RC24120 (W0611-168) November 28, 2006
Computer Science

IBM Research Report

Design and Evaluation of a Network Distance Based Planning Service

Sanghwan Lee
Kookmin University
South Korea

Sambit Sahu, Debanjan Saha
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Design and Evaluation of a Network Distance Based Planning Service

Sanghwan Lee
Kookmin University
Email: sanghwan@kookmin.ac.kr

Sambit Sahu
IBM T.J. Watson Research
Email: sambits@us.ibm.com

Debanjan Saha
IBM T.J. Watson Research
Email: dsaha@us.ibm.com

***Abstract*— In this paper, we design and evaluate the prototype of a network planning service utilizing the coordinate based embedding of network hosts. The kernel of our prototype consists of a scalable network distance embedding method, a core set of services built on top of this embedding, and a generic set of APIs exposed to the applications for utilizing these services. The implemented service core consists of four generic services that we argue to be common to a wide range of applications requiring management of services and monitoring of network distance among Internet hosts. The proposed service does not require any support from the end-hosts. We evaluate the implemented service core using real Internet data and demonstrate its efficacy. Our experience provides several key insights for the design and management of a suitable embedding scheme.**

I. INTRODUCTION

Scalable estimation of network distances among Internet hosts is the fundamental enabler for various content distribution applications such as BitTorrent, Skype, and Akamai. These applications often require scalable estimation of network distances among their serving client base to adapt resource usage and choice of services to changing network conditions. Recent approaches on scalable distance estimation embed Internet hosts into a Euclidean space based on the small amount of distance measurement from a set of machines called landmarks to the Internet hosts ([1], [2]). The network distances are estimated by the Euclidean distances between coordinates among the hosts.

Motivated by the promising results of such an approach, we propose and build a set of four distance based services utilizing the embedding approach. We believe that these core set of services can serve as the basic primitives for a large variety of CDN applications. While the services provided by the CDN/peer-to-peer based applications may differ significantly, there are lots of commonalities among the underlying service enablers. For example, the service provided by Akamai requires accurate estimate of nearest server selection for its clients for serving contents; Skype application requires the determination of relay nodes between the source and the destination; BitTorrent file distribution service requires set of peer nodes that are closer to the content downloader. While these requirements differ from one another, we argue that a few set of primitives can effectively address these requirements.

Adopting the landmark based network embedding schemes such as [1] and [2], we design a network planning service prototype, netGPS, that provides a core set of services to applications. netGPS consists of three parts : netGPS API, netGPS Core, and a set of services. netGPS API provides simple and generic APIs for applications to initialize and access its services. netGPS Core maintains all the measured distances from the Landmarks to the hosts and embedding information of the hosts. Based on the distance information from the Landmarks, the netGPS Core computes the embedded coordinates of the hosts. The coordinates are computed only when either a request arrives from an application or a change in network condition is observed among the registered hosts.

Furthermore, we evaluate the efficiency and accuracy of our prototype using the real Internet measurement data sets which consist of pairwise network distance measurements among up to 462 hosts. The main objective is to show that the generic services we propose are not very sensitive to the accuracy of the network embedding. To be specific, we run the same algorithm for the generic services with real distances and the estimated distances and we find that the difference in the results is minimal.

The paper is organized as follows. Section II provides background for co-ordinate based approaches. Section III describes the architecture of the prototype design. The algorithms for the implementation of netGPS service core is described in Section IV. Section V describes the experimental evaluation of our prototype. Finally, we present some insights for designing better embedding approach and summary in Section VI.

II. BACKGROUND AND MOTIVATION

In this section, we briefly discuss the coordinate based distance mapping of Internet hosts. Next we motivate the core set of services that we build utilizing such an approach. We believe that these set of services will serve as basic primitives that applications may use to facilitate a richer set of distance aware services.

A. Coordinate based distance mapping

The basic idea of coordinate based distance mapping is that each host has a set of synthetic coordinates for its position and the estimated distance is the Euclidean distance between the positions of the two hosts. The main difference among different coordinate based distance mapping schemes is how to assign the synthetic coordinates. GNP uses a fixed set of landmarks as the reference points ([1]). The landmarks measure the distances among themselves and assign coordinates by using simplex downhill optimization method. Basically, they assign coordinates such that the error between the actual distance and the estimated one is minimized. Then, based on the coordinates of landmarks and the measured distances from the landmakrs to a host, they compute the coordiantes of the host by using the simplex downhill method. One problem is the high computation time of the simplex downhill method. So other approaches such as Virtual Landmarks and ICS are proposed to reduce the computation time by using matrix transformation and Principal Component Analysis ([2], [3]).

Even though all the above methods use different approaches, their coordinates are based on the coordinates of the global landmarks. This infrastructure based approach may not be useful for some applications. Vivaldi method is fully distributed, requiring no fixed network infrastructure and no distinguished hosts ([4]). The initial coordinates start from the origin. By measuring the distances to small number of other nodes, they adjust their coordinates by using spring relaxation method. However, since Vivaldi requires the clients to run their algorithm, we do not consider Vivaldi for our purposes because in planning service, the clients could be any software accessing the servers so that we cannot assume any special capability of the software.

B. Core services

We propose and build four basic services utilizing the coordinate based mapping approach. These services are, namely, distance estimation between two arbitrary hosts/clients, estimated coverage of a server, server placement optimizing coverage, and scalable detection of changes in the network distance among a set of hosts. Before we describe these four services carefully, we motivate the reasoning behind our choice of these services as the building blocks for distance based planning services.

Typical content distribution services and overlay based peer-to-peer services often require the scalable determination of nearest server or peer nodes, determination of a set of leader nodes, and adaptation of such services to network changes. For example, Skype determines the best relay node between the source and the destination of a VoIP call, whereas BitTorrent based file dissemination requires the BT tracker to determine a set of nearby nodes to serve the file to the file downloader. A content provider such as Akamai is interested in scalable determination of the nearest server for a set of clients. A service provider may require to update its infrastructure such that it is able to address its changing client populations by optimizing its infrastructure placement near to the clients. Furthermore, since the network distance can change over time, they might need to monitor the distribution changes in a constant manner.

Analyzing these services, we identify a set of four services that are common among a variety of content distribution applications. We believe that if this set of services can be implemented in a generic manner, these can be served as the building blocks for designing more enhanced services. Following is the list of services that we have proposed and built in our prototype.

- Network Distance Estimation : The service provides the estimated distance between any pair of two nodes.
- Estimated Coverage of a server : The service provides the gain of placing the server to a place in terms of the clients who are within δ ms from the server. The server who has more such clients is more desirable.
- Optimization of the Server Placement : The service provides a solution on where to put the given number of servers to maximize the number of clients who are within δ ms from the nearest servers.
- A scalable and hierarchical method for detecting network changes : The service monitors the network status between the servers and clients so that the server placement can adapt to the changes.

III. NETGPS ARCHITECTURE

This section describes the architectural details of the netGPS service prototype. As mentioned earlier, netGPS utilizes a coordinate based distance mapping mechanism. netGPS consists of three parts : netGPS API, netGPS Core, and a set of services that we shall describe shortly.

netGPS API provides simple and generic APIs for applications to initialize and access its services. It should be noted that netGPS users are likely to be the service providers rather than end users. This API uses TCP communication to the netGPS core to get the service. This simple API only requires TCP connection, so it could be used in any environment. For example, mobile devices as well as the ordinary PC can use this service. netGPS Core is a server process that deals with the requests through the netGPS API. netGPS Core maintains all the distances from the Landmarks to the hosts and embedding information of the hosts. Based on the distance information from the Landmarks, the netGPS Core computes the embedded coordinates of the hosts. If the information about a host does not exists, it requests the Landmarks the measurement operation to get the distances from the Landmarks to the target hosts. Based on the distance information from the Landmarks, the netGPS Core computes the embedded coordinates of the hosts. The core services are provided in an event driven manner, i.e., computed and determined when either a request arrives from an application or a change in network condition is observed among the registered hosts.

Fig. 1 shows the overall structure of the netGPS service. netGPS Core and Landmarks are maintained by the netGPS service provider. netGPS API is used by the netGPS service users, which include the end users as well as the Internet Service Providers. There is only one type of communication between netGPS API and netGPS Core. It is that the API requests the coordinates to the netGPS Core by providing the list of hosts and netGPS Core returns the coordinates of the hosts. In the request, the API might set an option which makes netGPS Core run a forced new measurement to the listed hosts instead of returning the old measurement. For detecting network changes, the netGPS API sends the new measurement

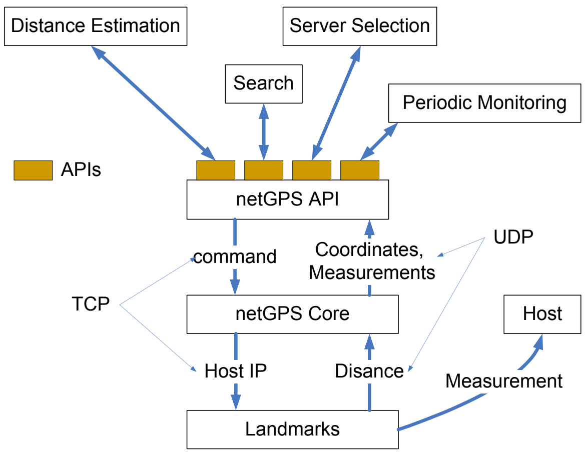


Fig. 1. Components of netGPS : netGPS API, netGPS Core, and Landmarks.

request periodically to the netGPS Core. The netGPS API provides the four services based on the coordinates of the hosts. In this architecture, all the complex operations are done on the service user's side and the netGPS Core does the simple embedding operation. This distinction between the API and Core enables a scalable service. In the next section, we describe the four services in detail.

IV. Core Service details

This section describes the algorithms and implementation design for the four proposed services.

Distance Estimation :

The estimation is based on the coordinates of the hosts returned from the netGPS Core. The Euclidean distance is the estimated network distance. The usefulness of the network distance estimation service has been argued by a large number of papers. In the planning service, the absolute accuracy of the network distance is not required due to the service objectives we describe later in this section.

Estimated coverage of a server :

A server can serve a large number of clients existing in a wide range of areas. The distribution of the network distances from the clients to the server is a metric to show the usefulness of the server. However, sometimes, the detailed distribution of the network distances may not be necessary. For example, in network games, the observed behavior of the server is non distinguishable when the response time is within a certain bound. So in this case, the coverage (i.e. the percentage or number of satisfied clients served by the server) of a server is a more meaningful metric to assess the usefulness of the server. Furthermore, it can reflect the distribution in some sense. So we claim that providing the estimated coverage of a server would be useful to many service providers. The computation is straightforward. From the coordinates of the clients and the server, we compute the number of clients that is within δ ms from the server. Depending on the application, we can provide the percentage or the absolute numbers.

Optimization of the Server Placement : Many service providers place multiple servers in the Internet to better serve their users. One possible criteria to select the right place is to use the demographic information of the area so that an area with a large population would get more servers than that with a small population. However, we think that a more systematic approach is needed for the server placement due to many reasons such as the locality. netGPS uses a simple greedy algorithm for this purpose. We first compute the estimated distance among all the hosts. For each host, we compute the number of hosts within δ ms from the host. We select the host with the largest number. Then, we remove the selected host and all the hosts within δ ms from the selected host. We repeat the procedure until we select the given number of hosts. Then we can put the same number of servers at the same location of the selected hosts.

A Scalable and Hierarchical Method for Detecting Network Changes : Monitoring the status of the client network is necessary for maintaining the performance of the deployed service. We describe a scalable and hierarchical method for quickly detecting network anomaly, i.e., large changes in the network delay using scalable end-to-end approach. Current Internet routing protocols actually do link status monitoring and adjust the routing table to detour the failed links. However, the routing change information of a remote network does not usually propagate to the other part of the Internet because of the BGP policy routing. For some applications like overlay networks or p2p networks, the number of nodes are very large and widely distributed. So it is quite difficult to quickly know the network status around remote nodes. One straightforward way is to monitor the interested nodes continuously. However if the number of interested nodes is high, it might not be scalable to do this way. We propose a scalable adaptive network monitoring scheme.

The basic idea is to group the target nodes into a number of clusters so that the network status change of a node implies the network status change of other nodes. Since the virtual coordinate system provides a method to estimate the distance among hosts, we can apply clustering algorithms such as K-mean to group the target nodes. Next, we select a node from each cluster and then monitor only the selected nodes. The monitoring requires the landmarks to periodically measure the distances from themselves to the selected nodes. Then new virtual coordinates for the selected nodes are computed. If the position of the selected nodes changes more than a threshold, it is likely that the network status around the node might also change. So we can further investigate the group that the node belongs to. We do not monitor all the nodes in the group. We subdivide the group into several subgroups by applying some clustering algorithms and then choose the centroid of each subgroup, then get the network status of the centroids again. If some of the centroids have changed their positions, we further subdivide the subgroups.

By the scheme we described above, the nodes with network status change can be identified. We call this set of nodes *identified nodes*. However, we do not know which area are actually the problematic part in the Internet. To get more detailed information, we use the traceroute data. Assume that we have the traceroute data from the landmarks to all the target nodes. We first compute the set of segments involved in the traceroute data. A segment is a consecutive set of links at which no branch is attached. Then, for each segment, we iterate the following steps.

(i) Choose one segment. (ii) Suppose the segment has increased its delay. (iii) List all the target nodes that would be affected by the delay increase of the segment. (iv) If the set of selected target nodes are *subset* of the *identified nodes*, the segment is a possible candidate which causes the network status change. If the set is a *superset* of the *identified nodes*, the segment cannot be a problematic node. (v) Do this iteration for other segments.

The set of segments selected in the above step is the candidates which might have caused the problem. Among the selected set of segments, we choose the smallest subset of segments that cause the changes of the *identified nodes*. We conclude that this smallest subset of the segments caused the network changes of the *identified nodes*. One heuristic in this algorithm is that the cause of network changes should be as small as possible.

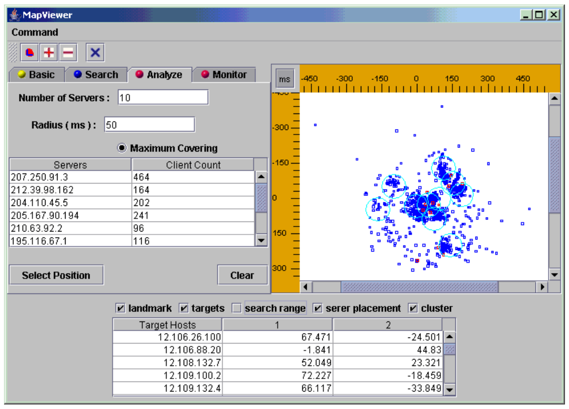
V. Preliminary Evaluation

In this section, we briefly describe the prototype system of netGPS. Then, we evaluate the efficacy of netGPS by showing the experiment result done with the prototype system.

A. Prototype Implementation

The prototype contains three different programs : Landmarks, netGPS Core and, MapViewer. MapViewer is an application that uses the four netGPS APIs shown in Fig. 1. netGPS APIs and MapViewer are implemented in Java. We will implement C coded netGPS APIs in the future work. Landmarks and netGPS Core are implemented in C and running on unix. *netGPS Core* does several tasks before it accepts requests from the netGPS APIs. One of them is to get the distances among the landmark nodes to compute the coordinates among the landmarks. In this prototype, the GNP method is used for the coordinate computation. The request from the netGPS API contains an option whether to run new measurements or to use the old coordinates. If the option is 0, it means that *netGPS Core* should return, if any, the old coordinates of the given hosts. If the option is 1 or there is no old coordinates of a host, new measurement is launched through the landmarks to the host. For the clustering operation in the scalable monitoring service, we implemented k-medoid algorithm. Other algorithms (such as k-mean) can be used without large changes.

Fig. 2 shows a snapshot of MapViewer, which is currently using the ”Optimization of the Server Placement” operation. Users can set the number of servers and the radius. The squares in the map represent the first two coordinates of the hosts. The circles show the coverage of the selected servers. As can be seen in Fig. 2, the dense area has more circles, which means more servers to be placed.



Number of Servers : 10

Radius (ms) : 50

(●) Maximum Covering

Servers	Client Count
207.250.91.3	464
212.39.98.162	164
204.110.45.5	202
205.167.90.194	241
210.63.92.2	96
195.116.67.1	116

✓ landmark ✓ targets ☐ search range ✓ serer placement ✓ cluster

Target Hosts	1	2
12.106.26.100	67.471	-24.501
12.106.88.20	-1.841	44.83
12.108.132.7	52.049	23.321
12.109.100.2	72.227	-18.459
12.109.132.4	66.117	-33.849

Fig. 2. A snapshot of MapViewer, which is running the ”Optimization of the Server Placement” operation.

One practical problem in the real experiment of the system is that we need to carefully place the Landmarks in the Internet to increase the accuracy of the embedding ([1]). Due to the limitation of machines we have, we use King ([5]) method instead of the Landmarks program for the real Internet experiments. In the experiment, we select 20 DNS servers as landmarks and measure the distances from the set of DNS servers to other DNS servers considering them as end hosts by King method. It should be noted that King method is relatively accurate for the distances among the DNS servers than estimating the distances among arbitrary hosts. So considering DNS servers as end hosts provides the same result as we place Landmarks on the 20 selected DNS servers and let them measure the distances to the other DNS servers. However, we can still use the current prototype for end hosts with a small loss of accuracy. Furthermore, this replacement does not affect the behaviour of the MapViewer because it only communicates to the netGPS core. The list of DNS servers is obtained from the King data set provided by Vivaldi research group ([6]).

B. Performance Evaluation

The performance of distance estimation has been analyzed by many papers such as [1], [2], and [3]. We would refer to their papers for more detailed evaluation information. Nonetheless, based on their result, the accuracy of the distance estimation is acceptable in terms of relative errors.

To evaluate the performance of Optimization of the Server Placement operation, we run simulation on the real Internet measurement data obtained from two independent measurement groups : Planelab ([7]) and King data set ([6]). The two data sets provide pairwise network distance among 202 and 462 hosts. In the Optimization of the Server Placement operation, we want to maximize the number of hosts within a given range from the nearest server.

Fig. 3 shows the fraction of hosts that is covered by the selected servers in a given range over different ranges. For a given range, we select 10 servers that can maximize the number of hosts that are within the range. Then, the hosts

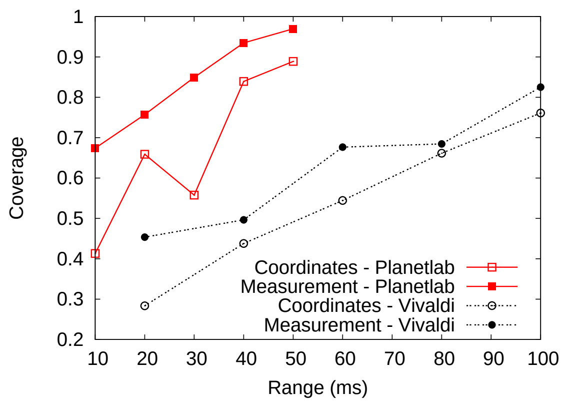


Fig. 3. Fraction of hosts within the given range from the nearest server over different ranges.

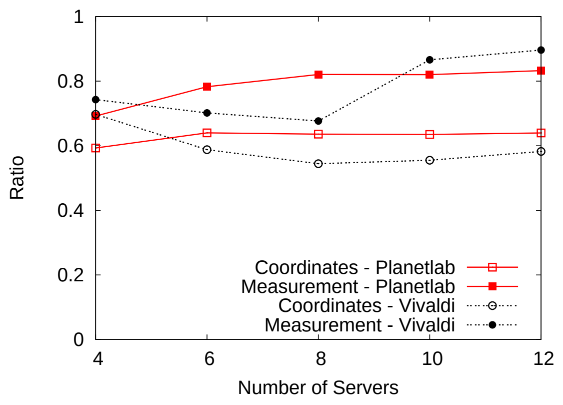


Fig. 4. Ratio of the coverage with Euclidean distances over the coverage with the real distances over the number of servers.

select the nearest server. There are two different ways to select the nearest server. One is to select the nearest one in the coordinate space. The other is to use extra measurements from each host to the 10 servers and choose the nearest one. For both ways, we compute the fraction of hosts that are covered by the servers in the given range. As the range increases, the coverage increases. When the nearest server is selected only based on the coordinates (Coordinates-Planetlab and Coordinates-Vivaldi), the coverage is lower than when the nearest server is selected by the measurements from the hosts to the 10 servers (Measurement-Planetlab, Measurement-Vivaldi).

In the next experiment, we show the performance of the server selection compared to the one that uses the real distances instead of the Euclidean distances. We compute the number of hosts within δ ms from the nearest servers in both cases and compute the ratio of the two numbers. The nearest servers are selected in two ways as described above. With the fixed range (50ms), we vary the number of servers. As can be seen in Fig. 4, the ratio stays almost constant about 0.6 to 0.9. It should be noted that in this experiment, the algorithm to select the given number of servers is the same, the greedy approach. Only the distances used in the algorithm are different. The result might be different if we use different algorithms, however, this experiment shows the performance of using estimated distances comparable to the real distances.

VI. Observations and Summary

The netGPS planning service highly depends on the accuracy of the network embedding scheme. Most of the Euclidean embedding scheme shows problems in estimating close distances. However, as shown in the evaluation, our service can bypass some of the problems of the embedding schemes. For example, in the server placement service, the inaccuracy of the small distances does not affect the performance much especially when we use the real measurement after the selection of the servers.

Nonetheless, good embedding scheme can improve the performance of our services. For example, the Euclidean assumption on the network distances causes most of the errors as can be seen in [8]. To provide some relaxation on the Euclidean assumption, we may want to add height vector. Since our service does not depend on the Euclidean assumption, any accurate distance estimation scheme can be used instead of GNP, which is currently adopted in our prototype. In addition, we only need to know whether the hosts are within a given radius from the servers. One way is to use the prefixes or AS numbers. In our experience, a pair of nodes with the same prefix sometimes has very large estimated distances. In this case, using prefix number may be useful to remove such false negatives.

In this paper, we proposed and prototyped a network distance based service utilizing coordinate based embedding of network distance. We argued that the four services that netGPS provides can be extremely useful and serve as primitives for facilitating the distance based service requirements of a large number of content distribution applications. Using real measurement data among a large number of Internet hosts, we evaluated the efficacy and accuracy of our prototype. Future work shall focus on balancing the load by adopting a distributed service infrastructure.

References

[1] T.S. Eugene Ng and Hui Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proc. IEEE INFOCOM*, New York, NY, June 2002.

[2] Liying Tang and Mark Crovella, "Virtual landmarks for the Internet," in *Proceedings of the Internet Measurement Conference(IMC)*, Miami, Florida, Oct. 2003.

[3] Hyuk Lim, Jennifer C. Hou, and Chong-Ho Choi, "Constructing Internet coordinate system based on delay measurement," in *Proceedings of the Internet Measurement Conference(IMC)*, Miami, Florida, Oct. 2003.

[4] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris, "Vivaldi: A decentralized network coordinate system," in *Proceedings of ACM SIGCOMM 2004*, Portland, OR, Aug. 2004.

[5] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble, "King: Estimating latency between arbitrary Internet end hosts," in *Proceedings of Internet Measurement Workshop 2002*, Marseille, France, Nov. 2002.

[6] "King462 data set," http://pdos.lcs.mit.edu/p2psim/kingdata.

[7] Jeremy Stribling, "Rtt among planetlab nodes," http://www.pdos.lcs.mit.edu/ strib/pl_app/.

[8] Sanghwan Lee, Zhi-Li Zhang, Sambit Sahu, and Debanjan Saha, "On suitability of euclidean embedding of internet hosts," in *Proc. ACM SIGMETRICS*, Saint Malo, France, June 2006.