

A Distributed Measurement Scheme for Internet Latency Estimation

Negar Hariri¹, Behnoosh Hariri², Shervin Shirmohammadi²,

¹College of Computing and Digital Media, DePaul University, Chicago, USA

²Distributed Collaborative Virtual Environment Research Laboratory

University of Ottawa, Ottawa, Canada

Email (hariri@cs.depaul.edu, bhariri@discover.uottawa.ca, shervin@discover.uottawa.ca,)

Abstract—Estimating latency between the hosts in the Internet can play a significant role in the improvement of the performance of many services that use latency among hosts to make routing decisions. A popular example is peer to peer networks that need to build an overlay between peers in a way that minimizes the message exchange delay among the peers. Acquisition of latency information requires a considerable amount of measurements to be performed at each node in order for that node to keep a record of its latency to all the other nodes. Moreover, measured latency values are frequently subject to change and need to be regularly repeated in order to be updated against network dynamics. This has motivated the use of techniques that alleviate the need for a large number of empirical measurements, and try to predict the entire network latency matrix using a small set of latency measurements. Coordinate-based approaches are the most popular solutions to this problem. The basic idea behind coordinates based schemes is to model the latency between each pair of nodes as the virtual distance among those nodes in a virtual coordinate system. This article proposes a new decentralized coordinate-based solution to the problem of Internet delay measurement. Simulation results demonstrate that the proposed system provides relatively accurate estimations.

Keywords- Distributed measurement, peer-to-peer networks, virtual coordinates, Latency estimation, distributed systems, Internet latency.

I. INTRODUCTION

In the Last few years there has been a fast growing trend towards the use of peer-to-peer based Internet services. Therefore, peer-to-peer distributed architectures have become the focus of countless studies. The goal of the majority of these studies is to either improve the achieved quality of service or to reduce the total amount of traffic that needs to be exchanged among the peers in a P2P networks. In early peer-to-peer networks like Gnutella[1], locating a data object was mainly based on flooding the query that results in a high amount of search traffic. Later, many other search algorithms

[2]-[5] were suggested to narrow down the search space and reduce the search traffic. Distributed Hash Tables (DHTs) are popular solutions to the P2P search problem in which the queries are guaranteed to be answered within a limited number of search hops. Although reducing the number of search hops directly affects the search efficiency, minimizing the transport latency of each hop is important as well. As each link in the overlay layer corresponds to one or more links in the physical layer (Internet), a short path may map to a long path in the physical layer. Ignoring this important issue may result in a considerable increase in communication costs and delay. Experiments in [6] show that only 2 to 5 percent of the total communications in a P2P network are between nodes in the same autonomous system(AS), while more than 40% of the nodes are within one of the main 10 ASs. This reveals the fact that most of the exchanged messages pass across AS borders and will therefore experience high latencies. Moreover, a large number links should be utilized in the underlying layer which causes more traffic in that layer. Hence, exploiting the topological information about the relative locations of the participating nodes can be useful in reducing the overlay network transport latency as well as underlying network bandwidth utilization. For example, latency information has been used in Pastry [7] where locality-aware algorithm has been introduced for routing queries over the network.

Although Latency information is useful to a variety of internet applications, latency estimation over the Internet as a large-scale network is a challenging problem due to the large amount of measurements that needs to be performed. Moreover, Internet is a dynamic environment where the measurement results are needed to be constantly updated to account for the changes.

The challenge of performing measurement over large scale dynamic networks such as Internet has introduced a new field in measurement research known as *distributed measurement*. The idea of distributed measurement has been applied in a number of related applications. For example, [8] proposes a

distributed digital measurement architecture for industrial applications. [9] uses distributed measurement for power quality calculations. A review of different solutions proposed for One Way Delay measurement (OWD) and their pros and cons is presented in [10], where various point-to-point measurement techniques such as Network Time Protocol, Global Positioning System, the IEEE 1588 Standard, and others are discussed.

While attempts have been made to improve delay estimation over large scale networks, the majority of existing works suffer from problems such as fixed landmarks, unproven convergence, or lack of locality-awareness. This has been the motivation behind this work to design a distributed delay estimation technique that overcomes these shortcomings and can be used to more accurately measure node-to-node delay across a large network. At a high level, our contributions are twofold. First, we propose a new distributed measurement algorithm for delay estimation which is more accurate than existing ones, can be mathematically proven to converge, and has a locality-aware design as requirement for many P2P applications. Second, we propose a P2P network based on our algorithm and show that our approach not only reduces network query time by as much as 45%, but also can reduce search traffic by about 5%.

This paper is organized as follows: section II discusses delay in P2P overlays and the problem of locality-aware overlay construction. Section III includes related works and existing solutions to the problem. In section IV we will introduce our approach to distributed measurement of internet latencies. In section V we will discuss methods to improve the accuracy of the measurements. The application of the coordinate system in locality-aware overlay construction will be investigated in Section VI, while Section VII will provide the results of the experimental evaluations. Finally section VIII will include summary and conclusion.

II. BACKGROUND: LATENCY IN P2P OVERLAYS

The goal in locality-aware overlay construction is to establish the logical links between peers based on their distances in the underlying physical network. The distance between any two nodes can be measured based on different metrics. Considering end-to-end latency as the distance metric, peers in a locality-aware overlay are supposed to be connected through relatively low latency links. Without this locality-aware overlay, there might be a situation where two close nodes, A and B, that frequently communicate with each other are forced to connect through an intermediate distant node C (instead of a more appropriate node). This will result in reduced performance. Moreover, in many real-time applications such as video conferencing or multiuser gaming, this will also result in violation of tolerable delay thresholds and makes the application unusable.

In structured peer-to-peer networks, each node is assigned a unique key in the key space that can be calculated by hashing one of its unique properties (for example, its IP address and port number). As Logical links between peers are established based on their hash keys, it is possible for distant nodes to be

neighbors in the overlay. A number of works [11][12] have addressed the issue of eliminating high delay connections or clustering nodes based on their positions in unstructured networks. A similar argument also exists in the context of P2P networks, where each node should be assigned an ID that is calculated based on its relative position to other network nodes.

In order to discover the network topology, each peer should be able to estimate its distance to any other peer in the network. In large networks, this problem is quite challenging as one-to-one distance estimation to all other nodes is a time-consuming and impractical operation. The simplest solution to this problem is the centralized approach where the topological information of the whole network is gathered and maintained in a central control server and made available to the peers. However, the use of a central server in the system can cause a potential bottleneck and will not scale as the size of the network grows. Moreover, such information is not easy to collect and update due to the large number of participating nodes and highly dynamic underlying network conditions. This has motivated techniques that attempt to estimate the end-to-end latency among peers using only a small set of local measurements. Coordinate-based approaches are among such techniques [13]-[17]. These approaches are based on the idea of assigning virtual coordinates to each node such that the distance between any two nodes can be estimated as a function of their coordinates. In this article, we propose a new coordinate-based algorithm by extending our initial idea published in [18] and applying it to the construction of a new P2P network. The initial algorithms as discussed in [18] have been elaborated and improved and also new algorithms have been introduced which increase efficacy and accuracy of delay measurement and positioning in P2P systems. Before getting into detailed discussion of the algorithms, let us first take a look at some related work in the field of coordinate-based delay measurement.

III. RELATED WORKS

Coordinate-based approach for distance measurement was first introduced in the GNP (Global Network Positioning) [19] where each node calculates its coordinates in a D-dimensional geometric coordinate space by using a small number of sample distances. In GNP, a fixed set of nodes, called landmarks, calculate their relative coordinates by first measuring the inter-landmark round trip times and then seeking to minimize the difference between geometric distances and measured distances. After this initialization step, newly joining hosts measure their distances to each of the landmarks and then try to infer their geometric coordinates by minimizing the difference between the real distances and the geometric distances which leads to an optimization problem. In order to improve the performance of GNP system, a virtual landmarks algorithm was suggested in [20] which uses Principal Component Analysis for extracting the topological information of the network. Similar to GNP, in virtual landmarks algorithm, a fixed set of landmarks are exploited.

Internet Iso-bar [21] is another example that uses landmarks for distance prediction among Internet nodes. In this system, each host measures its distance to L fixed landmarks to form a distance vector. A generic clustering algorithm is then applied to cluster hosts into K groups based on a correlation distance metric. Finally, a monitor is chosen for each cluster which periodically measures its distance to cluster members; based on these measurements, the distance between any pair of nodes can be estimated.

A landmark-based approach has been similarly used in order to construct a locality sensitive CAN (Content Addressable Network) [22]. Each node measures its distance to each of the m landmarks and orders the landmarks in ascending order of distances. With m landmarks, $m!$ ordering is possible and the key space is partitioned into $m!$ regions. Each incoming peer joins the network at a random point in the region that associates with its landmark ordering. Close nodes in the underlying IP network are expected to have similar landmark orderings; therefore, there is a high probability that they join the same region. Landmarks-based topology discovery has been addressed in many other researches as well [23][24][25]. While the above approaches demonstrate acceptable accuracy, the use of landmarks can introduce potential bottlenecks in the system as landmarks must be able to reply to the distance queries from all the peers. In addition, the choice of optimal landmark locations for accurate coordinate calculations is not trivial and is a problem of its own. Therefore, some studies have proposed pure distributed solutions to the problem. Lighthouse [26] follows this strategy, where each node selects a set of random nodes as its landmark set and forms its local coordinate system. A transition matrix is calculated for every local coordinate system that is used to map it to a global coordinate system shared among all the nodes. Although this approach alleviates the use of a fixed set of landmarks, the calculation of the transition matrices and keeping them up to date makes it very costly in terms of the message communications. Practical Internet Coordinates (PIC) [24], takes advantage of another strategy in order to avoid a fixed landmark set. In this approach, network nodes are divided into two categories. The first group contains the nodes that have already calculated their coordinates and the second group randomly takes a number of nodes from the first group as the landmark set. Despite the simplicity of the system, error propagation can dramatically reduce the accuracy of the calculated positions. As every node calculates its position with some inaccuracy, this error accumulates in the estimated position of another node when it is used as a landmark for that node.

In Vivaldi [27], the network is modeled as a mass-spring system in which each node represents a mass in the model. The measured distance between node j and node i , (d_{ij}) is assumed to be the rest length of the spring that connects their corresponding masses. Therefore, assuming the coordinates of nodes i and j to be C_i and C_j , the length of their connecting spring will be $d(C_i - C_j) = \|C_i - C_j\|$. Vivaldi tries to gradually minimize the energy of the system through an

iterative process. This is similar to minimizing the squared error function in equation (1).

$$f(C) = \sum_j \sum_i |d(C_i - C_j) - d_{ij}|^2 \quad (1)$$

In Vivaldi, the coordinate calculation algorithm starts by assigning random coordinates to each node. The difference between d_{ij} and $d(C_i - C_j)$ exerts a force on nodes i and j that causes them to move and change their coordinates accordingly. The movement relation is the result of the spring force between the nodes. The new coordinate of node i is found according to relation (2) where parameter δ is adaptively calculated based on the amount of confidence of i and j in their current positions and the unit vector $u(C_i - C_j)$ gives the direction of the force on node i .

As nodes keep updating their coordinates, their coordinates finally converge to values that can be used for distance prediction.

$$C_i = C_i + \delta \times (d_{ij} - d(C_i - C_j)) \times u(C_i - C_j) \quad (2)$$

Pcoord [28] is also a fully decentralized coordinate system that uses a similar coordinate estimation approach to Vivaldi. However, it aims at improving the convergence rate of Vivaldi by providing an enhanced message exchange protocol. Although no mathematical proof of convergence has been provided for Vivaldi and Pcoord, simulation results demonstrate that both of these systems convergence to the optimal solution within a reasonable time. However, the general convergence proof is still an issue where cases of trapping in local minimums are questionable. Similarly, Pharos [29] was suggested to refine Vivaldi's algorithm. After analysis of the coordinates calculated using Vivaldi's algorithm, it was shown that there is more error in estimating short distances. In order to alleviate this problem, it was suggested to assign global and local coordinates (that are two different sets) to each node. Global coordinates are computed using the same approach as in Vivaldi. For calculating the local coordinates, network nodes are clustered based on their distance with respect to each other. After clustering, Vivaldi algorithm will apply to each cluster independent of the rest of the network and in this way the local coordinates will be calculated. If nodes are located in the same cluster, their distance is estimated based on their local coordinates and, if they are not in the same cluster, global coordinates are used for distance estimation. Although this method outperforms Vivaldi in terms of accuracy, its major problem is that similar to [21] a fixed set of landmarks are used for clustering the nodes while one of the major goals of Vivaldi is to avoid a fixed set of nodes to be used as landmarks.

Finally, in Meridian [30] each participating host is responsible to keep track of the locations of a fixed number of peers and organize them into concentric rings according to their locations. For example, suppose that the host distance to peer j is d_j . j will be placed in the i^{th} ring if and only if $as^{i-1} < d_j \leq$

αs^i where α and s are algorithm specific parameters. Position-based routing in Meridian is performed using physical latencies and by utilizing the above-mentioned concentric rings. Although this approach can be exploited for routing queries, it is not applicable in locality-aware overlay construction.

In this paper, we present a fully decentralized measurement system, inspired mainly by Vivaldi, that doesn't rely on any infrastructure support or fixed landmarks from the underlying network. In addition, we will introduce an optimization technique (see section V) whose convergence can be mathematically proven. Finally, our algorithm has a locality-aware design which is important in P2P applications. Simulations in section VII show that our results are more accurate than existing techniques.

IV. DISTRIBUTED COORDINATE MEASUREMENT

In the proposed approach, coordinate estimation is performed in two steps. First, rough initial coordinates are assigned to each node. Then, these coordinates will be used as inputs to the second step where they will be refined using an iterative process.

In the proposed distributed measurement scheme, the nodes are divided into K groups, A_1 to A_k . The group membership is better to be uniformly distributed all over the network. Therefore we can choose a group number as a function of its IP address (let's say $M = \text{IP_address} \bmod k$) where $0, 1, \dots, K$ are assigned to groups A_1 to A_k respectively. The algorithm runs in K phases. In the j^{th} phase of the algorithm, each node in group A_i takes a fixed number of nodes from the group $A_{[(i+j) \bmod k]}$ as its own landmark set and updates its coordinates. Assuming the overall update interval to be KT , group A_j coordinates are updated at jT intervals where T represents the base update period. It should be noted that the coordinates of each group is assumed to be fixed during the update phase of the other groups.

In equation (1), let's partition the coordinates set, C , into K distinct subsets A_1 to A_k . Based on function $f(C)$ in (1), we define $f(A_i, A_j)$ as in (3). Note that each node must select its landmarks from the nodes in the other groups.

$$f(A_i, A_j) = \sum_{a_j \in \text{landmark set of } a_i} |d(a_i - a_j) - d_{ij}|^2 \quad (3)$$

Where a_i is an arbitrary member of group A_i that updates itself with respect to group j and a_j represents all the nodes in group A_j that are neighbors of node a_i .

At the beginning of the n^{th} step, the coordinates of the nodes in all groups are stored in vectors A_1^{n-1} to A_k^{n-1} respectively and the set of all the coordinates of the nodes are in vector A^{n-1} .

During the update phase of group A_i , all other group nodes are assumed to have fixed positions. Group A_i nodes update their coordinates to vector A_i^n such that:

$$A_i^n = \text{argmin}_{A_i} f(A^{n-1}) \quad (4)$$

Let's denote the landmark set of A_i with $L(A_i)$. In order to solve relation (4), each node A_i calculates A_i^n as follows:

$$A_i^n = \text{argmin}_{A_i} f(L(A_i)^{n-1}, A_i^{n-1}) \quad (5)$$

Since nodes in each group select their landmarks from the other group that are assumed to be fixed during their update time, the solution to the minimization problem in relation (5) can be found locally at each node. Equation (5) can be solved by various optimization methods. Here we use the Simplex Downhill method which is a numerical method used for finding the local optimum of an objective function with N variables. The method starts with forming a simplex based on an initial guess. In N dimensional space, a simplex S is a convex hull of $N + 1$ vertices. For example, in a 2 dimensional space, a simplex is a triangle. At each minimization step, this simplex is updated by applying certain transformations to it so that it "rolls downhill" until it finds a minimum. Interested readers are referred to [31] for more details on the algorithm. In order to apply the Simplex Downhill method, we use the coordinates obtained from the previous iteration as the initial solution estimation. After all the members of group A_i have performed the optimization in relation (5), the following relation will become valid:

$$f(A_1^{n-1}, \dots, A_i^n, \dots, A_k^{n-1}) \leq f(A^{n-1}) \quad (6)$$

In the next step of the algorithm, group A_i nodes are kept fixed and group A_{i+1} update their coordinates and this continues until all groups are updated. Hence,

$$f(A^n) \leq f(A_1^{n-1}, \dots, A_i^n, \dots, A_k^{n-1}) \leq f(A^{n-1}) \quad (7)$$

V. IMPROVING THE ROUGH INITIAL ESTIMATE

Similar to Vivaldi [27] and Pcoord [28], we also try to minimize the square-error function in equation (1). As mentioned before, the convergence of these systems has been demonstrated by means of simulation rather than being theoretically proven. This is in fact one of the shortcoming of these approaches. In order to overcome this issue, we have selected the majorization technique [34] to solve the optimization problem. The convergence of this approach can be mathematically proven and the accuracy of the results is also improved according to simulations.

The proper choice of landmarks in distance prediction methods is among the important issues that has been considered in many researches [24][27][28]. Before explaining the second phase, we will discuss this issue further in detail.

1) Landmark Set Selection Method.

As it was first demonstrated in PIC [24], landmark selection can affect the accuracy of a coordinate estimation algorithm. In PIC, three different strategies of landmark selection have been investigated. *Random selection* chooses the landmarks in a random way for all the nodes; this selection method has been used in the first phase of our algorithm. *Closest selection*

selects the elements of the landmark set for node n from those network nodes that are closest to n . The third method is the *hybrid selection* which selects a portion of the landmark set elements randomly and the rest of the set are selected using the closest selection method. The closest selection method has better performance in estimation of the relative position of the nodes with respect to their nearby neighbors. In contrast, random selection strategy is more beneficial for positioning the nodes relative to distant nodes. Hence, a *hybrid* strategy is preferred for achieving lower prediction errors for both short and long distances.

A problem with the use of the hybrid method is finding the nearby nodes to a given target. Many solutions have been proposed to this problem; however, most of them are costly due to the complexity of numerous distance measurements which increases the traffic overhead and the algorithm convergence time.

Although our algorithm is fundamentally different from PIC, both methods use a two-phase approach for coordinate calculation where network nodes are assigned rough coordinates in the first phase. Similar to PIC, we use the rough coordinates to estimate distances instead of measuring them through sending probe packets. For finding the k closest nodes to each node n , k random nodes are selected and a greedy walk towards the nearest node to n will start from each of these random nodes. In each step of the greedy walk, the current node sets the next destination to the closest node to n among its neighbors; if the current node is itself closer to n than to the next destination, then the greedy walk will stop and the current node will be returned as the closest candidate node to n .

2) The Iterative Refinement Process

At the beginning of this phase, each node i has coordinates $C_i = (x_i, y_i)$ which have been calculated in the first phase. We aim at minimizing $f(C)$ in equation (1) which cannot be solved locally at each node. Therefore, in our algorithm, each node i tries to locally minimize $F_i(C)$, instead of $f(C)$, as defined below:

$$F_i(C) = \sum_{j \text{ is a landmark for } i} |d(C_i - C_j) - d_{ij}|^2 \quad (8)$$

Majorization can be used to minimize our goal function in (8). This technique has been widely used in many applications, for example [33] uses majorization for the graph drawing problem: to construct the layout of a graph based on the information regarding the length of its edges. The solution to this problem generally reduces to the definition of a stress function that should be minimized. Our problem is different from the graph drawing problem in the sense that we require a completely distributed algorithm; however, the majorization technique can be modified to be fitted to a distributed environment. Here we first briefly review the majorization technique as explained in [32] and [33], and then we explain our algorithm.

In our algorithm, each node applies majorization in order to refine its coordinates. For applying majorization, another function $F_i(C, Z)$ is constructed such that it satisfies the

conditions in (9) and (10). $F(C, Z)$ should be selected in a way that its minimization would be easier than $F_i(C)$ and can give us a bound on the maximum value of $F_i(C)$.

$$F_i(C, Z) \geq F_i(C) \quad (9)$$

$$F_i(C, C) = F_i(C) \quad (10)$$

Minimization is done in an iterative manner. Here Z is the current best estimate for C . In each step, Z is assigned the value of C that minimizes $F_i(C, Z)$:

$$Z^{t+1} = \arg \min_C F_i(C, Z^t) \quad (11)$$

From relation (11), we can write inequality (12) which proves the convergence of the algorithm to a local minimum.

$$F_i(Z^{t+1}) \leq F_i(Z^{t+1}, Z^t) \leq F_i(Z^t, Z^t) \leq F_i(Z^t) \quad (12)$$

By decomposing $F_i(C)$ in Equation (8), we have:

$$\begin{aligned} F_i(C) &= \sum_{j \text{ is a landmark for } i} d^2(C_i - C_j) \\ &+ \sum_{j \text{ is a landmark for } i} d_{ij}^2 \\ &- 2 \sum_{j \text{ is a landmark for } i} d(C_i - C_j) d_{ij} \end{aligned} \quad (13)$$

Using the Cauchy-Schwartz inequality, relation (14) can be derived. More details on the proof of this is given in [32].

$$F_i(C) \leq F_i(C, Z) = \sum_{j \text{ is a landmark for } i} d_{ij}^2 + \text{Tr}(C^T L C) - 2\text{Tr}(C^T L^Z Z) \quad (14)$$

Where the $n \times n$ matrix L is defined as follows:

$$L_{ij} = \begin{cases} -1 & \text{j is a landmark for } i \\ 0 & \text{j is not a landmark for } i \\ -\sum_{i \neq k} L_{ik} & i = j \end{cases} \quad (15)$$

Also the matrix L^Z is defined as:

$$L_{ij}^Z = \begin{cases} -d_{ij} \text{inv}(d(Z_i - Z_j)) & \text{j is a landmark for } i \\ 0 & \text{j is not a landmark for } i \\ -\sum_{i \neq k} L_{ik}^Z & i = j \end{cases} \quad (16)$$

Where $\text{inv}(a)$ is defined as follows:

$$\text{inv}(a) = \begin{cases} \frac{1}{a} & a \neq 0 \\ 0 & \text{else} \end{cases} \quad (17)$$

Following the majorization technique, we find the minimum of $F_i(C, Z)$ by solving equation (18):

$$LC = L^Z Z \quad (18)$$

As mentioned before, Z is the best current estimate for C . In each step of the algorithm, the new coordinates are calculated based on Z ; so we can write:

$$LC(n+1) = L^{C(n)}C(n) \quad (19)$$

For a 2-dimensional coordinate system we need to solve the following equations:

$$\begin{cases} LC^x(n+1) = L^{C(n)}C^x(n) \\ LC^y(n+1) = L^{C(n)}C^y(n) \end{cases} \quad (20)$$

Where $C^x(n)$ and $C^y(n)$ represent the x and y coordinates of the nodes at the n^{th} iteration of the algorithm.

Node i can calculate $C_i^x(n+1)$ and $C_i^y(n+1)$ from the above system. The system can be solved locally because according to relations (15) and (16), L and $L_{ij}^{C(n)}$ equal zero if j is not a landmark of i . Note that this iterative process is performed in each node independently and asynchronously from the other nodes. By solving the equation system in (20) for x coordinate, relation (21) is derived. According to this relation, each node can update its x coordinate based on its coordinates in the previous step and the coordinates of its landmarks. Hence, each node i needs to communicate with its landmarks at the start of each iteration to obtain their positions for updating itself.

$$C_i^x(n+1) = \left(\frac{1}{K}\right) \sum_{j \in \text{landmarkSet}(i)} [C_j^x(n+1) + \delta_{ij}(C_i^x(n) - C_j^x(n))] \quad (21)$$

Where :

$$\delta_{ij} = d_{ij} \text{inv}(d(C_i^x(n) - C_j^x(n)))$$

Similarly, the y coordinate for each node i can be calculated from equation (22).

$$C_i^y(n+1) = \left(\frac{1}{K}\right) \sum_{j \in \text{landmarkSet}(i)} [C_j^y(n+1) + \delta_{ij}(C_i^y(n) - C_j^y(n))] \quad (22)$$

Where :

$$\delta_{ij} = d_{ij} \text{inv}(d(C_i^y(n) - C_j^y(n)))$$

As explained earlier, in each step of the algorithm the local function $F_i(C)$ decreases which guarantees the convergence of the algorithm.

3) Termination of the iterative refinement process

As the coordinate refinement process is performed independently at each node, the decision about algorithm termination should also be iteratively and independently made at each node. Therefore each and every node can only use its local data to determine the stability and accuracy of the estimated coordinates. In order to make this decision, each node should j collect some information about the accuracy of its current position during the algorithm run and use this information for determining the overall accuracy and stability of its coordinates. This procedure is accomplished in the following steps:

- 1- At each step of the algorithm, each node j calculates its position error with respect to all its neighbors. The error between j and neighbor i is calculated as in relation (23). In this relation d_{ij} represents the actual distance between i and j and \hat{d}_{ij} shows the estimated distance.

$$e_{ij} = d_{ij} - \hat{d}_{ij} \quad (23)$$

- 2- The average error of j position with respect to its neighbors is as follows:

$$e_j = \frac{\sum_{i \text{ is a neighbor of } j} e_{ij}}{\text{total number of neighbors}} \quad (24)$$

- 3- The average error of j position over time is defined as in relation (25).

$$\bar{e}_j = \alpha e_j + (1 - \alpha) \bar{e}_j \quad (25)$$

Where α is a constant coefficient that is set to 0.5 in our simulations. If a node whose coordinates are not accurate enough becomes j neighbor, using α in relation (25) helps preventing sudden changes in the value of \bar{e}_j .

- 4- If during t simulation steps, \bar{e}_j remains smaller than a desired accuracy threshold, j will decide to terminate the algorithm.

VI. LOCALITY-AWARE OVERLAY CONSTRUCTION

In the previous section, a distributed algorithm has been proposed for assigning virtual coordinates such that the distance between any pair of nodes could be estimated using their virtual coordinates. In this section, we will explain how virtual coordinates can be used to construct locality-aware peer-to-peer networks where logical neighbors are chosen as physically close as possible in the physical space. This is important because when the overlay is built in a way that the logical neighbors are also physically close, the message transfer time from one peer to a neighboring peer will become relatively small. This parameter is known as ‘‘per-hop latency’’. When a message passes K number of hops on its way to the destination, the overall latency will be the sum of the latencies over all those hops. Therefore, reducing the per-hop latency will result in decreased overall message transfer time.

While in our discussions we use CAN [22] as our test P2P network, without loss of generality, our proposed network construction method can be adapted for other structured and unstructured networks as well and is not limited to only CAN.

The rest of this section will describe the process of constructing a modified CAN network, which we call LA_CAN (Locality-aware CAN), such that the locality properties of the nodes are preserved meaning that the communication delay between logical neighbors are expected to be small. The key space in the CAN network is assumed to be a d -dimensional Cartesian space that is divided into different areas. Based on the ID of each node, a specific area in the key space will be assigned to each node and it will be responsible for replying to the queries that address the objects in its area.

In the CAN network, nodes are randomly assigned d-dimensional logical positions where a number of hash functions are used to produce the random d-dimensional coordinates. Each joining node generates a join request message that will be forwarded to the current owner of the area where the joining new node wants to reside. This area will then be divided between the current owner and the joining node. Assigning random positions to new nodes creates a situation where logical neighbors might be located far away from each other and will result in increased search response time and consequently will increase the traffic in the underlying network layer.

Based on our distributed method, virtual coordinates are assigned to network nodes such that close nodes in the Cartesian space are relatively close to each other in the physical layer. This idea can be used in the CAN network to select each node position in the key space such that the average distance to its logical neighbors is minimized. Since the join process in structured networks requires the exchange of traffic to update the neighbors of the joining node, it is neither beneficial nor practical to frequently relocate a node once it has joined the network. Therefore, similar to the approach used in CAN, each node in LA_CAN is initially located in a random area. The coordinate refinement algorithm will then start running; however, the location of the node will not change until the coordinates become stable. Upon the completion of the coordinate estimation phase, the node will be relocated in the network. Assume that the final result of coordinate estimation for an arbitrary node i is $SC_i = (x_i, y_i)$. In this case, i will send its join request to the area where SC_i resides. It will then leave its current location and move to the new area. This process is shown in Figure 1.

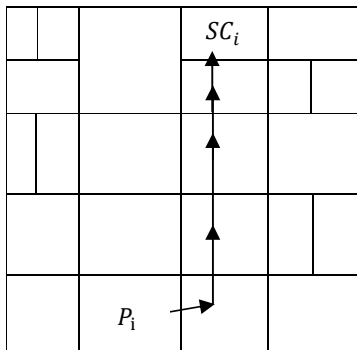


Figure 1 The position of node i in the LA_CAN network

VII. EXPERIMENTAL EVALUATION

In this section we provide a various simulation results in order to demonstrate the performance of the proposed approach. We have performed the simulations with the King Dataset from Vivaldi [27] which includes round trip latency among 1740 DNS servers. The simulations have been done using p2psim [34] which is a packet-level network simulator. We have run the simulation with different configurations and the results have been compared to some other network positioning systems.

It should be noted that for large scale P2P networks, real world implementation and testing can be costly due to large number of nodes involved in the network. Therefore, most of the works in this field use simulators to verify their design and compare it with existing systems, an approach taken in our paper as well. In addition to this verification, validation can also be used in the form of implementation. In this work, we suffice with verification and leave validation to future works.

Following the conventions in Pcoord [28] and Vivaldi [27], we define the link error as the difference between the actual Round Trip Time (RTT) and the estimated RTT. Based on the link error definition, the node error is defined as the median of the link errors for links that connect that node to other network nodes. The system error (also called the median prediction error) is defined as the median of the node errors. Also, Similar to GNP [19], we have used the relative error as an evaluation metric. For each pair of arbitrary nodes (i, j) the relative error, r_{ij} , is defined as follows:

$$r_{ij} = (d_{ij} - \hat{d}_{ij}) / \min(d_{ij}, \hat{d}_{ij}) \quad (26)$$

Where d_{ij} is the measured distance between i and j and \hat{d}_{ij} is the estimated distance based on their coordinates.

In order to determine the impact of the first phase on the convergence behavior of the algorithm, we have performed a simulation scenario where only the second phase of the algorithm is applied. (i.e. we eliminated the first phase) and compared the results with the original proposed distributed coordinate system (abbreviated as DCS). The result of this comparison is shown in Figure 2. The simulations have been done in a network of 1000 nodes. 10 reference points are used at each coordinate update and hybrid method is used for landmark selection. According to Figure 2, at the start of the algorithm and before time 5ms, the second phase performs better. However, as time passes, the convergence rate of the second phase decreases and finally it converges after the DCS algorithm. Besides, the final results of the DCS algorithm are more accurate, because the first phase can provide the second phase with a good initial guess that increases the convergence rate and in some cases can help the algorithm to escape the local minima. Also, we have used the initial results of the first phase in the landmark selection method which decreases the system error.

Figure 3 compares the accuracy of the system by exploiting different methods of landmark selection. As discussed earlier, for updating the coordinates of a particular node, choosing landmarks from nearby nodes improves the prediction accuracy for short distances but this also can result in “folding” the coordinate space where the distance between two nodes may be predicted to be short while they are far away from each other. As indicated in Figure 3, the random selection method performs better than the closest selection approach while the hybrid method outperforms the other two strategies. This is consistent with the results in [24] which claims that the hybrid strategy performs better because it improves the accuracy of short distances and also helps the algorithm to avoid folding the coordinate space.

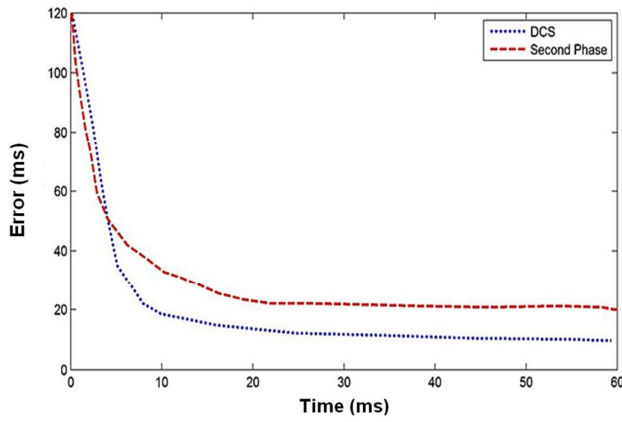


Figure 2 The comparison of System Error for DCS and DCS without applying the first phase of the algorithm.

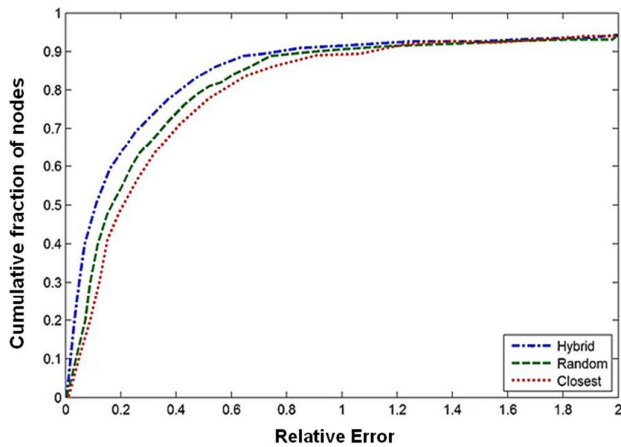


Figure 3 Cumulative Distribution Function of relative error for different types of landmark selection methods

Figure 4 compares the Cumulative Distribution Function (CDF) of relative error for Vivaldi and the suggested Distributed Coordinate System (DCS). Both Vivaldi and DCS have used 2-dimensional Cartesian coordinates and the simulation has been performed on the King Dataset. In the suggested method, each node takes 10 landmarks on each coordinate update and the selection of these landmarks is based on the hybrid method. We have configured Vivaldi to use an adaptive time step with $C=0.25$ which yields fast error reduction compared to other values of C . The results in the figure demonstrate the fact that the accuracy of Vivaldi and our approach are pretty similar for relative errors greater than 0.6. However, our proposed scheme shows better performance for relative errors that are less than 0.6.

Figure 5 and Figure 6 compare the convergence behavior of Vivaldi and DCS in terms of time and number of samples used for coordinate updates. The simulation has been performed with the same configurations as in Figure 4. According to Figure 5, DCS converges faster than Vivaldi and also achieves better accuracy. According to Figure 6, DCS needs more

samples to converge. That is due to the fact that in Vivaldi, each node takes only one sample at each time it updates its coordinates while in DCS each node takes 10 samples during each update iteration.

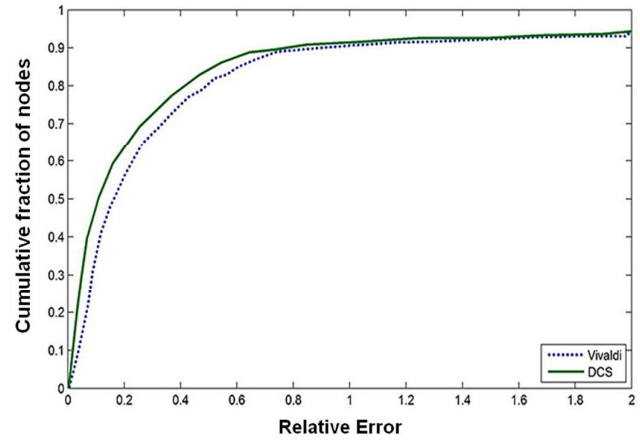


Figure 4 CDF of relative error for Vivaldi and the proposed Distributed Coordinate System (DCS)

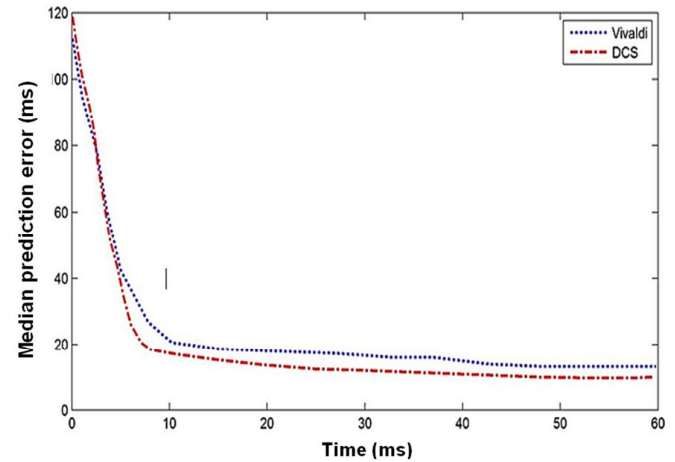


Figure 5 The convergence behavior of Vivaldi and DCS

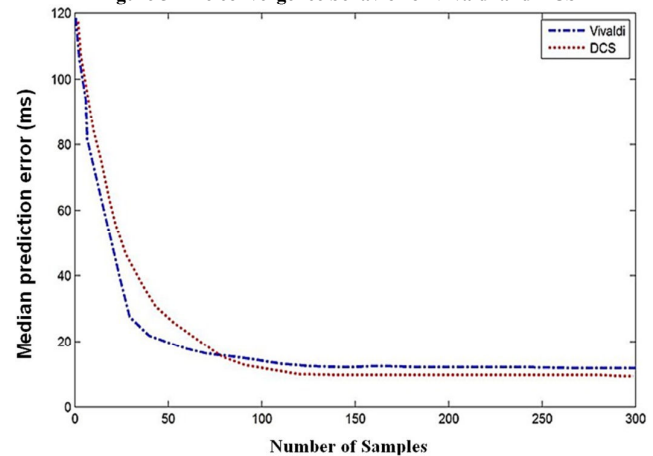


Figure 6 Median prediction error for different number of samples

In order to evaluate the accuracy of our algorithm in finding the closet neighbors, we calculate coordinates for each of the 1740 nodes in the King dataset using DCS and Vivaldi. Then we select 100 nodes as targets and find their neighbors nodes based on their coordinates. Figure 7 plots the CDF of the relative error of closet neighbor selection for Vivaldi and DCS.

As indicated, DCS performs better than Vivaldi in applications that require closest neighbor selection.

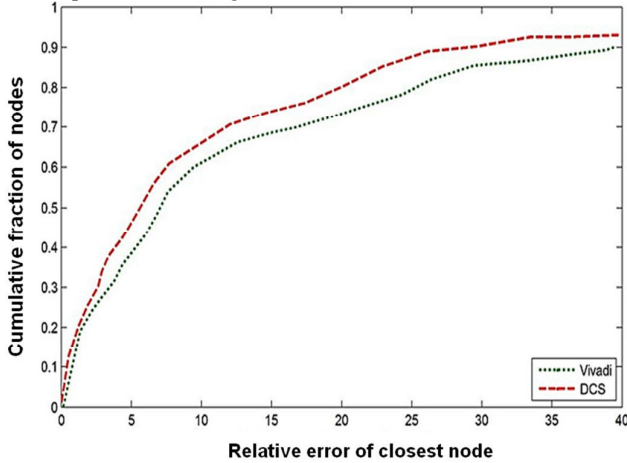


Figure 7 CDF of the relative error of closet node selection for Vivaldi and DCS

1) Application of DSC to CAN

The following simulation scenarios will demonstrate the impact of the using the proposed scheme on the efficiency of the CAN network. The simulations are performed on a network of 500 nodes where each node owns a set of objects and has the capability of requesting a new object at each simulation step. Based on the study in [35], the number of search requests sent by each node follows a lognormal distribution. The result from studying the behavior of users in different parts of the world shows that only the parameter for the lognormal distribution differs among the users. Therefore, we have chosen to use this distribution for modeling the maximum number of requests sent by each user during its lifetime.

At each simulation step, each and every node issues a search request with probability of P_r . After sending the maximum number of requests, the node leaves the network with probability of P_l in subsequent steps. Also, a number of nodes join the network at each step to keep the network size fixed.

In LA_CAN and CAN networks, the search algorithms are the same. Figure 8 shows the percentage of reduction in search traffic (which is the number of communicated messages for answering a query) for LA_CAN network in comparison with CAN network. The average of search traffic is almost the same in both of these networks as expected.

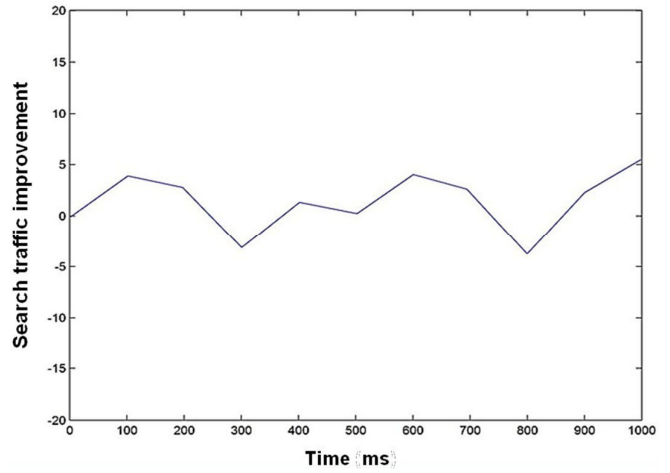


Figure 8 percentage of reduction in search traffic for LA_CAN network with respect to CAN network.

The simulation results show that the average distance between each node and its neighbors is 60 ms in LA_CAN network while it is 182ms in CAN networks, which is a significant improvement.

Figure 9 shows the query response time reduction in LA_CAN with respect to CAN. As expected, improving the network structure has significantly improved the search efficiency.

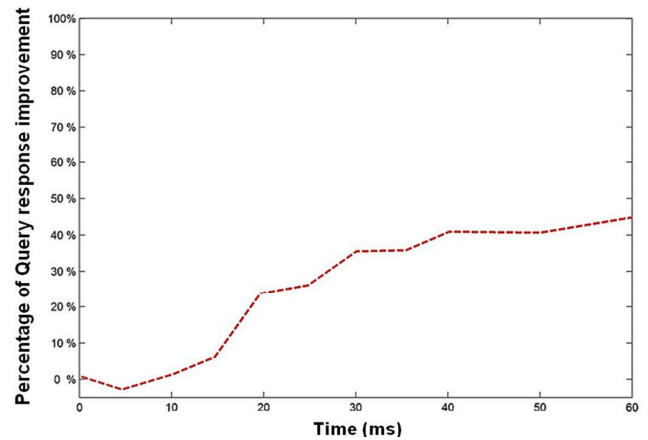


Figure 9 Query response time reduction in LA_CAN with respect to CAN network

VIII. CONCLUSION

This article addresses the problem of locating hosts over the internet in terms of physical proximity, and proposes a distributed measurement technique as a solution to alleviate the need for performing a large amount of latency measurements. The basic idea is to assign an n-dimensional coordinate to each node on the Internet such that the distance between the nodes in the coordinate space reflects their proximity over the latency space. Moreover, the proposed distributed measurement scheme does not require the use of fixed landmarks for the coordinate estimation and hence does

not need to deal with several scalability issues related to the use of fixed landmarks. The suggested algorithm has been compared to similar approaches like Vivaldi in terms of accuracy and time to stability and it was shown to outperform those methods.

Moreover, we demonstrated the use of the proposed distributed coordinate system in the construction of a locality-aware overlay for the CAN network. As observed in the simulation results, preserving locality of the overlay network greatly reduces the search response time. It should be noted that the proposed scheme could also be adapted to be used in any other peer-to-peer networks.

REFERENCES

- [1] The Gnutella website: <http://www.gnutella.com>, 2008.
- [2] B. Yang, H. Garcia-Molina, Improving search of peer-to-peer networks, In Proc. of the 22nd IEEE International Conference on Distributed Computing (IEEE ICDCS'02), pp. 5-14, 2002.
- [3] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, Search and replication in unstructured peer-to-peer peer networks, In Proc. of ACM international conference in Super computing, pp. 84-95, 2002.
- [4] A. Crespo, H. Garcia-Molina, Routing indices for peer-to-peer systems. In Proc. of 28th Conference on Distributed Computing Systems, pp. 23- 32, 2002.
- [5] D. Tsoumakos, N. Roussopoulos., Adaptive probabilistic search in peer-to-peer networks, technical report, CS-TR-4451, 2003.
- [6] M. Ripeanu, I. Foster, A. Iamnitchi., Mapping the Gnutella network: Properties of large scale peer-to-peer systems and implication for system design, In Proc. of the 1st International Workshop on Peer-to-Peer Systems, pp. 85-93, 2002.
- [7] A. Rowstron, P. Druschel, "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems," In Proceeding of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), pp.329-359, 2001.
- [8] G. Bucci and C. Landi, "A Distributed Measurement Architecture for Industrial Applications," IEEE Transactions on Instrumentation and Measurement, vol. 52, pp. 165-174, 2003.
- [9] L. Cristaldi ,A. Ferrero, and S. Salicone "A distributed system for electric power quality measurement.", IEEE Transactions on Instrumentation and Measurement, vol. 51, No.4, pp. 776-781, 2002.
- [10] L. De Vito, S. Rapuano, and L. Tomaciello, "One-Way Delay Measurement: State of the Art", IEEE Transactions on Instrumentation and Measurement, Vol. 57, No. 12, pp 2742-2750, 2008.
- [11] X.Y. Zhang, Q. Zhang, Z. Zhang, G. Song, W. Zhu, "A construction of locality-aware overlay network: mOverlay and its performance," IEEE journal on Selected Areas in Communications, Vol. 22, No. 1, pp 18-28, 2004.
- [12] Y. Liu, X. Liu, L. Xiao, L.M. Ni, X. Zhang, "Location-Aware Topology Matching in P2P Systems," In Proceedings of IEEE INFOCOM, Vol. 4, pp. 2220-2230, 2004.
- [13] J. Ledlie, P. Gardner, M. Seltzer, "Network coordinates in the wild," In Proceeding of USENIX NSDI'07, pp. 299-311, 2007.
- [14] B. Gueye, A. Ziviani, M. Crovella, S. Fdida, "Constraint-based geolocation of internet hosts," IEEE/ACM Transactions on Networking, Vol. 14, No. 6, pp. 1219-1232, 2006.
- [15] P. Pietzuch, J. Ledlie, M. Seltzer, "Supporting network coordinates on PlanetLab," In Proc. of the 2nd conference on Real, Large Distributed Systems, Vol. 2, pp. 19-24, 2005.
- [16] Y. Shavitt, T. Tankel, "Big-Bang Simulation for embedding network distances in Euclidean space," IEEE/ACM Transactions on Networking (TON), Vol.12, pp. 993-1006, 2004.
- [17] M. Castro, P. Druschel, Y.C CharlieHu, A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," Tech. Rep. MSR-TR-2002-82, Microsoft Research, 2002.
- [18] N. Hariri, B. Hariri, S. Shirmohammadi, "A Distributed Measurement System for Internet Delay Estimation", Proc. IEEE International Instrumentation and Measurement Technology Conference, Austin, Texas, USA, pp. 1556-1560, 2010
- [19] T.S.E. Ng, H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," In Proceedings of INFOCOM'02, Vol. 1, pp. 170-179, 2002.
- [20] L.Tang, M. Crovella., Virtual landmarks for the internet, In Proc. of the 3rd ACM SIGCOMM conference on Internet measurement, pp. 143-152, 2003.
- [21] Y. Chen, C. Overton and R. H. Katz, "Internet Iso-bar: A Scalable Overlay Distance Monitoring System," Journal of Computer Resource Management, Computer Measurement Group, Spring Edition, 2002.
- [22] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content Addressable Network," In Proceedings of ACM SIGCOMM 2001, Vol. 31, No.4, pp. 161-172, 2001.
- [23] H. Lim, J.C. Hou, and C.H. Choi, "Constructing internet coordinate system based on delay measurement," IEEE/ACM Transactions on Networking (TON), Vol. 13, pp. 513- 525, 2005.
- [24] M. Costa, M. Castro, R. Rowstron, and P. Key, "PIC: Practical Internet Coordinates for Distance Estimation," In Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04), pp. 178-187, 2004.
- [25] L.Tang, M.Crovella, "Virtual landmarks for the internet," In Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, pp. 143-152, 2003.
- [26] M. Pias, J .Crowcroft, S .Wilbur, T. Harris, S. Bhatti., Lighthouses for scalable distributed location, In Proc. of Int. Workshop on Peer-to-Peer systems, pp. 263-313, 2003.
- [27] F. Dabek, R. Cox, F. Kaashoek, R. Morris, "Vivaldi: a decentralized network coordinate system," In Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM), Vol. 34, No. 4, pp. 15-24 2004.
- [28] L. Lehman, S. Lerman, "PCoord: Network Position Estimation Using Peer-to-Peer Measurements," In Proceedings of the Third IEEE International Symposium on Network Computing and Applications (NCA), pp. 15-24, 2004.
- [29] Y. Chen, Y. Xiong, X. Shi, J. Zhu, B. Deng, X. Li., Pharos: Accurate and Decentralised Network Coordinate System, IET Communications, Vol.3, Issue. 4, pp. 539-548, 2009.
- [30] B. Wong, A. Slivkins, and E. Gun Sirer, "Meridian: a lightweight network location service without virtual coordinates ACM SIGCOMM Computer Communication Review, Vol. 35., pp. 85-96, 2005.
- [31] A. Mordecai, "Nonlinear Programming: Analysis and Methods," Dover Publishing, 2003.
- [32] I. Borg, P. Groenen, "Modern Multidimensional Scaling: Theory and Applications," Springer-Verlag, 1997.
- [33] E. Gansner, Y. Koren, S. North, "Graph Drawing by Stress Majorization," Proceedings of 12th Int. Symp. Graph Drawing (GD'04), pp. 239-250, 2004.
- [34] P2PSim, available at <http://pdos.csail.mit.edu/p2psim/>
- [35] A. Klemm, C. Lindemann, M. K. Vernon, and O. P. Waldhorst. Characterizing the query behavior in peer-to-peer file sharing systems, In Proc. of the 4th ACM SIGCOMM conference on Internet measurement, pp. 55-67, 2004.