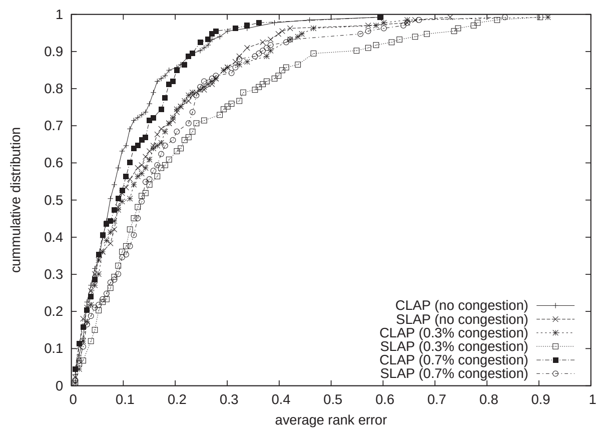

Figure 22: Cumulative distribution of rank error at various amount of network error.

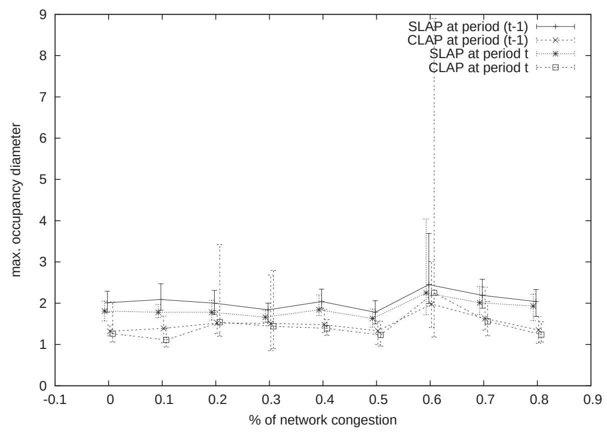

Figure 23: Variation of maximum node movement in the system with network error.

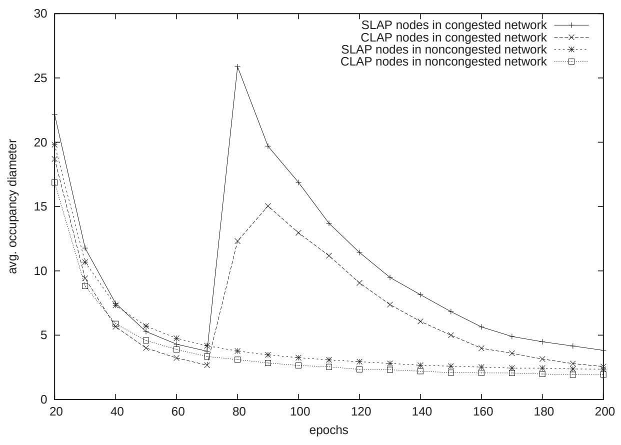

Figure 24: Variation of average occupancy diameter when a network segment become erroneous.

routing overlays, DHT key creation in RAN is much different than the others'. RAN uses a special data structure called *Hilbert curve* to create the DHT keys. The DHT keys produced in this way retain the locational information of the nodes and hence become the nodes' *location identifiers* (LIDs). RAN is the first research work that try to incorporate the locational information of a node into its ID.

The nodes in the type rings are ordered according to their LIDs, thus constructing a location aware overlay. To constrain the discussions for location-based discovery, for the rest of this section, it is assumed that there is only one ring in the system. This assumption is not overly simplifying, because the location-based discovery always follows the profile-based discovery and these two phases do not overlap.

The rest of this section is organized as follows: Section 5.1 describes the Hilbert curve data structure and the creation of LIDs, Section 5.2 presents the routing table structure used by the RAN's location-based discovery mechanism, and the procedure of creating the routing table is described in Section 5.3.

5.1 Hilbert Curve and Location IDs

Location based discovery can be considered as a search in multidimensional space (the d-dimensional Cartesian space). There is a dedicated area of study called *spatial data structures* [39] for this specific purpose. However, my aim is not to just search in multidimensional space, but to do so by creating a structured search space out of the locality information. I chose *Hilbert space filling curve* as the data structure for the location-based discovery in RAN.

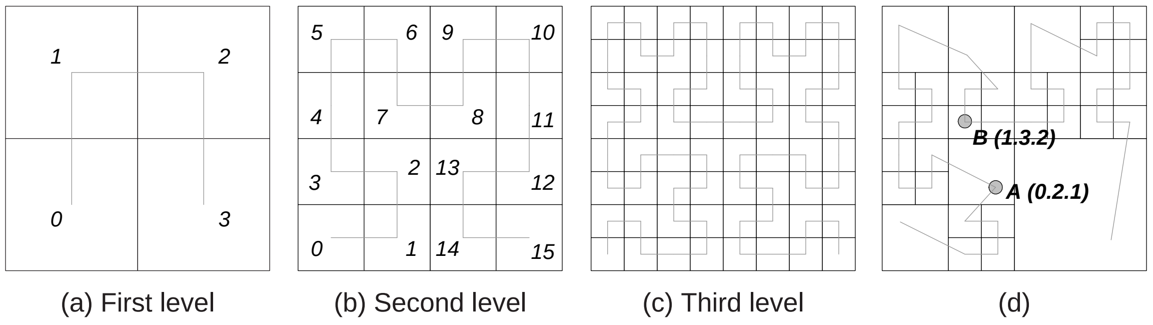

Figure 25: 2-dimensional Hilbert space filling curves at various approximation levels.

A *space filling curve* [40] is a curve that passes through all the points in a bounded space. When the points in a bounded space are approximated to areas partitioning the space, the space filling curve passing through them is said to be an approximate one. The number of the areas into which the whole space is divided determines the *approximation level* or *resolution* of the curve (Figure 25). When the approximation level tends to infinity, the approximate curve becomes the real curve. There are many types of space filling curves, but the *Hilbert curve* (HC) (Figure 25) is selected for RAN for its following characteristics:

- It maps *d*-dimensional data to a single dimensional space (curve) that provides natural ordering. This ordering is referred as *Hilbert indices* (Figure 25(a) & (b)). Generally, spatial data structures organizes the objects in a hierarchical structure and therefore the nodes higher in the hierarchy become bottleneck at the time of search. This is problem is alleviated in RAN due to the flat name space provided by HC.

- Careful observation of Figures 25(a), (b), and (c) shows that a higher approximation level structure is composed of lower level structures in a recursive manner. Using this property, the nodes in the HC can be given hierarchical names as in Figure 25(d), which can be used for creating DHT-like overlay.

- Proximity property is preserved in HC; i.e. the nodes closer in the multi-dimensional space are placed nearby in the Hilbert ordering. Hilbert ordering is not prefect; it can produce “false negatives”, which means that it can conclude two objects are far apart even though they are actually neighbors in the multi-dimensional space (indices 1 and 14 in Figure 25(b)). But it never produces “false positive”, which means, if it concludes that two nodes are closer to each other, they are really near by in the multi-dimensional space too. (Recall that the landmark binning schemes (Section 3.1.1) can produce false positives).

- Index of a node for a chosen approximation level is always fixed; it does not change due to the addition/deletion of the other nodes. It provides a consistent naming.

- Simple mathematical constructs exist [41, 42] for producing Hilbert indices at different approximation level. As the Hilbert index of an object is not affected by the operations on the other nodes, there is no need for separate algorithms for addition, insertion, or deletion operation as required in the other spatial data structures.

Once LAP has assigned coordinates to the nodes, RAN fits a fixed depth HC in the coordinate space and assigns the hierarchically structured Hilbert indices of the nodes as their LIDs. As the coordinate of a node always shows an oscillation (Figures 19 and 23), a HC above a certain resolution is not practical. Any resolution higher than this will result in continuously changing LIDs for the nodes, that results in higher management overhead. In RAN, the resolution of the HC is fixed such that the the smallest Hilbert zone covers the average occupancy diameter of the nodes in the system.

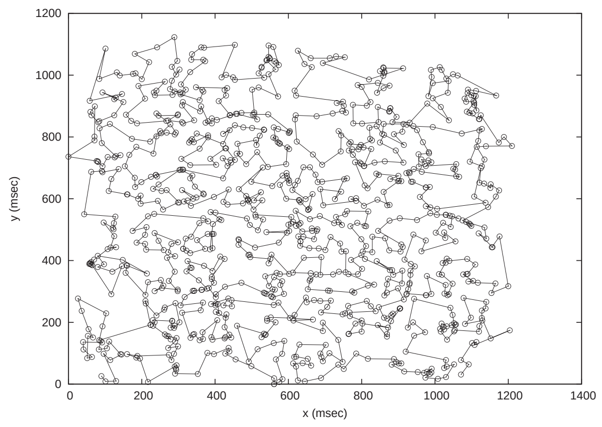

Figure 26: Nodes of a random network arranged in a Hilbert curve ($N = 1000$).

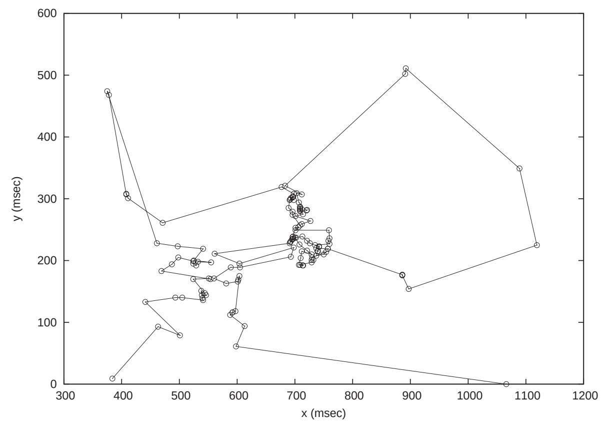

Figure 27: Nodes of the Planetlab network arranged in a Hilbert curve ($N = 133$).

Figures 26 and 27 show the nodes of a random network and Planetlab testbed organized using a HC after their coordinates have been found by CLAP. The figures show that the nodes located nearby are arranged next to each other in the HC. It is also prominent that when the system is more dense the proximity-based arrangement using HC is better; i.e., the nodes next to each other in the HC are actually close to each other in the network.

5.2 Location-based Routing Architecture

The architecture of an overlay structure is determined by the functionality and the connectivity of the nodes. For example, nodes in the overlay have to maintain connectivity with nodes with special function (operating as directory servers). Connectivity is determined by the routing pointers each node maintains. As RAN implements a fully autonomous P2P architecture where the functionality of the nodes is the same across the system, it is the connectivity between the nodes that defines the architecture of the RAN.

RAN uses both the linear and hierarchical properties of the Hilbert indices. The nodes in the RAN are arranged in a ring ordered by their Hilbert indices. Thus, the HC fit into the LAP Cartesian space (Figures 26 and 27) becomes the RAN ring when its both ends are connected together. The *routing table* in a RAN node maintains routing entries to construct and manage this RAN overlay (ring). Each routing entry maps a node's LID to its IP address. RAN routing table consists of two sections: the *ring pointer* section and *jump pointer* section. A sample routing table is shown in Table 2.

Each node in the RAN ring maintains at least two route pointers, called *ring pointers*, to create the RAN ring. One ring pointer points to the node on the left side and the other points to the node on the right. The ring structure guarantees connectivity between any two nodes in the system. Searching for a node along the ring results in $O(N)$ route-hop complexity. RAN uses the hierarchical structure of the Hilbert indices to construct the jump pointer section of the routing table that reduces the search complexity. While the ring pointer section is always complete, the jump pointer section can be partially empty. The jump pointer section is similar to the routing table section of a Pastry [9] node. It has 2^d columns and h number of rows, where h is the resolution of the HC used. If the LID of a node is $< I_0 \cdot I_1 \cdot \ldots \cdot I_h >$, the entries $[i, I_i]$ of its routing table are always empty (the shaded boxes). If the LID of the entry at $[i, j]$ is $< R_0 \cdot R_1 \cdot \ldots \cdot R_h >$, the followings are true:

$$\begin{aligned} R_i &= j && \forall i && (6) \\ R_k &= I_k && \forall k; 0 < k < (i-1), 0 < i \leq h && (7) \end{aligned}$$

In other words, for every entry at $[i, j]$ of the jump pointer section, the i^{th} digit of the LID of the remote node will be equal to j and the first $i-1$ digits of the remote LID matches with local LID. For the first row, only the fist rule can hold.

Ring Pointer Section			
Left Pointer LID	IP Address	Right Pointer LID	IP Address
2.3.3.0.0	...	2.3.1.2.0	...

Jump Pointer Section							
0		**1**		**2**		**3**	
LID	IP Address	LID	IP Address	LID	IP Address	LID	IP Address
0.2.1.0.0	...	1.2.3.3.3	...	▨	▨	3.2.2.0.1	...
2.0.2.3.2	...	2.1.3.1.2	...	2.2.3.2.0	...	▨	▨
–	–	2.3.1.2.0	...	2.3.2.0.1	...	▨	▨
2.3.3.0.0	...	▨	▨	–	–	–	–
▨	▨	–	–	–	–	–	–

Table 2: A sample RAN routing table at a node with LID of 2.3.3.1.0.

When a resource is required at a specific location, the location is converted into a hierarchically structured Hilbert index. This destination index is matched with entries in the jump pointer section. The entry with the longest prefix match is chosen as the destination node. The request is passed to the destination node. If the chosen node is not the node at the required destination, it would be the closest node to the required destination among all the nodes in the routing table. In case no entries in the jump pointer section matches the search entry, the ring pointers are used to convey the request in a direction that takes it closer to the destination. This situation can occur because the jump pointer section is allowed to be empty or expired (node pointed by the entry no longer in the system). RAN routing scheme returns the node at the specified location or, if no such node exists, returns the node that is closest to the desired position. With the jump pointers defined by the Equations 6 and 7, RAN provides a $O(h)$ search hop complexity. As the resolution of the Hilbert curve h is always fixed, the search hop complexity becomes $O(1)$.

The fixed resolution of the HC can result in mapping more than one node into a single Hilbert index and hence assigning them with the same LID. It is discussed earlier that, due to the variations in the ping values, the position of a node rarely stays fixed in time. Therefore, the node at a position at an instance may not be in the same position at another instance. Hence, the system using a HC with infinite resolution will return different nodes at different time for the same query. Service overlays using RAN as discovery substrate create location-based bindings with the resources RAN return for a query. When different resources are returned at different times, these bindings are forced reconstructed which increases the management overhead in the service overlay. Therefore, I argue that fitting a HC with infinite resolution has worse effect than fitting a HC with fixed resolution which returns multiple resources for a single query. Fixing the resolution of the HC provides additional advantage of producing $O(1)$ search hop complexity.

RAN can support both point and range queries. A *point query* requests for a single node at a particular position. It is handled as described above. A *range query* requests for a set of resources at a particular area. One or more final digits of the target LID of a range query contains wild-cards (“*”) to mark the interested area. Note that increasing the number of wild-cards in LID has the effect of lowering the resolution of the

HC and hence denoting larger areas. A range query is handled by a special range query message which is first routed to the center of the area and then finds the nodes in the required range using a *ring traversal* (routing using only the ring pointers).

If the fluctuations in the node position are small, they can be turned into an advantage in RAN. One of the issues in the HC data structure is that it produces false-negatives as described in Section 5.1. Due to this issue, the range queries can fail to include the best nodes in the returned set. For example, the node 2 in Figure 25(b) might not include 13 in a range query. However, due to the fluctuations in the position the node 2 itself might have been in the position 13 in the past. RAN nodes keep a set of *history pointers* to remember their locations in the past. These history pointers can be used to extend the normal result for a range query.

5.3 Location-based Routing Table Creation and Updates

In a RAN routing table, the jump pointer section can be incomplete or the entries can be invalid (i.e. pointing to a node that is no longer in the system). However, the entries gets filled and updated as the node stays in the system longer.

When a new node joins RAN, it first runs the LAP procedure to find its coordinates. Then, it contacts an *entry node* (any existing node in RAN) with the `join_find` request. The entry node is responsible for finding the position in the RAN ring where the new node has to be inserted. It uses LID of the new node as the target LID of a discovery message. This discovery message will return a node that shares the same LID or is closest to the required LID. The returned node provides the join-point for the new node. The join process consists of creating a initial routing table at the new node and adjusting the ring pointers in the *join-point node* and its present neighbor. The jump pointer section at the new node is initialized with the jump pointer entries in the join-point node. The join-point node decides on which side in the ring the new node has to be inserted comparing the LID of the new node with its own. Then it informs its present neighbor at that side and the new node about each other so that ring pointers are properly modified/created in both the nodes. The join-point node also updates its ring pointer to point the new node. This completes the join process.

Jump pointer section of a routing table continuously gets updated during the life time of the node. All the RAN messages includes LID of the source and that of the required destination. Nodes monitor the messages passing through them and update the proper entries of their jump pointer sections using the source LID information. RAN nodes use an aggressive update policy, where the jump pointer entries are always updated with the new information. This aggressive update ensures the jump pointer entries are mostly valid as they most likely point to live nodes (nodes that are still in the system). This confirmation is essential in a P2P environment with volatile memberships.

This jump pointer update procedure ensures that the jump pointer section of a routing table gets filled up as the system progresses. A node can enjoy the $O(1)$ search complexity only if the jump pointer section is complete. Therefore, this jump pointer update scheme creates an incentive for the nodes to stay longer in the system. The longer the nodes stays in the system, the better the performance the service overlay (that uses RAN) can extract from a resource pool. Further, when the nodes stays longer in the system, the possibility for the validity of entries in the jump pointer sections increases.

6 Profile-based Discovery

RAN's location-based discovery scheme discover resources at or near a specified location. But it does not discriminate between different types of resources. Therefore, RAN implements another augmenting discovery mechanism to discover resources based on their attributes.

Discovery mechanisms rely on naming schemes to identify the objects to be discovered. Depending on what they want to discover, they implement different types of naming schemes. For example, Internet uses IP address to uniquely identify the resources, that also implicitly represents the administrative structure (in the form of IP address classes); on the other hand, directory services like LDAP [1] are used to discover resources based on their characteristics. Existing naming schemes can be classified into two major classes: *label-based naming* (LBN) and *description-based naming* (DBN) systems. LBN systems affix a label, for example a DNS name, with an object and use it to locate and access the object. DBN systems, on the other hand, use a set of attribute-value tuples to name or to *describe* an object. Even though it provides flexibility in answering resource queries, it comes at the cost of additional overhead. Most of the overhead is associated with maintaining databases of the attribute-value tuples, enforcing consistency among a network of such databases, and resolving queries using values within these databases. Often the overhead increases dramatically with increasing size of the network [32].

The significant difference between the two classes of systems is that the LBN systems normally result in unique names for the objects. But the uniqueness of the description-based names is determined by the attributes and values used for the descriptions. Therefore, while the label-based names are used for locating and accessing a specific object, description-based names are used for locating and accessing an object belonging to a given family.

I introduce a new naming strategy called the *profile-based naming* (PBN) that combines the benefits of DBN and LBN naming systems. In simple terms, PBN considers the set of attribute-value pairs describing an object to be the *profile*[1] of the object. The profiles are labeled with *profile IDs* (PIDs) that become the name of the corresponding objects. PBN does not label all combinations of attribute-value tuples, but only the ones that are popular in the system. It prevents the name space from getting too big to handle. DBN systems are flexible and capable of handling any type of resource query, while LBN systems are known for their scalability (e.g. DNS). PBN's novelty is its ability to trade off flexibility for scalability. Tagging resources and queries with PIDs reduces the system overhead in a number of ways: (a) compact resource dissemination and query messages; (b) avoidance of information loss due to aggregation or summarization of resource descriptions; and (c) ability to incorporate an efficient profile-based discovery mechanism.

The motivation behind the PBN scheme is that the popularity of resources are skewed to a small set of resources. Our work [43] presents a case study supporting this argument and evaluates the performance of a primitive PBN system in terms of scalability.

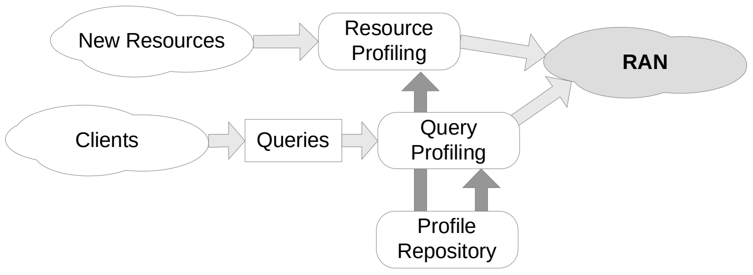

Figure 28: Architecture of a profile-based naming system.

Figure 28 illustrates the architecture of the *profile-based naming* (PBN) system. Incoming resources undergo a profiling phase which tags them with appropriate profile IDs. Similarly, queries are profiled based on the same profile repository. The resources and queries that do not match one of the recognized profiles will not be accepted in the system. However, as PBN uses the HC to name the PIDs, it can return the closest matching profiles for the queries failed to be accepted. The profile repository is distributed among

[1]the terms *profile* and *type* mean the same in this report

the nodes so that each node. Therefore, the resource/query profiling process also is decentralized alleviating bottleneck problem.

The rest of this section is organized as follows: Section 6.1 explains the profile-based naming process in RAN; Section 6.2 presents the PID-based routing procedure; and the construction of type rings and PID-based routing tables is given in Section 6.3.

6.1 Profile IDs

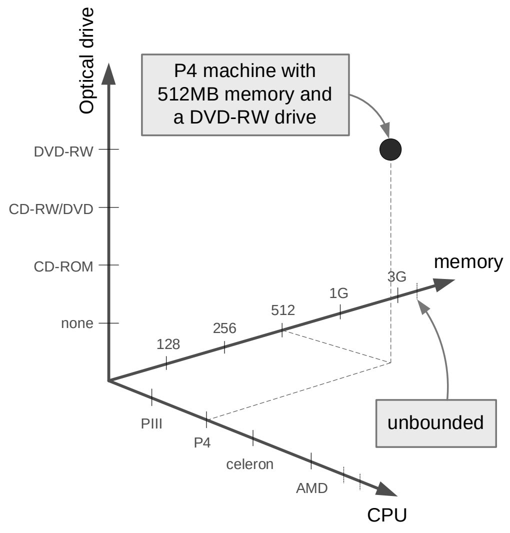

Figure 29: Attribute-value space.

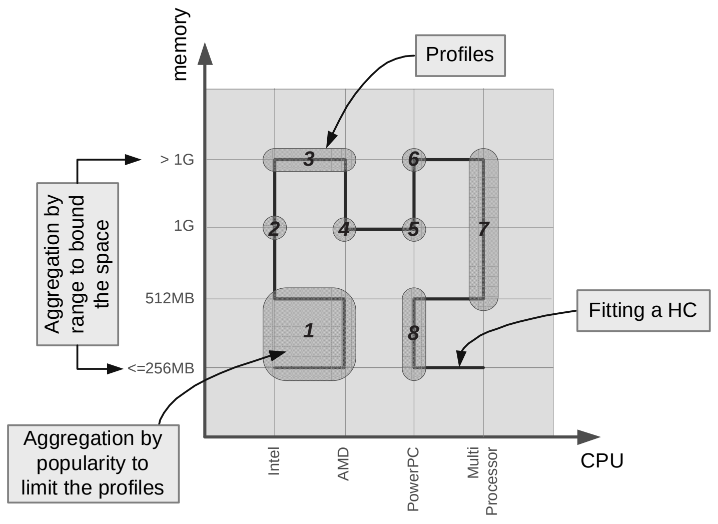
memory
Profiles
Aggregation by range to bound the space
> 1G
1G
512MB
<=256MB
3
6
2
4
5
7
1
8
Fitting a HC
Aggregation by popularity to limit the profiles
Intel
AMD
PowerPC
Multi Processor
CPU

Figure 30: Profile space showing the creation of profiles.

Resource descriptions can be considered as points in an attribute-value space where each attribute becomes an axes of the space and the values become the tics in those axes (Figure 29). The axis in this space is unbounded producing infinite number of points and hence an unmanageably huge name space. PBN transforms this attribute-value space into a bounded *profile space*. The profile space can be bounded both in terms of number of axis and the values considered as shown in Figure 30. This bounded profile space provides a metric space for attribute-based discovery, analogous to the network delay space for location-based discovery. A Hilbert curve is fit into this metric space to name the points in the space. However, unlike the location-based naming, not all the the points are named in profile-based naming. The number of points names, i.e. the allowed *profiles*, are restricted based on the popularity. Figure 30 shows an example where out of 16 possible points only 8 profiles are created. Profiles are assigned with HC-based PIDs. The PIDs can be of three types: (a) fully resolved like the profile 2 or 6 which have the PIDs 1.0 and 2.1 respectively; (b) list-based like the profile 3 or 7 which have the PIDs $[1.1, 1.2]$ and $[2.2, 2.3, 3.0]$; and (c) wild card-based like the profile 1 which has the PID $0.*$.

As shown in the Figure 3, RAN overlay consists of type rings. These rings are named by a PID. In other words, each popular set of resources in the system own a type ring in RAN overlay.

6.2 Profile-based Routing Architecture

Profile-based routing is very similar to the location-based routing described in Section 5.2. Each RAN node, in addition to the location-based routing table, maintains a similar profile-based routing table. An example profile-based routing table is shown in Table 3. In this example, a 3-D profile space is assumed. Therefore,

Ring Pointer Section

Left Pointer PID	Left Pointer LID	IP	Right Pointer PID	Right Pointer LID	IP
7.4.*	2.3.3.1.2	...	7.5.2, 7.5.3	2.3.2.0.1	...

Jump Pointer Section

0			**1**			**2**			**7**		
PID	LID	IP	PID	LID	IP	PID	LID	IP	PID	LID	IP
0.7.3	2.2.1.0.0	...	1.3.0	2.1.0.1.1	...	2.2.0	2.2.1.3.0	...	▨	▨	▨
7.0.0	2.3.1.2.0	...	–	–	–	7.2.4	2.2.3.3.0	...	7.7.3	2.3.3.2.0	...
■	■	■	▨	▨	▨	7.5.2	2.3.3.1.1	...	–	–	–

Table 3: A sample profile routing table at a node with PID of 7.5.1.

each digit in the Hilbert index can have 2^3 possible values. The resolution of the HC in this example is 3. Note that the Table does not show the full width of the table; cells from 3–6 are hidden.

The ring pointers of the profile-routing table construct the neighborhood rings of the RAN overlay structure shown in Figure 3. Therefore, the stacking order of the type rings is determined by Hilbert ordering of the PIDs. As in the location-based routing table, while the ring pointers of the profile-based routing table point to the nodes in the adjacent type rings in the Hilbert ordering, the jump pointers points to the nodes in some distant type rings.

The primary differences between profile-based and location-based routing tables are (a) ring pointers can point to nodes that represent aggregated Hilbert indices as explained in Section 6.1; (b) the route entries also include the LIDs of the pointed nodes. This LID is not used in profile-based routing, but used in the routing table construction (Section 6.3); and (c) it is possible to have multiple "home" cells (gray boxes) in a single row of a profile-based routing table due to the list-based and wild card-based PIDs.

Profile-based discovery in RAN can return a closest matching profile if the exact match is not included in the profile repository. In addition to the system enabled list-based wild card-based aggregation, users can launch range queries using the proximity property of the HC.

6.3 Profile-based Routing Table Construction

Similar to the location-based routing table (Section 5.3), profile-based routing table is initialized at the time of the node join and continuously updated throughout the life time of the node.

When there is only one node in the system, it retains the whole profile repository information. When another node joins, depending on the profile of the new node, the first node decides whether the new node should be included in the same type ring, or allowed to spawn a new type ring. If a new type is to be spawned, the first node instructs the new node the decision and transfers part of the profile space information to the new node. The spawning process creates proper ring pointer and jump pointer entries in the profile-based routing table in each node. When there are a few nodes in the system, each of them maintains a large portion of the profile repository. As more resources join the system and new type rings are spawned, the profile space information is more and more divided among the resources/type rings and, hence, creating a scalable architecture. Figure 31 shows this information distribution over multiple resource join.

Profile-based discovery can suffer from triangular routing in terms of network proximity. For example, assume that the network locations are named as $0, \ldots, 10$ and types as $A, \ldots, F$. Note that an alphabet represents a type ring, while an alphabet-number combination refers to a node. If $A0$ wants to find $C0$, $A0$ has to reach the type ring C through type ring B. If $A0$ has a pointer only to $B3$ and $B3$ only to $C5$, to reach $C0$ from $A0$ (who are in the same location), the query will be routed through location 3 and 5 and comes back to location 0. To eliminate this problem, it is ensured that ring and jump pointers are connects the rings

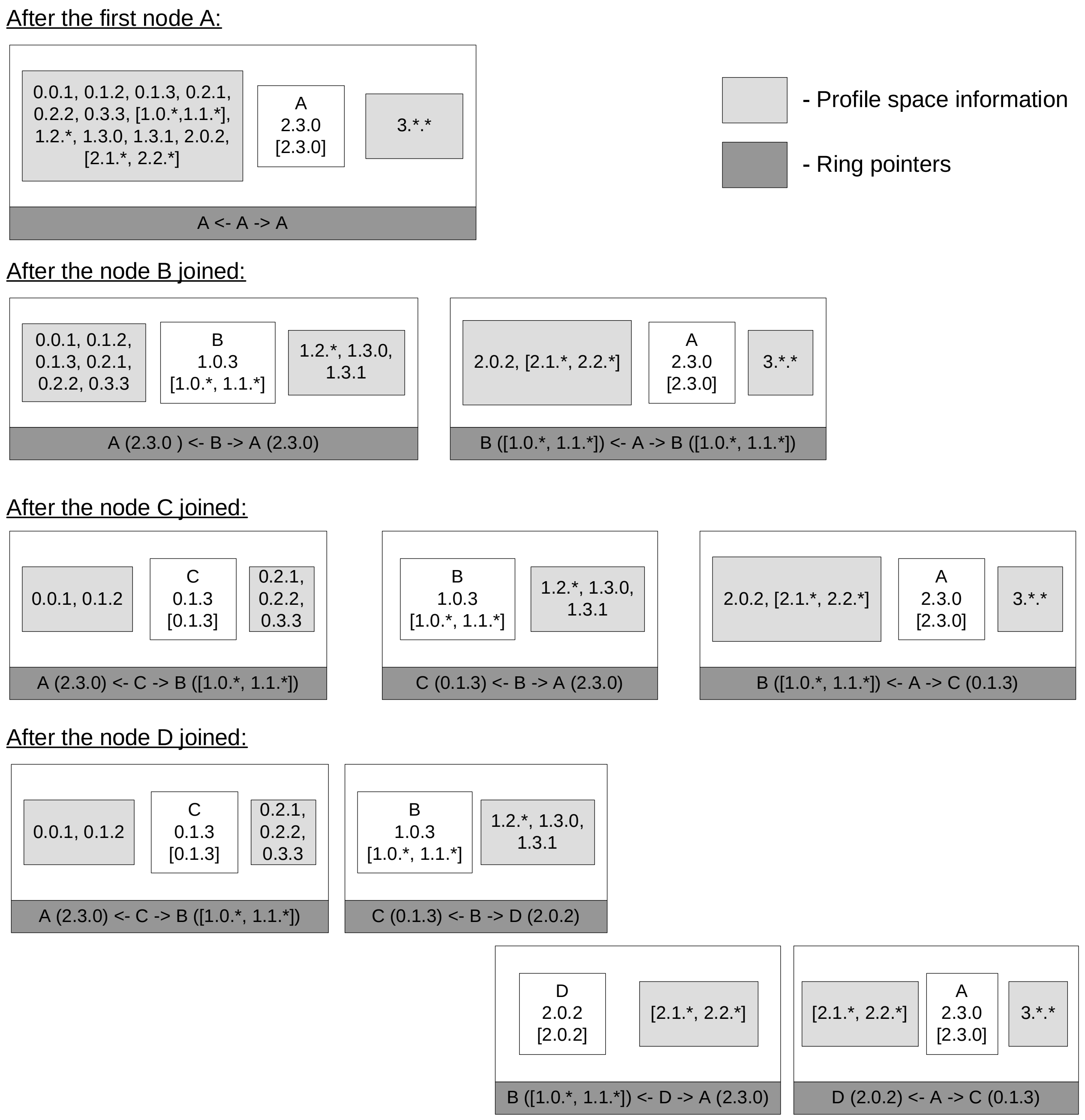

Figure 31: Distribution of profile space information with spawning of new type rings.

through the nodes within the same locality. That is why the interconnecting rings are called “neighborhood” rings. The route pointers keep LIDs of the pointed nodes and when nodes with closer LIDs are encountered in the by-passing query messages, the routing entries are updated with the more appropriate node.

7 Conclusion

This research work focuses on developing a scalable P2P resource discovery framework called resource addressable network or RAN. It is scalable in terms of the message overhead, loading on the participating resources, and the number of resources. This is achieved by a fully decentralized implementation with efficient algorithms. RAN can discover resources along two axis: it can discover a resource with specified attributes and/or at a specified location in the network. RAN is designed to function as a discovery substrate for any P2P overlay.

This report presented the design of RAN in three major parts: The landmark aided positioning (LAP) which creates a network map of the participating resources supporting a location-based discovery; the location-based discovery structure that can discover resource at specified locations in the Internet; and a profile-based discovery structure that can discover resources based on the specified attributes. The design of the LAP and its concepts are clearly described and a performance evaluation study is presented. The next subsection describes contributions of of my research to the resource discovery area of research and provides a road map to be followed to complete my Ph.D.

7.1 Research Road Map and Expected Contributions

RAN makes number of contributions to the area of P2P resource discovery. In this section, I summarize the major RAN issues, issues addressed and the approach followed, and a brief road map for proposed work.

7.1.1 Architecture for resource discovery:

Structured document discovery mechanisms randomly embed documents onto different points of a metric space and then construct search graphs in this space along which the document discovery process is carried out. Because the random embedding destroys the proximity information, some discovery mechanism introduce proximity notion in an implementation specific manner. The RAN architecture, on the other hand, uses a proximity space among the resources (e.g., the delay space) as the metric space.
Implemented: The concept of creating a structured metric space from an unstructured one is shown feasible by using Hilbert curve.
Proposed work: In a document discovery system, the objects are uniformly distributed in the metric space enabling an efficient routing design. In resource discovery, the objects can be unevenly distributed in the metric space as the mapping is given. Further work is planned to study the deficiencies the routing mechanism will face due to this non-uniformity. The deficiencies can be increase in search complexity or route maintenance.

7.1.2 Network positioning:

Even though network positioning is a well researched sub-area in location-based discovery systems, some practical issues such as message overhead, validity of the ping values, and stability of the coordinates are not well studied. RAN develops new algorithms or schemes to improve the performance of network positioning in terms of these aspects.
Implemented:

- An improved network positioning system called clustered landmark aided positioning (CLAP) is developed. CLAP makes reasonable assumptions that the nodes within a locality can trust each other and the pings within a local network segment are not affected by congestion in the network backbone. Using these assumptions, CLAP is shown to produce more accurate and robust network positioning system even in congested network conditions which offset the ping measurements.
- This research recognizes the need for designing an algorithm for landmark positioning that takes fewer iterations to converge so that the inter-landmark pings, and thus the message overhead, can be reduced. A new algorithm called SpringEq is already developed and is shown to produce fewer iterations to converge than the Simplex and Spring algorithms. At present, a centralized implementation is studied.
- Number of simulation studies have been conducted to compare the Spring and the widely used Simplex algorithms. The Spring is showed performing better in node positioning case than the Simplex.
- A prototype implementation is almost finished on the Planetlab network. This implementation is carried out by a colleague based on my design.

Proposed work:

- As the network positioning problem is formed as an optimization problem, it is always a concern that whether the solution converge to the global minima. An study on the SpringEq algorithm is planned to analyze its convergence and stability characteristics.
- Results from the prototype implementation will be used to further study the proposed schemes.

7.1.3 Location-based discovery:

RAN's location-based discovery applies the concept of unstructured → structured metric space transformation for finding the resources based on their location. The network delay space provides the metric space for location-based resource discovery. The placement of the objects (resources) onto this space is performed by the LAP scheme and the Hilbert curve data structure is fit into this network delay space to extract a structure out of it.
Implemented:

- Algorithms are implemented to fit the Hilbert curve structure onto the network delay space so that each coordinates can be given a location ID (LID) in the form of Hilbert indices.
- An efficient routing overlay mechanism is designed using these LIDs.
- It is mentioned before that non-uniformity can affect the search complexity of scheme. However, it is shown that when the locating resolution is fixed, the search complexity can be fixed as well.
- A prototype implementation is almost finished on the Planetlab network.

Proposed work:

- A detailed simulation study is to be conducted to evaluate the performance of this scheme.
- The practical value for locating resolution is to be analyzed based on the errors in ping values and the requirements of the applications that benefits from the location-based discovery service.
- The results from the prototype implementation will be considered for improving the scheme.

7.1.4 Attribute-based discovery:

The concept behind the attribute-based discovery in RAN is similar to the location-based discovery. Analogous to LAP, RAN uses a naming scheme called profile-based naming to map resources on to the attribute-value space and then uses Hilbert curve to assign the resources with profile IDs (PIDs).

Implemented: The profile-naming scheme is designed exploiting the observation that the popularity of resources are always skewed in resource queries. Also designed the profile-based routing overlay to discover resources based on profiles. The search complexity in this scheme depends on the popularity distribution of the resources.

Proposed work: Unlike the network delay space, the attribute-value space is not naturally bounded. But, a bounded space is a requirement for creating a structured space. At the current implementation, the space is shrunk down using popularity-based aggregation. However, I recognize that there can be other ways for this mapping that can be more efficient. An in-depth study is needed to analyze the characteristics of possible mappings and their effect in search complexity. Theoretically, profiles can be abstract providing a possibility of resources based on their functional descriptions (for example, finding the landmark nodes). The impact of including such virtual profiles into the profile space has to be studied.

References

[1] T. A. Howes, “The lightweight directory access protocol: X.500 lite,” Technical report, Center for Information Technology Integration, University of Michigan, July 1995.

[2] W. Hoschek, “The web service discovery architecture,” in *International IEEE/ACM Supercomputing Conference*, Nov. 2002, pp. 1–15.

[3] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, “SETI@home: An experiment in public-resource computing,” *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, Nov 2002.

[4] Onion Networks, “Swarmcast,” http://swarmcast.net/.

[5] I. Foster and A. Iamnitchi, “On death, taxes, and the convergence of peer-to-peer and grid computing,” in *2nd International Workshop on Peer-to-Peer Systems (IPTPS’03)*, Feb. 2003.

[6] A. Nakao, L. Peterson, and A. Bavier, “A routing underlay for overlay networks,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, Aug. 2003, pp. 11–18.

[7] “Gnutella,” http://www.gnutella.com/.

[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in *ACM SIGCOMM*, Aug. 2001.

[9] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Nov. 2001.

[10] M. Maheswaran, B. Maniymaran, S. Asaduzzaman, and A. Mitra, “Towards a quality of service aware public computing utility,” in *1st IEEE NCA Workshop on Adaptive Grid Computing (in the proceedings of 3rd IEEE Symposium on Network Computing)*, Aug. 2004.

[11] P. Francis, S. Jamin, C. Jin, et al., “IDMaps: a global internet host distance estimation service,” in *IEEE/ACM Transactions on Networking (TON)*, Oct. 2001, vol. 9, pp. 525–540.

[12] Y. Shavitt and T. Tankel, "Big-Bang simulation for embedding network distance in Euclidean space," in *IEEE Infocomm*, April 2003.

[13] S. M. Hotz, *Routing information organization to support scalable interdomain routing with heterogeneous path requirements*, Ph.D. thesis (draft), University of Southern California, 1994.

[14] Z. Xu, M. Mahalingam, and M. Karlsson, "Turning heterogenity to an advantage in overlay routing," in *INFOCOM*, April 2003.

[15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *Proceedings of IEEE INFOCOM'02*, June 2002.

[16] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Internet Measurement Conference*, Oct. 2003, pp. 143 – 152.

[17] T. S. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *INFOCOM 2002*, June 2002.

[18] T. S. E. Ng and H. Zhang, "A network positioning system for the internet," in *USENIX Conference*, June 2004.

[19] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris, "Practical, distributed network coordinates," in *The 2nd Workshop on Hot Topics in Networks (HotNets-II)*, Nov. 2003.

[20] F. Dabek, R. Cox, F. Kaahoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *SIGCOMM 2004*, Aug. 2004.

[21] M. Costa, M. Castro, A. Rowstron, and P. Key, "PIC: Practical internet coordinates for distance estimation," Techincal Report MSR-TR-2003-53, Microsoft Research, Cambridge, UK, Sept. 2003.

[22] M. P. J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti, "Lighthouses for scalable distributed location," in *IPTPS 2003*, 2003.

[23] L. Lehman and S. Lerman, "PCoord: Network position estimation using peer-to-peer measurements," in *Third IEEE International Symposium on Network Computing and Applications (NCA'04)*, Aug. 2004, pp. 15–24.

[24] N. J. A. Harvey, M. B. Jones, S. Saroiu, M. Theimer, and A. Wolman, "Skipnet: A scalable overlay network with practical locality properties," in *4th USENIX Symposium on Internet Technologies and Systems*, March 2003.

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

[26] Z. Xu and Z. Zhang, "Building low-maintenance expressways fro p2p systems," Technical Report HPL-2002-41, HP Laboratories Palo Alto, March 2002.

[27] S. Fitzgerald, I. Foster, C. Kesselman, G. V. Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," in *The 6th IEEE Symposium on High-Performance Distributed Computing*, 1997, pp. 365–375.

[28] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in *The 10th IEEE International Symposium On High-Performance Distributed Computing*, Aug. 2001.

[29] "The Globus alliance," http://www.globus.org/.

[30] X. Zhang, J. Freschl, and J. Schopf, "A performance study of monitoring and information services for distributed systems," in *The 12th International Symposium on High-Performance Distributed Computing*, Aug. 2003, pp. 270–282.

[31] P. A. Dinda and D. Lu, "Nondeterministic queries in a relational grid information service," technical report, Computer Science Department, Northwestern University, Apr. 2003.

[32] P. Dinda and B. Plale, "A unified relational approach to grid information services," Informational Draft GWD-GIS-012-1, Grid Forum, Feb. 2003.

[33] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," in *17th ACM Symposium on Operating Systems Principles*, Dec. 1999.

[34] M. Balazinska, H. Balakrishnan, and D. Karger, "INS/Twine: A scalable peer-to-peer architect ure for intentional resource discovery," in *International Conference on Pervasive Computing*, Aug. 2002.

[35] F. J. Blom, "Considerations on spring analogy," *Interanational Journal for Numerical Methods in Fluids*, vol. 32, no. 6, pp. 647–668, March 2000.

[36] J. Stribling, "All pairs pings data for Planetlab," http://pdos.csail.mit.edu/˜strib/pl_app/.

[37] "Planetlab project," http://www.planet-lab.org/.

[38] "Route Views project," http://routeviews.org/, University of Oregon.

[39] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley Publishing Company, Inc., 1989.

[40] H. Sagan, *Space-filling curves*, Springer Verlag, 1994.

[41] A. R. Butz, "Alternative algorithm for hilbert's space filling curve," in *IEEE Transactions on Computers*, April 1971.

[42] S. Shekhar and S. Chawla, *Spatial Databases : A Tour*, Prentice Hall, 1st edition, 2002.

[43] B. Maniymaran and M. Maheswaran, "On the benefits of profile-based naming for large network computing systems," in *16th International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, Nov. 2004.