Technical Report

UCAM-CL-TR-576
ISSN 1476-2986

Number 576

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

An implementation of a coordinate based location system

David R. Spence

November 2003

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/TechReports/

Series editor: Markus Kuhn

ISSN 1476-2986

An Implementation of a Coordinate Based Location System

David Spence
University of Cambridge Computer Laboratory
J J Thomson Avenue, Cambridge, UK, CB3 0FD
David.Spence@cl.cam.ac.uk

Abstract

This paper explains the co-ordinate based location system built for XenoSearch, a resource discovery system in the XenoServer Open Platform. The system is builds on the work of GNP, Lighthouse and many more recent schemes. We also present results from various combinations of algorithms to perform the actual co-ordinate calculation based on GNP, Lighthouse and spring based systems and show our implementations of the various algorithms give similar prediction errors.

1 Introduction

XenoSearch [13] is the initial Resource Discovery (RD) system which is being developed for the XenoServer Open Platform. For a full explanation of the XenoServer Open Platform see the companion paper [6].

The XenoSearch system is required to locate the best possible XenoServer to run a particular task on, out of the approximately ten thousand envisioned XenoServers. The primary motivating attribute in a search is location – we wish to run tasks at remote network positions, either close to resources we wish to use or at an interesting point in the network or to minimise the maximum latency to a set of hosts using a collaborative tool.

To this end we have developed the `ucam.location` classes. We take up a co-ordinate based approach to network locality, as first suggested by GNP [10]. This approximates the network topology as a d-dimensional Euclidean space. First a basis is set up between more than $d + 1$ Positioning Servers. This basis is then used to calculate the position of arbitrary hosts by measuring the latency to that host from $d + 1$ of the Positioning Servers, and then the approximate position of the host with respect to the basis is calculated.

For XenoSearch we may have hosts not participating in the overlay to deal with, so entrusting the measurements and calculations to the measured hosts can not be used, as it was by GNP and lighthouse [11]. Instead the Positioning Servers measure the latency, connected by an overlay network, and we introduce a load balancing system to make sure Positioning Servers do not get overloaded. We allow Positioning Servers to join and leave freely: the location system is still viable, as long as there have always been at least $d+1$ Positioning Servers in the overlay since it was created.

We envision this system to be deployed as part of the XenoSearch system, with each XenoSearch node running a Positioning Server as a background task. Each time the XenoSearch node needs to find a host's position, it contacts its local Positioning Server, which contacts d other Positioning Servers which make measurements and return their latency measurements and their own location. The local Positioning Server can then calculate the host's location, and return it to the XenoSearch node.

The code for this location system is available free on request from the author.

The following Section explores related work, Section 3 explains the various calculation engines we have implemented, Section 4 briefly describes the overlay we use as a base for the co-ordinate based location service and Section 5 how this integrates with the higher level location service to perform load balancing. Section 6 details the interface that is presented to the user of the location system, Section 8 presents an evaluation of the calculation engines, Section 8 details the ongoing work and Section 9 concludes.

2 Related Work

The work in this area is based on the assumption that network latency is a quantity that is not constant over long timescales and this is backed-up by work such as [18]. Given this assumption co-ordinate based location systems make sense.

The idea of co-ordinate based location systems was first put forward in the GNP [10]. In this case

Simplex Downhill Error minimisation was used to first construct a basis and then using that basis to find the positions of hosts relative to that basis. This was shown to have better average predictions of latency than IDMaps [4].

Lighthouse [11] was a development of the GNP idea which seeks to solve a number of the scalability issues of the GNP system by allowing any previously positioned server to act as a "lighthouse". Further scalability issues have been addressed with the SCoLE system [15], which introduces clustering on the BGP Address Prefix and does not require the participating hosts to agree on a global basis.

Virtual Landmarks [16] is a further development of the basic GNP idea. In Virtual Landmarks a Lipschitz transformation is used instead of a mapping to Euclidean Space. A Lipschitz transform is performed by simply using the distance to 20 landmarks as the co-ordinate vector. This high dimensional vector is then reduced to 7-9 dimensions by a global transformation calculated by Principle Component Analysis (PCA). Similarly [8] also uses PCA to extract topological information from the distance vector.

The simulation of physical springs to calculate relative network positions, with the rest length of the springs being the observed latency has been used to place nodes on overlays [17]; we use this for co-ordinate calculation as was also suggested in [2, 3]. The spring model was also used as a first stage for the work on Big Bang simulation [12], but this scheme also goes on to use more complex vector fields than the one produced with the spring model.

The most similar work to ours is the PIC system [1], which was developed to allow more efficient building of Pastry overlays. It is similar to lighthouse but also, like our work, incorporates a P2P overlay so the other peers can be found.

Location systems do not have to be co-ordinate based: a number of other systems have been proposed; IDMaps [4] has a network of tracers which monitor the latency between themselves, uploading this to a HOPS server which calculates a virtual topology for the network. A query for a distance between two hosts is then satisfied as the latency from the Address Prefix of each host to the nearest tracer and the distance in the virtual topology. King [5], alternatively, uses authoritative DNS name servers close to hosts to estimate the latency between two hosts, by bouncing a recursive query of one DNS server to the other.

3 Calculation Engines

There are two main calculations which need to be made within the locating system. Firstly we need to calculate a basis (a set of $d+1$ locations for the initial $d+1$ Positioning Servers) from a matrix of $d+1$ latency measurements between the initial Positioning Servers. This is performed by the `BaseSystemCalculation` class. Secondly for each new Positioning Server, each recalculation of a Positioning Server's location and each host location measurement, we need to calculate a single location from $d+1$ locations and latency measurements. This is performed by the `NewPointCalculation` class.

Many algorithms for these problems are described in the papers mentioned in Section 2. In our work we evaluate a number of these and also look how combinations of them perform. The major difference between the GNP and Lighthouse approaches is that GNP uses a single global basis, while Lighthouse uses many local bases. In our system, we use one basis, but do not rely on a single set of nodes to define it.

3.1 `BaseSystemCalculation`

We have implemented a number of different algorithms for performing the Basis Calculation, each with their own strengths and weaknesses. These methods can be combined to reach the required speed-accuracy trade-off. Each take a starting point as a assignment of each co-ordinate. The default is the origin – as in the Big Bang paper [12].

3.1.1 Exact Calculation (Lighthouse)

This method is often used to calculate the initial global basis; we take the matrix of latencies and try to calculate, by simplified Gaussian elimination, an exact set of co-ordinates if they exist. This system is based on the observation that we have many more degrees of freedom than constraints, so we can set the first Positioning Server's co-ordinates to $(0, ..., 0)$, the second to $(0, ..., 0, x_1)$ and so on. Therefore the system becomes trivial, with the system already being in the triangular form.

In the majority of cases there is no exact system available, therefore we iteratively change the co-ordinates toward a Euclidean set, shortening the longer sides of 'triangles' of latencies and lengthen the two shorter sides. As all the sides of these 'triangles' are also the sides of other 'triangles' we are careful to weight the changes on the distance

of the 'triangle' to a valid Euclidean triangle, and consider clashes for any particular side.

As with GNP and Lighthouse, we expect a relative error of less than 25% for 95% of latencies and less than 5% for 70% of latencies. Therefore it is in the main part not far to a valid Euclidean set and so this approximate method performs well, as it is only calculating a start point for a gradient descent method.

The Exact Calculation method always passes its results through a very simple gradient descent method based on Newton's Method. The hope for this method is to find a good start point near the global minimum, to prevent the error-minimisation algorithms getting stuck at a local minimum. In fact this is a separate module and can be use as an alternative to Simplex Downhill.

3.1.2 Damped Spring Simulation

Recently topology-aware overlays have been built which use simulation of physical springs to find co-ordinates relative to other nodes in the overlay [17].

In these systems the round-trip time (RTT) is measured between the tracers calculating the basis and these are used to set the rest length of the springs between all the basis tracers. The simulation is allowed to run and the co-ordinates of the settled system can be read off at the end of the simulation. We wait until the total acceleration has reached a value below a certain threshold (ie there is no force on any of the "particles").

3.1.3 Simplex Downhill (GNP)

The Simplex Downhill Algorithm [9] is a versatile multi-dimensional error minimisation method. It tends to produce good results in a quick time, although it is suggested to restart the method about three times to check that the minima has been found. This is how the GNP paper performs error-minimisation. As we use a very simple non-optimised version we chose to re-run the algorithm until we get exactly the same results twice. This can sometimes take many repeats so we cap this. The maximum number of times we allow repeats gives us a parameter using which we can trade off the maximum execution time and the likely accuracy of the results.

3.2 `NewPointCalculation`

Similarly to calculate the position of a host we have several methods which can be combined to achieve the required trade-offs.

3.2.1 Exact Calculation (Lighthouse)

The exact method is largely as described in the Lighthouse paper, it relies on the solving of the set of equations:

$$\begin{cases} c_1\|\mathbf{l_1}\| + \cdots + c_k\|\mathbf{l_k}\|cos(\widehat{\mathbf{l_1},\mathbf{l_k}}) = \|\mathbf{n_j}\|cos(\widehat{\mathbf{n_j},\mathbf{l_1}}) \\ \vdots \\ c_1\|\mathbf{l_1}\|cos(\widehat{\mathbf{l_1},\mathbf{l_k}}) + \cdots + c_k\|\mathbf{l_k}\| = \|\mathbf{n_j}\|cos(\widehat{\mathbf{n_j},\mathbf{l_1}}) \end{cases}$$

The lengths $\mathbf{l_n}$, are the calculated vectors from the nominated 'origin' Positioning Server which is the origin of the local basis to each of the other Positioning Servers. $\mathbf{n_j}$ is the position of the host which we are calculating. $\|\mathbf{n_j}\|$ is the measured latency from the origin. The solution is the set c_i and is used to calculate $\mathbf{n_j}$:

$$\mathbf{n_j} = c_1\mathbf{l_1} + \cdots + c_k\mathbf{l_k}$$

We then add in the position of the origin Positioning Server in the global basis.

As we use the cosine rule without converting to an angle, we get a result, even if there is no valid one.

3.2.2 Damped Spring Simulation

For `NewPointCalculation` we model these hosts and our new hosts as points connected by Springs. The hosts already in the system are fixed and the new host is freely movable. The springs between this host and the others has a rest length of the RTT of that link. The system is then simulated until it comes to rest (the total acceleration is below a threshold).

3.2.3 Others

As with the `BaseSystemCalculation` we have both a Downhill-Simplex and simple error-minimisation methods also implemented.

3.3 Error Function

We tried out two error functions:

$$\sum_{i=0}^{k+1}\left(\frac{measured_i - estimate_i}{measured_i}\right)^2$$

$$\sum_{i=0}^{k+1}\left|\frac{measured_i - estimate_i}{measured_i}\right|$$

the first being the error function used in the GNP paper. These are known as "Relative Squared" (RelSqr) and "Relative Absolute" (RelAbs) in the paper. The Relative Absolute function is always used in the calculation of the basis.

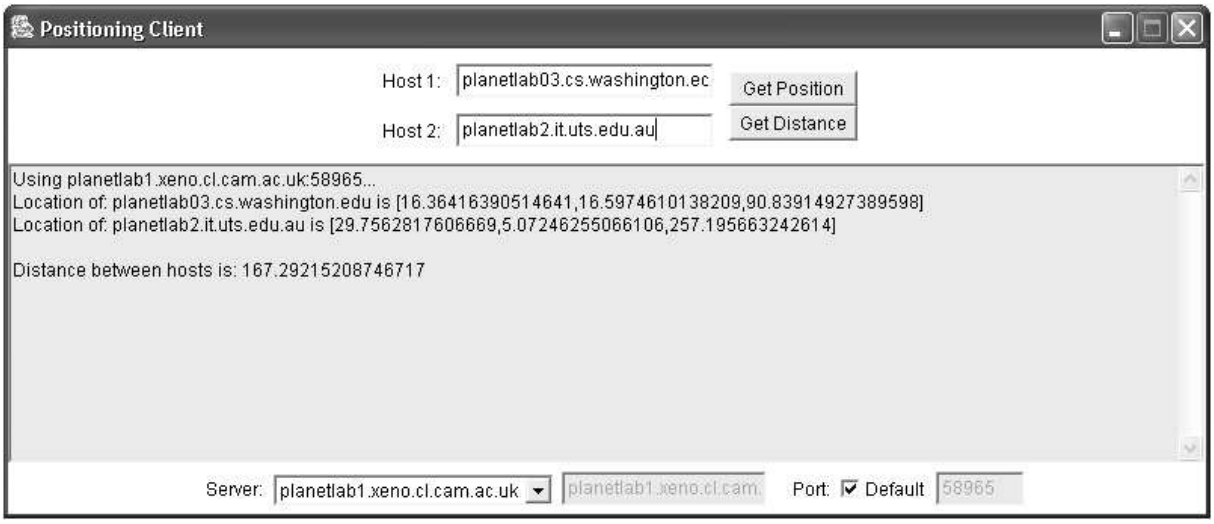

Figure 1: The Client Interface

```
[cambridge6@planetlab03 location]$ ./ping -c 5 -q planetlab2.it.uts.edu.au
PING planetlab2.it.uts.edu.au (138.25.15.195) from 128.95.219.194 : 56(84) bytes of data.

--- planetlab2.it.uts.edu.au ping statistics ---
5 packets transmitted, 5 received, 0% loss, time 4044ms
rtt min/avg/max/mdev = 157.542/158.275/159.084/0.588 ms
```

Figure 2: Same query using ping

4 Structure of Overlay

We have developed a simple peer-to-peer overlay. It stores the IP address and port (UDP) of every other peer in the system. This has been been developed as a quick, simple and lightweight DHT, for situations when we have few nodes in the system. The location classes use this overlay to keep track of the other Positioning Servers in the system and to communicate.

Each node in the overlay has a unique 128bit nodeId. The system models the key space as a ring, with nodes managing the key space after it's identifier, up to the next one, much in the same way as Chord [14].

4.1 Joining

A node joins the overlay by contacting a node already in the peer-to-peer system. From this peer we retrieve the list of all the peers in the system and their addresses (called its *routing table*). This is then used to calculate the new nodes predecessor. The new node sends a join message to this node (which could be forwarded on several times round the ring, to support concurrent joins). The predecessor node adds the new node to its routing table and at this point the node has joined the network. It replies with a JoinAck message, this may contain a new routing table entry if the Join message went via another message. Finally in the background the new node notifies all the other nodes in its routing table of it's presence.

4.2 Routing within the Overlay

Routing is simply a matter of consulting the local list of active nodes and sending to the appropriate node. Nodes can request an acknowledgement to check the receipt of the message otherwise the overlay automatically resends.

There is no guarantee that a routing table is complete, but a message sent to the wrong node is always sent to a predecessor of the correct node and as long as the node knows about the successor it will be forwarded on and the sending nodes tables corrected.

Using this method it is a very short time before every node knows about a new node.

4.3 Node Death

When a node dies, it is not noticed until a node tries to send it a acknowledged message (there is no need to actively probe as this consumes bandwidth). If any node, A, detects that another

node, B, is not acknowledging the messages it sends to it, A removes B from its routing table and sends background messages to all the other nodes it knows about. After this it retries the messages, sending it to the node that is now responsible for that area of the key space.

4.4 Rate Limiting

The traffic from nodes is shaped: ordinary traffic is passed through a token bucket rate limiter, before being put on a per remote host queue.

Background traffic is sent at a constant rate of 5 packets per second.

5 Distribution and Load Balancing

Within the location system there are two factors which affect the choice of which Positioning Server we use to sample a host's distance:

- Widely distributed locations of probes, to enable a good estimation
- Load balancing: we do not wish a Positioning Server located in a sparsely populated area to be involved in every query.

To this end we register each Positioning Server on the overlay with a topologically dependant nodeId. We know the position of the Positioning Server and we normalise each dimensions of this to a number in the range $[0, 1]$. We then use the top $\lfloor 128/d \rfloor$ bits (where d is the number of dimensions) for each dimension as the input to a Hilbert-transformation to a single $d * \lfloor 128/d \rfloor$ bit integer. This is used as the top bits of the 128 bit nodeId. The algorithm used is the one described in [7].

The Hilbert mapping gives clustering: when points are close on the Hilbert curve they tend to be close in real space. In fact the clustering is such that the d-dimensional space is split into 2^d equal hypercubes and the curve totally fills one of these before going on to the next. Further, in each of the hypercubes the curve has identical behaviour. This can be seen in the simple case of the two dimensional version shown in Figure 3.

Therefore we split the curve (or entries in the routing table) into 2^{d*c} equal length segments, where c is a variable cluster factor (2–3 in the current system). We then have values for the number of Positioning Servers in each of the clusters and so choose $d+1$ of the clusters, with the probability for each cluster weighted by the number of items in it and then chose a random Positioning Server in each of the chosen clusters. Hence we have the $d+1$ probes and on average each receives the same amount of traffic and we do not normally choose servers which are close to one another. If there are not enough clusters with Positioning Servers in then we revert to choosing $d + 1$ severs uniformly, if there are not $d + 1$ servers the overlay is dead and needs to be restarted.

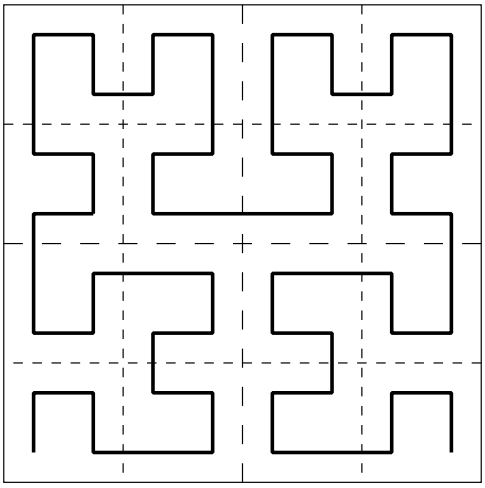

Figure 3: 2D Hilbert curve example

6 Client Interface

The client interface is simply an advertised TCP port. Clients send the line: `FindHost` followed by the IP address on the following line. The Positioning Server responds as follows:

- `LocationReturn`: followed on the next line by the calculated co-ordinates of the host
- `ErrorDead`: the systems no longer has enough servers.
- `ErrorNoOverlay`: the system has not finished being set up.
- `ErrorNoHost`: the Positioning Server can not resolve the host name or ping the host.
- Disconnect: unknown error.

When a Positioning Server wishes to join the overlay, it opens a connection to the same port and sends a `FindHostNewServer` and its IP address on the next line. the possible responses are `ErrorDead`, `ErrorNoHost` or Disconnect as above or:

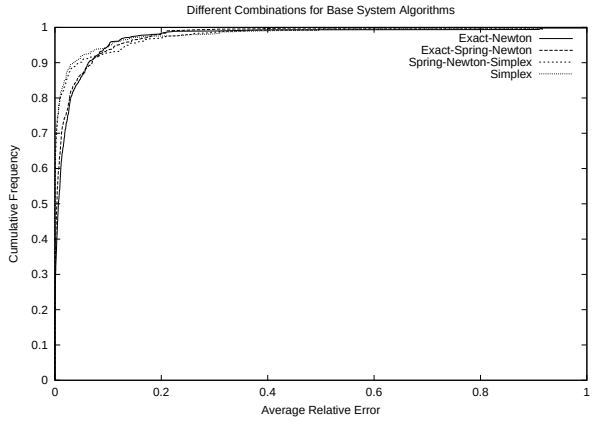

Figure 4: Accuracy in Extrapolation: Comparing various methods

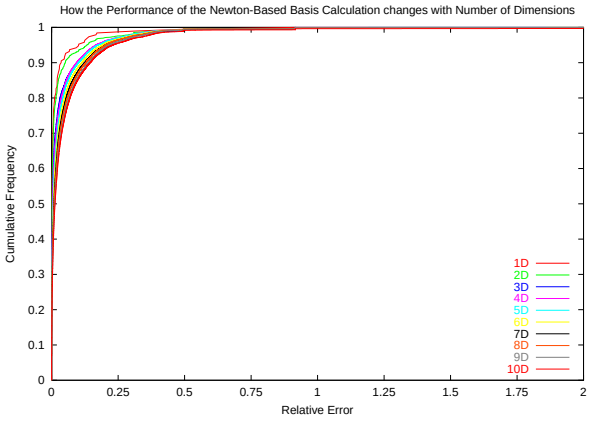

Figure 5: Accuracy in Extrapolation: How it changes with Dimensions (Note the x-axis is not at y=0)

- `ServerLocation`: followed on subsequent lines by the position and an overlay boot-point. The boot-point returned should be the Positioning Server Topologically closest to the new server.

- `ErrorNoOverlay`: the system has not yet been set up – in this case it is followed by the IP and port address of the server to contact to help to create the overlay.

To help in the overlay creation a server opens a TCP connection to the given port and sends `HelpInit`. If there is already enough servers then `HelpObtained` is returned and the server backs off for a few seconds and then tries to join normally. Otherwise `WaitForInfo` is sent, then after a delay `GetDistanceSet`, with the number of dimensions and the hosts to probe. The server pings these hosts, replying with `DistanceSetReturn`, and the latencies. The co-ordinating server then calculates the Location of each server and sends each server a `JoinOverlay` message with its location and where to bootstrap for the overlay.

6.1 Client

We have written a simple client, whose display is shown in Figure 1, with the equivalent ping shown in Figure 2. This client simply uses the text based TCP interface to query a chosen Positioning Server, which returns a location. The distance between the two hosts are then calculated. For this example we set up a three dimensional basis (for which we need $d+1 = 3+1 = 4$ servers to calculate the basis) with one server in England, one the Western coast, one on the Eastern and one in the centre of the United States. With this set up we queried the distance from a server in Washington to one in Australia and then performed the equivalent ping. The value was 167.292 for our system and 158.275 for the ping a 5% error. All the servers used were part of the PlanetLab[1] testbed.

7 Results

In these results we present a sub-set of the results we have obtained, they seek to show that the various algorithms give similar results to each other and to GNP. This paper as a whole seeks to give an overview of the practical system we developed to perform co-ordinate based location, we wish to show it is a frame-work that many algorithms can integrated. The data we use is the GNP data used in both the GNP and lighthouse papers.

7.1 Basis Calculation

Figure 4 compares various combinations of algorithms and how well they fit the model. These results use every possible permutation of 4 tracers from the GNP set and then plot the Cumulative Frequency of the relative error between the measured distance and the basis model created. The two algorithms which in the last case use the Simplex algorithm perform roughly the same as do the two having the Newton steepest slope as the last step. The Newton step seems to do best to reduce the really large outliers where the Simplex seems to overall increase the percentage of errors which have zero error (70% rather than 40%).

[1] `http://www.planet-lab.org/`

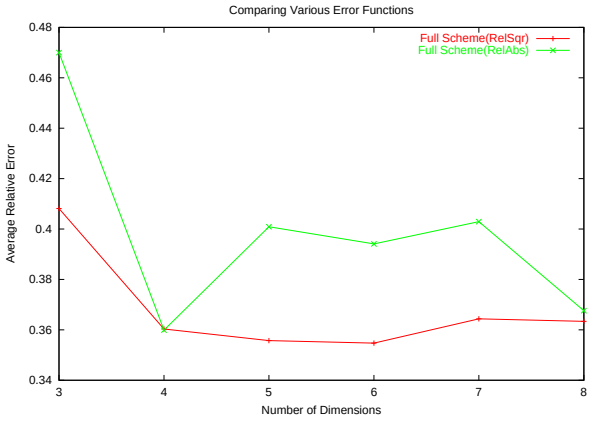

Figure 6: Accuracy in Prediction: comparing various error functions

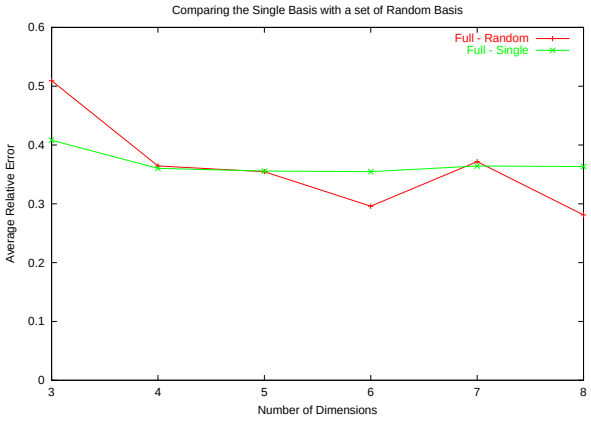

Figure 7: Comparing the situation used in the paper for one set of basis servers to results taken over a wide number, randomly chosen

This is a cumulative Frequency of the Average Relative Error between the predicted RTT between two servers in the model and the measured value. This is done for each pair of servers in the model calculated for each possible set of tracers we can use as a basis.

Figure 5 looks at one of these cases (the Exact-Newton case) and considers how it performs with the number of dimensions. It seems that the algorithm gets worse with the larger number of dimensions, but this is not the case as the number of constraints tends to go up with $O(N^2)$ so the model is harder to fit, it still produces better prediction as seen in Section 7.3.

7.2 Error Functions

Figure 6 shows a comparison between various different error functions. These results were obtained using our "full" scheme (described in Table 1). We chose a set of $d+1$ landmarks from the 19 tracers in the GNP set, these were used to calculate a basis. The rest of the tracers were then placed by the "full" New Point Calculation routine as were all the 869 targets. The results are the average relative error between the predicted distance between every target and every non-basis tracer and the actual measured RTT between the same two servers.

It can be seen that the "Relative Square" error function out performs the "Relative Abs". We have chosen to use the "Relative Square" function in the rest of this paper.

The"Relative Abs" behaves strangely anomaly at 4 and 8 dimensions (or at 5, 6 and 7) where the "Relative Abs" function drops right down to the value of the "Relative Square" square functions. We think this is due to the particular data-sets we have chosen to use from the GNP data sets

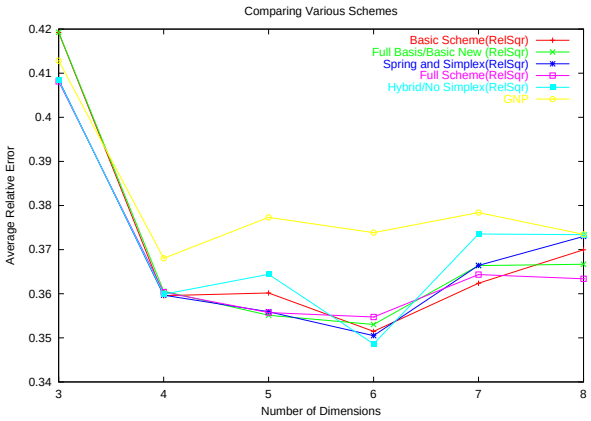

Figure 8: Accuracy in Prediction: comparing various methods (Note the x-axis is not at y=0)

(the choice of which landmarks to use) and will be resolved by taking wider results.

7.3 New Point Calculation

The results shown in Figures 8 and 9 summarise some of the results we obtained, to compare the various combinations of schemes.

The method is the same as described in the previous section for comparing error functions. We take the results from a single choice of basis servers, we compare this in Figure 7 to taking results for lots of different choices of basis at Random, giving a total of 1000 results. As can be seen they closely match, except for the 3 dimensional case - which seems to be too few dimensions for this data set. With higher dimensions the random line tends to be below the single case, but it seems to have a larger variance, this is because although we have the same number of results in each ex-

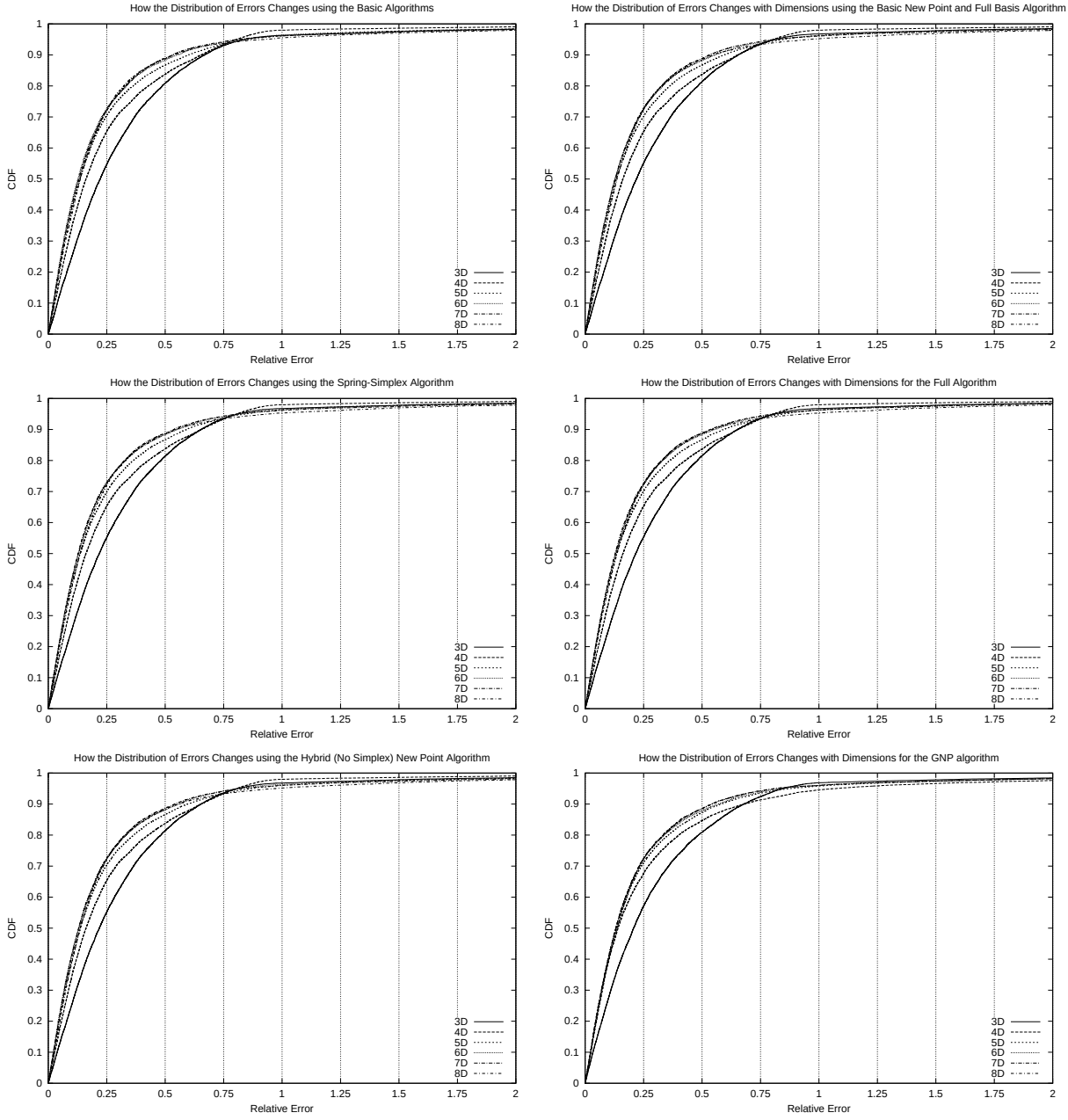

Figure 9: Accuracy in Prediction: comparing various error functions

Name	Method used for Base System	Method used for New Point
Basic Scheme	Simplex Downhill	Simplex Downhill
Full Basis/BasicNew	Spring Newton Minimise Simplex Downhill	Simplex Downhill
Spring and Simplex	Spring Newton Minimise Simplex Downhill	Exact Co-ordinates Spring Simplex Downhill
Full Scheme	Spring Newton Minimise Simplex Downhill	Spring Simplex Downhill
No Simplex	Spring Newton Minimise Simplex Downhill	Spring Newton Minimise
GNP	GNP	GNP

Table 1: Key to Methods

periment, we choose a much smaller proportion of the basis, for 3 dimensions there are 969 possible basis, for 8 dimensions there are 75582. Therefore we expect a larger variance in results.

Figure 8 compares at a glance how the various schemes compare, while Table 1 explains which algorithms are used and in what order. The graph shows that all our algorithm combinations perform better than GNP, this is probably due to us doing more restarts of the Simplex Downhill. It seems that the best results are to be found at 6 dimensions, but I think this maybe the fact that the simplex algorithm we use seems to need more restarts to get that last bit of accuracy. The increase in average error for 7 and 8 dimensions is just us not getting that last bit of accuracy and so performing more like GNP.

Figure 8 shows the results in more detail, with the Cumulative Distribution of the Relative Error (absolute) for each dimension in each scheme. It is most interesting to note that in all the results of our algorithms the 4D performs much better in the tail (and the tail is generally better than GNP). And so in this case 4D may be better than higher dimensions if it is really detrimental to have massively wrong results. It seems that for 4D we have 80% of errors less than 0.4 and only 5% of them are more than a factor of 1 out.

There seems to be differences in opinion over how many dimensions best model the Internet topology, some suggest three dimensions [2] while others go as high as seven [15] or eight [16]. This is both a question of further research and of the required accuracy of the resulting approximations, but our results seem to suggest that, for the GNP set we chose, 6 dimensions seems to give the best results gives a pretty optimal solution.

8 Further Work

We would like to investigate further in to several parts of the location system. We would definitely wish to look at how the system performs with larger data sets, along side this we would like to perform try a more wider set of landmarks/lighthouses using the GNP data-set.

A second issue we would like to investigate further is which actual hosts and positions cause the large outliers. We believe this will give insights on how to develop better algorithms.

Finally we wish to perform test of the distributed location system in the wide-area to evaluate it's performance.

9 Conclusion

We presented a distributed wide area co-ordinate system for use in the XenoSearch Resource Discovery system of the XenoServer Open Platform. We also presented results from various possible algorithms that can be used within the framework to calculate the co-ordinates, each with different trade-offs between average prediction error, distribution of the errors (whether there are many really bad predictions) and time. The co-ordinates calculated would be used to search and index XenoServers.

10 Acknowledgements

We are grateful to the States of Jersey Education Department for funding the work of David Spence.

References

[1] COSTA, M., CASTRO, M., ROWSTRON, A., AND KEY, P. Pic: Practical internet coordinates fo distance estimation. Tech. Rep. MSR-TR-2003-53, Microsoft Research, Cambridge, September 2003.

[2] COX, R., AND DABEK, F. Learning euclidean coordinates for internet hosts. http://www.pdos.lcs.mit.edu/~rsc/6867.pdf, December 2002.

[3] COX, R., DABEK, F., KAASHOEK, F., LI, J., AND MORRIS, R. Practical, distributed network coordinates. In *the proceedings of the ACM SIGCOMM Scond Workshop on Hot Topics in Networks (HotNets-II)* (November 2003).

[4] FRANCIS, P., JAMIN, S., JIN, C., JIN, Y., RAZ, D., SHAVITT, Y., AND ZHANG, L. IDMaps: a global internet host distance estimation service. *IEEE/ACM Transactions on Networking (TON) 9*, 5 (2001), 525–540.

[5] GUMMADI, K. P., SAROIU, S., AND GRIBBLE, S. D. King: Estimating Latency between Arbitrary Internet End Hosts. In *SIGCOMM Internet Mesurement Workshop 2002, Marseille, France* (November 2002).

[6] HAND, S., HARRIS, T., KOTSOVINOS, E., AND PRATT, I. Controlling the XenoServer Open Platform. In *Proceedings of the 6th International Conference on Open Architectures and Network Programming (OPENARCH)* (April 2003).

[7] LAWDER, J. K. Calcualation of mappings between one and n-dimensional values using the hilbert space-filling curve. Tech. Rep. JL1/00, School of Computer Science and Information Systems, Birbeck College, University of London, August 2000.

[8] LIM, H., HOU, J., AND CHOI, C.-H. Constructing internet coordinate system based on delay measurement. In *Internet Measurement Conference 2003* (October 2003).

[9] NELDER, J. A., AND MEAD, R. A simplex method for function minimization. *Computer Journal 7* (1965), 308–313.

[10] NG, E., AND ZHANG, H. Predicting Internet network distance with coordiantes-based approaches. In *INFOCOM'02, New York, USA* (2002).

[11] PIAS, M., CROWCROFT, J., WILBUR, S., HARRIS, T., AND BHATTI, S. Lighthouse for Scalable Distributed Location. In *proceedings of IPTPS '03 - 2nd International Workshop on Peer-to-Peer Systems* (February 2003).

[12] SHAVITT, Y., AND TANKEL, T. Big-bang simulation for embedding network distances in euclidean space. In *the proceedings of IEEE INFOCOM 2003* (April 2003).

[13] SPENCE, D., AND HARRIS, T. Xenosearch: Distributed resource discovery in the xenoserver open platform. Proceedings of the 12th IEEE symposium on High Performance Distributed Computing (HPDC-12).

[14] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, F., AND BALAKRISHNAN, H. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01)* (New York, August 2001), R. Guerin, Ed., vol. 31, 4 of *Computer Communication Review*, ACM Press, pp. 149–160.

[15] SZYMANIAK, M., PIERRE, G., AND VAN STEEN, M. Scalable cooperative latency estimation. Submitted for publication, July 2003. http://www.globule.org/publi/SCOLE_draft.html.

[16] TANG, L., AND CROVELLA, M. Virtual landmarks for the internet. In *Internet Measurement Conference 2003* (October 2003).

[17] WALDVOGEL, M., AND RINALDI, R. Efficient topology-aware overlay network. In *proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)* (October 2002).

[18] ZHANG, Y., PAXSON, V., AND SHENKER, S. The Stationarity of Internet Path Properties: Routing, Loss, and Throughput. *ACIRI Technical Report* (2000).