Building Topology-Aware Overlays using Global Soft-State

Zhichen Xu
Hewlett-packard Laboratories
zhichen@hpl.hp.com

Chunqiang Tang
Dept. of Computer Science
Univ. of Rochester
sarrmor@cs.rochester.edu

Zheng Zhang
Microsoft Research Asia
zzhang@microsoft.com

Abstract

Recent peer-to-peer (P2P) networks, represented by CAN, Chord, and Pastry, offer an administration-free and fault-tolerant application-level overlay network. For these systems to function efficiently, they must make effective use of the underlying network topology.

Existing techniques for discovering network proximity information, such as landmark clustering and expanding-ring search, are either inaccurate or expensive. Moreover, the lack of global proximity information in overlay construction and maintenance results in either bad proximity approximation or excessive communication.

To address these problems, we propose the following: (1) Combining landmark clustering and RTT measurements to identify the closest node, achieving both efficiency and accuracy. (2) Controlled placement of global proximity information on the system itself as soft-state, such that nodes can independently access relevant information efficiently. (3) Pub/sub functionality that allows nodes to subscribe to the relevant soft-state and get notified as the state changes necessitate overlay restructuring.

Keywords: peer-to-peer, overlay, soft-state, network protocol, topology, landmark, pub/sub system, heterogeneity

1 Introduction

Recent peer-to-peer (P2P) networks, represented by CAN [11], Chord [15], and Pastry [13], offer an administration-free and fault-tolerant application-level overlay network. The basic functionality these systems provide is a distributed hash table (DHT). For these systems to function efficiently, they must effectively take advantage of the conditions of the underlying physical network. These conditions include storage capacity, forwarding capacity, and network topology. In this paper, we describe a novel approach that effectively utilizes physical proximity information.

Effectively utilizing topology information involves two aspects: techniques to generate proximity information and ways to use this information. There are three ways to generate proximity information: *expanding-ring search, heuristics*, and *landmark clustering*. Expanding-ring search has at least two limitations: it has to blindly flood a large number of nodes to obtain a reasonable result. In addition, a node needs to measure round-trip time (RTT) to all the nodes directly or indirectly contacted and hence has the potential to become a bottleneck. To reduce the degree of blindness in expanding-ring search, heuristic based approaches such as hill climbing have been proposed [17]. A common limitation of heuristic approaches is local minimum pitfalls, which may fail the search for the closest node. Landmark clustering is based on the intuition that nodes close to each other are likely to have similar distances to a few selected landmark nodes. As a coarse-grained approximation, it is not very effective in differentiating nodes within close distance, according to our study.

Techniques to exploit topology information in overlay routing include *geographic layout*, *proximity routing* and *proximity neighbor selection* [3]. With geographic layout such as topology-aware CAN [12], the overlay structure is constrained by underlying network topology. This technique, unfortunately, can create uneven distribution of nodes in the overlay, increasing the chances of overloading nodes and rendering the maintenance cost formidable. Our study shows that, for a typical 10,000-node topology-aware CAN, 5% nodes occupy 85-98% of the entire Cartesian space, and some nodes have to maintain 450-1500 neighbors. In Proximity routing, physical topology is not considered when constructing the overlay. Instead, a message is forwarded to the topologically closest node among the next hop candidates in the routing table [15]. The choices for each routing hop, unfortunately, are limited to entries in the routing table. In proximity neighbor selection, routing table entries are selected according to proximity metric among all nodes that satisfies the constraint of the logical overlay. For instance, in Pastry, the constraint is the nodeId prefix.

In theory, proximity neighbor selection is superior than the other two approaches, but existing overlay construction algorithms taking this approach have their own limits. For instance, Pastry assumes *triangle inequality* in the topology. It relies on the ability to find the physically closest node at node join and uses expanding-ring search or heuristic for this purpose. Studies [14] have shown that triangle inequality may not hold in Internet topology. In fact, study from Pastry has shown that the proximity approximation is much worse when using the Mercator topology that is based on the real measurements of the Internet [3].

A further problem relates to the dynamism in the overlay. As nodes join (depart) or network conditions flux, routing tables of existing nodes need to be repaired to reflect the changes. Finding all affected nodes is a challenging task. Without timely fixes, the structure of the overlay will digress from optimal as inefficient routes gradually accumulates in routing tables. The main limit of existing approaches is the lack of global state of the system when constructing and repairing the overlay, which could result in either bad proximity approximation or excessive communication.

In this paper, we address problems related to both generating and using proximity information. To eliminate the blindness in expanding-ring search and heuristic-based approaches, and also impression of landmark clustering, we propose to use landmark clustering only as a preselection process to locate nodes that are possibly close, and then measure RTTs to identify the node that is actually the closest, achieving both efficiency and accuracy. Our experiments show that when guided by landmark clustering, 20-30 RTT measurements can be enough to locate the closest node to a particular node in a topology with approximately 10,000 nodes.

To effectively use the proximity information generated, we propose to store information of the system as *soft-state* in the system itself, taking advantage of its self-organizing and fault tolerant nature. In particular, we use landmark clustering to control the placement of proximity information such that information about nodes that are physically close to each other are stored logically close to each other in the overlay. Each node is assigned a landmark number that reflects its physical position in the network. A node uses its landmark number as the key to access relevant proximity information in the overlay.

In this paper, we make the following contributions:

- Combining both landmark clustering and actual measurement to generate proximity information, achieving both efficiency and accuracy.
- A novel landmark clustering scheme to group nodes close to each other, and using space-filling curve to reduce the dimensionality of landmark cluster to generate a single landmark number.
- Use the overlay itself to store proximity information as soft-state such that nodes in the system act as rendezvous points for each other to discover nodes that are physically close.
- Pub/sub functionality that allows nodes to subscribe to relevant soft states using its landmark number as the key, and get notified as state changes necessitate neighbor re-selection.
- Last, a quantitative breakdown of sources of performance penalty, including those imposed by the structural constraints of the overlay, and those due to inaccuracy of proximity generation techniques.

We evaluate our techniques using eCAN [19], a hierarchical variation of CAN. The remainder of the paper is organized as follows. We discuss related work in Section 2, and give background in Section 3. Section 4 describes our techniques for proximity search that combines both landmark clustering and actual measurements. In Section 5, we describe how global information of the system can be stored in the overlay network in a controlled way to facilitate overlay construction and maintenance. We discuss in Section 6 other uses of global states. We conclude in Section 7.

2 Related Work

We compare our work with related work in proximity generation and proximity-aware overlay construction.

Several techniques have been proposed to estimate Internet distance. IDMaps [6] places *tracers* at key points in the Internet. These tracers measure the latency among them and advertise the measurements to clients. The distance between two clients A and B is estimated as the sum of the following: the distance between A and its closest tracer A', the distance between B and its closest tracer B', and the distance between tracer A' and B'. The accuracy of IDMap improves as the number of tracers increase.

A second approach is the *landmark ordering* technique used in topology-aware CAN [12], a node measures its round-rip time to a set of landmarks and sorts the landmarks in terms of increasing RTT. Thus, every node has an associated order of landmarks. Nodes with the same (similar) landmark order(s) are considered close to each other. This technique cannot differentiate nodes with the same landmark orders.

A third approach is coordinate-based [5]. In this approach, landmark nodes measure the RTTs among themselves and use this information to compute a coordinates in a Cartesian space for each of the landmark. These coordinates are then distributed to clients, which measure RTTs to landmark nodes and also compute a coordinates in the Cartesian space for itself, based on the RTT measurements and the coordinates of landmark nodes. The Euclidian distance between nodes in the Cartesian space is directly used as an estimation of the network distance.

Comparing with above algorithms, our approach does not rely on any centralized server or global knowledge, and the landmark numbers generated using space filling curve [1] can be mapped to points in overlays of any dimension.

Miguel Castro *et al* [3] divide techniques used to exploit network proximity into three categories: geographic layout, proximity routing and proximity neighbor selection. Proximity neighbor selection is superior in terms of load balancing and proximity approximation. The existing algorithms that belong to this category, however, rely on expanding-ring search or heuristics for bootstrap and a gossiping protocol for maintenance. Both may require extensive message exchanges to achieve reasonable accuracy, especially when the proximity information kept in the overlay has already digressed from optimal.

Even with proximity neighbor selection, the nearest neighbor selection is still constrained by the logical structure of the overlay. Without this constraint, P2P routing protocol [20] similar to the distance vector routing algorithm can achieve efficiency comparable to IP routing, but it is not suitable for a very dynamic environment because of the frequent propagation of routing information.

In existing P2P networks, our contribution of using the archival nature of the system to store and retrieve relevant system information to gain performance advantage is unique. Self-archiving of system information has been explored in other areas, e.g., GLS [9]. However, their goal is to assign an appropriate number of location servers for each mobile node, rather than efficient routing.

3 Background

This section provides a short description of CAN and eCAN. In theory, eCAN is equivalent to overlay networks such as Pastry. The Cartesian space abstraction of CAN and eCAN, however, makes them more attractive in places where the application directly demands such an abstraction, e.g., document ranking using latent semantics [16].

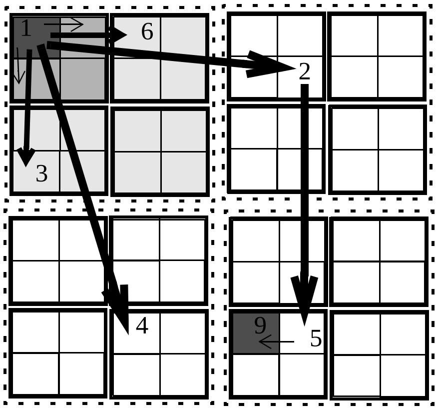


Figure 1: An example of eCAN

3.1 CAN

CAN stands for *content-addressable network*. It abstracts the problem of data placement and retrieval over large scale storage systems as hashing that maps “keys” onto “values” [4]. CAN organizes the logical space as a d-dimensional Cartesian space (a d-torus). The Cartesian space is partitioned into zones, with one or more nodes serve as owner(s) of a zone. An object key is a point in the space. The node, which owns the zone that contains the point, owns the object. Routing from a source node to a destination node boils down to routing from one zone to another in the Cartesian space. Node join corresponds to picking a random point in the Cartesian space, routing to the zone that contains the point, and split the zone with its current owner(s). Node departure amounts to having the owner(s) of one of the neighboring zone take over the zone owned by the departing node. In CAN, two zones are neighbors if they overlap in all but one dimension along which they abut each other.

3.2 eCAN

eCAN auguments CAN’s routing capacity with routing tables of larger span to achieve logarithmic routing performance. Every k CAN zones represent an order-1 zone, and k order-i zones represents an order-(i+1) zone. As a result, a node is an owner of a CAN zone and is also a member of the higher-order zones that encompass the CAN zone. Besides its default routing neighbors that are CAN zones, a node also has high-order routing neighbors that are representatives of its neighbors in the high-order zones.

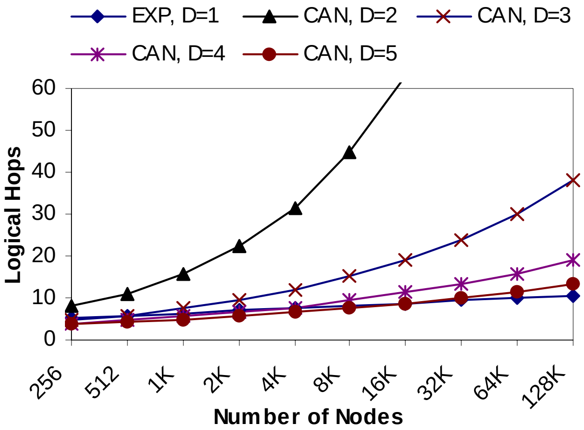


Figure 2: eCAN compared with CAN with different d

Figure 1 illustrates eCAN with an example. The default CAN zones are order-1, and each of the CAN zones is 1/64 of the entire Cartesian space. In this example, four neighboring CAN zones make one order-2 eCAN zone and four order-2 zones make an order-3 zone. For example, node 1 owns a CAN zone (the zone with dark shading in the upper-left corner), and it is also a member of the order-2 and order-3 eCAN zones that enclose the CAN zone. The routing table of node 1 consists of the default routing table of CAN (represented by the thin arrows) that link only to node 1’s immediate CAN neighbors, and high-order routing tables (represented by the thick arrows) that link to one node in each of node 1’s neighboring high-order zones. Figure 1 also illustrates how node 1 can reach node 9 using eCAN routing (1-2-5-9). Figure 2 shows that eCAN with the lowest dimension (d=1) easily outperforms the basic CAN up to d=5.

Introducing eCAN is not the main point of the paper, please refer to [19] for details on eCAN construction as well as its routing algorithm. Among the current proposals, eCAN is probably the simplest in reaching O(logN) routing performance by riding on the basic CAN protocols.

It should be noted that, eCAN is similar to Pastry in that there exists flexibility in selecting the high-order neighbors. When selecting a high-order neighbor, it can select the node that is closest to the current node among all nodes that are a member of the neighboring high-order zone.

4 Generating Proximity Information

Finding an effective way to generate proximity information is crucial for topology-aware overlay networks to work well. The proximity information is usually used to partition nodes into clusters [12], or to estimate the distance between nodes directly [5].

As described in Section 1, three techniques have been proposed to address this problem: *expanding-ring search*, *heuristics*, and *landmark clustering*. With expanding-ring search, to find a node that is closest to a particular node A, node A first contact the nodes that it knows and then have those contacted nodes in turn contact nodes they know until a radius (in terms of network hops) is reached. The major limitation of expanding-ring search is that node A has to measure RTTs to a large number of nodes to obtain a reasonable result. Heuristic based approaches are likely to contact a smaller number of nodes, but they may stumble at local minimum and fail in finding the closest node.

Landmark clustering is based on the intuition that nodes close to each other are likely to have similar distances to a few selected landmark nodes, although details may vary from system to system. With landmark ordering (topology-aware CAN), a node measures RTTs to each of these landmarks and sorts the landmarks in terms of increasing RTTs. Nodes with the same or similar landmark order are considered close to each other. In coordinate-based approaches[5], the measured RTTs to landmarks are used to compute a position in a Cartesian space for each node. The Euclidian distance between nodes in the Cartesian space is directly used as an estimation of network distance.

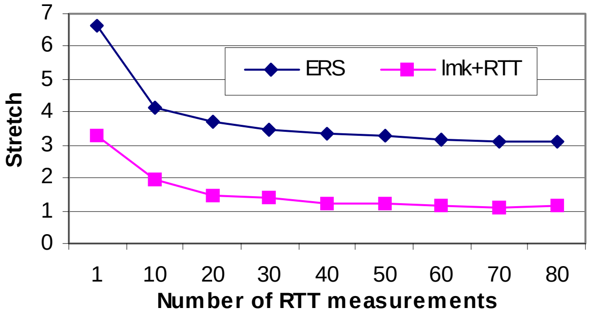


Figure 3: Comparison of expanding-ring search (ERS) and our hybrid approach in finding the nearest neighbor, using topology “ts10k-large”.

Because landmark clustering is a coarse-grain approximation, our study shows that it is not very effective in differentiating nodes within close distance. To solve this problem, we propose a hybrid approach that uses landmark clustering only as a preselection process to locate relatively close candidates, and then sorts the candidates based on the landmark metric. Finally, it measures RTTs to a few top candidates to select the closest node.

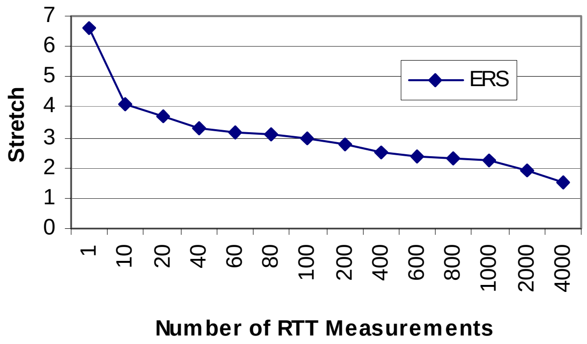


Figure 4: The effect of expanding-ring search in finding nearest neighbor, using topology “ts10k-large”

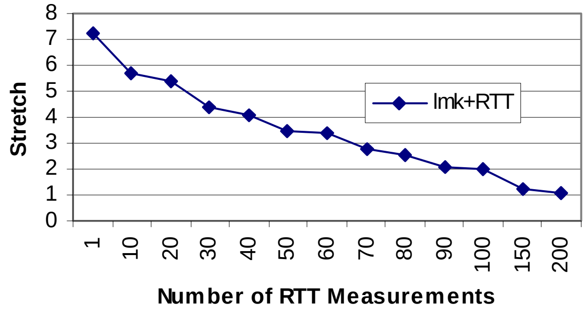


Figure 5: The effect of our hybrid approach in finding the nearest neighbor, using topology “ts10k-small”

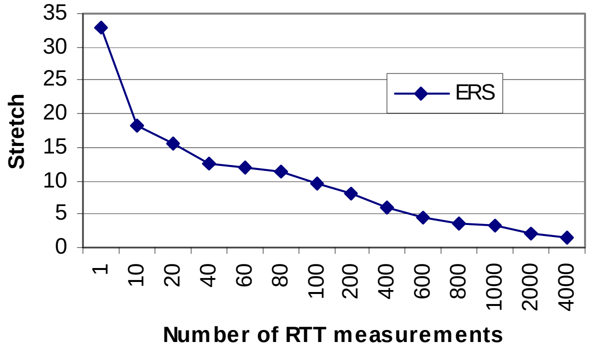


Figure 6: The effect of expanding-ring search in finding the nearest neighbor, using topology “ts10k-small”

To evaluate the effectiveness of the various approaches described above, we compare three approaches with simulation: expanding-ring search (ERS), landmark only, and our hybrid “landmark+RTT” approach. The evaluations in topology-aware CAN show that its performance is comparable to a variant of the coordinate-based approach. Our study confirms this finding. For the sake of clarity, we present only the result for landmark ordering here. For heuristic based approaches, there exists a great diversity among them. Since they can be viewed as a kind of guided flooding, we can get a flavor of their performance from the simple expanding-ring search. The metric used to evaluate the algorithms is *stretch*, defined as the ratio of the distance between a node A and its nearest neighbor found by the algorithms to the distance between A and its ideal nearest neighbor.

We use GT-ITM[18] to generate two transit-stub topologies with approximately 10,000 nodes each. The first topology, “ts10k-large”, has 228 transit domains, 5 transit nodes per transit domain, 4 stub nodes attached to each transit node, and 2 nodes in each stub domain. The second topology, “ts10k-small”, differs from “ts10k-large” in that it has only 25 transit domains, but there are 20 nodes in each stub domain. Intuitively, “ts10k-large” has a larger backbone and sparser edge network (stub) than “ts10k-small”. We choose “ts10-large” to represent a situation in which the overlay consists of nodes scattered in the entire Internet and only very few nodes from the same edge network join the overlay. (The effect of generating a very small stub domain is similar to first creating a stub domain with a large number of nodes and then choosing only a few of them to join the overlay.)

In the landmark approaches, we randomly choose 15 nodes from the topology as the landmarks. For expanding-ring search, we construct a 2-dimensional CAN consisting of all nodes in the topology. We randomly pick 1000 nodes from the topology and report their average stretch. The results are presented in Figure 3-6, where “lmk+RTT” is the result of our hybrid approach. The first points on the “lmk+RTT” series—the one with one RTT measurement—corresponds to the results for the “landmark only” approach.

Three conclusions can be drawn from the figures. First, expanding-ring search is not effective in finding the nearest neighbor unless a large number (thousands) of nodes have been tested, implying that heuristic approaches are also unlikely to work well by visiting only a small number of nodes. Second, landmark techniques on its own is not effective in finding the nearest neighbor, but our hybrid approach greatly improve its accuracy with only a medium amount of RTT measurements. Finally, finding the nearest neighbor in a dense edge network is harder than that in a sparse edge network, but the performance of our hybrid algorithm improves quickly while the number of RTT measurements increases.

On the whole, finding the nearest neighbor is a difficult problem. For the “ts10k-small” topology, even our hybrid algorithm has to test about 150 nodes to achieve a result close to the ideal case, because the landmark technique cannot differentiate nodes in stubs close-by. In designing topology-aware overlays, this property must be considered. Taking Pastry as an example. It heavily relies on the ability to find the nearest neighbor for bootstrap, but its expanding-ring search or heuristic algorithm cannot work well, as is demonstrated in our experiments. Moreover, because of the lack of global information, the bootstrap process transitively relies on every node on the bootstrap route to find its nearest neighbor, resulting in routes with increasingly accumulated inefficiency.

5 Tuning towards Network Conditions via Global State

Our experiments have shown that combining landmark clustering with actual measurement is quite effective. The challenge is to effective use this information. Although Pastry’s algorithms utilize the proximity information that is kept in the overlay’s routing tables, the gossiping protocol they use for overlay maintenance may require extensive message exchanges to achieve reasonable accuracy in proximity approximation, especially when the proximity information kept in the overlay has already digressed from optimal. The major limitation of their approach, to our opinion, is the lack of global state.

We propose an alternative approach based on controlled placement of global state to achieve good proximity approximation without excessive massaging. In particular, we use landmark clustering to control the placement of proximity information such that information about nodes that are physically close to each other are stored logically close to each other in the overlay. Each node is assigned a *landmark number* that reflects its physical position in the network. A node uses its number to access relevant proximity information in the overlay. Nodes in the system act as rendezvous points for each other to discover nodes that are physically close. To allow the overlay to effectively adapt to the dynamism in the network, a node subscribes to relevant soft states using its landmark number, and get notified as the state changes necessitate neighbor re-selection. Based on these techniques, we have implemented a topology-aware overlay, and a scalable simulator on Linux machines.

In the sections that follow, we first describe the structure and content of the global state and how nodes use the global state to perform proximity neighbor selection. We then describe a pub/sub system that enables efficient overlay maintenance. Last, we present an evaluation of our techniques via simulation. (In the appendix, we show how a *landmark number* that approximates its position in the physical network can be generated.)

5.1 Structure and Contents of Global State

Without loss of generality, we use eCAN as the example, but the techniques described here can be applied directly to other overlay networks such as Pastry and Chord.

The basic idea is to use landmarks to generate proximity information and build “maps” of the proximity information for various “logical regions” in the overlay. For eCAN, a region is part of the Cartesian space (e.g., a high-order zone), whereas for overlays such as Pastry, a region is a set of nodes sharing a particular prefix. For each region, a map is constructed. It contains proximity information about all nodes in the region, and is also stored on those nodes. When such maps are available, any node *y* can find its physically closest neighbor in a particular region *Z* by consulting an appropriate map.

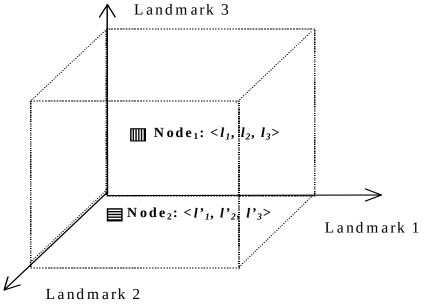


Figure 7: Landmark space for 3 landmark nodes

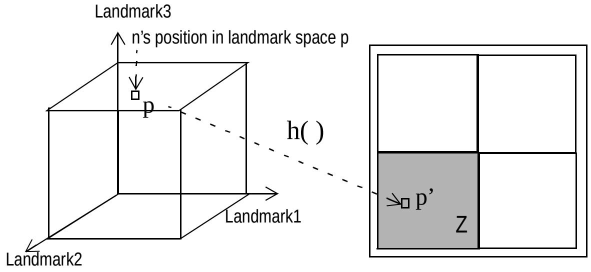
As is described in Section 4, we can use a position *p* in a coordinate space to approximate a node's position in the physical network. We call such coordinate space, the *landmark space*. We show a simple way to do this, although more sophisticated techniques [5] can be used. We pick *n* landmark nodes that are randomly scattered in the Internet. These nodes can be part of the overlay itself or standalone. Each node measures its latencies to the *n* landmarks. For node A, suppose that the measured latencies are $<l_1, l_2, \ldots, l_n>$. We then position node A in an *n*-dimension Cartesian space using $<l_1, l_2, \ldots, l_n>$ as its coordinates. We call the points *landmark vectors*. Figure 7 shows an example with three landmarks.

Usually, a sufficient number of landmarks are needed to reduce the probability of false clustering where nodes that are far away in network distance are clustered close to each other. As a result, the dimension of *p* is typically higher than the dimension of the overlay itself. To solve the dimension mismatch problem, we introduce a hash function

`p'= h(p,dp, dz, z)`

where *dp* is the dimension of *p*, *z* is the region in which p's proximity information is about to be stored, *dz* is the dimension of region z, and *p'* is a position in region z. We call p' called the *landmark number* of the node. With the hash function, if p1 and p2 are two points close in the landmark space, they will be mapped to two points that are close in region z. We will show an example hash function in the appendix.

Figure 8 illustrates this using eCAN as an example. In this example, we store the information of a node *n*, whose position in the landmark space is *p*, onto zone *Z*. We first compute n's position in Z by invoking the hash function `p'=h(p, 3, 2, Z)`. We store the triple <Z, *n*, *p*> as an object in the node that owns *p'*.



(1) Position of n in landmark space (2) Position of n on Map

Figure 8: Compute n's position in a Map

As describe in Section 3, any node *x* is an owner of a CAN zone and is also member of all the high-order zones that enclose its CAN zone. For other nodes that are physically close to *x* to select *x* as a high-order neighbor, *x*'s information needs to be published in maps corresponding to those high-order zones. Therefore, there is one map for each high-order zone in the system. (For Pastry, there is a map for nodeId prefix for each level of the routing table). It follows that each node will appear in a maximum *of* $\log_K N$ such maps. Assume that the total number of nodes *N* is 2^{20} and *K* is 4, this number is 10. We believe that this is not a big issue.

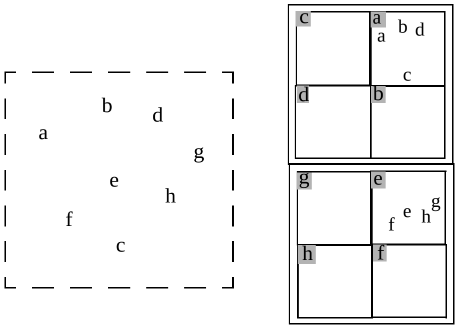
```
Let px be x's position in the landmark space;
Map px to px' in Z;
Route to the node y in Z that owns px';
If (y's map content is not empty)
    Return map content
Else
     Define a TTL to search outside y's map content range.
```

Table 1: Procedures for locating the closest node in a zone

The original node join procedure for CAN is slightly modified, and we refer the readers to our technical report [19] for more details. Now, when a node is looking for candidates in a high-order zone *Z* that is close to it, it uses its own landmark number to index into *Z*'s map, as is shown in Table 1

Note the map is stored among the nodes comprise the target region. When a node uses its own landmark to index into the map, it's possible that the node it reached owns a piece of the map recording no nodes. Techniques to deal with this are discussed in [19]. Due to space limit, we only explain the "condensed map" idea here. Simply put, the map is stored in a fraction of region it covers. We define the ratio of map size to the size of the hosting zone as *condense rate* of coordinate map.

Figure 9 puts everything together with an example. Figure 9-1 depicts 8 nodes (*a* to *h*) and their positions in the



(1) (2)

Figure 9: Storing and retrieving coordinate maps

landmark space with two landmarks. These nodes are distributed in a 2-d eCAN as shown in Figure 9-2, where *a-d* and *e-h* correspond to two neighboring high-order zones. Each node's CAN zone are those small squares with owner's ID in shaded box. Without using the global state, each node simply randomly pick one node from the neighboring zone as its high-order neighbor. For instance, *a* can select either *e, f, g* or *h*, without considering physical locality. With the global state in place and with a map condense rate of 1/4, we can do much better. In this case, *a-d* publish their positions in the grid owned by *a*, where *e-h* publish in the grid owned by *e*. Now when *a* selects its high-order neighbor, it uses its own network coordinate and consults the global state of its neighbor which is stored in *e*, and find that *e* is physically closest. Thus *a* uses *e* as its representative for the zone that comprise *e-h*. Likewise, *c* will select *f*.

5.2 Overlay Maintenance using Pub/Sub

Because of the dynamic nature of the network, a node should periodically check the target high-order zone's map to see whether more favorable nodes are available. The frequency of the checking ideally should be conducted in a demand-driven fashion when the network condition has changed to an extent that necessitates a node to make a re-selection of the neighbors. To accomplish this goal, we propose to introduce pub/sub functionality to the global state. A node specifies the conditions under which it should get notified. This condition could be "notify me when 5 more nodes have joined the zone", etc. With the overlay already in place, when the conditions are triggered, the notifications can be efficiently disseminated to all subscribers through distribution trees embedded in the overlay itself.

The accuracy of the global state can be lazily maintained. In the most reactive case, departed nodes are deleted from the global state only when they are selected as high-order replacements and later found un-reachable. Alternatively, each owner of the map information can periodically poll the liveliness of the nodes. The most proactive measure is to update the map when a node is about to depart.

5.3 Evaluation of Algorithms

We use the topologies described in Section 3 to evaluate our algorithms. With a given topology, "ts10k-large" or "to10k-small", we experiment with two ways to set latency for links in the graph. The first one uses the default latency generated by GT-ITM. In the second setting, the latency is set manually according to the following rules: 100ms for cross transit links, 20ms for links connecting nodes inside a single transit, 5ms for links connecting a transit node and a stub node, 2ms for links connecting nodes inside a single stub.

We choose CAN with d=2 to give a reasonable fault-tolerance capability. We conduct several sets of experiments. Table 2 summarizes the parameters that we vary, and default values we use throughout the experiments. The only metric that we use is *stretch* defined as the ratio of accumulated latency in the actual routing path to the shortest path latency from the source to destination. Unless otherwise stated, measurements are made for twice the number of nodes in the overlay.

Parameters	**Default**	**Range**
# nodes	4096	512-8192
# landmarks	15	5-15
# RTTs	20	0-30
landmark vector index	5	5
Map condense rate	0.1	0.1

Table 2: Parameters for the experiments

In the first set of experiments, we study the effectiveness of varying the number of landmarks and number of RTTs. In Figures 10-13, we show the results for landmarks number 5 and 15, and varying the number of RTT measurements from 0-30.

Figures 10 and 11 compares the difference between topologies with latencies set by GT-ITM and manually. The *optimal value* corresponds to the results when the number of RTT measurements is infinity, meaning that the routing neighbor is the closest one in the target zone. As we can observe from the figures, increasing the number of landmarks is more effective for topology with latencies set manually. This is because the landmarks can better cluster the nodes when the latencies are more regular. When latencies are more regular, the distances to different landmarks can better differentiate the positions of nodes in the system. For the same reason, the stretch better approximates the optimal for the topology with link latencies set manually.

Figures 12 and 13 shows the stretches when varying number of landmarks and RTT measurements for the topologies with small transits. As we can see from the figures,

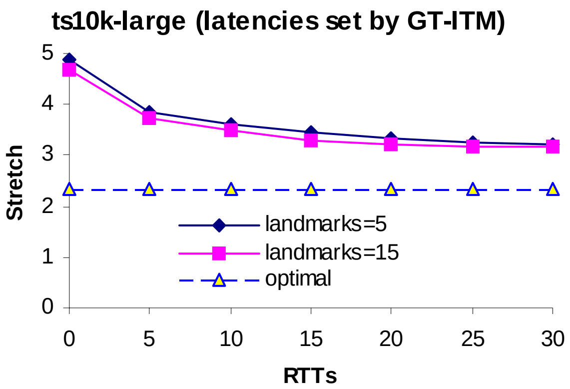


Figure 10: 10k nodes with large transits. Number of nodes in the overlay is 4096. Latencies set by GT-ITM

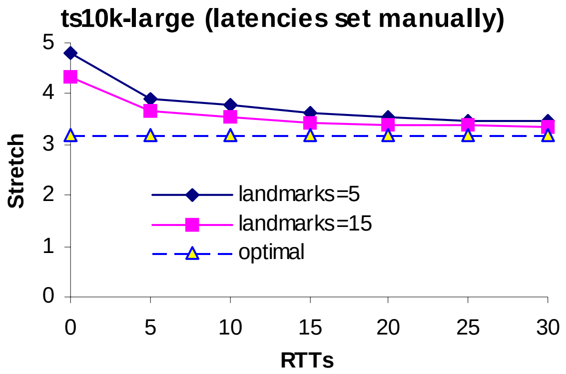


Figure 11: 10k nodes with large transits. Number of nodes in the overlay is 4096. Latencies set manually

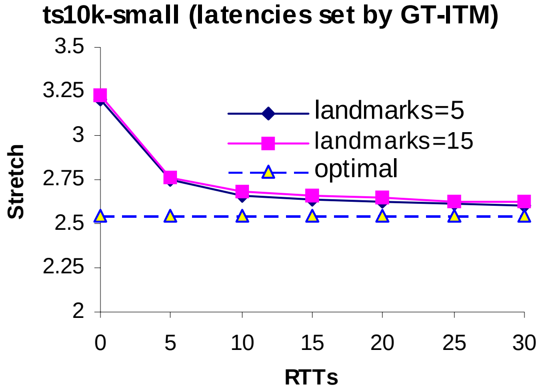


Figure 12: 10k nodes with small transits. Number of nodes in the overlay is 4096. Latencies set by GT-ITM

varying the number of landmarks is not as effective for topology with small transits as for topology with large transits. This is because the distance variation in a small network is smaller than that in a large network, requiring smaller number of landmarks to differentiate nodes at a coarse grain. Because the penalty of choosing a suboptimal route in a small network is less severe than that in a large network, its performance is also closer to optimal. Same as topologies with a larger transit, topology with latencies set manually tend to perform better.

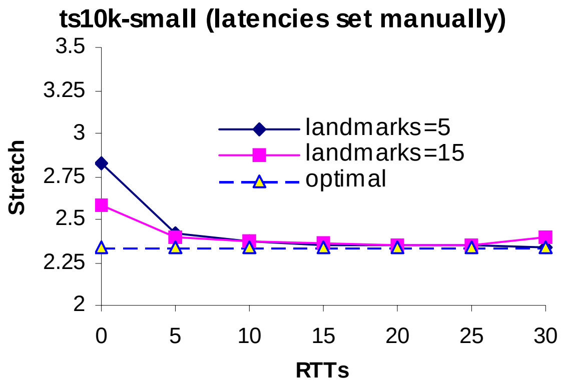


Figure 13: 10k nodes with large transits. Number of nodes in the overlay is 4096. Latencies set manually

In the second set of experiments, while fixing the number of landmarks to 15, and the number of RTT measurements to 20, we vary the number of nodes in the system and compare the performance improvement over the default case where routing neighbor is selected randomly from the target zone. The results are shown in Figure 14 and Figure 15.

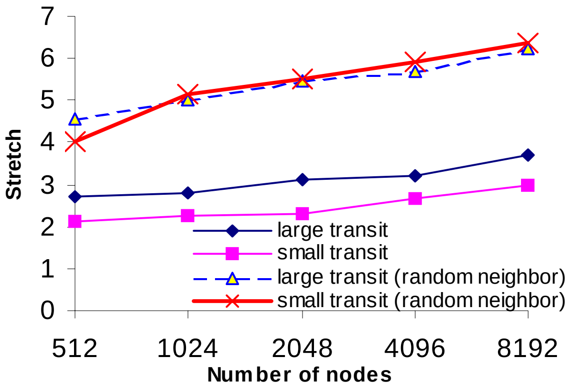


Figure 14: Topology with latencies set by GT-ITM.

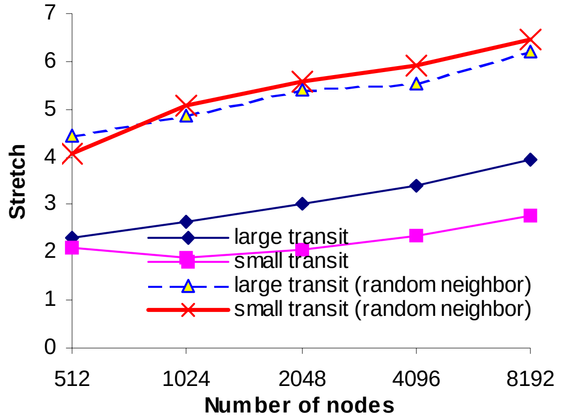


Figure 15: Topology with latencies set manually.

We can observe the following from the two figures: (1) introducing global state via landmark clustering improves the stretch by 50~75%. (2) The improvement is more significant for topologies with small transit and large stub graphs. This is because the less severe penalty for choosing a suboptimal route. (3) The performance difference between topologies with small and large transit is more prominent when the latencies are set manually. This is because the distance among nodes in a stub graph is more regular.

We also studied the effect of map condense rate and found that as long as there are about 30 entries on each node, the performance impact is negligible. To give the reader a flavour, Figure 16 shows an example for the topology ts10k-large with latency set by GT-ITM. In the figure, the dashed line shows the number of map entries per node and the solid line shows the corresponding stretch. Because landmark clustering tend to cluster nodes together in the landmark space, we have to set map reduction rate larger than 1 to actually enlarge the map to cut down number of map entries per node.

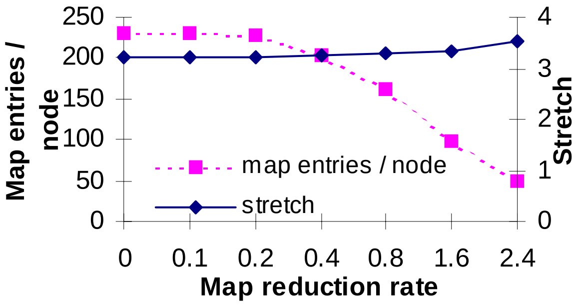


Figure 16: Effect of varying number of map entries / node. 4096 measurements are made for 4096 nodes in the overlay

We have planned to quantitatively compare our results with other topology-aware overlays such as Pastry. Unfortunately, the simulator that we downloaded from the Pastry site ran out of memory when running configurations comparable to those used in our experiments. The readers may notice that some of the Pastry numbers are better than ours. This is mainly because of two reasons: First, the backbone sizes of the topologies we have used are perhaps the largest in most studies, which is closer to the real Internet. Comparing the results for large and small transits in Figure 14 and Figure 15 conforms that it is easier to achieve good stretches with smaller backbones. Second, the optimal numbers reported in Pastry assumes that a node is always able to find the physically closest node at node join, and the network is able to completely repair itself as nodes join and depart. This requires excessive message exchanges without a global state.

5.4 Pushing Limits of Overlay Performance

In an ideal world, the performance of a topology-aware overlay should be able to approximate that of IP routing. In reality, we can observe two performance gaps in Figure 10 to Figure 13.

The first gap is between shortest paths and the optimal cases where eCAN can always find the nearest high-order neighbor that satisfies the prefix constraint. This is the value of the stretch curve corresponds to the “optimal.” The increase is about 100~150%. This is the price for meeting prefix constraint in selecting neighbors. Without this constraint, P2P routing stretch can be reduced to 1, using a protocol [20] similar to the distance vector algorithm, but it has limitations as described in Section 2. This gap is the price for the DHT abstraction and tolerance for network dynamism.

The “landmark+RTT” approach we used adds the second performance gap on top of the “optimal” stretch imposed by overlay constraint. The good news is that our technique cuts down 50~75% latency when compared with random neighbor selection, and approaches the “optimal” for topology with smaller backbones. Additional optimizations can only improve this second gap. We include some of the ideas below.

The first approach is to divide a large number of landmarks into groups, and each node computes a set of landmark positions. All these positions are then joined together to eliminate false clustering. A second approach is to perform hierarchical measurements, a small widely scattered landmarks are used to do a preselection, and localized landmarks are then selected to refine the result.

Our third approach is more radical. We propose to use a large number of randomly selected landmarks and rely on classical data analysis techniques such as Principal Component Analysis and Singular Value Decomposition to automatically extract useful information from the large number of RTTs and to suppress noise. Given the preprocessed landmark information, we use artificial neural network to automatically learn an optimal function to estimate Internet distance. Our preliminary results on this approach has shown one order of magnitude improvement in the accuracy of distance estimation. We are currently working on integrating it into our topology-aware overlay and expect its performance to approximate that of the optimal case.

6 Other Uses of Global States

The advantage of global state can be explored in other areas as well. Examples include congestion control, meeting quality of service (QoS) guarantee, taking advantage of heterogeneity in storage capacity and forwarding capacity, etc.

Nodes that are situated close to routers and gateways tend to have better forwarding capacity than other nodes in the system. The dynamic nature of the Internet traffic also causes the load at nodes to flux, which may cause temporarily congested bottleneck for the system. To better balance the traffic based on each node's capacity and current load, a node periodically publishes these statistics along with its proximity information. Nodes can trade off network distance with forwarding capacity and current load while selecting neighbors. A full set of algorithms balancing forwarding capacity with traffic is offered elsewhere [21].

If a node concerns QoS, it can subscribe not only to proximity information but also to the load statistics, specifying the conditions under which it should be notified, e.g., “the selected neighbor is handling 80% of its maximum load capacity”. When such a condition is triggered, the node starts a new round of neighbor selection in order to find better routes.

7 Conclusion

The central concepts of our proposals include the following:

1. Combining landmark clustering and RTT measurement for proximity information generation.
2. Controlled placement of system information (such as proximity and load information) as objects stored on the system itself, in a way that is easy to update and retrieve.
3. Pub/sub functionality that allows nodes to subscribe to the relevant soft-state using its landmark number as the key, and get notified as the state changes necessitate neighbor re-selection.

Our techniques are essential in exploiting the underlying conditions for overlay network construction and maintenance. The techniques are generic for overlay networks such as Pastry, Chord, and eCAN, where there exists flexibility in selecting routing neighbors, and for constructing unconstrained auxiliary network as described by Xu et al which can deliver optimal routing performance [20].

8 Acknowledgements

Artur Andrzejak suggests using space-filling curves. We thank Dejan Milojicic for valuable comments to a draft of this paper.

References

[1] T. Asano, et al. Space Filling Curves and Their Use in the Design of Geometric Data Structures. *Theoretical Computer Science*, 181:(1), pp.3-15.

[2] W. Bolosky, J. Douceur, D. Ely, and M. Theimer. Feasibility of a Serverless Distributed File System Deployed on an existing set of Desktop PCs. In *Proceedings of SIGMETRICS 2000*, Santa Clara, CA, June 2000.

[3] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting Network Proximity in Peer-to-Peer Overlay Networks. *International Workshop on Future Directions in Distributed Computing (FuDiCo), Bertinoro, Italy, June, 2002*

[4] F. Dabek, M. Frans Kaashoek, D. Karger, R. Morris, I. Stoica. Wide-area cooperative storage with CFS. *ACM SOSP 2001*, Banff, October 2001.

[5] T. S. Eugene, Ng, H. Zhang. Towards global network positioning. *ACM SIGCOMM Internet Measurement Workshop 2001*. San Francisco, CA.

[6] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz and Y. Jin. An Architecture for a Global Internet Host Distance Estimation Service. *IEEE INFOCOM 1999*, pp. 210-217", New York, NY".

[7] B. Karp and H. Kung. Greedy Perimeter Stateless Routing. In *Proceedings of ACM Conf. on Mobile Computing and Networking (MOBICOM)*, Boston, MA, 2000.

[8] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An Architecture for Global-scale Persistent Storage. *In Proceedings of ASPLOS 2000*, Cambridge, Massachusetts.

[9] J. Li, and J. Jannotti, D.D. Couto, D. Karger, and R. Morris. A Scalable Location Service for Geographic Ad-hoc Routing. In *Proceedings of ACM Conference on Mobile Computing and Networking (MOBICOM)*, Boston, MA, 2000.

[10] A. Medina, A. Lakhina, I. Matta, J. Byers. BRITE: Universal Topology Generation from a User’s Perspective. *Technical Report BUCS-TR-2001-03*. Computer Science Department, Boston University.April 2001.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker. A scalable content-addressable network. *SIGCOMM’01*, August 27-31, 2001, Scan Diego, CA.

[12] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. *IEEE Infocom’2002.*

[13] A. Rowstron and P. Druschel,. PAST: Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. SOSP’01.

[14] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. *ACM SIGCOMM* (August 1999).

[15] I. Stoica, R. Morris, D. Karger, M. F. Kasshoek, H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM’01*, August 27-31, San Diego, CA.

[16] C. Tang, Z. Xu, and M. Mahalingam. PeerSearch: Efficient Information Retrieval in Peer-to-Peer Networks. *HotNets 2002, SIGCOMM/CCR 2003*.

[17] M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. *HotNets 2002, SIGCOMM/CCR 2003*

[18] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an INternetwork. *IEEE Infocom’96*, San Francisco, CA, May 1996.

[19] Z. Xu and Z. Zhang. Building Low-maintenance Expressways for P2P Systems. 2001, *Hewlett-Packard Labs: Palo Alto.*

[20] Z. Xu, M. Mahalingam, M. Karlsson. Turning Heterogeneity to an Advantage in Overlay Routing. *HPL-2002-126R1, July 2002.*

[21] Z. Zhang, S. M. Shi and J. Zhu. Self-Balanced Expressway: When Marxism Meets Confucian. Technical report MSR-TR-2002-72. Submitted for publication.

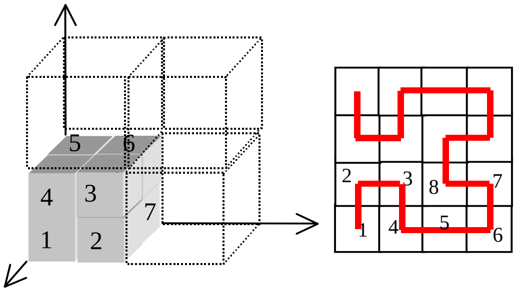
APPENDIX: Space-Filling Curves as Hash Function

As we mentioned earlier, the difficulty in storing the position in landmark space of the nodes is that the landmark space is of relatively high dimension, whereas the overlay itself can be of a relatively low dimension. We show an example of how to solve this problem using space-filling curves.

Space-filling curves map points in the domain R^1 (the domain of real numbers) into R^d (a d-dimension Cartesian space) such that the closeness relationship among the points is preserved. If two points are close to each other in R^1, they will also be close to each other in R^d. One example of space-filling curves is the Hilbert Curve [1]. The Hilbert curve is defined recursively. For an approximation level equal to 1 it is a point. For an approximation level equal to 3, it looks similar to Figure 17-2. For each higher approximation level, we subdivide the entire space into four sub-zones and copy a shrunken and possibly rotated version of the current approximation into each sub-zone.

We partition the landmark space into 2^{nx} grids of equal size (where n refers to number of landmarks and x controls the number of grids used to partition the landmark space), and number each expressway node according to the grid into which it falls. We call this number the *landmark number* of the node. Closeness in landmark number indicates physical closeness. The smaller the x, the larger the likelihood that two nodes will have the same numbering, and the finer grain the physical proximity information.

Given the landmark numbers, they can be used as keys to store information of nodes such that information about nodes that are physically close are stored logically close to each other on the overlay. For CAN, we can partition a zone into grids, and store the information about a node in a grid depending on its landmark number, again using a space-filling curve (see Figure 17-2). In the case of Chord, we can simply use the landmark number as the key to store the information of an expressway node on a node whose ID is equal to or greater than the landmark number. In the case of Pastry, we can use a prefix of the node IDs to partition the logical space into grids.



(1). Landmark numbers

(2). Stored landmark numbers on overlay

Figure 17: Mapping 3-dimensional landmark space to 2-dimension using space curve filling

Using space-filling curve to reduce a high dimension landmark vector can introduce inaccuracy. As an optimization, in stead of using the entire landmark vector to generate the corresponding landmark number, we use only a few components of it (say 5) to compute a landmark number. We call this subset the *landmark vector index*. A node uses its landmark number as key to access a map. Once it a map lookup request reached the destination node, the full landmark vector of the requesting node is used to sort the information of nodes published on that node. A maximum of X nodes that are closest to the requesting node is sent back. The requesting node then measure RTTs to this X nodes and record the node that has the smallest RTT value.