PIC: Practical Internet Coordinates for Distance Estimation

Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key
Microsoft Research, Cambridge, CB2 3NH, UK

September 2003

Technical Report
MSR-TR-2003-53

This paper introduces PIC, a practical coordinate-based mechanism to estimate Internet network distance (i.e., round-trip delay or network hops). Network distance estimation is important in many applications, for example, network-aware overlay construction and server selection. There are several proposals for distance estimation in the Internet but they all suffer from problems that limit their benefit. Most rely on a small set of infrastructure nodes that are a single point of failure and limit scalability. Others use sets of peers to compute coordinates but these coordinates can be arbitrarily wrong if one of these peers is malicious. While it may be reasonable to secure a small set of infrastructure nodes, it is unreasonable to secure all peers. PIC addresses these problems: it does not rely on infrastructure nodes and it can compute accurate coordinates even when some peers are malicious. We present PIC's design, experimental evaluation, and an application to network-aware overlay construction. Our results indicate that PIC can reduce the overhead of network-aware overlay construction by almost an order of magnitude.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
`http://www.research.microsoft.com`

I. Introduction

An efficient mechanism to estimate distance in the Internet could benefit many large scale distributed applications. For example, the performance of overlay networks can be greatly improved by exploiting information about the underlying network but the cost of sending probe messages to estimate distances can be significant (e.g., [1], [2], [3]).

This paper introduces PIC, a practical coordinate-based mechanism to estimate distances in the Internet. PIC assigns a point in a d-dimensional Euclidean space to each node and uses the distance between two points in the space as an estimate of the network distance between the corresponding nodes. Nodes compute their coordinates in the Euclidean space when they join the system. Given the coordinates for two nodes, any node can predict the distance between them. Our results show that PIC can reduce the cost of overlay construction in Pastry [2] by almost an order of magnitude.

In the last few years there has been much interest in this area, but previous proposals for network distance estimation suffer from problems that limit their practicality. Most proposals [4], [5], [6], [7] rely on a small set of infrastructure nodes that are a single point of failure and can limit scalability if they become communication bottlenecks. For example, GNP [5], which pioneered coordinate-based distance estimation, uses a set of fixed landmark nodes that are probed whenever a node joins the system.

Other proposals [8], [9] use sets of peer nodes in the system to compute each node's coordinates but they are vulnerable to malicious peers that can cause coordinates to be arbitrarily wrong. While it may be reasonable to secure a small set of infrastructure nodes, it is unreasonable to assume that no peers will behave maliciously. Additionally, the proposal in [8] incurs a large communication overhead and both proposals appear to be less accurate than GNP.

PIC addresses these problems. It scales well because it does not rely on infrastructure nodes; any node whose coordinates have already been computed can act as a landmark. Therefore, it can distribute communication and computation load evenly over all the nodes in a system. Additionally, it computes coordinates efficiently and we describe a technique for choosing landmark nodes that can predict distances as accurately as GNP. Finally, PIC can compute accurate coordinates even when some peers are malicious.

The rest of the paper is organised as follows. Section II describes PIC and presents some results. Section III describes how to implement PIC efficiently. Section IV describes and evaluates the PIC approach to security and Section V discusses an application of PIC in proximity-aware overlay construction. Section VI presents related work and Section VII concludes.

II. PIC coordinate computation

PIC maps each node to a point in a d-dimensional Euclidean space. When a node n joins the system, it computes the coordinates of its corresponding point. It probes the network distance to each element of a set of landmarks, L, where L must have at least $d+1$ members. Then it obtains the coordinates of each landmark, and uses a multi-dimensional global optimization algorithm (e.g., Simplex Downhill [10]) to compute its coordinates such that the errors in the $|L|$ predicted distances between n and each node in L are minimized. The errors are computed using the measured and estimated distances. The probe could use ICMP, application-level round-trip time, or number of IP hops. This is similar to GNP [5] but GNP uses a fixed set L for all the nodes that join the system.

In PIC, the joining node can pick any node whose coordinates have already been computed to be a landmark. Let N be the set of nodes whose coordinates have already been computed. When a node n joins the system, it can select any set L that is a subset of N with size $|L| > d$. We experimented with three different strategies to choose L:

- *random:* pick the elements of L randomly with uniform probability from N;
- *closest:* pick the elements of L to be the elements of N closest to n in the network topology;
- *hybrid:* pick some elements as in random and others as in closest.

We define L for a node n to be the union of two sets L_r and L_c. The elements in L_r are chosen randomly from N, whilst the elements in L_c are the $|L_c|$ members of N which are closest to node n in the network.

When bootstrapping the behaviour of the system is slightly different. If $|N| < |L|$ then n selects all the nodes in N. Then it obtains the measured distances between *all* pairs of nodes in N (a $|N| \times |N|$ matrix). In this case, the global optimization algorithm computes new coordinates for all the nodes in N by minimizing the error in the predicted distances between all pairs of nodes in $N \cup \{n\}$.

We use the Simplex Downhill [10] algorithm to compute coordinates as in [5]. We experimented with several target error functions to minimize. The one that performed the best was the sum of the squares of the relative errors:

$$\sum_{i=1}^{|L|}(\frac{d_i^m - d_i^p}{d_i^m})^2$$

where, d_i^m is the distance measured between node n and the ith node in L and d_i^p is the distance predicted between node n and the ith node in L.

The intuition behind the different strategies to choose L in PIC is the following. The *closest* strategy should provide the Simplex algorithm with better information to position the joining node correctly in the Euclidean space relative to nearby nodes in the network. The *random* strategy should provide the Simplex algorithm with better information to position the joining node correctly in the Euclidean space relative to distant nodes in the network. Therefore, the *closest* strategy should achieve lower relative errors when predicting short distances whereas the *random* strategy should achieve lower relative errors when predicting long distances. The *hybrid* strategy should achieve something in the middle. The experimental results in the next section confirm this intuition and show that the *hybrid* strategy achieves lower relative errors than the other strategies.

The current version of Lighthouses [9] uses the *random* strategy to select L. The *closest* strategy is similar to the approach used in Mithos [8].

The *closest* and *hybrid* strategies require a mechanism to find the closest nodes to a node in the network. This can be done in several ways, e.g., using expanding ring multicast or the algorithms described in [11], [12], [8]. In Section III, we describe an efficient closest node discovery algorithm. The algorithm to find the closest nodes in Mithos is significantly more expensive than ours.

A. Experimental evaluation

We ran a number of experiments to evaluate the different strategies to pick landmarks in PIC and to compare them with GNP.

1) Experimental setup: The experiments used network distance data from the following three network topologies. In each topology, there is a core set of routers and we ran PIC on 40,000 end nodes that were randomly assigned to routers in the core with uniform probability. Each end node was directly attached by a LAN link with a 1ms delay to its assigned router.

GATech is a transit-stub topology generated with the Georgia Tech [13] random graph generator. This network topology has 5050 routers arranged hierarchically. There are 10 transit domains at the top level with an average of 5 routers in each. Each transit router has an average of 10 stub domains attached, and each stub has an average of 10 routers. The network distance in this topology is round-trip delay. The delay between routers is computed by the graph generator and routing is performed using the routing policy weights of the graph generator. As in the real Internet, the triangle inequality does not hold for all round trip times among end nodes in this topology.

Mercator is a topology with 102,639 routers and it was obtained from real measurements of the Internet using the Mercator system [14]. The authors of [15] used real data and some simple heuristics to assign an autonomous system to each router. The resulting AS overlay has 2,662 nodes. Routing is performed hierarchically as in the Internet. A route follows the shortest path in the AS overlay between the AS of the source and the AS of the destination. The routes within each AS follow the shortest path to a router in the next AS of the AS overlay path. The network distance in this topology is the number of hops in the route and we use shortest path routing.

CorpNet is a topology with 298 routers and is generated using real measurements of the World-Wide Microsoft Corporate network. The network distance in this topology is the minimum round-trip delay.

We experimented with different values for the number of landmarks and dimensions. Increasing the number of dimensions improves accuracy for GNP and all PIC strategies but we did not observe any benefit above 12 dimensions. Increasing $|L|$ also improves accuracy for GNP and the *random* strategy but has little effect for *closest* and *hybrid*. The experiments described here used $d = 8$ dimensions and $|L| = 16$ landmarks.

GNP relies on a fixed set of landmarks so its accuracy is very sensitive to their placement in the network. To provide a fair comparison between PIC and GNP, we ran an optimization procedure to determine the best landmark placement for GNP on each topology. We ran 100 different GNP experiments with 1,000 nodes, 1,000 test distances, and different randomly picked sets of landmarks L. The mean relative error in these experiments varied significantly: it varied between 0.17 and 0.29 in GATech, between 0.17 and 0.23 in Mercator, and between 0.11 and 0.32 in CorpNet. The results that we present for GNP were obtained using the landmarks that produced the minimum average relative error. The average relative error obtained using PIC did not vary significantly across these experiments, which is a desirable property.

2) Results: Figures 1, 3, and 2 show the cumulative distribution of relative errors in 100,000 random test distances for the *GATech*, *Mercator*, and *CorpNet* topologies. Each figure has lines for GNP and each of the PIC strategies. The *hybrid* strategy used 4 nearby landmarks

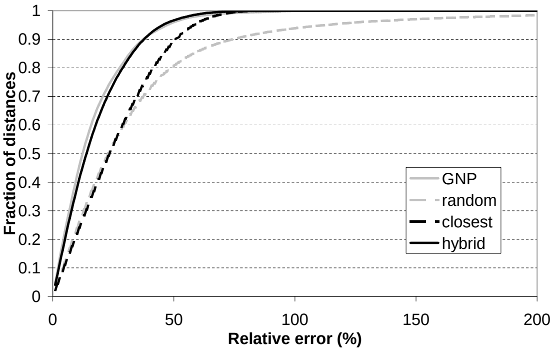

Fig. 1. Cumulative distribution of relative errors over random distances in GATech.

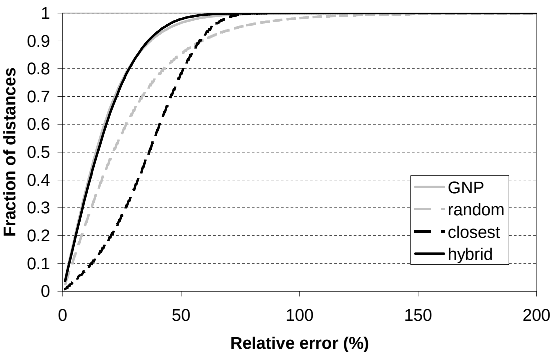

Fig. 2. Cumulative distribution of relative errors over random distances in Mercator.

and 12 randomly selected ones. The results show that PIC can match the accuracy of GNP with the *hybrid* strategy but performs significantly worse using either the *random* or *closest* strategies. It is interesting to note that GNP is using an optimized landmark placement; PIC with *hybrid* provided better accuracy than GNP before we optimized landmark placement.

Figure 4 provides some intuition to explain the previous results. It shows the cumulative distribution of relative errors for short test distances. For this experiment, we randomly selected 2,000 nodes and for each

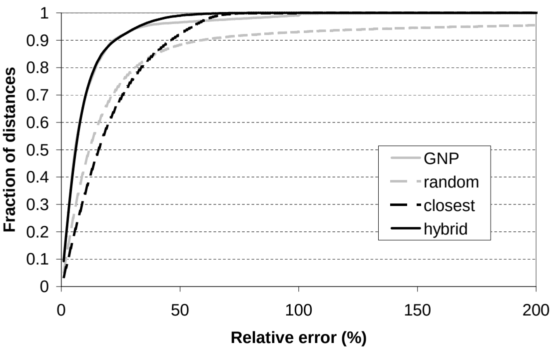

Fig. 3. Cumulative distribution of relative errors over random distances in CorpNet.

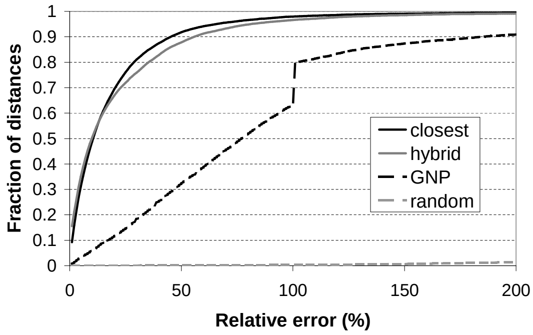

Fig. 4. Cumulative distribution of relative errors over short distances in GATech.

node we generated test distances between the node and the 50 closest nodes in the network. This generated a test set of 100,000 test distances. The results show that *closest* performs significantly better than GNP in this range whereas *random* performs badly. They support the intuition that picking landmarks that are close on the network reduces relative errors in the prediction of short distances. Since *closest* does worse than GNP over random test distances, this also shows that using distant random landmarks reduces relative errors in the prediction of long distances. It is unclear why the GNP curve has a discontinuity around 100%. We suspect that this is due to the delay discontinuity between LAN and inter-router links.

The results also show that *hybrid* performs similarly to *closest* over short distances. Therefore, using 4 nearby landmarks appears to be a good configuration; it achieves the benefits of *closest* over short distances and matches the performance of GNP over random test distances.

Table I shows summary metrics for the relative error distributions of 100,000 random test distances in all topologies for GNP and all PIC strategies. The results are qualitatively similar across all topologies. They are even quantitatively very similar for the mean and 90-th percentile of GNP and *hybrid* on the GATech and Mercator topologies.

		GNP	random	closest	hybrid
GATech	Max	19.11	391.60	1.62	2.13
	90-th	0.37	0.75	0.51	0.38
	Mean	0.17	0.41	0.26	0.17
Mercator	Max	2.50	8.63	1.55	1.64
	90-th	0.37	0.58	0.57	0.37
	Mean	0.17	0.28	0.36	0.17
CorpNet	Max	13.34	404.92	1.00	1.33
	90-th	0.23	0.60	0.47	0.23
	Mean	0.11	0.61	0.20	0.09

TABLE I

RELATIVE ERROR DISTRIBUTION SUMMARIES.

The results also show that the *closest* and *hybrid* strategies reduce the maximum relative error significantly relative to the *random* strategy and GNP, which is not surprising given that this is likely to correspond to a short distance and they perform well over short distances.

III. Finding close nodes

The previous section shows that PIC performs best using the *hybrid* strategy but this strategy requires some mechanism to find the closest nodes in the network. In the previous section, we used an oracle to find the closest nodes with global knowledge. This section describes algorithms that can be used to implement this oracle efficiently in a distributed system. We also evaluate the impact on PIC's accuracy of replacing the oracle by one of these algorithms.

PIC can replace the oracle by one of several algorithms that have been proposed to find the closest node to a particular node in a network, for example, [11], [12]. These algorithms all share a similar overlay structure. Each node in the overlay maintains a set of pointers to other nodes in the overlay that we call its *neighbors*. These algorithms prescribe a particular mix of near and far away neighbors to ensure that a node n can find the closest node in the overlay in $O(logN)$ steps provided the topology satisfies certain conditions. To find the closest node, n starts by setting its estimate of the closest node in the overlay, c, to a random overlay node. Then it probes the distance to all of c's neighbors and picks the closest neighbor. If this closest neighbor is closest than the current value of c, c is updated to point to this neighbor and the process is repeated. Otherwise, the algorithm stops and c is an approximation to the closest node to n in the overlay. These algorithms can also find the k closest nodes to n by keeping track of the k closest nodes visited.

PIC could use one of these algorithms but they require a significant number of probes to estimate the distance between nodes. For example, the algorithm in [12] requires approximately 250 probes for a 60,000 node network.

We can reduce the overhead by using PIC to estimate distances rather than using probes to measure the distances. The problem is that we need to find the closest nodes to a node that does not have coordinates yet. Our improved algorithm solves this problem as follows. A joining node starts by using PIC with the *random* strategy to generate a rough estimate of its coordinates from a set of random overlay nodes. Then these rough coordinates are used to estimate distances in the algorithm to find the closest k nodes. After finding the closest k nodes, the joining node uses the PIC *hybrid* strategy to refine its coordinates.

```
probed = Probe(random nodes)
allVisited = probed
CalculateCoordinates(probed)
numWalks = 0
do
  numWalks++
  nearNode = PickRandom(allVisited)
  visited = nearNode
  do
    currentClosest = nearNode
    visited += GetNeighbors(nearNode)
    nearNode = GetPredictedClosest(visited+{nearNode},1)
  while(currentClosest != nearNode)
  allVisited += visited
while (PredictedDistanceTo(nearNode) > t
      and numWalks < maxWalks)

probed += Probe(GetPredictedClosest(allVisited, k))
CalculateCoordinates(probed)
```

Fig. 5. Optimized close node discovery algorithm.

We implemented and evaluated two variants of this strategy. The pseudo code for the first one is in Figure 5. In this variant, the joining node, n, computes rough coordinates from a set of random nodes and performs a series of greedy walks in the overlay towards the closest node. It starts each walk from a random node and uses the rough coordinates to estimate distances to guide the walk. This search stops when the number of walks reaches a maximum value or when the predicted distance to the closest node found is below a threshold t. Our current implementation sets the threshold to the average distance to the closest neighbor computed over the set of all nodes visited during the search process. When the search stops, the node probes the k predicted closest nodes and uses the measured distances and their coordinates to recompute its coordinates. We use $k = 4$ because our previous results indicate that PIC *hybrid* performs well with this value.

The second variant is in Figure 6. It incurs a higher overhead but it achieves better accuracy because nodes refine their coordinates at each step of the search process. More precisely, the joining node n probes the m predicted closest neighbors at each step and it uses the measured distance to select the closest neighbor. n also recomputes its coordinates using all nodes probed so far. Our current implementation uses $m = 2$. Additionally, n probes the k predicted closest nodes at the end of each

```
probed = Probe(random nodes)
allVisited = probed
CalculateCoordinates(probed)
numWalks = 0
do
   numWalks++
   nearNode = PickRandom(allVisited)
   visited = nearNode
   probed += Probe(nearNode)
   do
      currentClosest = nearNode
      visited += GetNeighbors(nearNode)
      nodes = GetPredictedClosest(visited, m)
      foreach node in nodes
         probed += Probe(node)
         nearNode = closerToMe(node,nearNode)
      CalculateCoordinates(probed)
   while (currentClosest != nearNode)
   allVisited += visited
   probed += Probe(GetPredictedClosest(allVisited,k))
   CalculateCoordinates(probed)
while (DistanceTo(nearNode) > t
      and numWalks < maxWalks)
```

Fig. 6. Optimized close node discovery algorithm with progressive coordinate refinement.

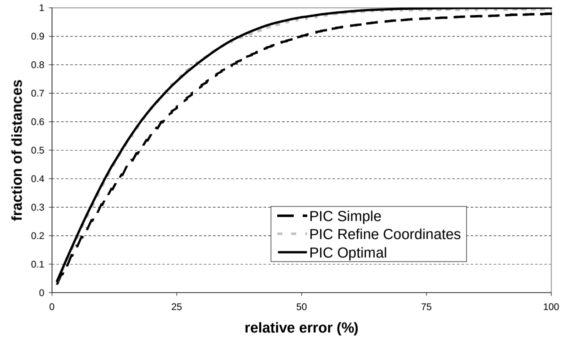

Fig. 7. Cumulative distribution of relative errors over random distances in GATech, using two algorithms to find close nodes.

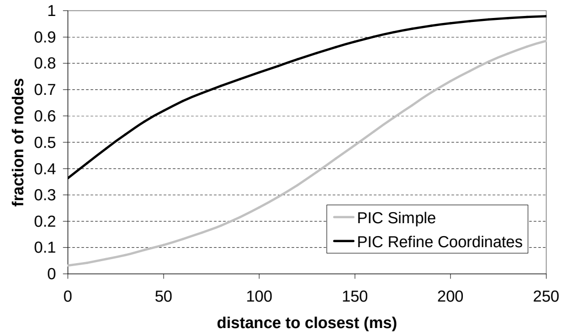

Fig. 8. Cumulative distribution of distances to closest nodes.

walk (if not probed yet) and recomputes its coordinates.

It is also interesting to note that we can can improve the accuracy of the algorithms to find close nodes, and reduce their associated cost, by multicasting discovery messages in a range of one or two network hops. This simple mechanism will work very effectively when the density of PIC nodes in the network is high. In the particular case where all the nodes in the network are part of the PIC system, a joining node can find the closest nodes simply by broadcasting a discovery message over all of its physical interfaces.

A. Experimental evaluation

We ran some experiments to evaluate PIC's accuracy when using both algorithms described above. Our experiments ran on the Pastry overlay described in [12]. Each Pastry node keeps a set pointers to neighbors. These pointers form the overlay that is used to locate closest nodes. The results should be similar for the overlay described in [11]. The experiments ran on the GATech topology in the experimental setting described in Section II-A.1. We set the maximum number of walks to 5, $m = 2$, and we use $k = 4$ to match the experiments presented in previous sections. The pastry overlay was configured with $b = 4$ and $l = 16$ [12].

Figure 7 shows the cumulative distribution of relative errors for PIC using the oracle and the two algorithms to find close nodes. The line labeled *PIC Simple* corresponds to the version of the algorithm in Figure 5, and the line labeled *PIC Refine Coordinates* corresponds to the one in Figure 6. The line labeled *PIC Optimal* corresponds to the version of PIC that uses the oracle to find the closest node as in the previous section.

The results show that PIC's accuracy is essentially the same using the oracle or the algorithm with coordinate refinement but the accuracy drops when using the algorithm without refinement. This drop in accuracy is explained by the results in Figure 8. This figure shows the cumulative distribution of absolute errors when finding close nodes with both algorithms. The algorithm with coordinate refinement is significantly more effective at locating the closest node or a close approximation, which explains the improved accuracy. This increased accuracy comes at the cost of an increased number of probes: the algorithm with refinement probes 55.6 nodes on average and the algorithm without refinement only probes 21 nodes.

IV. Secure coordinate computation

The version of PIC that we described in the previous section and previous solutions to network distance estimation are vulnerable to malicious nodes. If a malicious node is selected as a landmark, it can lie about its coordinates or interfere with the distance measurement. The result of this attack is a set of coordinates that can be arbitrarily wrong. For PIC to be practical, we need to be able to compute accurate coordinates even when some of the nodes chosen to be landmarks are malicious.

We devised a *security test* to eliminate malicious nodes from the set of landmarks chosen to compute the coordinates of each node n. The test relies on the observation that the *triangle inequality* holds for most triples of nodes in the Internet. Since the accuracy of PIC and the other distance estimation proposals relies on this condition [5], [4], it is a reasonable assumption for our security test. Therefore, we assume that for most triples of nodes a, b, c, $d_{a,b} + d_{b,c} \geq d_{a,c}$, where $d_{i,j}$ denotes either the measured network distance between i and j or the predicted distance.

The intuition behind the security test is as follows. An attacker that lies about its coordinates or its distance to the joining node is likely to violate triangle inequality. The joining node uses the distances it measured to each landmark node and the coordinates of the landmarks to check for violations of the triangle inequality. It then removes from L the nodes that most violate the triangle inequality.

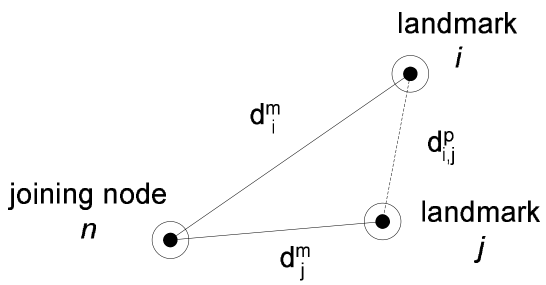

Fig. 9. Triangle inequality with measured and predicted distances.

Let L be the set of landmark nodes chosen to compute n's coordinates and let i,j be two distinct elements of L. This is illustrated in Figure 9. If d_i^m is the distance measured between n and i and $d_{i,j}^p$ is the distance predicted (using the coordinates) between i and j, all of the following should hold for a correct i:

$$d_i^m \leq d_j^m + d_{i,j}^p \ (1)$$

$$d_i^m \geq d_j^m - d_{i,j}^p \ (2)$$

$$d_i^m \geq d_{i,j}^p - d_j^m \ (3)$$

The first inequality imposes an upper bound on d_i^m and the other two impose a lower bound of $|d_j^m - d_{i,j}^p|$.

For each element i in L, the security test checks whether the upper bounds and lower bounds defined by each element j in L are satisfied by i and computes the following two metrics:

$$upper_i = \sum_{j=1}^{|L|} \begin{cases} d_i^m - (d_j^m + d_{i,j}^p) & \text{if } (d_j^m + d_{i,j}^p) < d_i^m, \\ 0 & \text{otherwise} \end{cases}$$

$$lower_i = \sum_{j=1}^{|L|} \begin{cases} |d_j^m - d_{i,j}^p| - d_i^m & \text{if } |d_j^m - d_{i,j}^p| > d_i^m, \\ 0 & \text{otherwise} \end{cases}$$

$upper_i$ is the sum of the deviations above the upper bounds and $lower_i$ is the sum of the deviations below the lower bounds.

The security test computes the maximum value of both metrics for all nodes in L and removes the corresponding node. Then, the joining node uses the Simplex to compute its coordinates with the remaining landmarks. This process is repeated a fixed number of times (less than $|L| - d - 1$) or until the average relative error in the predicted distances between the joining node and the remaining landmarks is below a threshold (currently, 5%).

A. Experimental evaluation

We ran experiments to evaluate the accuracy of PIC under attack with the security test.

1) Attacker model: We model a very powerful attacker. We assume that a fraction f of the nodes in the overlay is malicious and that all the malicious nodes collude to cause the most damage to the system.

When a node joins the system, all malicious landmarks collude to produce coordinates for the joining node that are the furthest away possible from the correct ones. We give the attacker total knowledge to achieve its goal; all malicious landmarks know the distances between all the landmarks and the joining node, n, and all the landmark coordinates. The attacker uses this information to compute a set of fake coordinates and distances for all malicious landmarks that maximize the error in the coordinates computed for the joining node.

We impose a restriction on the shortest distance provided by a malicious landmark. It cannot be shorter than the distance between the joining node and the closest attacker. This is a realistic assumption if probes include a nonce (an unpredictable value that identifies the probe). The nonce ensures that the closest point at which a credible reply to a distance probe can be faked is the position of the closest attacker.

Picking fake coordinates and distances that maximize the error in the coordinates computed for the joining node is a multi-dimensional optimization problem similar to the coordinate computation problem in PIC. The Simplex algorithm [10] is a good approach to solve this type of problem. We implement the error maximizing attack using simplex to minimize the following function:

$$f(C_a, D_a) = \begin{cases} +\infty & \text{if } \exists d \in D_a : d < d_{ca}, \\ 1/dist(p_{correct}, p(C_a, D_a)) & \text{otherwise} \end{cases}$$

Here, C_a is the set of attacker coordinates, D_a is the set of attacker distances to the joining node, d_{ca} is the distance from the joining node to the closest attacker, $p_{correct}$ is the correct position of the joining node, and $p(C_a, D_a)$ is the position of the joining node computed with the current values of C_a and D_a (and the coordinates and distances of the correct landmarks that are all known to the attacker).

When some malicious nodes are chosen as landmarks, they compute the joining node's correct position using their correct coordinates and distances and the distances and coordinates of the correct landmark nodes. Then they find the minimum of the function above using the Simplex Downhill algorithm and they supply the joining node with the coordinate values C_a and distance values D_a at the minimum value of $f(C_a, D_a)$.

We reiterate that this is a very powerful attacker. In practice, we expect that it will be difficult for the malicious nodes to learn the distances and coordinates of the honest landmarks. Additionally, faking distances to malicious landmarks equal to the distance to the closest malicious node is hard because it requires the closest attacker to intercept the probes to all malicious landmarks. Finally, we give the attacker 10 minutes to compute fake coordinates and distances that maximize the error in the coordinates of the joining node. In practice, the attacker would have only seconds or less to run this computation.

2) Experimental results: We ran some experiments to evaluate the accuracy of PIC under the attack model outlined above. We also experimented with less powerful attacker models, for example, attackers that try to attract joining nodes to their position by providing small distances, and attackers that provide random distances and coordinates. The attacker model discussed above was significantly more effective than all of these.

Figure 10 shows the cumulative distribution of relative errors for PIC *hybrid* with 32 landmarks, 4 of which are nearby landmarks, and security. These results were obtained with the GATech topology in the experimental setting described in Section II-A.1 but with 2,000 end nodes. The two topmost lines show the performance of the system without attacks. They show that using the security test does not result in any degradation on accuracy.

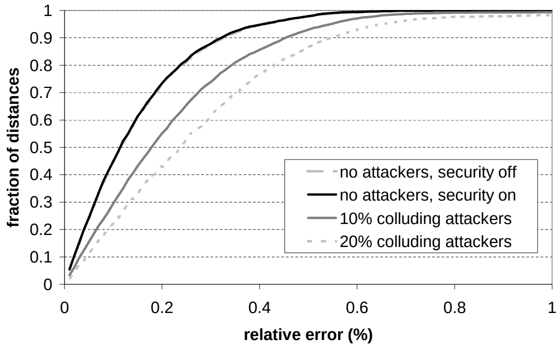

Fig. 10. Cumulative distribution of relative errors over random distances with security in GATech.

The lines labelled *colluding attackers* were obtained using the attacker model described above when 10% and 20% of the nodes in the system are malicious. The results show that the accuracy of PIC is very good even when the fraction of malicious nodes is high and even with the very powerful attacker that we modelled.

V. Network-aware overlay construction with PIC

Recently there has been much interest in structured peer-to-peer overlay networks like CAN, Chord, Pastry, Tapestry, and others [16], [17], [2], [3], [18], [19]. They assign application-defined keys to overlay nodes and provide a primitive to route a message to the node responsible for a key. Structured overlays conform to a specific graph structure that allows them to route in $O(\log N)$ hops while maintaining at most $O(\log N)$ routing state where N is the number of nodes in the overlay.

It is important for overlay routing to exploit proximity in the underlying network. Otherwise, each overlay hop has an expected delay equal to the average delay between a pair of random overlay nodes, which stretches route delay by a factor equal to the number of overlay hops and increases the stress in the underlying network links. There are several techniques for proximity-aware routing proposed in the literature [20], [3], [16], [2], [21], [22]. Recent work [23], [24], [25] identifies proximity neighbor selection (PNS) as the most promising technique. Tapestry, Pastry [12], and a recent version of Chord [25] implement PNS.

PNS can be used to achieve low delay routes and low bandwidth usage. It selects routing state entries for each node from among the closest nodes in the underlying topology that satisfy constraints required for overlay routing. Currently, building the routing state of each node requires probes to estimate network distances when nodes join and while maintaining the overlay. This contributes a significant overhead, for example, there are approximately 297 distance probes on average when a node joins a Pastry network with 20,000 nodes. These overheads are lower than in other algorithms.

Using PIC to estimate network distances reduces this overhead by almost an order of magnitude because distance probes can be replaced by PIC distance estimates. PIC only requires distance probes to compute node coordinates. This technique can be applied to any of the systems mentioned above. We implemented and evaluated a version of PNS on Pastry that uses PIC to estimate network distances.

We modified Pastry to use PIC as follows. Each Pastry node already maintains a set with 16 other nodes (called the leaf set) which are random and uniformly distributed across the network. A joining node uses the leaf set of its contact node as the set of landmarks L to compute a rough estimate of its coordinates with *random* PIC. Then it uses this rough estimate to find the closest node in the Pastry overlay while refining its coordinates using the algorithm in Figure 6. When it finds this node, it uses the algorithm in [12] to join with the closest node as seed. We modified the join and routing table maintenance algorithms in Pastry to replace probes by distances estimated with PIC.

A. Experimental evaluation

We ran an experiment to evaluate the impact of using PIC on overhead and route delays. We used the GATech topology model described before and we ran Pastry on 20,000 end nodes that were randomly assigned to routers in the core with uniform probability. Each end node was directly attached by a LAN link with a delay of 1ms to its assigned router. We compare the version of Pastry described in [12] with the version of Pastry that uses PIC, which is labeled *Pastry-PIC*.

Figure 11 shows the number of distance probes per node when building the overlay in both versions of Pastry. The results show that PIC can reduce the overhead by almost an order of magnitude; Pastry-PIC requires only an average of 32 probes per node while Pastry requires an average of 297.

The second experiment routed 100,000 lookup messages from randomly chosen overlay nodes to randomly chosen keys. Figure 12 shows the delay stretch relative to the direct IP route between source and destination for both versions of Pastry. The results show that PIC can reduce the overhead dramatically while increasing route delays by only 13% on average.

Furthermore, there are several applications built on top of Pastry that can benefit from the coordinates generated

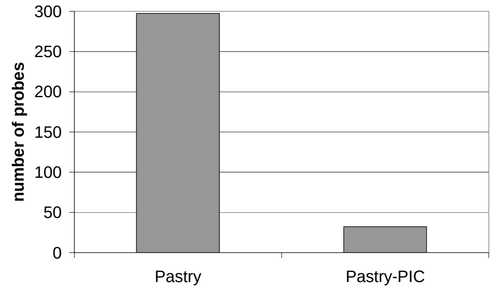

Fig. 11. Distance probes per node to build a 20,000-node Pastry overlay in GATech with and without PIC.

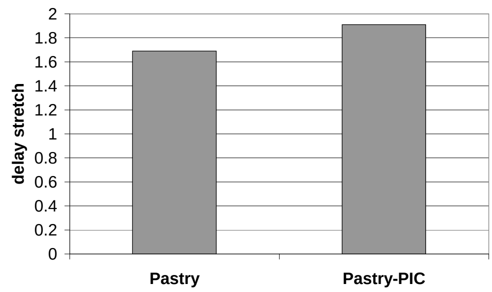

Fig. 12. Delay stretch in a 20,000-node Pastry overlay in GATech with and without PIC.

using PIC. For example, they can reduce the cost of the bottleneck remover in the Scribe [26] application level multicast algorithm that currently requires probes to estimate network distances. They can also improve the performance of anycast on Scribe trees [27].

VI. Related work

Most mechanisms that have been proposed to estimate network distances [4], [5], [6], [7], [28] rely on a set of special infrastructure nodes that are a single point of failure and that bear most of the communication load required to predict distances. For example, GNP [5], which pioneered coordinate-based distance estimation, uses a set of fixed landmark nodes that are probed whenever a node joins the system. These landmarks can limit the scalability of the system if they become communication bottlenecks and the system's accuracy is very sensitive to their placement. PIC does not require any dedicated infrastructure and it matches the accuracy of GNP with optimized landmark placement.

Mithos [8] does not rely on dedicated infrastructure but it selects the closest nodes as landmarks. Therefore, our results indicate that it is less accurate than GNP and *hybrid* PIC. Additionally, the algorithm used by Mithos to locate the closest nodes is expensive because it requires a large number of probes to measure network distance. We described an efficient algorithm to locate the closest nodes using PIC. Mithos does not include any defense against malicious landmarks.

Lighthouses [9] was designed concurrently with PIC. It also does not rely on a fixed infrastructure but selects landmarks randomly. Therefore, our results indicate that it is less accurate than GNP and *hybrid* PIC. Lighthouses computes coordinates differently from GNP and PIC. It uses exactly $d+1$ landmarks and solves a set of linear equations to determine a nodes's coordinates. It lacks the robustness to measurement noise and malicious peers of a multidimensional optimization algorithm with more landmarks. The techniques that we describe to secure PIC could potentially be used to secure Lighthouses.

VII. CONCLUSIONS

This paper described PIC, a coordinate-based mechanism to estimate Internet network distance (i.e., round-trip delay or network hops). PIC is scalable and robust because it does not rely on infrastructure nodes and spreads load evenly over all the nodes in the system. PIC is also secure because it can compute accurate coordinates even when some nodes are malicious. Therefore, PIC can be used to improve the performance of many large-scale distributed applications like network-aware overlay construction and location of nearby resources in the network. We modified Pastry to use PIC instead of direct distance probing. Our results indicate that PIC reduces the cost of joining the Pastry overlay by almost an order of magnitude.

ACKNOWLEDGMENT

The authors would like to thank Hongsuda Tangmunarunkit, Ramesh Govindan, and Scott Shenker for the Mercator topology. The authors would also like to thank Marteen van Steen for interesting discussions.

REFERENCES

[1] Yang hua Chu, Sanjay G. Rao, and Hui Zhang, "A case for end system multicast," in *Proc. of ACM Sigmetrics*, Santa Clara, CA, June 2000, pp. 1–12.

[2] Antony Rowstron and Peter Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Middleware 2001*, Heidelberg, Germany, Nov. 2001.

[3] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph, "Tapestry: An infrastructure for fault-resilient wide-area location and routing," Tech. Rep. UCB//CSD-01-1141, U. C. Berkeley, April 2001.

[4] Paul Francis, Sugih Jamin, Vern Paxson, Lixia Zhang, Daniel Gryniewicz, and Yixin Jin, "An architecture for a global internet host distance estimation service," in *IEEE Infocom99*, March 1999.

[5] T. S. Eugene Ng and Hui Zhang, "Predicting internet network distance with coordinates-based approaches," in *IEEE Infocom02*, June 2002.

[6] S. Hotz, "Routing information organization to support scalable interdomain routing with heterogeneous path requirements," 1996.

[7] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker, "Topologically-aware overlay construction and server selection," in *Infocom*, 2002.

[8] M. Waldvogel and R. Rinaldi, "Effi cient topology-aware overlay network," in *HotNets*, 2002.

[9] Marcelo Pias, Jon Crowcroft, Steve Wilbur, Tim Harris, and Saleem Bhatti, "Lighthouses for scalable distributed location," in *IPTPS'03*, Feburary 2003.

[10] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.

[11] David R. Karger and Matthias Ruhl, "Finding nearest neighbors in growth-restricted metrics," in *ACM Symposium on Theory of Computing (STOC '02)*, May 2002.

[12] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," Tech. Rep. MSR-TR-2002-82, Microsoft Research, May 2002.

[13] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM96*, 1996.

[14] Ramesh Govindan and Hongsuda Tangmunarunkit, "Heuristics for internet map discovery," in *Proc. 19th IEEE INFOCOM*, Tel Aviv, Israel, March 2000, IEEE, pp. 1371–1380.

[15] H. Tangmunarunkit, R. Govindan, D. Estrin, and S. Shenker, "The impact of routing policy on internet paths," in *Proc. 20th IEEE INFOCOM*, Alaska, USA, Apr. 2001.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in *Proc. of ACM SIGCOMM*, Aug. 2001.

[17] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.

[18] Petar Maymounkov and David Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS'02*, Boston, MA, Mar. 2002.

[19] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: A scalable and dynamic emulation of the butterfly," in *PODC'02*, Aug. 2002.

[20] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *Proc. 9th ACM Symp. on Parallel Algorithms and Architectures*, June 1997, pp. 311–320, Newport, Rhode Island, USA.

[21] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao, "Distribted data location in a dynamic network," in *SPAA'02*, Aug. 2002, Winnipeg, Manitoba, Canada.

[22] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker, "Topologically-aware overlay construction and server selection," in *Proc. 21st IEEE INFOCOM*, New York, NY, June 2002.

[23] Miguel Castro, Peter Druschel, Y. Charlie Hu, and Antony Rowstron, "Exploiting network proximity in distributed hash tables," in *FuDiCo 2002: International Workshop on Future Directions in Distributed Computing*, June 2002.

[24] Miguel Castro, Mike Jones, Anne-Marie Kermarrec, Antony Rowstron, Marvin Theimer, Helen Wang, and Alec Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlay networks," in *Proc. 22nt IEEE INFOCOM*, Mar. 2003.

[25] Krishna P. Gummadi, Ramakrishna Gummadi, Steven D. Gribble, Sylvia Ratnasamy, Scott Shenker, and Ion Stoica, "The impact of dht routing geometry on resilience and proximity," in *ACM SIGCOMM 2003*, 2003.

[26] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on*

Selected Areas in Communication (JSAC), vol. 20, no. 8, October 2002.

[27] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron, "Scalable application-level anycast for highly dynamic groups," in *NGC'2003*, 2003.

[28] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for internet hosts," in *ACM SIGCOMM 2001*, August 2001.