ABSTRACT

KANDEKAR, KUNAL. TAO: A Topology-Adaptive Overlay Framework. (Under the direction of Dr. Khaled Harfoush.)

Large-scale distributed systems rely on constructing overlay networks in which nodes communicate with each other through intermediate overlay neighbors. Organizing nodes in the overlay while preserving its congruence with the underlying IP topology (the underlay) is important to reduce the communication cost between nodes. In this thesis, we study the state-of-the-art approaches to match the overlay and underlay topologies and pinpoint their limitations in Internet-like setups. We also introduce a new Topology-Adaptive Overlay organization framework, TAO, which is scalable, accurate and lightweight. As opposed to earlier approaches, TAO compiles information resulting from traceroute packets to a small number of landmarks, and clusters nodes based on (1) the number of shared hops on their path towards the landmarks, and (2) their proximity to the landmarks. TAO is also highly flexible and can complement all existing structured and unstructured distributed systems. Our experimental results, based on actual Internet data, reveal that with only five landmarks, TAO identifies the closest node to any node with 85% - 90% accuracy and returns nodes that on average are within 1 millisecond from the closest node if the latter is missed. As a result, TAO overlays enjoy very low stretch (between 1.15 and 1.25). Our results also indicate that shortest-path routing on TAO overlays result in shorter end-to-end delays than direct underlay delays in 8-10% of the overlay paths.

TAO: A Topology-Adaptive Overlay Framework

by

Kunal Kandekar

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh
2006

Approved by:

Dr. George Rouskas	Dr. Carla Savage

Dr. Khaled Harfoush
Chair of Advisory Committee

To my family, friends and fellow geeks…

Biography

Kunal Kandekar was born in Satara, India in 1979. He moved a lot with his family while growing up, completing his high school in Kuala Lumpur, Malaysia. He received his Bachelor's degree in Instrumentation and Process Control Engineering from Vishwakarma Institute of Technology, Pune University, Pune, India in 2001. He worked two years as a software engineer at Cognizant Technology Solutions, Pune before pursuing further studies. Since 2003, he has been a Master's student in the Department of Computer Science at North Carolina State University, Raleigh, NC. He worked as a software development intern at IBM Corp., RTP, NC full time in Summer 2004 and Fall 2004, and part-time in Spring 2005.

Acknowledgements

I would like to thank Dr. Khaled Harfoush, my academic advisor, for all his patience, guidance and direction while working on my thesis. From him I learned how to carry out research with a focused, analytical and systematic approach.

I am grateful to Dr. George Rouskas and Dr. Carla Savage for being on my thesis committee and providing valuable insights.

Many thanks to all my friends here for their constant support, help and unwavering commitment to enjoying life, including my room-mates Rahul, Bhavna (especially for taking all the cooking turns this past couple of months) and Sushmita, and ex-roommates Nari, Vinay and Karan. Special thanks to Ratna for her regular firings that got all my work completed in time. I greatly appreciate the endless entertainment provided by Ghatol, Neel, Bapat and Maalik in the Fall of 2004. Regards to the IBM gang for all those long lunches. An honorable mention must go to Sammy's for always being there.

Most significantly, thanks to Sam for all her love, patience and pampering (as well as the occasional funding) that sustained me through my graduate studies. Last but not least, lots of love and gratitude to my parents for making all this possible!

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Overlay networks offer the potential for unparalleled scalability and robustness in distributed systems by harnessing the resources of any computer connected to the Internet that is willing to participate. In overlay networks [8, 13, 22, 30], nodes communicate through intermediate overlay hops, each consisting of multiple links in the underlying Internet Protocol (IP) network (the *underlay*). A lack of congruence between the overlay and the underlay structures can severely amplify the communication latency between nodes and degrade the system's performance. Measurement studies estimate that only 2 - 5% nodes in the Gnutella network [8] have overlay neighbors in the same Autonomous System (AS), while 40% of the nodes lay in only 10 ASes [24]. Additionally, the largest overlays currently deployed are *unstructured* peer-to-peer networks like Gnutella [8] and Kazaa [13], which rely on flooding or random overlay walks and hence are routing-inefficient. Furthermore, from the perspective of an Internet Service Provider (ISP), inter-AS traffic is more expensive than intra-AS traffic. Therefore, topology-aware overlay organization is essential to reduce this mismatch without sacrificing scalability, modifying existing applications, or changing other aspects of the overlay network.

Consequently, there has recently been extensive research focused on devising a technique to accurately and efficiently construct topology-aware overlays. Previous approaches to this problem typically use end-to-end latency measurements and can be classified into two broad categories: *recursive probing*, where joining nodes recursively discover and probe online nodes to find the closest nodes, and *virtual coordinate* methods which embed nodes as points in a multidimensional Euclidean space to represent their relative locations in the Internet.

Both categories do not explicitly consider the inherent Internet structure and thus have serious shortcomings, and cannot guarantee topology congruence.

To address this problem, we have devised a new Topology Adaptive Overlay organization framework, TAO.[1] The concept behind TAO is simple and intuitive: Two nodes are expected to be closer to each other than to others if they share a longer path towards a *landmark* node. This heuristic is based on the hierarchical structure of the Internet topology and on the observation that nodes that share a longer path hop through more of the same ISP networks (service providers) and hence are expected to be closer. Therefore TAO nodes collect *traceroutes* to a small set of landmarks and organize themselves such that nodes having greater sharing in their IP landmark paths are placed closer together in the overlay network. For each landmark, nodes construct an overlay tree reflecting the IP shortest path tree formed by their traceroutes to that landmark, and the overlay mesh is formed by interleaving these trees for all landmarks. This is done in a completely distributed manner, and results in the formation of interconnected clusters of closest nodes. Clusters are organized in order of ascending path sharing with respect to each landmark, and clusters with identical path sharing are placed in order of increasing distance to the landmarks. Connectivity information is maintained such that new nodes joining anywhere in the overlay can be systematically routed to their appropriate clusters using their traceroute information. Thus, besides simply discovering and clustering closest nodes, TAO also provides a systematic method of creating inter-cluster connections. This framework can then be leveraged for a variety of application-specific objectives such as content distribution and application-level multicast. Despite being used universally for network topology inference, traceroute has always been considered too

[1] *Tao* is Chinese for “path”.

network intensive for use in large-scale overlays. It is a common mistake to be swayed by traditional wisdom and to overlook the potential of an unconventional approach.

The rest of this thesis is organized as follows: In section 2 we discuss related work and briefly identify the main reasons behind their limitations in sections 3 and 4. We then illustrate the feasibility and advantages of using traceroute in section 5, identifying several shared path semantics and the topology information we can infer from them. In section 6 we describe the operation of TAO in detail and briefly address practical issues like churn, which is the process of nodes continuously joining and leaving the overlay. In section 7 we describe a few possible applications of TAO. Section 8 details the experiments using simulations of some representative TAO-enabled applications using real-world Internet datasets [20] that we used to evaluate TAO, and we discuss their results in section 9. We briefly compare TAO with previous work in section 10, discuss future work in section 11, and conclude in section 12.

2. BACKGROUND

2.1. Overlay Networks

The most popular use of overlays nowadays is in peer-to-peer (P2P) networks, which are basically large-scale distributed systems to locate shared resources across the Internet. Overlays can be classified as *structured* and *unstructured*.

Unstructured overlays: These consist of peers connected in a more or less ad-hoc manner. They have no systematic resource location or routing strategy, and hence must do so by

flooding query messages more or less indiscriminately across the overlay. The lack of structure results in low maintenance overhead and more robustness, but at the expense of enormous bandwidth consumption [25]. The earliest peer-to-peer systems, like Napster [18], avoided flooding by taking a centralized approach to locate shared content. However this led to a single point of failure, prompting the development of decentralized, randomly constructed systems such as Gnutella [8]. Kazaa [13] adds a small optimization by using super-peers, which are peers with greater capacity that assume greater responsibility on behalf of their less capable peers. This introduces a hierarchy in the overlay, improving the scalability of flooding, but still suffers from topology mismatch.

Structured overlays: These provide systematic routing for distributed resource location services to avoid flooding. Structured P2P overlays like CAN [22] and Chord [30] use the Distributed Hash-Tables (DHT) paradigm to allocate overlay nodes the responsibility of serving requests for specific resources. Peers assume IDs which are used to specify their location in a rigid graph structure, as well as to determine responsibility for storing or indexing a shared resource. Thus they can deterministically route to the node hosting a requested resource. However, the rigidity of the structure increases maintenance overhead and complexity and limits robustness.

Currently, the most popular use of peer-to-peer networks are in file-sharing applications like Gnutella and Kazaa, which are unstructured overlays with millions of users, typically swapping large media files and consuming a substantial proportion of network bandwidth [28, 24]. This indicates that topology mismatch is a significant problem.

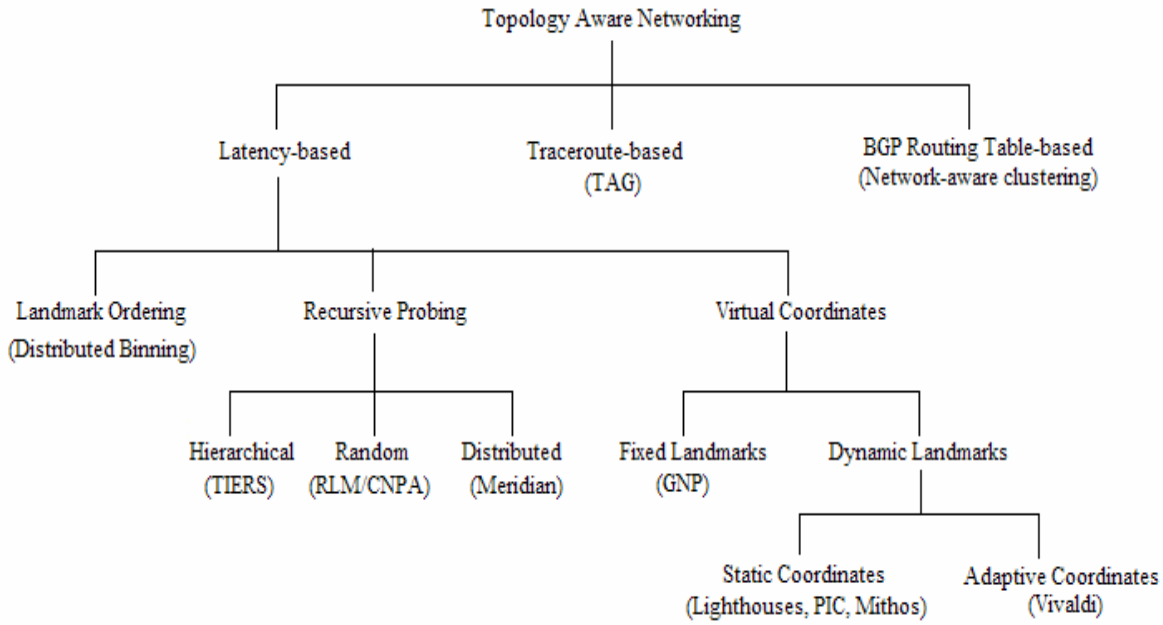


Figure 2.1: Classification of Topology Aware Overlay Techniques.

2.2. Overlay Organization

This section briefly describes the existing work done in topology-aware overlay construction. The conventional approach has been simply to discover and peer with the closest online node, thus organizing the overlay into a number of closest-node clusters. While the effectiveness of this method is uncertain (and is discussed briefly in section 6) as it provides no guidance at all on interconnecting these clusters, it would reduce the number of long overlay hops and hence certainly is a reasonable first approximation. Probing every node is the most obvious solution, but it simply will not scale beyond a few dozen nodes, and hence most of the work in this area has focused on finding the closest node in a distributed and scalable manner. The concept of scalable network distance prediction was pioneered by Francis et al in IDMaps [7], an infrastructure-based solution, followed by King [10] which used recursive DNS queries. Both provide rough distance estimates between any two nodes, and do not scale to organize nodes in large overlays.

One of the earliest approaches proposed specifically for overlays was distributed binning [23], a topology-aware variation of CAN. In [23], nodes sort landmarks in order of ascending ping latencies. Nodes with the same landmark ordering are assumed to be relatively closer together in the network, and hence should be placed closer together in a "bin" in the hash-space. Within the bin, nodes place themselves randomly per the original CAN. Some systems, such as Chord and Pastry [27], use *proximity routing* rather than constructing a topology aware overlay, whereby nodes forward messages to the topologically closest of the logical next-hop candidates in the routing table.

There have recently been a number of proposals which assign coordinates to nodes and embed them as points in an N-dimensional Euclidean space. These coordinates represent the relative locations of the respective nodes in the Internet. The pioneering work in this area was Global Network Positioning (GNP) [19]. GNP and its variations generate coordinates for nodes by geometric triangulation of their ping latencies to a small set of landmark nodes. The main advantage of these methods is that they are easy and intuitive to use, being analogous with the use of maps for inferring geographical proximity.

Other approaches use recursive probing of peers (typically discovered by gossiping) and their neighbors to discover closer nodes. These work by recursively probing the neighbors of the nearest known nodes to identify good candidates for further probing to discover even closer nodes. Tiers [2] arranges peers in a hierarchy of clusters and new peers start at the top level and iteratively identify the next cluster to target by probing every node in the current cluster. Other methods, such as [36] for Pastry take a more random approach, offering more

scalability but less accuracy. LTM [17] applies to churn-prone, unstructured overlays like Gnutella, having nodes maintain closer neighbors by adaptively dropping slower connections and peering with the closest of those discovered by flooding of 2-hop overlay neighbors every minute.

There are also schemes that use an adaptive hybrid approach for distance estimation and peer selection. Mithos [33] uses recursive probing to discover closer neighbors, applies a spring-force system between the new node and the closest discovered neighbors, and solves it to determine the coordinates for the new node. In Vivaldi [6], on the other hand, each node starts with random coordinates and continuously updates them based on measurements during the course of normal operation, emulating a spring-force system, thus avoiding the overhead of recursive probing. Meridian [37] uses a loosely structured, lightweight overlay employing *distributed* recursive probing to provide network localization. It uses Euclidean geometry with latencies as virtual coordinates as a heuristic to increase the topological diversity of its neighbors, but not for localization. It works by probing a given target node, requesting neighbors with similar latencies to probe the target and then those closest to the target recursively request their appropriate neighbors to probe the target. It can locate the closest node to a client, the central node to a set of clients, as well as nodes that satisfy multiple latency-based constraints.

Topology Aware Grouping [16] for Overlay Multicast works simply by taking a traceroute to the multicast source, and recursively arranging nodes with lesser path overlap to be the parents of those with greater overlap, minimizing link stress. To the best of our knowledge,

this is the only other self-organizing overlay to date that leverages traceroute information. Network aware-clustering of web clients [15] uses IP address prefix matching with information extracted from BGP routing tables to cluster web clients for better placement of server content.

Most of the schemes mentioned above have significant shortcomings. Binning is a very coarse-grained approach, and no systematic method has been proposed to optimize placement of nodes within bins or the placement of bins relative to each other. More seriously, our studies show this to be a poor measure of locality; we found cases in the NLANR dataset [20] where distant nodes actually mapped to the same bin, thus actually *increasing* the topology mismatch. As more landmarks result in more bins, there are a large number of "empty" bins, leading to non-uniform distribution. In fact [34] suggests that the improvement in performance is more the result of extreme zone imbalance than of peer locality. Proximity routing provides very localized optimization at each hop, since the choice of next-hop candidates is limited to nodes in the routing table. Virtual coordinate methods are lightweight but incur large relative errors. Both binning and virtual coordinates cannot distinguish between far-apart nodes with similar landmark distances, and offer no distributed technique to locate closest nodes efficiently. Recursive probing methods can effectively locate closer nodes, but involve heavy network probing for every node join, which is not feasible given the churn in practical P2P overlays. Also, besides clustering closest nodes they provide no guidance for inter-cluster organization. It should be noted that a majority of the proposals are geared towards structured overlays, whereas the most widely-deployed and bandwidth-consuming applications are unstructured overlays that rely on flooding of queries. Moreover,

most schemes have been evaluated using simulated topologies, which do not accurately depict the peculiar geometry of the Internet, and as such their real-world performance is largely untested.

It is clear that while significant work has been done in this area, despite reducing network overhead to varying degrees, almost all approaches consider only end-to-end latencies, and so were not proven to alleviate topology mismatch effectively.

3. THE ASYMMETRIC NODE SELECTION PROBLEM

Node selection is often asymmetric, meaning one node may find another node to be its closest neighbor, but the other node may not find the former to be closest to itself. A common problem we noticed in most previous closest node selection schemes is that they do not explicitly consider the asymmetric node selection problem. It is typically the joining node that discovers a subset of online nodes and decides which ones are closest to itself, hence the asymmetry of node selection may lead to loss of accuracy. For instance, node *A* may be the closest overlay neighbor for node *B*, but *A* finds node *C* closer to itself. In this case, a greedy algorithm would have node *A* ignore node *B* and prefer node *C*. Thus if node *A* joins later and discovers both *B* and *C*, it would only select *C*, thus depriving *B* of its actual closest neighbor. In the case of overlay networks this may possibly cause the overall performance to degrade, as indicated in our results. We refer to this as the *Asymmetric Node Selection Problem*.

4. THE CURSE OF GEOMETRY

Our earlier attempts at constructing topology aware overlays used a combination of binning with recursive probing for intra-bin placement and virtual coordinates to indicate optimal placement of bins relative to each other. We simply could not get significant improvements with this combined approach, so we studied the deficiencies of each technique. We discovered that due to the distinctive geometry of the Internet, with its hierarchical core-based structure, asymmetric routing and the triangle inequality violations, all latency-based approaches will invariably have limited accuracy. We refer to this problem as the *Curse of Geometry*.

Virtual Coordinate Techniques: These represent relative locations of nodes by coordinates in Euclidean space, typically calculated by triangulation of their landmarks distances. Inter-node latencies are predicted by calculating geometric distances between their respective coordinates, and hence are affected by the Curse of Geometry. To reconcile the Euclidean geometry of their calculations with the highly non-rectilinear paths followed by network probes, these techniques use higher dimensions (six to eight) in an attempt to sufficiently "warp" the Euclidean space. However, despite several variations [21, 5], these methods incur large relative errors.

We realized that these methods failed mainly because the Internet is basically a graph, and the most common motifs in graphs, such as star-topologies, simply cannot be accurately embedded in a Euclidean space with any number of dimensions. When we talk of accuracy of

embedding, we refer to the accuracy with which inter-node distances can be predicted using geometric coordinate distances.

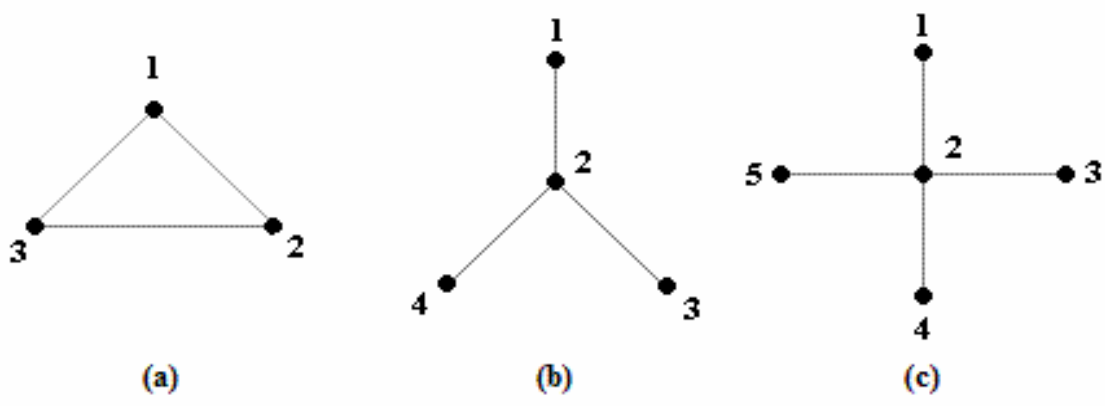


Figure 4.1: Common network motifs.

Consider some common motifs observed in the Internet, depicted in Figure 4.1. If three vertices form a triangle as depicted in Figure 4.1(a), they all can be accurately embedded in a 2-dimensional plane. However, if four vertices form a star as in 4.1(b) at most only three of them can be accurately embedded in a 2-dimensional space. The fourth vertex can not be represented without incurring a substantial error, no matter how many dimensions are used. If only the leaf vertices are to be embedded, as in GNP, the topology can be reduced to a triangle. As soon as there are more than four vertices in a star topology as shown in Figure 4.1(c), it becomes impossible to accurately embed even the leaf vertices in a Euclidean space of any dimension. At most three vertices can be accurately embedded with respect to each other but each additional vertex can be accurately embedded with respect to at most only two other vertices. If it uses the coordinates of any two as the reference, it will be pushed farther from the remaining vertex, thus incurring an unavoidable error unless higher dimensions are used. This problem will only get worse as the size, complexity and degree of the graph

increases, and increasing the number of dimensions is not an adequate solution for Internet-like graphs.

Star-like structures are quite common in the Internet, with the high degree backbone routers at the core and lower-degree routers and end hosts towards the periphery. Hence the star topology in the above example is apt, and easily explains the relative errors observed in GNP. Further problems with geometric approaches are identified below.

- **Triangle Inequality Violations:** These approaches rely on geometric techniques, and hence result in very large errors in the presence of triangle inequality violations. These are known to exist in the Internet and their causes have been studied in [38], which found significant number of violations in a number of topology datasets. We found that 26% of all distinct triangles in the NLANR dataset violate the triangle inequality.
- **Limited Relativity:** Co-ordinates which are meant to represent the relative location of a node with respect to the entire Internet are derived using measurements to a small subset of nodes in the Internet, which could be fixed (as in GNP) or chosen dynamically (as in Lighthouses, PIC, Vivaldi). Hence the resultant coordinates can represent node location relative only to these landmarks and not the rest of the nodes in the Internet.
- **Non-unique Coordinates:** GNP does not guarantee unique coordinates, and nodes with identical coordinates may not necessarily be as close as their coordinates would imply. This is obvious considering there is no reason why two nodes that are topologically far apart would not have similar distances to landmarks. Vivaldi [6]

addresses this problem by assigning an additional height vector to the calculated 2-dimensional co-ordinates representing the distance between any nodes with similar co-ordinates, and incorporating that in its predicted distance estimates.

- **Asymmetry of RTT Latencies:** A principle assumption is that latencies in RTTs are symmetric, that is, the latency each way is half of the RTT. Due to routing anomalies in the Internet, this is not always true, and leads to errors in calculations.

One approach to improve the accuracy of virtual embeddings is to take into consideration the curvature of the Internet as in [29]. This model notes the geometry of the Internet and non-rectilinear paths taken by network probes, and hence models it as a hyperbolic space rather than a Euclidean space. Connections between nodes are assumed to follow a hyperbolic curve instead of a straight line. Similarly, distances between nodes are predicted by calculating the length of a hyperbolic curve between their coordinates. It is more accurate than GNP, but the mathematics involved are rather complex.

Recursive Probing Techniques: Limited relativity is the major problem since the nodes finally selected as the closest are closest amongst only those discovered and probed. Although it may not be obvious, these techniques suffer from triangle inequality violations as well. They work on the assumption that the neighbors of the topologically nearest of the currently known nodes would be good candidates for further probing to discover even closer nodes. But due to a triangle inequality violation, the closest neighbor of a closest neighbor of a node could actually be quite far, and would actually lead it away from the closest nodes in successive probes. For instance, if node A finds node B to be its closest known peer with an

RTT of 10 ms, and if node C is B's closest neighbor with an RTT of 20 ms between them, A would expect C to be less than 30 ms away. However, due to triangle inequality violations the actual latency between nodes A and C could be much larger than that, and the largest such error we found in the NLANR dataset was 84 ms. Probing C's neighbors would probably not discover closer nodes. The effect of Internet geometry on recursive probing methods can be seen in [4] where closest node selection performs much worse for the Mercator topology, which is based on real Internet measurements, than for simulated topologies. This is a strong indication that the Curse of Geometry adversely affects recursive probing methods as well.

We concluded that the best way to represent a graph is, well, a graph. We also realized that end-to-end latencies do not provide enough information about the underlying topology and are often misleading, as they are affected by several other factors such as heterogeneity, queuing delays, and bandwidth. Clearly, a paradigm shift was needed. Hence we decided to use traceroutes to structure the overlay into a graph resembling the Internet.

5. TRACEROUTE

TAO nodes derive topology information from traceroutes to a small set of landmarks, and metrics based on shared path semantics are used to infer locality in the topology. Traceroute [32], along with ping, is amongst the earliest tools developed to provide information about the network. We show that it is actually comparable in terms of network load to previous approaches that rely on ping latencies, and proves to be an invaluable tool to extract the internal geography of the Internet.

5.1. Feasibility Analysis

Traceroute is widely available, being included by default in almost every operating system for PCs. It has several open-source variations optimized for various purposes. First, we show that traceroute is not as network-intensive as previously thought. We evaluate network load incurred in terms of "packet events", where a packet event occurs every time a given packet is forwarded on a unique link towards the router. For the sake of simplicity we assume symmetric routing. Hence a single ping from a given node to a landmark which is m hops away in the IP topology incurs 2*m* packet events (*m* each way). Pings to n landmarks incur 2*nm* packet events. Since a traceroute is essentially a ping to each subsequent hop along the IP route, a single traceroute to the same landmark incurs $m(m+1)$ packet events ($m(m+1)/2$ each way). The average number of hops observed in the Internet is also about 15, with a maximum of around 30, according to [35], while the average in the NLANR dataset is almost 11.

Most latency-based techniques, especially recursive probing schemes, need to ping a large number of machines to provide any accuracy at all. Those requiring the fewest pings are probably binning and virtual coordinate schemes, as they typically use a fixed number of specific landmarks, numbering between 4 and 15 to get sufficient accuracy. Some recursive probing techniques require at least 30 pings and others even more. Assuming an average of 15 hops between nodes and landmarks, ping-based techniques incur between 120 to 450 packet events on average. On the other hand, a single traceroute incurs an average of 240 packet events. Since we show later that five or less traceroutes are more than adequate to

achieve very high accuracy, and are only required for the very first node joins, traceroute is at least comparable to latency-based approaches. In addition, a traceroute can be represented very *concisely*. We can represent the IPV4 address and RTT at each hop in 8 bytes or less. Since the average number of hops observed in a traceroute is about 15, a typical traceroute can be encapsulated in less than 120 bytes, and only 60 bytes if RTT latencies are not used, making it easy to "piggyback" this information on application-specific messages.

Traceroutes, as with ping-based approaches, may focus network load at landmarks but we identify simple techniques to address this. Firstly, a traceroute need not be run periodically or at every node join, since the Internet topology is not that dynamic. Even if it does, there may be insignificant changes in the path to a landmark, and these can be easily accommodated. Only when a node joins for the very first time must it run a traceroute, caching the path for subsequent joins, as opposed to recursive probing methods that require probes at every join. Later, only the first few hops need be probed to ensure that the route has not changed drastically, due to host mobility or changes in the Internet.

A common problem is the failure or corruption of traceroute due to policy routers, firewalls or routers that disable traceroute. This can be solved simply by selecting landmarks that are not firewalled. Typically if ISPs disable traceroute at all they block *outgoing* ICMP packets at the gateway router and allow incoming ones, so traceroute is still feasible *from* end hosts in such ISPs. If a traceroute *to* a firewalled node is required, TCP-based tools such as tcptraceroute [31] and paratrace [12] can be used. Corrupt traceroutes can also occur because

of faulty implementation on routers, but are easily handled by fault-tolerant algorithms that are aware of possible errors.

5.2. Analysis

A single traceroute contains a wealth of information compared to simple end-to-end latency measurements:

- IP addresses of intermediate routers. These are used to compare two traceroutes and derive shared path information.
- RTTs at each hop. It is also possible to estimate the per-hop latency by deducting the RTT of the previous hop from the RTT of each hop.

Often router DNS names are available as well, which can be used to group routers belonging to the same domain. Traceroutes, however, are often corrupt in many ways, and any approach that relies on them for organization must be aware of these problems:

- Incomplete traceroutes: some paths are partially correct and then followed by missing replies up to, and often including, the last hop due to firewalls or problematic routers.
- Missing hops, probably because certain routers along the path have disabled traceroute.

Even uncorrupt traceroutes are sometimes misleading:

- Routing anomalies: sometimes two paths are partially internally disjoint, having unshared hops between segments of shared hops, where packets are forwarded along another, possibly suboptimal, path.

- Inconsistent latencies. Traceroute only provides the RTT for each hop, some of which could be misleading due to congestion and queuing delays. This leads to incorrect estimation of per-hop latencies for all subsequent hops.

These problems are resolved fairly easily. Selecting landmarks intelligently is the obvious first step. Corrupt traceroutes can be handled by programming the path comparison algorithm to anticipate and compensate for errors.

5.3. Shared Path Semantics

Here, we describe an algorithm to compare traceroutes of nodes to landmarks and identify the resulting shared path metrics that can be used to infer topology information.

Path Comparison Algorithm: When comparing the paths of two nodes to the same landmark, start at the last hop (the landmark) and compare the IP addresses at each hop towards the nodes. Hence the discovered topology is "seen" from the perspective of the landmarks. The number of routers with the same IP address gives the count of hops shared along both paths. Routing anomalies and corrupted traceroutes are accounted for by scanning both paths entirely for common routers, instead of stopping at the first unshared hop. The last shared hop is a much more reliable indication of true topological proximity, and is preferred for determining the count of shared path hops, discarding any intermediate disjoint paths.

Despite having access to extensive data that could be used to substantially improve the accuracy of traceroutes to landmarks, we chose not to do so, since we cannot expect to have access to such information in a real-world deployment. Instead we focused on making the

path comparison algorithm more robust and fault tolerant to account for various traceroute anomalies.

Based on the IP addresses of intermediate routers and per-hop RTT latencies, this algorithm provides several metrics which impart additional topological information. The most significant of these are:

- **Path Shared Hop Count (PSHC)** - the number of hops shared along both paths to the landmark. Intuitively, a higher PSHC implies greater topological proximity of two nodes.
- **Path Unshared Latency Estimate (PUSL)** - the sum of estimated latencies of the unshared hops along a given path. This provides an estimate of the latency from the node to the last shared router.
- **Total Unshared Latency Estimate (TUSL)** - the sum of PUSL along both paths. TUSL provides an upper bound on the latency of a direct path between two nodes, assuming there is no shorter path available between both nodes other than along the unshared hops to the landmark, which is usually not the case given the rich connectivity of the Internet. Nonetheless, a lower TUSL implies greater relative topological proximity.

However, the per-hop latencies in traceroutes may be inaccurate due to rate-limiting of ICMP packets at routers from certain vendors, which limits the reliability of the shared and unshared latency estimates (PUSL, TUSL) metrics. Hence only end-to-end latency measurements that do not rely on ICMP packets can be considered reliable.

With traceroutes to multiple landmarks we can infer even more information about the topology. We formed the following metrics for multiple traceroutes by using the following operators on the single-traceroute metrics:

1. Maximum: the simplest operator, returning the maximum metric used.
2. Ordered: Sorts the chosen metric for paths to all landmarks in descending order. This provides a more comprehensive utilization of the traceroute data, and hence more precise topological inference.
3. Weighted Sum: This simply assigns a weight to each path, for instance a higher weight for a higher PSHC, and returns the weighted sum of metrics for all paths. However, we could not determine any method to assign weights to paths that gave more accurate results, and hence the effectiveness of this method is unknown.

Probably the most significant advantage of shared path metrics over simple latency measurements is that they are unaffected by triangle inequality violations, unless the violation is caused by a severe routing anomaly.

5.4. Effectiveness

From our preliminary experiments with the NLANR dataset, we found that the Maximum PSHC using three landmarks provided the most reliable metric of topological proximity, identifying the closest node with about 75% accuracy, and average worst case relative error of around 20 - 40 ms. However the Maximum PSHC metric suffers from the “long hop” problem described below.

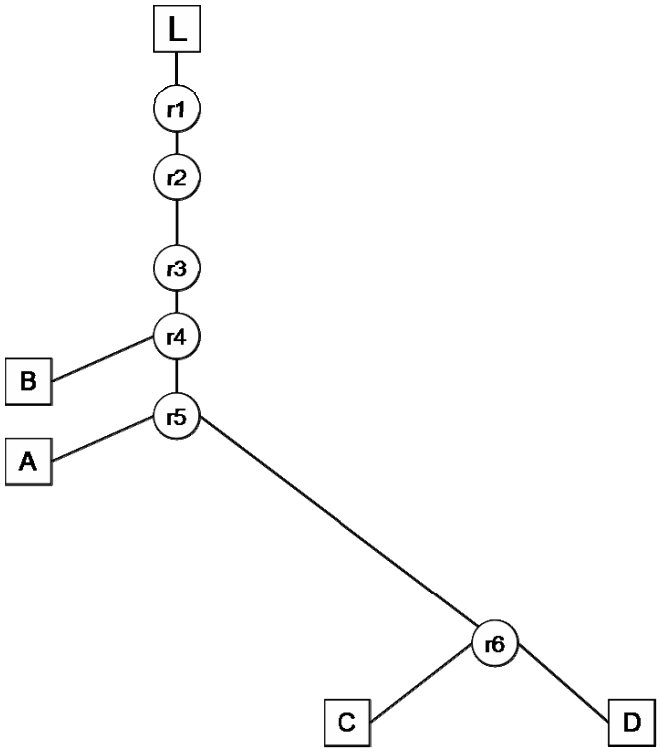


Figure 5.1: The Long Hop Problem.

The "Long Hop after Maximum Shared Path" Problem: While the preliminary results themselves are impressive, there is a major impediment to getting any further improvements. The problem is that nodes with maximum shared path can still be very far. Those that incur the highest relative errors typically have one very long (high-latency) hop on the unshared portion of the path with maximum sharing, as depicted in Figure 5.1. This long hop is also on the direct path between both nodes and hence they are not actually topologically close to each other. For instance, in Figure 5.1, node *A* is actually closer to node *B* but shares an extra hop (r3-r4) with node *C* on the path to landmark *L*. Based on greater path sharing alone, node *A* would select *C* as the closest neighbor, but the path from A to C has a long link (r4-r5), which causes the latency between nodes *A* and *C* to exceed the latency between nodes *A* and *B*. This long hop can not be identified from the PSHC alone, and this is the drawback of

using such a simple metric. This indicates that more sophisticated techniques are required to improve accuracy.

It is logical that this problem would only occur when the problematic node is very far from all the landmarks. This is typical when some nodes that are far from all other nodes, especially those that are across an intercontinental link. However if the distant node is closer to even one landmark than other nodes, this problem would occur less frequently, since this would imply that the other nodes have a greater sharing along their paths to this landmark. For instance if node *D* was used as a landmark, nodes *A* and *B* would have a greater sharing along their paths to *D* and hence would select each other. Hence landmark selection can help some nodes overcome the log-hop problem solely using maximum PSHC as a metric. However, this implies that a landmark is needed before and after every long hop to reduce its adverse effects on the overlay, which would be unpractical for large, geographically distributed networks. One idea in this case is to dynamically select nodes in the overlay itself as landmarks, and use them for better overlay organization and higher accuracy. We explore the use of this principle for *recursive tracing* as described in section 6.2

It must be noted that even the long hop problem is not always an undesirable effect. Although in the context of closest node selection this reduces the accuracy, the overlay as a whole it actually identifies the optimal nodes to use for peering between distant clusters of nodes. In Figure 5.1, nodes *A* and *B* belong to one closest-nodes cluster, and nodes *C* and *D* to another, and the long-hop situation indicates that *A* and *C* are the ideal nodes to

interconnect their respective clusters, since they have the longest shared path between two nodes that would belong to different clusters.

Used intelligently, these semantics provide the potential of almost unlimited granularity and precision in organizing the overlay for a given topology. For instance, if a node can locate another with a maximum shared path, it can reasonably assume that it is the closest available node. How to locate such nodes in a scalable, distributed manner is the focus of this paper.

5.5. Additional Metrics

Since the PSHC is prone to the Long Hop problem, additional metrics are required to identify an occurrence of this problem. One option is to use the Total Unshared Path Latency (TUSL) as an additional metric, since it would include the long hop latency. Hence a larger TUSL would indicate the presence of a long hop. However, as mentioned previously, unshared path latencies are unreliable we need to rely on end-to-end latencies. Hence we explored the use of Relative Latency Error (RLE) as a metric. The relative latency error is defined as:

$$R_{AB} = \sum_{l \in L} | D(A, l) - D(B, l) |$$

where,

$D(x, y)$ is the latency of node x to node y, and L is the set of landmark nodes.

Since the long hop latency would also be a major part of the landmark latency, it can be used to judge whether the path between an overlay node and a prospective nearest neighbor contains a long hop as compared to other prospective nodes. Hence if node A has to decide between two nodes B and C having the same maximum PSHC without direct probing, the

condition $R_{AB} >> R_{AC}$ indicates that there may be a long hop between *A* and *B*, and hence *A* should prefer *C*.

6. THE TAO FRAMEWORK

TAO identifies "appropriate" nodes based on shared path semantics derived from traceroutes. Although the concept of "appropriate node" is mostly application-specific and TAO can support various objectives, we discuss TAO in terms of locating the closest node, which is typically the most common objective. As with any distributed system, TAO can be implemented in three ways.

Centralized: The simplest way to collect and use topology information would be to use a centralized server to which all participating nodes forward their traceroutes to common landmarks. The server would store this traceroute and then compare it with previously received traceroutes to identify the probable closest nodes. This approach however suffers from the usual problems of centralized systems: limited scalability and single point of failure. Furthermore, this offers only a limited, static view of the Internet, resulting in lesser accuracy when used for a dynamic overlay. However, this approach applies well when the number of participating nodes is relatively small, so that the server has to store fewer traceroutes and coarse-grained accuracy is sufficient; for instance, for a client to locate the closest amongst a number of geographically distributed servers that are under single administrative control. This is typical in the case of Content Distribution Networks where the number of servers is relatively much smaller than typical peer-to-peer overlay sizes, and each server (or server farm) is usually assigned to serve a large area of the network topology or geography. A client

first contacts a well-known central server and provides its traceroute to a landmark (possibly the server itself). The server compares the client's traceroutes with those of the other servers, identifies the appropriate one, and redirects the client to it. Although the load requirements on a centralized server make this approach unsuitable for typical peer-to-peer overlays where most nodes are lower-end PCs, it could be used in bootstrap servers or host caches, which cache the addresses of nodes thought to be online as in Gnutella host caches, providing a more topology aware choice for joining nodes to connect to.

Unstructured: Joining nodes forward their traceroutes to all other nodes by flooding and select those with appropriate shared path semantics. This approach is simple, avoids centralization and increases robustness, but would suffer from the large bandwidth consumption of typical unstructured overlays. Further, it would only locate closest nodes clusters, and provide inadequate information for systematic and topology-aware interconnection to these clusters.

Structured: To avoid flooding as well as centralization, we devised an algorithm to organize TAO into a structure emulating the underlying topology in a systematic and distributed manner. TAO can be briefly described as a distributed graph structure representing the Internet router-level topology. Each node in the overlay "represents" some or all the routers along its traceroutes to the landmarks, and is aware of IP-level branches at each router leading to other sections of the Internet topology. In the next section, we describe how a node joining anywhere in the overlay can be routed towards close nodes by "tracing" its path along this virtual topology.

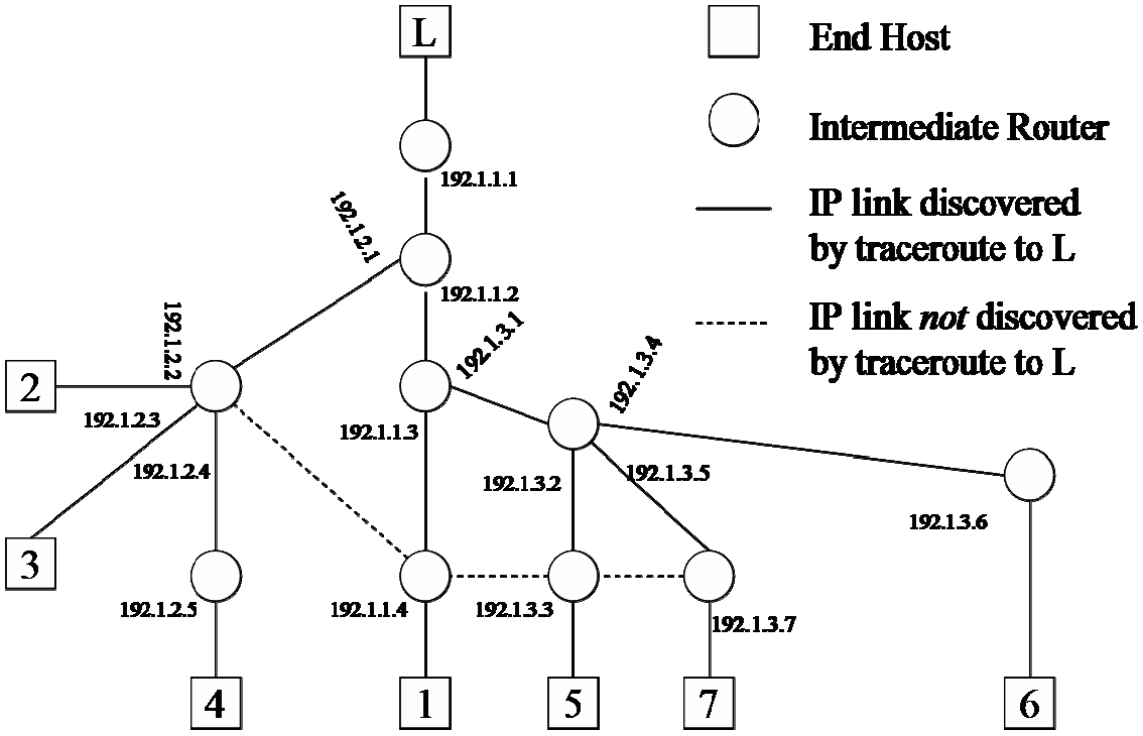
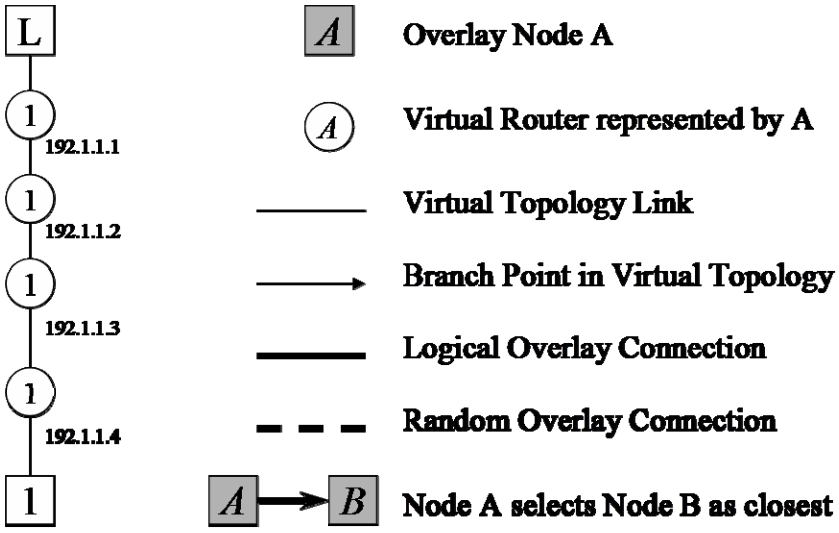


Figure 6.1: Sample IP Topology with Six Overlay Nodes and Landmark L.

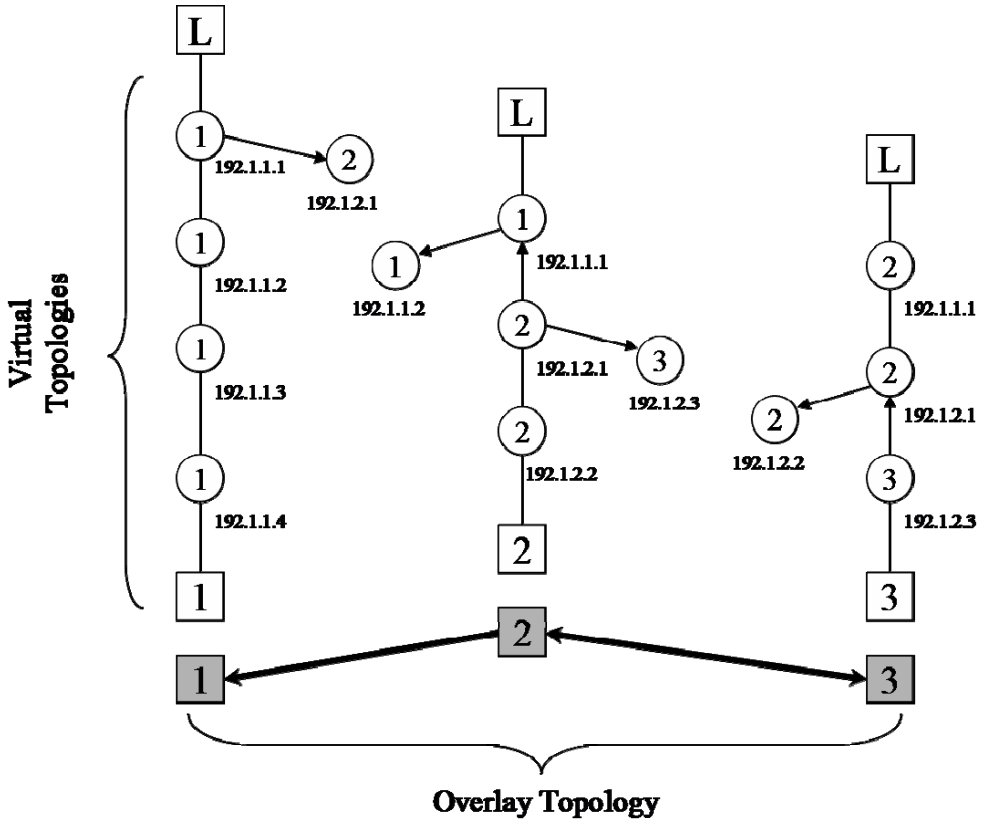
6.1. Overlay Organization

TAO constructs a virtual graph depicting the Internet topology discovered by traceroutes from joining nodes to the landmarks and organizes overlay nodes to reflect this virtual graph. The Internet shortest paths from all nodes to each landmark form a reverse shortest path tree rooted at the landmark. The complete graph is formed by interleaving all landmark trees by identifying vertices (router IP addresses) and edges (adjacent pair of router IP addresses) in each individual tree which are also present in all or some of the other trees, and using these shared vertices to interconnect individual trees. The overlay network is constructed by having TAO nodes uniquely "represent" vertices (routers), and edges (links) in this virtual graph are represented by establishing overlay connections between the overlay nodes that represent the endpoints of each edge. Thus the overlay topology *adapts* to the underlying IP topology as it

is discovered from the traceroutes of joining nodes. Moreover, a node joining anywhere in this overlay can be routed to its appropriate place by *tracing* the new node's traceroutes along the virtual topology.



(a) Virtual Topology Key



(b) Overlay and Virtual Topologies

Figure 6.2: Virtual and overlay topologies for the IP topology.

It is worth noting that a router typically has more than one interface, each with a unique IP address. Traceroute only discovers the IP addresses of the interfaces on the reverse path from the landmark, but limited router alias resolution can be achieved when comparing two paths to the same landmark: the first unshared hop along each path would be interfaces on the same router. However, for the purposes of comparing shared path metrics, this is not significant as it does not change the relative path shared hop count. Hence, to preserve simplicity, alias resolution is not performed, and each interface is considered as a unique router in the path and represented by a unique vertex in the virtual topology.

Each TAO node only represents routers in the virtual topology that are not already represented by the nodes it connects to. Thus each router is represented uniquely by an overlay node, but each node may represent more than one router. Routers are represented by nodes that are closest to them to increase the accuracy with which the overlay topology reflects the underlying Internet. The distance between any node and a given router can be estimated from the per-hop latencies determined from the traceroute to a landmark. However, as router latencies increase monotonically with each hop, the node-to-router latency is directly proportional to the landmark latency. Thus the landmark ping latency (effectively the latency of the last hop in the traceroute) can be used directly as a metric to determine which node can better represent a router.

TAO is constructed incrementally, so the first node to join the system would initially represent all discovered routers. However, as subsequent nodes join, the overlay node currently representing routers near a landmark can *hand over* the responsibility of representing these routers to a new node that happens to be closer to the landmark. Note that

it is only necessary to hand over representation of routers on the shared path, since the unshared routers are obviously not on the other node's shortest paths. This *restructuring operation* can be done either proactively at each node join, or reactively when repairing the overlay due to churn. This allows TAO to adapt to the IP topology more accurately and reduce the average stretch of the whole overlay, besides simply forming clusters of closest nodes. Thus TAO presents a systematic method to form topology-aware interconnections between clusters of closest nodes, something that we believe has not been addressed in-depth before.

Figure 6.1 depicts a sample IP topology with six overlay nodes and a single landmark, and Figure 6.2 shows the corresponding virtual and overlay topologies adapted by TAO. It can be seen that connections in the overlay represent those edges in the virtual topology whose endpoints are *not* represented by the same overlay nodes. These edges occur at the branch points in each landmark virtual tree, and form the basis of organization and routing in TAO.

6.2. Algorithm

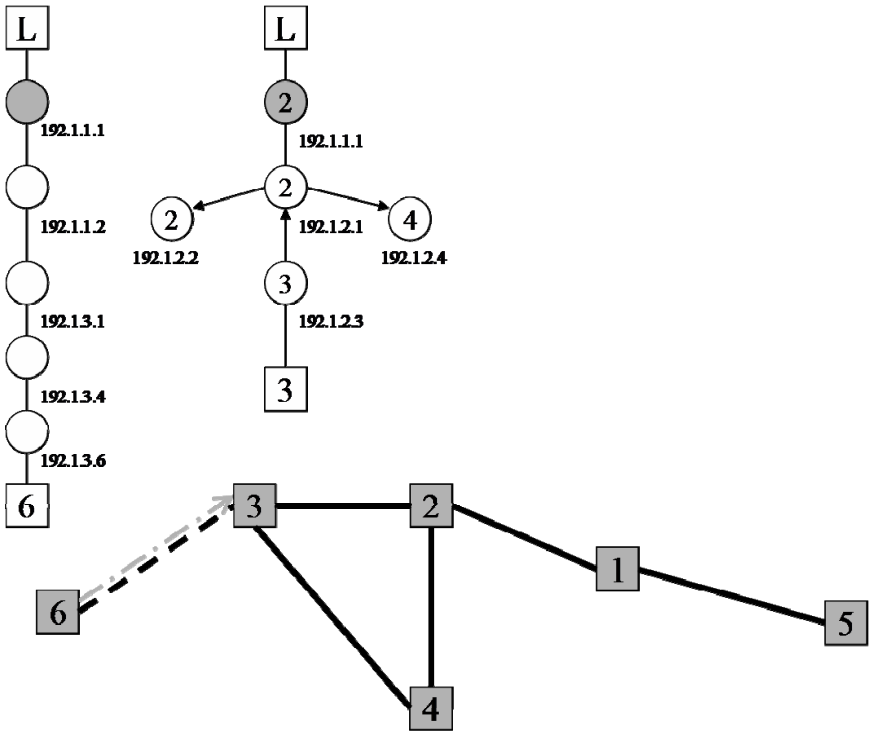
TAO uses *locality queries*, consisting of traceroute information of joining nodes, to discover their topologically close nodes in the overlay. These queries are routed by forwarding them *up* each landmark tree *towards* the root until a node is found that represents a router along the corresponding traceroute in the query. Then this query is forwarded *down* the tree until nodes with maximum sharing are found. The TAO construction and tracing algorithm consists of five operations: Join, Locality Query Routing, Select, Connect and Restructure. Each

operation is illustrated in Figure 6.3 for the case of a single landmark. In case of multiple landmarks, the same operations are performed iteratively over each landmark path.

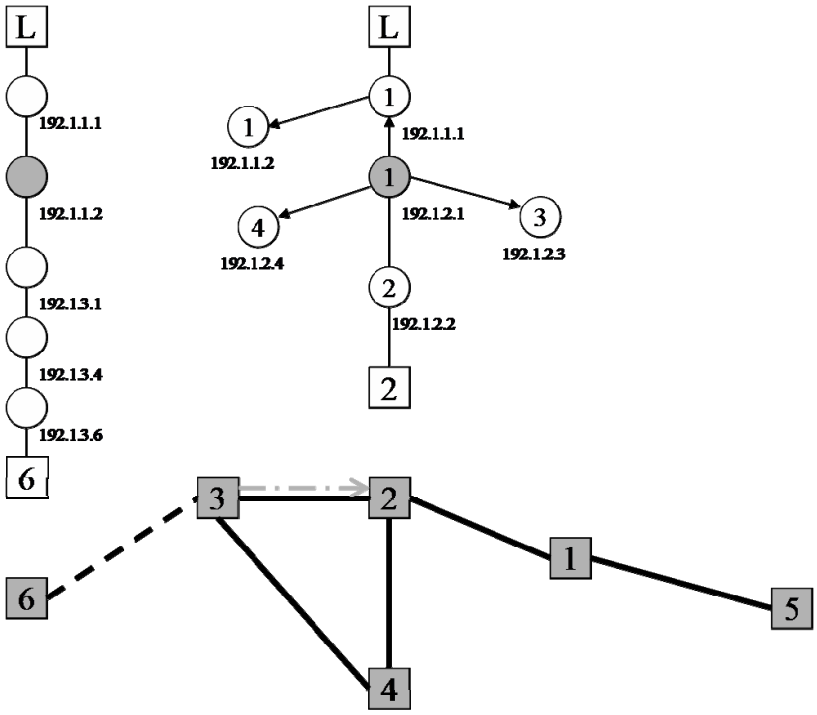
Node Join: When a TAO node joins the system, it collects its traceroutes to a small predefined set of landmarks, which we call *landmark paths*. It then locally constructs a *virtual topology* graph from this information and "represents" each router on these paths itself, by labeling these virtual hops with its own ID. This new node discovers and connects to a random TAO node, to which it sends a locality query encapsulating its set of traceroutes, as shown in Figure 6.3(a). The locality query hence represents the topological location of the joining node. This query is then routed across the overlay to the closest online node by "tracing" over the virtual topology at each hop.

Routing: The locality query is forwarded based on shared path semantics. To prevent cycles, the overlay node at each hop appends its ID and IP address to the query message. At each hop, the TAO node runs the path comparison algorithm on the landmark paths in the query message and its own and those of its overlay neighbors to trace the correct next hop. For each landmark, the algorithm determines the IP address of the first unshared hop along the corresponding path. This IP address identifies the vertex in its virtual topology to forward the query along, and the ID this vertex is labeled with identifies the corresponding neighbor.

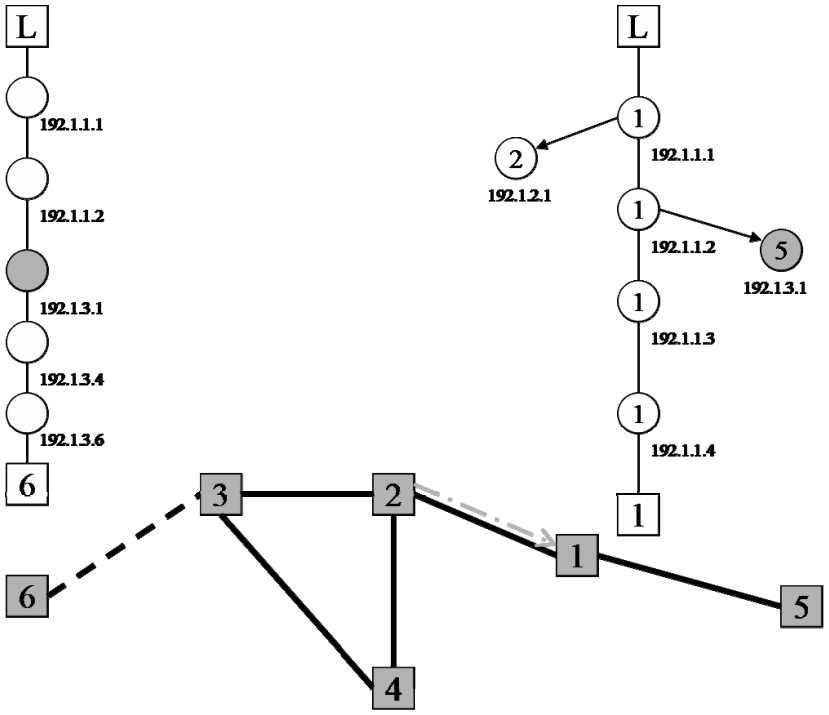
Figure 6.3: TAO Routing in Action.



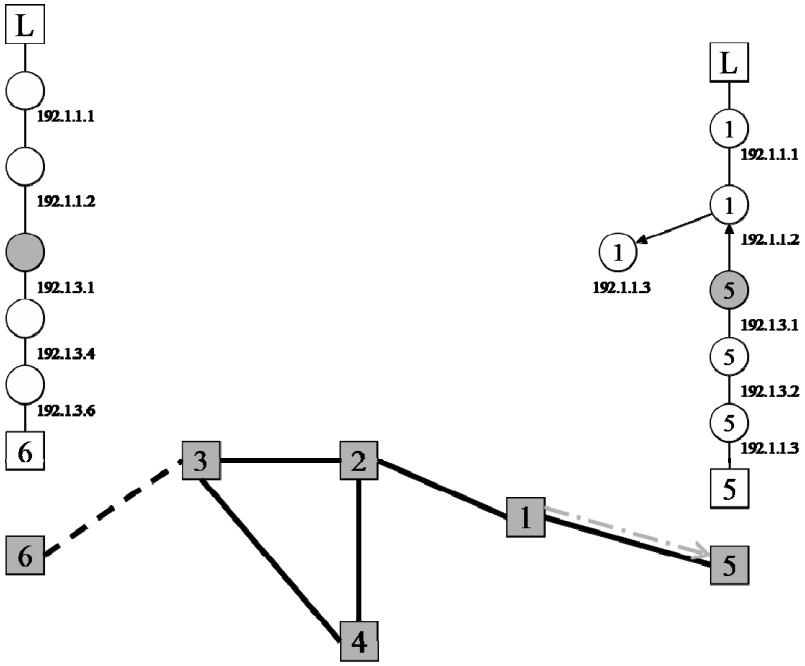
(a) Node 6 joins randomly at node 3 and sends its locality query. Node 3 identifies node 2 as the next logical hop.



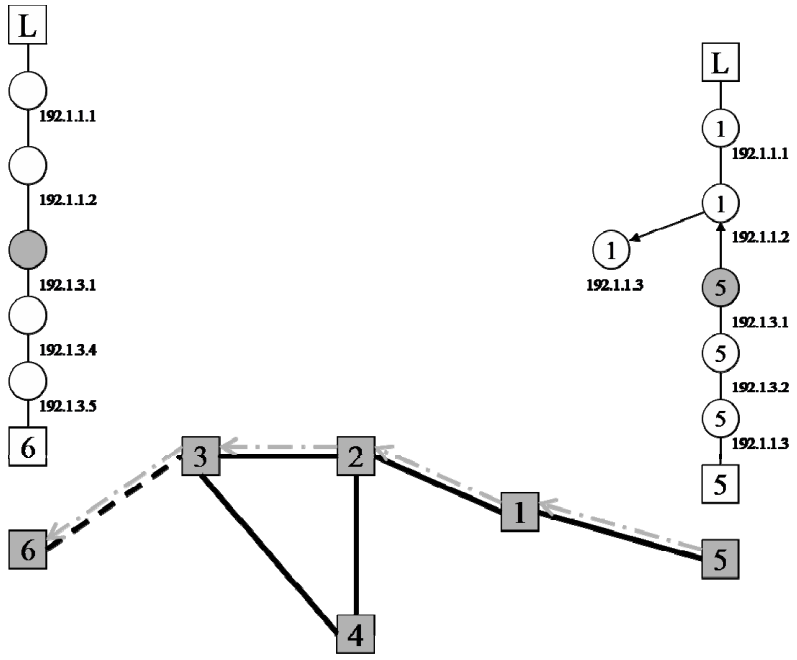
(b) Node 3 routes the query to node 2. Node 2 identifies node 1 as the next logical hop.



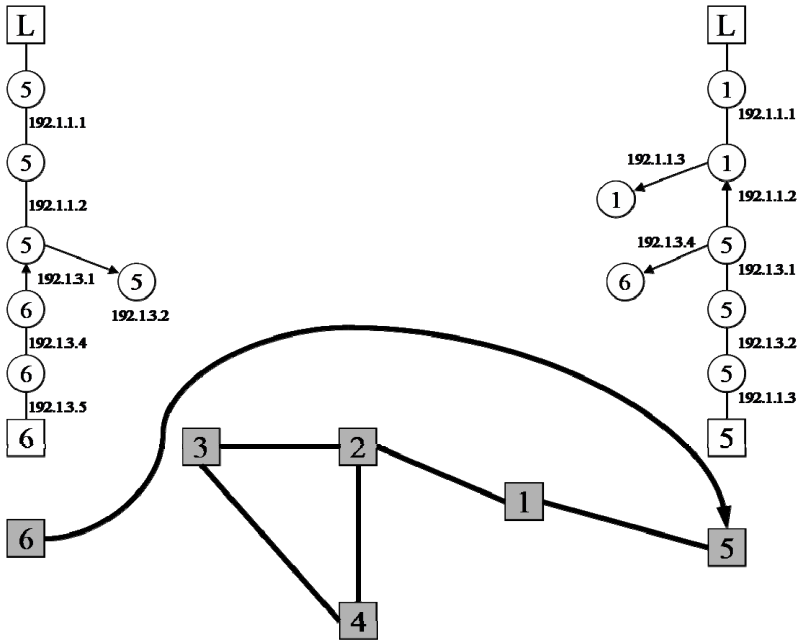
(c) Node 2 forwards query to node 1. Node 1 identifies node 5 as the next logical hop.



(d) Node 1 forwards query to node 5, which identifies itself as the node with maximum path sharing.



(d) Query results are back-propagated over query path.



(e) Node 6 selects node 1 from the results and connects to it, and both nodes update their virtual topology to reflect the new connection.

Unless the message was forwarded from the node identified by this branch point label, it forwards the query to that node. For instance, in Figure 6.3(a), node *3* finds *192.1.1.1* to be the last shared hop with joined node *6* for landmark *L*, which is represented by node *2* in node *3*'s virtual topology. Hence node *3* forwards the query to node *2*. Similarly, in Figure 6.3(b), node *2* finds *192.1.1.1* to be the last hop, which is represented by node *1*, thus identifying the next logical hop. Thus each node only needs to be aware of its immediate parent and children for each landmark tree for routing to be successful.

If the vertex is labeled with its own ID, the node is the *branch point* in the virtual topology for the landmark in question. It then compares the query path information with that of those nodes representing the children of the branch point vertex. If it identifies any node having greater path sharing than itself, it forwards the query to this node. Thus, in Figure 6.3(c), node *1* is the branch point, which identifies node *5* to be the next logical hop, since node *5* shares at least one hop greater with node *6* than itself. On the other hand, if this vertex has no other child vertices, or the message came from the identified node, it assumes itself to be the TAO node with the maximum shared path, and forwarding terminates here, as seen in the case of node *5* in Figure 6.3(d).

The TAO nodes thus located by the query include their IP addresses and other contact information in the query result. Each result node also sets an *isClosestToPeer* flag in its result message if the traceroute information from the locality query suggests that the new node may be a good candidate for closest neighbor to itself. This mechanism is essential to handle the Asymmetric Selection Problem. The query results can be returned either by directly

unicasting them to the source of the query (since anonymity is not a concern), or back-propagation along the query path, as shown in Figure 6.3(e). If the number of results is expected to be large, results can be efficiently aggregated by back-propagation. Besides the nodes with maximum PSHC, the result set also includes nodes with the second maximum PSHC, to give a wider choice and account for the long-hop problem. In the example illustrated in Figure 6.3, this would mean that node 1 also includes itself in the result, since it has the second maximum sharing with node *6* after node *5*.

Select: The joining node then selects certain nodes from those discovered by the query based on shared path metrics. Alternately, the same metrics can be used at each hop to narrow the choice during routing itself, but since peer selection is often an application-specific strategy, we let the joining node to do it. We explore the performance of different shared path metrics later. The joining node also selects nodes that have the *isClosestToPeer* flag set in their result messages. This solves the *asymmetric node selection problem*, allowing discovered online nodes to express their preference in their result messages to joining nodes, which can then make a better informed, and hence more complete, neighbor selection decision. The joining node then connects to the selected overlay nodes.

Connect: Each TAO node represents a portion of the underlying network in its virtual topology that is not already represented by the nodes it connects to. The first TAO node to go online represents all hops in its virtual topology, including the landmarks themselves. Each subsequent node typically discovers a new part of the Internet topology, so there is always some portion that is not yet represented elsewhere by its predecessors. This portion is the set of hops *after the last shared hop*. The new node represents this portion in its virtual topology

itself, and labels the remaining routers (the last shared hop and before) with the IDs of its predecessor. Both nodes cache each other's virtual topology and interconnect their respective virtual topologies at the appropriate branch points. Each child node at each branch point is labeled by the respective neighbor's ID, and identified by the IP address of the *first unshared hop* along the *neighbor's* landmark path. This identifies a *branch point* to the newly discovered topology. This process is illustrated in Figure 6.3(f). When two overlay nodes peer with each other, the RTT between these nodes is measured during the process of exchanging topology information messages, and this is used by both nodes to confirm or, if necessary, update their closest node selection.

Restructure: Here, nodes analyze their virtual topologies and reorganize themselves to form an overlay that is more efficient, robust, load-balanced or optimized for a specific application. This function is not essential for the correctness of TAO routing, but is necessary to reduce the average stretch of the entire overlay itself, and may also be needed to deal with practical issues. As long as the correctness of their virtual topology is maintained, routing would not be affected. The most important aspect of the restructuring operation is having routers represented by overlay nodes closest to them. Initially, the first node that discovers a set of routers on its landmark path represents those routers in the virtual topology. However, later if a node joins that is closer to these routers than the node currently representing them, it takes over the responsibility of representing the relevant routers. This can be viewed as inserting the new node at a given depth (corresponding to its landmark latency) in the overlay tree for each landmark instead of attaching it as a leaf node. Hence the overlay has to be reorganized to keep the virtual topology consistent, whereby nodes that had the earlier node

as their branch point would disconnect from it and connect to the newer node instead. If restructuring is done re-actively, the appropriate node simply notifies the node currently representing the relevant routers of its presence, and the former takes over when the latter leaves the system. Restructuring also leads to better load balancing. We also use the delegate function to allow a node to hand-off some of its virtual topology to another node, typically one that is less loaded, or one more centrally located within that section of the topology. This reduces the average number of connections per node without affecting the correctness of TAO routing, although it can also increase the stretch.

This process can be viewed as a distributed implementation of Algorithm 1 below. It is basically an algorithm to construct a sorted search tree where nodes are sorted in ascending order according to two metrics, PSHC and landmark latency. This algorithm results in a tree for each landmark with path sharing as well as landmark latencies increasing monotonically with the depth of the tree. The distributed implementation can be made more efficient than the algorithm itself by performing the restructure operation reactively when nodes leave and join during normal overlay churn rather doing so than actively. Furthermore, since a restructure is equivalent to inserting a value, or rather a sorted sub-tree, into an already sorted tree, it should have low algorithmic complexity.

```
constructMesh( nodeList, landmarkSet )
   for each node in nodeList do
      for each landmark L in landmarkSet do
         node.path(L) = shortest path from node to L
         node.latency(L) = latency of shortest path from node to L
         rootNode = root(overlayTree(L))
         insert(node, rootNode, L)
         restructure(node, overlayTree(L), L)
      end for
   end for
return
insert( node, searchNode, Lm )
   searchPSHC(Lm)  = comparePaths (searchNode.path(Lm), node.path(Lm))
   for each childNode in searchNode.children(Lm) do
         childPSHC(Lm)  = comparePaths (childNode.path(Lm), node.path(Lm))
         if childPSHC(Lm) > searchPSHC(Lm) then
            insert(node, childNode, Lm)
            return
         end if
   end for
   searchNode.children(Lm).insert(node)
   node.parent(Lm) = searchNode
return
restructure(node, T, Lm)
   depthNode = node.parent(Lm)
   while node.latency(Lm) < depthNode.latency(Lm)
      depthNode = depthNode.parent(Lm)
   end while
   if depthNode != node.parent(Lm) then
         parentNode = node.parent(Lm)
         parentNode.children(Lm).remove(node)
         depthNodeParent = depthNode.parent(Lm)
         depthNodeParent.children(Lm).remove(depthNode)
         node.children(Lm).insert(depthNode)
         for each childNode in depthNode.children(Lm) do
            insertHere(childNode, node, Lm)
         end for
   end if
return
insertHere(testNode, insertedNode, Lm)
   parentTestNode = testNode.parent(Lm)
   insertedPSHC(Lm)  = comparePaths (childNode.path(Lm), insertedNode.path(Lm))
   parentPSHC(Lm)  = comparePaths (childNode.path(Lm), insertedNode.path(Lm))
   if insertedPSHC(Lm) >= parentPSHC(Lm) then
         parentTestNode.children(Lm).remove(testNode)
         node.children(Lm).insert(testNode)
   else
         for each childNode in testNode.children(Lm) do
            insertHere(testNode, node, Lm)
         end for
   end if
return
```

Algorithm 1: The TAO Algorithm

6.3. Correctness

Each TAO node maintains a local virtual topology reflecting the underlying Internet along its landmark paths, which together form a tree rooted at itself. Collectively, these distributed virtual topologies would make a graph consisting of the Internet topology discovered by their landmark paths, which we call the *global virtual topology*. Admittedly, since the number of landmarks is small, it is a narrow view of the Internet, but it is sufficient in accurately providing basic services such as closest node selection and topology-aware inter-cluster organization. It is this global virtual topology that enables systematic routing of locality queries, and as long as this state is consistent, queries will be routed correctly.

In the simplest case, if there is only one landmark, individual virtual topologies will consist of a single path to the landmark, and the global virtual topology graph would resemble a tree rooted at this landmark. Thus, logically a locality query simply has to be routed recursively from a random node in the tree along nodes *towards* the landmark root to ensure that eventually *at least* one node will be found that has a branch leading to nodes with greater shared path away from the root. In the worst case, this branch would be at the root of the virtual topology tree, indicating that the querying node joined at a very distant node. Ideally, there will be only one such node for each landmark tree, but in the case of multiple landmarks, a node that is not the branch point in one landmark tree could be a branch point in another.

If such a branch is found, the query is forwarded recursively down this branch *away* from the landmark until no more such branches are found. If and when no such branches are found, it

indicates that the remaining section of the query's landmark path is not represented in the virtual topology so far, and the new node would be the first to do so. At this point the new node creates a branch and extends the global virtual topology with its own. Routing is thus systematic, and flooding is avoided unless faced with highly incomplete or anomalous traceroutes. For the case of multiple landmarks, this process is simply repeated for each individual landmark. However, the TAO algorithm is implemented to avoid forwarding the same query to the same node more than once as far as possible, which can happen if a node represents the branch points on more than one landmark trees.

Because TAO emulates the router-level topology, the upper bound on number of overlay messages per join is the sum of the number of IP hops on the paths of the joining and closest nodes to the landmark that gives the maximum path sharing, assuming the worst case that each router along these paths is represented by a different node. This bound is constant with increasing overlay size, but can be reduced by simple techniques such as maintaining information of more than just the 1-hop neighbors in the overlay.

6.4. Resultant Overlay Structure

It is apparent that this incremental construction would eventually lead to clusters of topologically proximal nodes interconnected with other such clusters. The overlay would thus have a hierarchical nature similar to that of the Internet topology it adapts to. Each cluster would have a "cluster leader" for each landmark, a single node to which all other cluster members are connected. This cluster leader would typically be the one that joined the earliest or is closest to the landmark, and hence represents the last shared router (for instance

a gateway router) on the path of all nodes in that cluster to the given landmark. It is also possible that a node can be a cluster leader for multiple landmarks. These cluster leaders are important since they connect all other nodes in their cluster to the rest of the overlay, and hence should employ failure detection techniques and appoint replacement nodes to take over in case of failure.

If only a single landmark is used, these clusters would have a hierarchical, tree-like structure. However, each additional landmark would introduce its own cluster hierarchy, and interleaving these trees would identify nodes that belong to multiple clusters. These nodes are significant, since they are close to all the clusters they belong to. Hence when exchanging messages between nodes in these clusters, they should be routed along these common nodes to reduce the overlay stretch.

6.5. Landmarks

Landmarks can be chosen randomly as long as traceroutes to these hosts can be correctly collected. However the main advantage of having more landmarks is that they help discover more inter-cluster connections. Hence intuitively it would be better to select topologically distant landmarks.

6.6. Node Selection

A basic TAO query could return a number of topologically co-located nodes, out of which those that represent routers at the branch points must be selected and peered with in order to maintain the correctness of TAO. However, results may return distant nodes due to the long-

hop problem. Here we present techniques to identify the actual closest nodes with better resolution.

- **Other Shared Path Semantics:** Obviously, it is not possible to distinguish between nodes with identical PSHC metrics. PSHC is also inadequate to identify a case of the long-hop problem. This is where the other shared path semantics prove useful. As mentioned before, a large total unshared latency (TUSL) estimate is a good indication that the node is farther away than the PSHC may suggest.
- **Direct Probing:** This is the simplest approach to derive maximum accuracy. Nodes located by the TAO query are targeted for individual ping measurements to identify closest nodes. However, to solve the asymmetric node selection problem efficiently, TAO takes a reverse approach. Each TAO node that identifies itself as the result of a locality query pings the joining node, and includes the RTT in the result message. Furthermore, this node also compares this RTT with that of its current closest nodes, and if it finds that the joining node is closer to itself than others, it sets the *isClosestToPeer* flag. Thus the joining node not only discovers the latency to each node in the result messages, it also finds out which of these nodes consider itself to be closest to them. Since the query discovers all nodes in the maximum PSHC cluster as well as the next-to-maximum PSHC, this is more of a brute force approach and may not be scalable to larger result-sets.
- **Recursive Tracing:** Here, joining nodes dynamically select one of the discovered TAO nodes as a landmark, and collect traceroutes to it. In our implementation we select the cluster leaders as the dynamic landmarks, but other metrics can be used for dynamic landmarks selection as well. Besides organizing clusters into sub-clusters

with finer resolution, this can also identify topologically closer nodes based on PSHC alone by tracing along these secondary traceroutes. This process can be repeated recursively for almost arbitrary precision. However, recursive tracing does not necessarily solve the Long Hop problem and hence may not increase closest node selection accuracy. Rather, it would help to fragment large clusters into smaller sub-clusters with better resolution.

Our preliminary experiments suggested the following node selection metrics provide the best accuracy:

- **Max-PSHC:** The nodes with the maximum PSHC with respect to any landmark are selected.
- **Ordered-PSHC:** For each node, the PSHCs with respect to all landmarks are sorted in descending order. The nodes with the numerically maximum PSHC ordering are selected as closest. For instance, suppose node A has to decide between node B which has PSHCs {3, 6, 5}, node C with PSHCs {4, 5, 6} and node D with PSHCs {5, 3, 5} with respect to three landmarks L_1, L_2 and L_3. It sorts the PSHCs for each node in descending order, giving B {6, 4, 3}, C {6, 5, 2} and D {5, 5, 3}. Hence C is chosen, since it has the maximum overall PSHC of all other nodes.
- **Reverse-Ping:** This is the direct probing method, whereby nodes discovered by the query ping the joining node and include the RTT in the result message. The joining node then uses the latencies included in the query message to select the closest node.

6.7. Practical Issues

The main problems facing any real-world peer-to-peer overlay are churn and partitions. We discuss these issues and suggest feasible mechanisms to address them, although we have not implemented them in our simulations.

Churn: Branch nodes are vital in maintaining the structure and correctness of TAO, since it can lead to other segments of the topology. A node leaving suddenly could lead to inconsistencies in the global virtual topology state, where certain routers are not represented by any nodes, which would affect proper forwarding of a locality query, or represented by multiple nodes that are unaware of each other, causing partitions.

Partitions: A partition is especially likely if a crucial node (the cluster leader for instance) leaves suddenly, thus isolating all nodes within its cluster. Partitions may also occur during TAO construction due to inconsistencies between paths. Because of router reconfigurations or re-routing within the Internet due to congestion or network failure, there may be dissimilar routes between topologically close nodes to the same landmarks, which may lead to partitions between clusters of nodes that should ideally be interconnected.

Load: Nodes representing landmarks and high-degree routers at the core of the Internet would eventually have a large number of branches (and hence connections) as the overlay grows, and would get overloaded.

We have not studied the effects of churn on TAO, but we believe the use of standard fault-tolerance and load-balancing approaches would suffice. Furthermore, high churn would present an opportunity to restructure the overlay as mentioned in the algorithm.

Failover, Redundancy and Replication: Each TAO node could connect to, or at least be aware of, multiple TAO nodes apart from its logical neighbors (either n-hop neighbors or random peers) and replicate their traceroutes to keep track of their relative topological locations. Replication would incur negligible storage requirements due to the conciseness of traceroute information. Coupled with standard failure detection techniques such as heartbeat messages, these redundant connections would provide alternative paths to repair the TAO overlay in case of intermediate node failure. The network load of failure detection mechanisms can be minimized by restricting them to critical branch nodes (for instance, cluster or gateway router representatives) and a few selected failover nodes. Repairing the overlay is easy, since if a branch node goes down suddenly, the portion of the virtual topology it represents is annexed by its adjacent nodes. The repair algorithm is simply to have one of the departing node's branch neighbors join its predecessor, and failure of leaf nodes need not be handled. TAO repair mechanisms can be incorporated into that of the application.

Load-balancing: A single router could be represented by multiple nodes which coordinate to represent different branches in the virtual topology and cooperate to efficiently forward queries. This would be useful for nodes representing routers with high degree, so that a single node is not overloaded by all the connections from this router.

Super-Peers: This extends naturally to modern unstructured overlays, where a more reliable super-peer can act as a "leader" for a large cluster of normal peers and provide a reliable point of contact for new nodes routed to this cluster.

Traceroute Refresh to Detect IP-Topology Changes: To check whether its view of the Internet topology is still up-to-date, and to repair partitions due to path inconsistencies, a node could perform traceroutes at periodic intervals. This refresh rate is a heuristic that can be tuned to provide a trade-off between validity of topology information and network load incurred. A change in network topology can easily be handled by having nodes re-join the overlay at the node representing the router after which the topology has changed. The node that discovers the change in the route to a landmark sends a *topology update* message to its cluster neighbors. This message is propagated down the overlay landmark tree of the changed path to all the nodes that may be affected by this change (that is, the node representing the last router after which the path has changed.) As Internet routes do not change drastically, the impact of a topology change would be minimal and localized, probably requiring only one node in each affected overlay landmark tree to connect to a different parent node. This technique effectively updates all affected nodes, and greatly reduces the need for a high traceroute refresh rate. Only one node in every cluster can be elected to perform periodic traceroutes, or each node would carry out traceroutes at large, random intervals, to further reduce the network load incurred by traceroutes.

Firewalls: These would prevent a new node from connecting to its closest neighbor directly, but can be solved using techniques from Gnutella like PUSH messages, where nodes connect to the appropriate un-firewalled TAO node instead.

In addition, some simple refinements are possible with TAO. Adaptive Traceroute: When a new node joins for the first time, it requests the bootstrap node's traceroute information before beginning its own traceroute. It then targets intermediate router IP addresses towards the end of the bootstrap node's traceroute to avoid overloading the landmark node itself. This technique may also discover some of the direct path between both nodes if the router probed does not happen to lie on the shared path. Query path and result caching can be used to make redundant connections or forwarding queries along a shorter overlay path. Topology information of distant nodes and joining nodes could be disseminated and cached to locate farther areas of the topology faster. Traceroutes can be piggybacked with normal application-specific messages to disseminate them with negligible overhead.

7. APPLICATIONS

The only service TAO provides is a framework to systematically route a randomly connected node to a topologically appropriate peer. Distributed resource discovery, application-specific routing and other services are left to the distributed application that uses it. TAO has direct relevance to any distributed application that can benefit from localization and is applicable to both structured and unstructured overlays. Aside from accurate localization, the incremental nature of TAO enables a high degree of flexibility in constructing an overlay for a given

application and physical network topology. Subsequent sections explore the application of TAO to several well-known problems in distributed and peer-to-peer systems.

7.1. Intelligent Peer Selection

The criteria for selecting peers are application-specific, manifold, varied and often contradictory. TAO offers the flexibility to intelligently select peers to meet a number of objectives with a minimum of measurements.

Closest Neighbor Location: This is the simplest objective, and a straightforward application of TAO, where a node joins the set of nodes returned by the locality query. The closest node can be identified from these by direct probing if the result set is small, else by recursive tracing. This has a wide variety of applications:

- Closest server selection.
- Clustering of distributed servers.
- Efficient locality-aware grouping and TTL-estimation for scoped IP multicast schemes.

Multi-objective Peer Selection: An overlay may have resilience as the primary objective, and peers would be selected based on a combination of maximum and minimum path sharing to provide greater path diversity. Alternatively, applications that enable "swarming" downloads, such as Gnutella and BitTorrent [3], could have node proximity and shared bandwidth as contradictory objectives. Here, a node downloads the same content from multiple peers, allocating a different portion to each uploading peer. Hence uploading peers must be selected such that they are closest to (have maximum shared path with) the

requesting node, yet have minimum path sharing amongst themselves to maximize total available bandwidth and minimize congestion. TAO can also be used in conjunction with other metrics, such as bandwidth.

7.2. Peer-to-Peer Networks

Here, we explore how TAO can be applied to provide topology-aware construction to existing overlay applications. Although peering with the closest node is a very simplistic approach, it makes some sense in the context of unstructured peer-to-peer networks. We also show how this can be applied to structured peer-to-peer systems.

Unstructured Overlays: TAO applies naturally to unstructured overlays, where nodes simply join the closest nodes. Since these are typically ad-hoc in nature anyway, they can simply adopt the TAO structure to reduce topology mismatch. It is directly applicable to several distributed applications such as:

- **File-sharing peer-to-peer networks**, like Gnutella.
- **Super-peer based overlays**, like Kazaa. TAO would be used to discover the closest super-peer. In this case, the locality query is simply restricted to super-peers. The closest super-peer that also has sufficient capacity to accept another connection is chosen. Organizing super-peers amongst themselves is a harder problem, but a good approximation would be to connect each super-peer with others located by TAO, yet maintain extra connections for redundancy and to avoid partitions.

- **Content Delivery Networks (CDN):** Identifying the optimal distribution paths is an important problem, and the peer-selection approach described for swarming downloads could be useful first approximation.

The effect of localization in the overlay network on the search efficiency of unstructured networks is largely unknown, but intuitively, nodes that are topologically co-located are typically also geographically co-located, and hence users of these nodes would probably have similar interests. Thus having topologically co-located P2P nodes may discover content of interest with better search efficiency as well as low network overhead. For instance, a study of the Kazaa file-sharing at the University of Washington in [9] suggests that geographical locality has significant relevance to the interests of users, and they claim that up to 85% of queries leaving the campus network could have been answered by nodes within that network itself.

The main limitation of TAO in the context of simple file-sharing networks like Gnutella is the unsuitability of its structured nature for the high churn rates typically observed in these overlays. TAO can still be used, but performance would degrade when faced with high churn. However, as mentioned before, this churn can be used to some advantage by taking this opportunity to perform the restructure operation in TAO.

Structured Overlays: In the case of DHTs, locality can be used for ID-based or prefix-based routing schemes such as Chord or Pastry, as shown in [36], where a joining node assumes an ID similar to that of the closest node it discovers through recursive probing. Since TAO queries discover the closest nodes with high accuracy, it is directly applicable to this

approach. This is especially useful in Chord, for instance, where most of the hops in each lookup are between nodes with successively closer IDs. Further, due to its incremental and systematic construction TAO would enable nodes to choose their IDs based on their estimated proximity in the topology and hence clusters of close nodes would be more accurately placed in ID space relative to distant clusters as well. Such overlays also use proximity routing, where latencies are minimized at each hop by forwarding the message to the closest of the logically correct next hop candidates. These techniques assume the availability of a function that provides accurate locality information, but usually rely on latency measurements. TAO has the potential to enable accurate topology aware overlay substrates as well as proximity routing.

- **TAO/Chord:** Here, the similarity of IDs between two nodes is directly proportional to their proximity in TAO. The ID with a prefix matching the ID of the closest node discovered by TAO is assigned to joining nodes.
- **TAO/CAN:** We construct a 2-dimensional CAN peer-to-peer overlay using TAO for topology awareness. It follows a very simple algorithm. Instead of randomly assuming coordinates in CAN hash-space a joining node first discovers the closest online node using TAO. The online node then determines the best way to split its hash-space by comparing the new node's traceroute information with that of its neighbors'. Basically it splits the hash-space such that the new node would have maximum shared path metrics with (and hence lower latency to) all its resultant hash-space neighbors. Since TAO nodes cache the traceroutes of their neighbors and locality queries contain the new nodes' traceroutes anyway, the network overhead and complexity for CAN node join reduces to that of the TAO locality query.

Overlay Multicast: Traceroute has already been leveraged in construction of application-level multicast overlays in [16], but this work dealt with only a single source and constructed a tree rooted at this source. Using TAO more sophisticated techniques can be devised to construct multi-source meshes, but this is an area of future work.

Resilient Overlays: One objective of overlays such as [1] is to quickly provide alternative paths between any two nodes in case of intermediate network outage. Nodes that are disconnected due to network failures connect to such an overlay, deployed as an infrastructure-based service, which tries to reroute packets along unaffected paths. TAO can be used to provide an adaptive topology aware RON. Depending on the extent of traceroute information available some significant optimizations are possible. Firstly, traceroute can be used to identify the affected section of the network. Secondly, if all nodes involved have traceroutes to the landmarks (ideally taken prior to the network failure), TAO can be used to discover the closest RON node to each affected Internet node. One of the RON nodes can then iteratively request farther nodes in the TAO/RON network until one is able to complete the circuit to the other. Alternatively, a single ideal RON node can be selected as the one with maximum shared path with both affected nodes which is also able to connect to both of them. RON typically employs active, full-mesh probing to keep track of closest nodes in the overlay, but using TAO, this can be done reactively in response to a failure. This is also an area of future work that has much promise.

8. EXPERIMENTS

To evaluate the effectiveness of TAO in the Internet, we carried out our experiments using the NLANR [20] dataset of 117 nodes, dated 31st January 2003. We condensed the multiple measurements in this dataset into a 117x117 matrix of minimum observed latencies, and traceroute information between all pairs. Of these, one node (amp-wvu) is a pathological case where all traceroutes *from* it are highly corrupt. Hence we did not use it in our experiments, since in the absence of correct landmark path information, the efficiency of TAO reduces to that of a random join. Further, approximately 16% of paths between all pairs of nodes are corrupt to some extent, and TAO has been implemented to function accurately in spite of these traceroutes. We assume that each node in the network is uniquely identifiable and can reach any other node.

We constructed a TAO overlay for 1 – 5 landmarks using the Max-PSHC, Ordered-PSHC and Reverse-Ping metrics for node selection. Landmarks are selected randomly from the set of nodes, with preference given to those to which all other nodes have the least corrupt traceroutes, and are excluded from the overlay construction process. We restructure the TAO overlay actively at each join. We annotate experiments with the number of landmarks used, for instance, TAO-5L indicates an experiment with 5 landmarks.

We compare the TAO overlay with the following techniques:

- **Random:** This is constructed using Gnutella-like random connection, where a joining node randomly discovers some online node and connects to it. To provide a fair comparison with TAO, the random overlay is constructed such that it has a similar node degree distribution as the corresponding TAO overlay.

- **GNP/CN:** Closest-Node Selection. We select the closest node for every node based on GNP coordinates alone. We ran GNP with 8, 10 and 12 landmarks with 7 dimensions, which according to [19] gives amongst the highest accuracies.
- **GNP/DT:** Overlay construction. To evaulate the TAO overlay for stretch, we constructed an overlay network by using GNP with three to five landmarks with dimensions in the range [2:L), where L is the number of landmarks, in conjunction with Delaunay Triangulation to interconnect the nodes. Delaunay Triangulation connects nodes with their closest neighbors in all directions based on coordinates assigned by GNP. This is the simplest way of connecting closest nodes in a multi-dimensional coordinate space while ensuring that there are no partitions.

We annotate GNP experiments with the number of landmarks and dimensions per the original paper [19] and its application. Hence a Closest-Node Selection experiment with 5 landmarks and 3 dimensions is referred to as GNP-5L-3D/CN, whereas the same experiment for the Delaunay Triangulation overlay is referred to as GNP-5L-3D/DT.

For each overlay construction technique, we simulated a realistic scenario whereby nodes join in arbitrary order and discover their closest neighbor. We assume the presence of a bootstrapping technique to provide joining nodes with one or more random nodes already in the overlay to connect to. TAO nodes then discover their closest nodes and their appropriate location in the overlay using locality routing. For both GNP/DT and Random overlays, the corresponding discovery mechanism is assumed to exist, and nodes simply connect with their appropriate neighbors.

We evaluated each topology-aware overlay technique mentioned above for the following real-world problems:

- **Closest Node Selection:** Each node connects to a set of nodes dictated by the overlay construction technique, and selects the neighbor with the lowest RTT as its closest node. For each node we compared the set of closest nodes discovered by each technique to the set of actual closest nodes based on latency, and calculated the relative delay penalty as the difference (in milliseconds) between the RTT to the closest actual node and the RTT to the closest discovered node. For GNP/CN, there is no overlay construction technique, and hence the closest node is simply selected as the one with the minimum geometric distance.
- **Unstructured Overlays:** We measured the average overlay stretch between all pairs for the overlays constructed. The stretch is calculated as the ratio of the total latency of the shortest overlay path to the latency of the direct IP shortest path. We determine the shortest overlay path by running Dikjstra's shortest path algorithm on the overlay graph with each edge weighted by the RTT between the nodes represented by the endpoints of that edge. TAO itself does not provide any logical routing services besides routing the locality query.

We measure performance for each overlay in terms of the following metrics:

- **Closest Node Accuracy:** The percentage of nodes that discover at least one of their actual closest neighbors.
- **Fraction of All Closest Nodes Discovered:** This is the ratio of the number of closest nodes discovered to the total number of closest nodes actually online.

- **Relative Delay Penalty (RDP):** This is the difference (in milliseconds) between the latency to the selected closest node and that to the actual closest node.
- **Latency Stretch:** The ratio of the overlay routing latency between two nodes to the direct network (IP) latency between them. This is a measure of how well the overlay organization reflects the underlying topology.
- **Average Latency Stretch:** The average of the latency stretch between all pairs of nodes in the system. This is an indication of how well the overlay is organized with respect to the underlying topology.

Cost is another important metric to be considered. Since the cost of collecting traceroutes is a constant O(N), and typically a one-shot expense, we do not consider it as part of the cost. Instead, we measure cost in terms of network load incurred.

- **Messages:** The number of inter-node messages exchanged during routing per join.
- **Pings:** The number of pings per join.
- **Node Degree:** The number of connections required to be maintained by a node to keep the overlay structure consistent and avoid partitions.
- **Connects Per Node:** The number of connections initiated by a node during the overlay construction. This includes the connections required to keep the overlay consistent, the closest node connections as well as connections incurred during the overlay restructuring operation. Each connection also provides RTT information that can be used to make better choices.

For each cost metric, we determine the average as well as the cumulative distribution function (CDF) of all nodes. We averaged the results of each experiment over 10 runs.

9. RESULTS

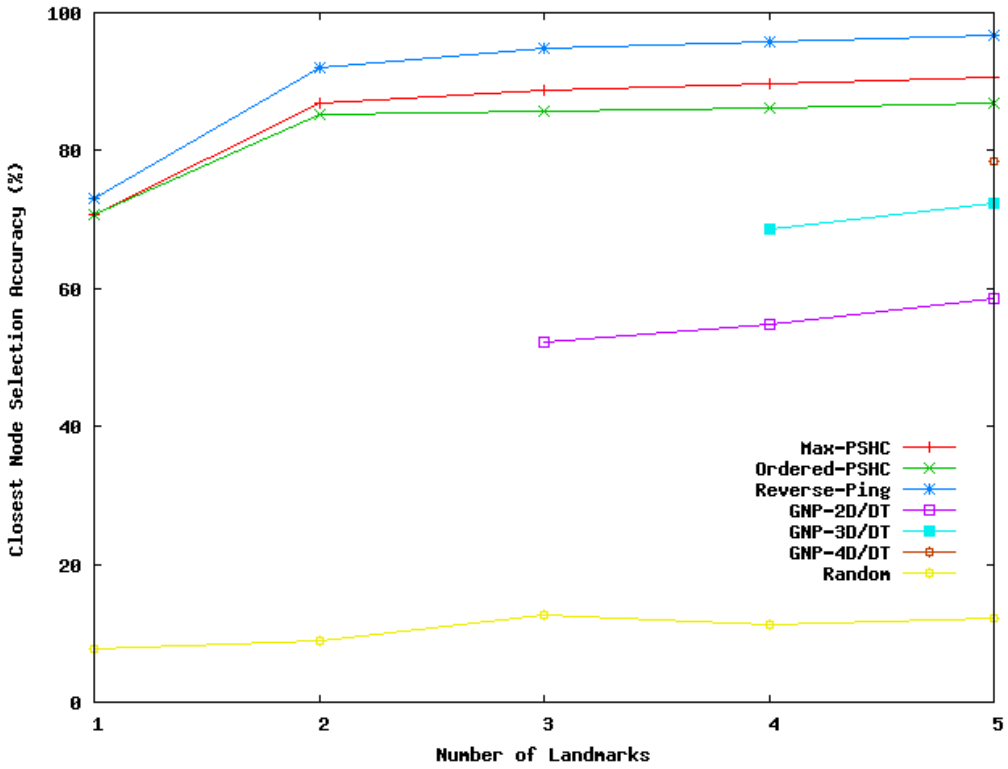
As the results show, TAO performs very well for both closest node selection as well as overlay construction applications.

9.1. Closest Node Selection

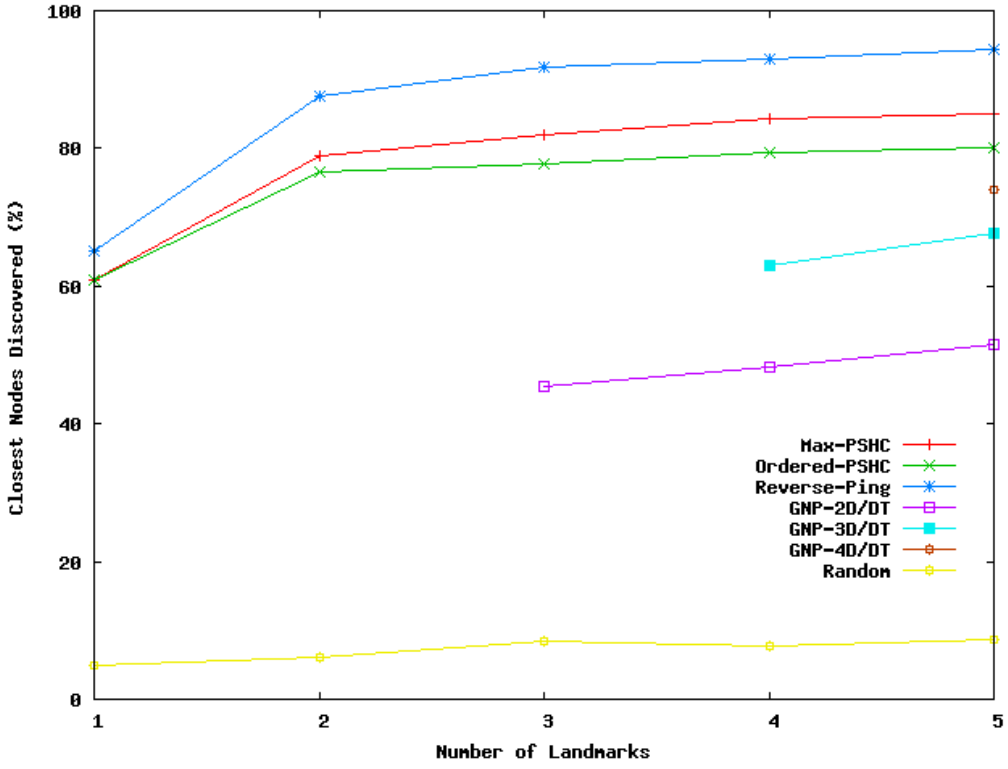
TAO outperforms GNP for all experiments, giving 80% to 95% accuracy depending on the metric used for node selection and the number of landmarks used for constructing the overlay. Obviously, the Reverse-Ping metric gives the highest accuracy, up to 95% with five landmarks. Max-PSHC is the next best metric, achieving 85% accuracy for two landmarks and approximately 90% accuracy for five landmarks. Ordered-PSHC achieves approximately 85% accuracy for two landmarks but does not improve significantly as the number of landmarks increases.

Figures 9.1(c) and (d) show that TAO also outperforms GNP in terms of RDP. GNP/CN, where nodes are selected purely on the basis of geometric distance, performs poorly even for 12 landmarks and 7 dimensions, with a maximum accuracy of 44%. However, the GNP-5L-4D/DT overlay has accuracy almost as high as TAO-5L, but at the cost of a very high average node degree, as shown in Figure 9.3(a). Since each node is connected to a large number of peers, it has a wider choice to look for its closest neighbor in.

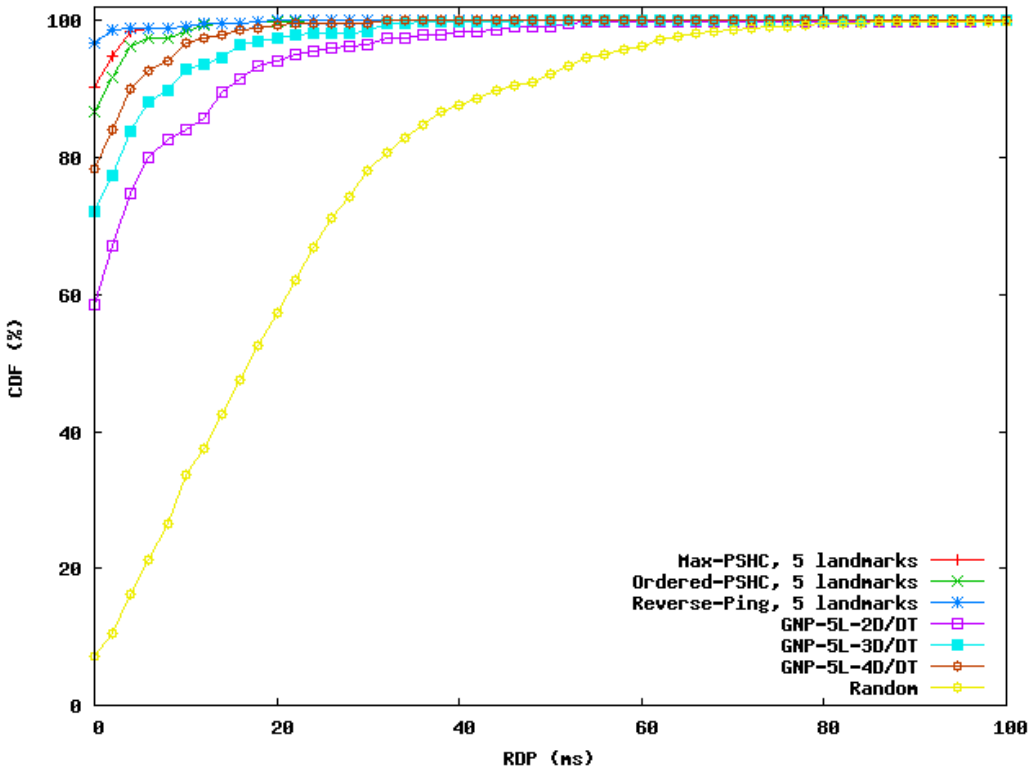
Figure 9.1: Results for Closest Node Selection for TAO, GNP/DT, GNP/CN and Random.



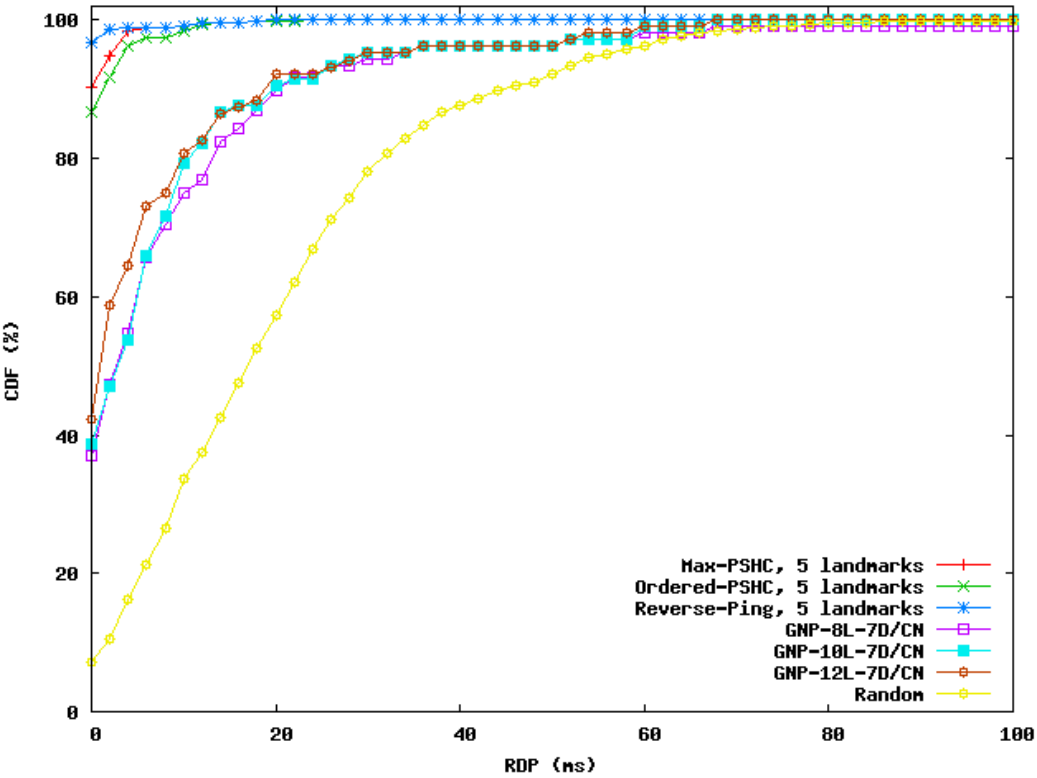
(a) Percentage of nodes discovering at least one closest neighbor.



(b) Average of percentage of all closest nodes discovered for every overlay node.



(c) CDF of RDP for TAO, GNP+DT/Overlay and Random.



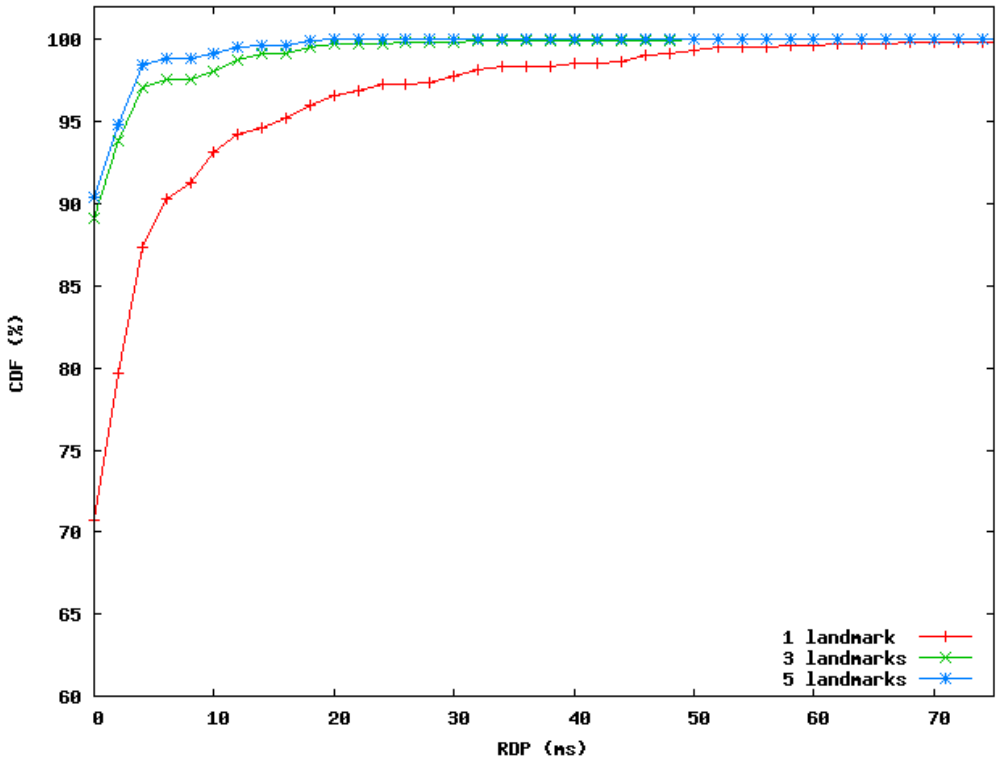
(d) CDF of RDP for TAO, GNP/Closest-Node and Random.

Using a single landmark gives the least accuracy as well as the highest RDP for all metrics. This indicates that one landmark is not sufficient to deal with the Long Hop. Even one additional landmark increases the accuracy greatly (from about 70% to 85% and more for all selection metrics.) Max-PSHC and Ordered-PSHC are obviously equivalent to each other in the case of one landmark, and hence perform identically. Using more landmarks does increase the average accuracy of node selection, but the improvement decreases with each additional landmark, as seen in Figures 9.1(a), 9.1(b) and 9.1(c).

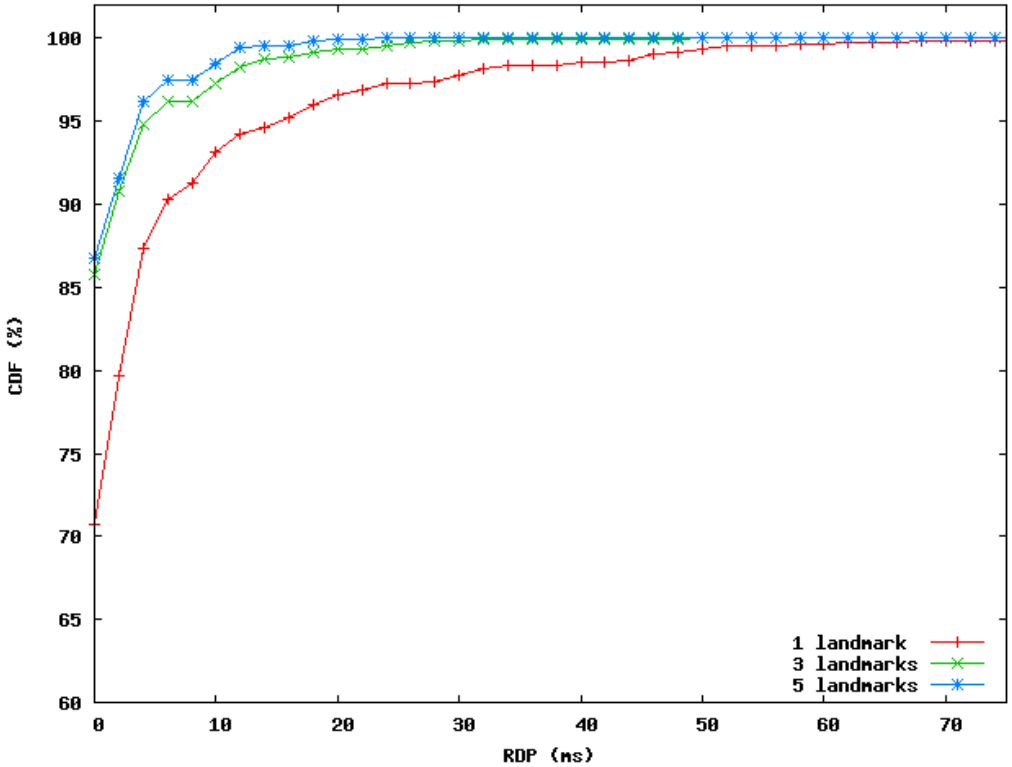
The number of results per query is the same for different metrics, as seen in Figure 9.3. The node selection metric does not change the structure of the overlay significantly, as explained in the next section. Hence the accuracy of the Reverse-Ping technique signifies that a TAO query does locate up to 95% of the closest nodes, it is only the selection metric used that limits the accuracy of the selection. Hence a better use of the information from the path comparison algorithm would give better results.

The reason Max-PSHC performs better than Ordered-PSHC is simply because the former returns a larger result-set, and hence each node connects to more nodes and hence has a greater probability of discovering the closest node amongst them. Order-PSHC, on the other hand, returns a much more restricted result-set, as seen from the number of connects in Figure 9.6(c), and hence reduces the probability that the closest node is included. Hence using one metric over another is simply a trade-off between limiting the number of nodes to connect to and the accuracy of the selection. The Max-PSHC is a more optimal choice as it increases accuracy without significantly increasing the number of connects.

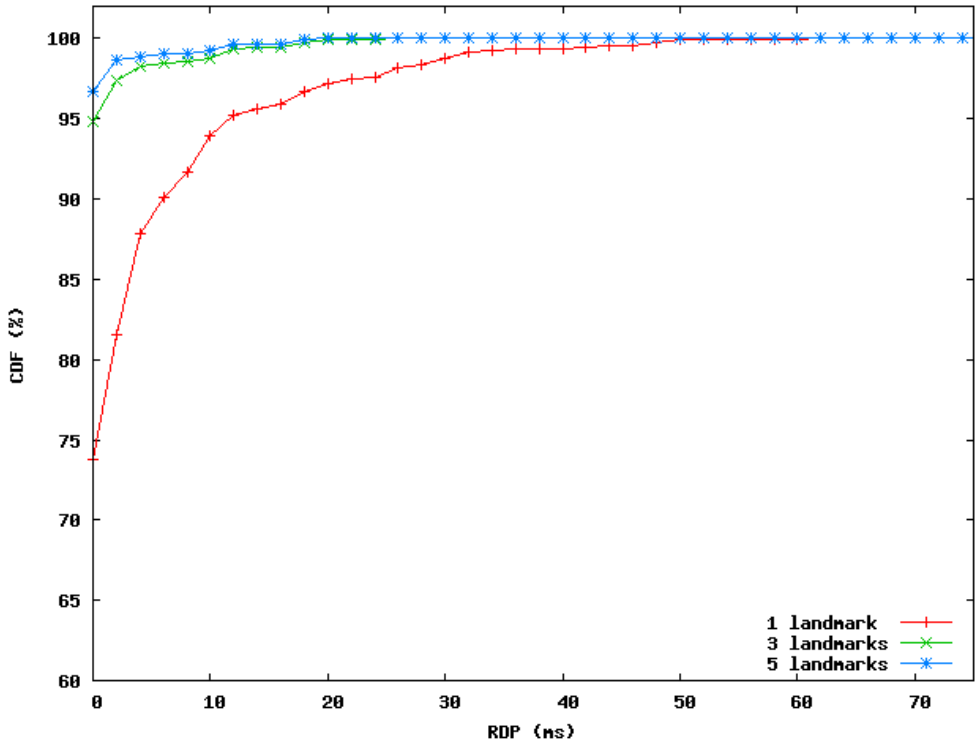
Figure 9.2: Results for Closest Node Selection for Different TAO Metrics.



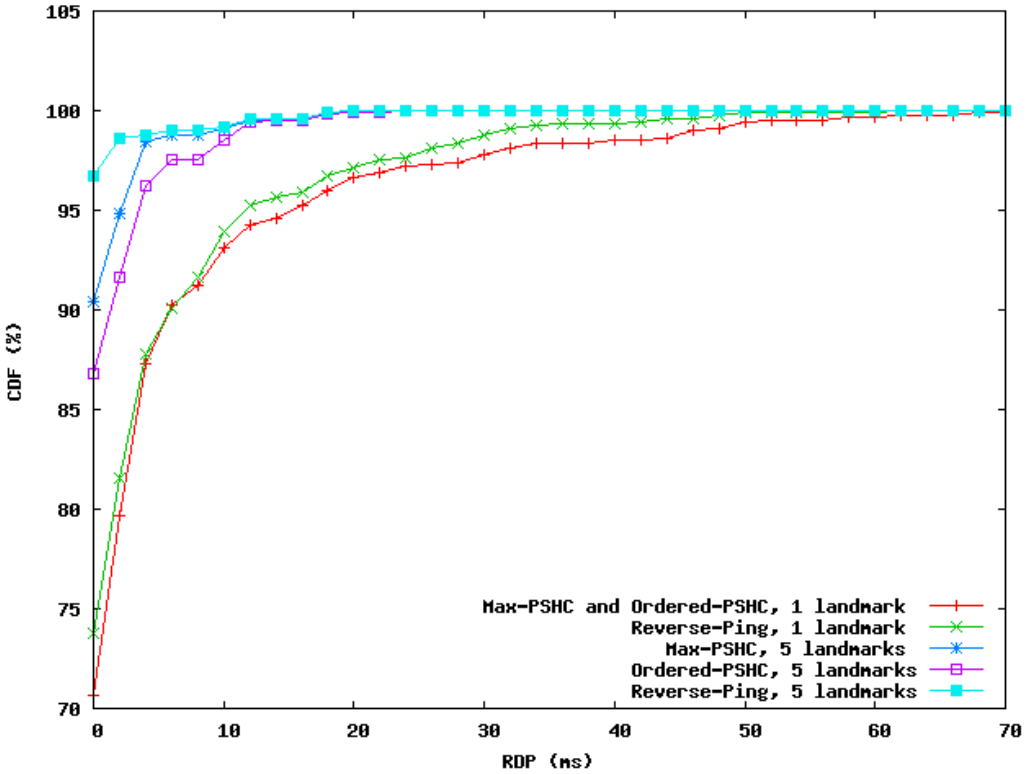
(a) CDF of RDP for Max-PSHC Metric.



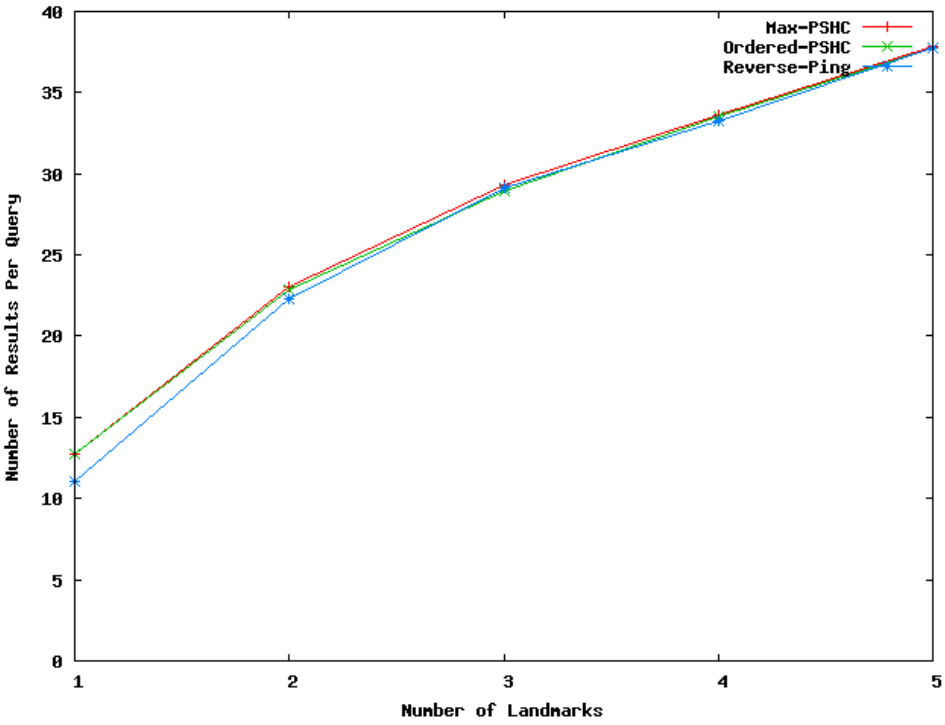
(b) CDF of RDP for Ordered-PSHC metric.



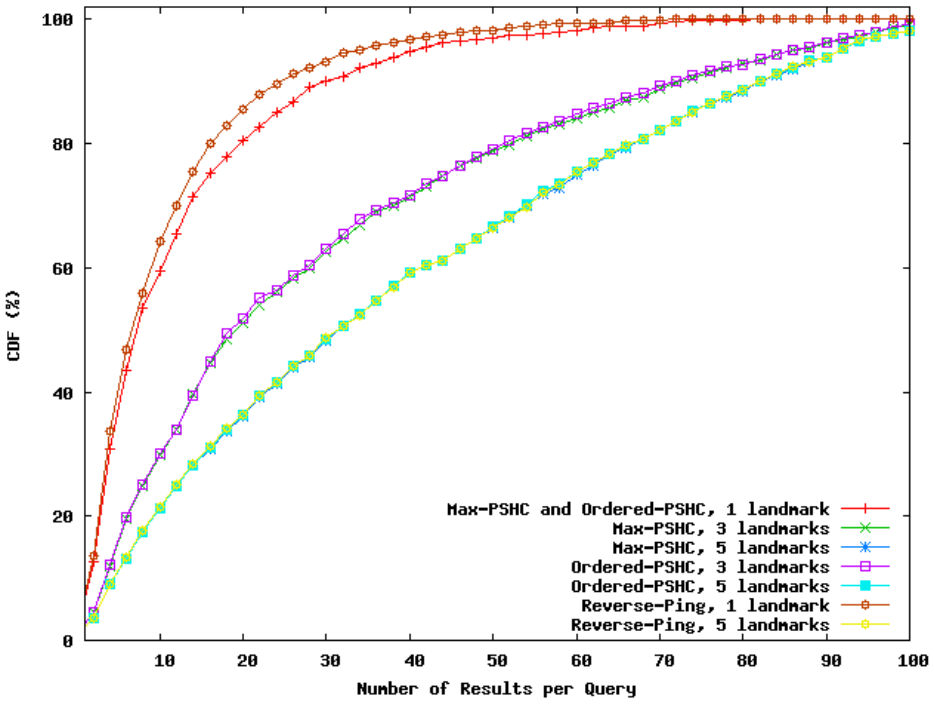
(c) CDF of RDP for Reverse-Ping.



(d) CDF of RDP for All TAO Metrics.



(a) Average



(b) CDF

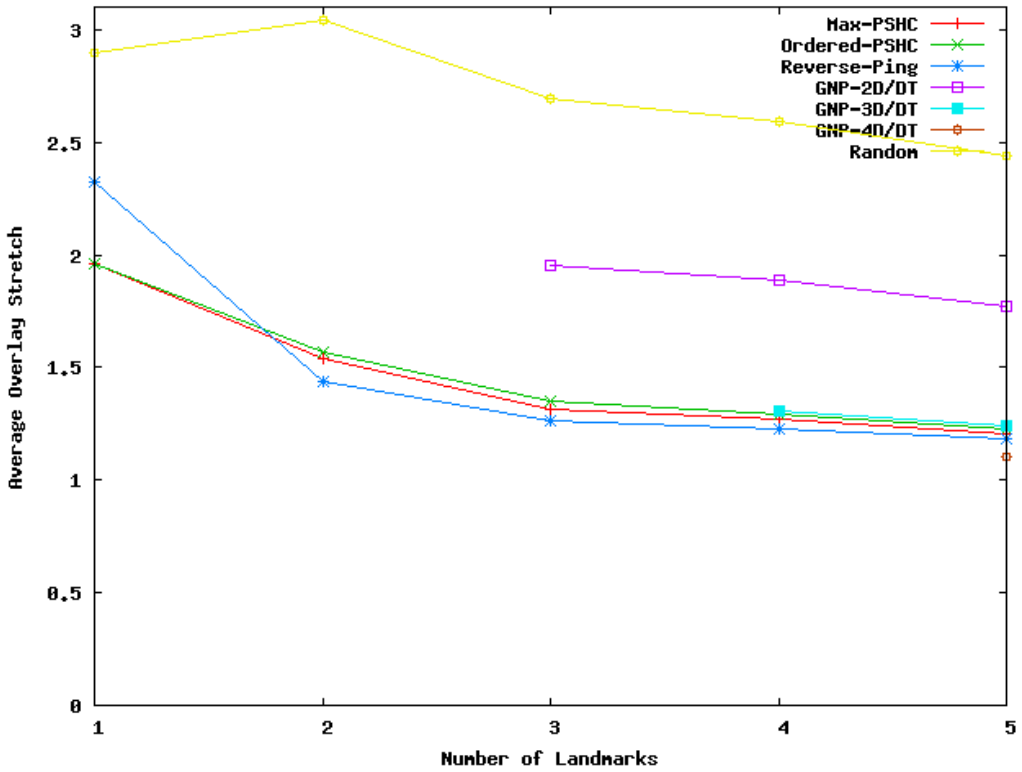
Figure 9.3: Number of Results per Query.

TAO also discovers the highest fraction of all closest nodes online, as shown in Figure 9.6(b), which is probably why the average RDP is low. No previous work has measured, or even considered, this statistic, and we believe it is a significant metric to evaluate the accuracy of any technique. Figures 9.2(a) – (c) show the CDF of RDP for each TAO metric, and 9.2(d) compares the CDF of all TAO metrics, for different number of landmarks. Figures 8(a), (b) and (c) shows that the accuracy and RDP for TAO-3L and TAO-5L are very similar. It shows that even though Max-PSHC has lower accuracy than Reverse-Ping, they have a similar CDF of RDP after about 5 ms. Ordered-PSHC matches the other two metrics after about 12 ms. The low value of the maximum RDP indicates that TAO solves the Long Hop problem for very high latency links, such as intercontinental links.

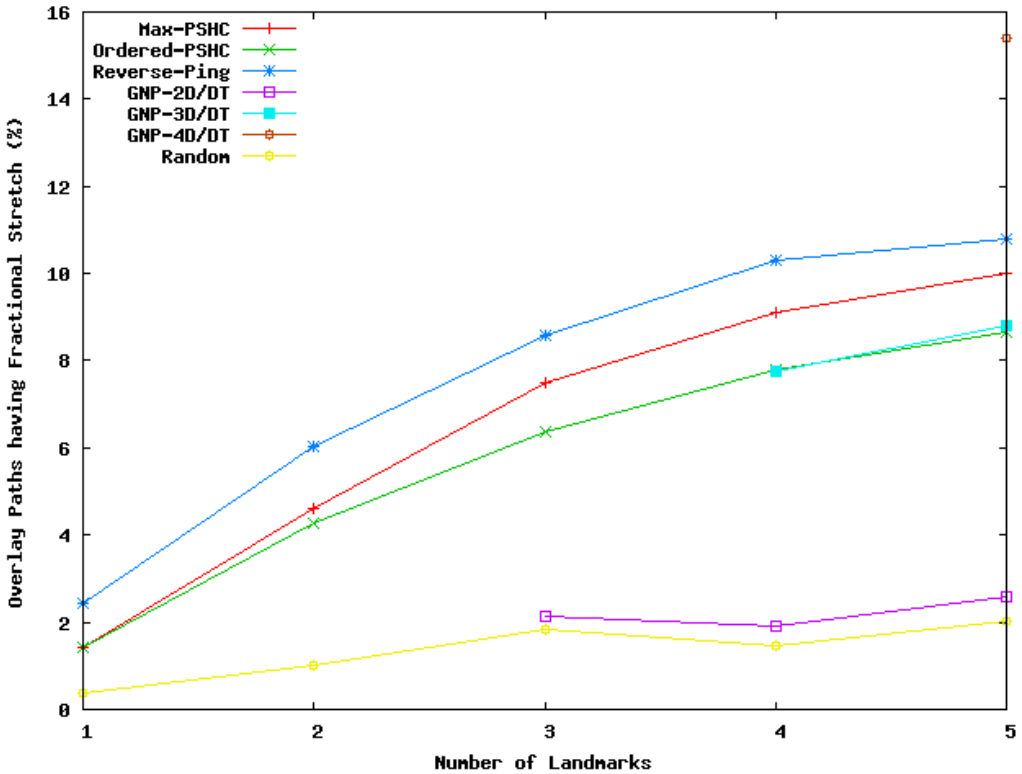
9.2. Overlay Construction

TAO results in a very low stretch of about 1.2 for five landmarks, while maintaining a low average node degree of about 5.6 peers per node. More landmarks result in lower stretch, but the increase in stretch with every landmark reduces after three landmarks, as seen in Figure 9.4(a). The graph also shows that the stretch for the different TAO metrics is very similar for three landmarks or more.

TAO also discovers some overlay paths that have a total latency less than the direct IP path between the corresponding nodes. Since these paths effectively have a stretch less than 1.0, we call them *Fractional Stretch Overlay Paths*.



(a) Stretch



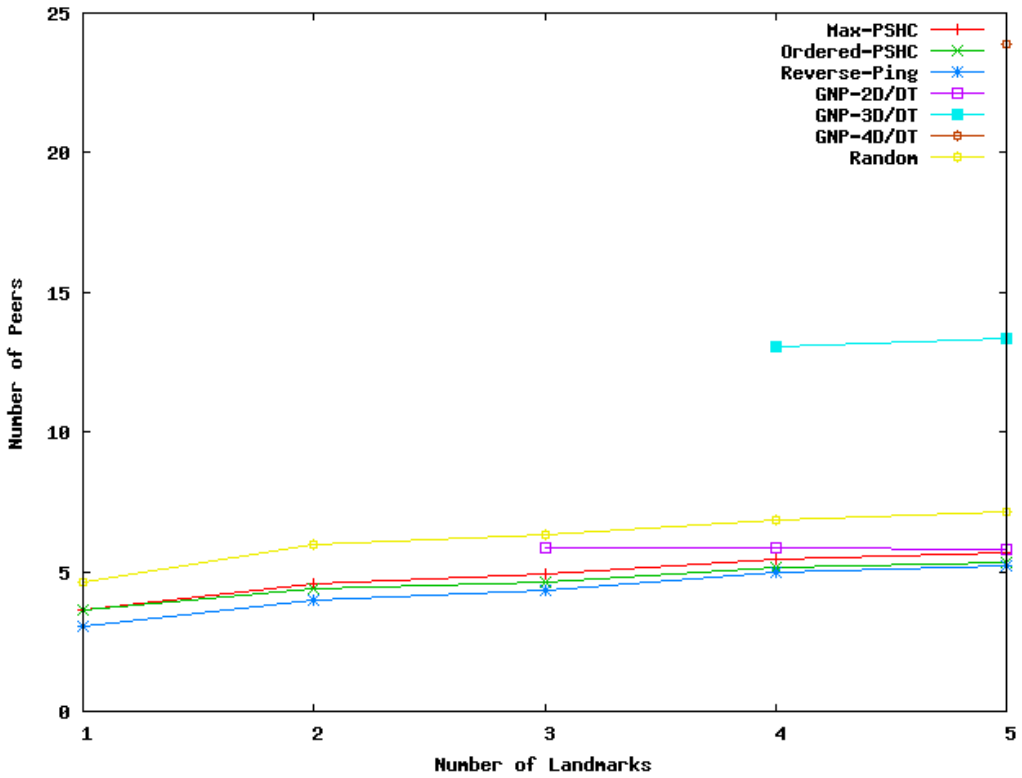
(b) Number of Fractional Stretch Overlay Paths.

Figure 9.4: Overlay Metrics

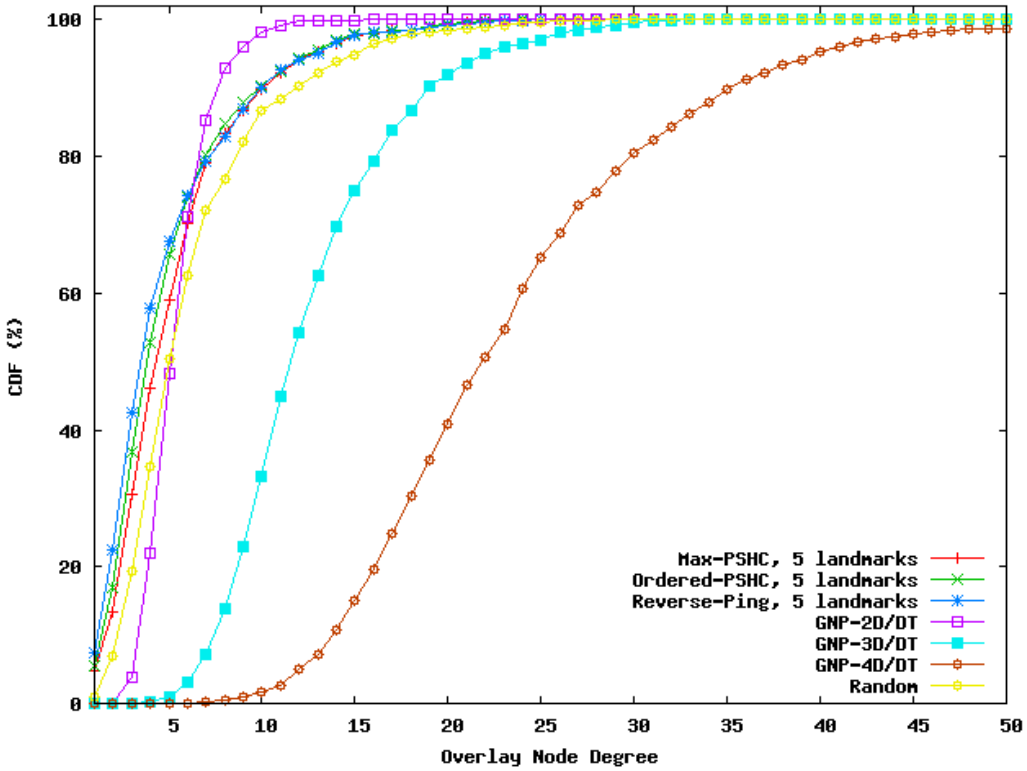
Figure 9.4(b) plots the percentage of all overlay paths that are fractional stretch paths, and shows that extra landmarks lead to the discovery of more fractional stretch paths, with five landmarks resulting up to 10% of all overlay paths having fractional stretch. Our studies of the NLANR dataset reveal that a staggering 35% – 40% of all node pairs have a overlay path that is shorter than the corresponding IP path, and of TAO discovers approximately 18% of all fractional stretch paths possible in the overlay. These fractional stretch paths result in the very low overlay stretch observed in TAO.

This clearly validates the conventional approach of discovering the closest nodes and peering with them, since these closer connections do result in lower stretch. As with other results, Max-PSHC performs better than Ordered-PSHC but worse than Reverse-Ping. Apparently the percentage of fractional paths discovered, and hence the overlay stretch, are directly proportional to the closest node selection accuracy, which makes sense, since for each closest node that is not discovered, the overlay path to it will only be that much longer .

For GNP the stretch reduces and the number of overlay paths shorter than IP paths increases as the number of landmarks and number of dimensions increases. GNP-5L-4D/DT actually results in lower stretch and discovers more fractional stretch overlay paths than TAO, but, again, this is because it has a very high average node degree (about 23.8 peers per node, as shown in Figures 9.5(a) and 9.5(b)), whereas TAO has a very low average node degree.



(a) Average



(b) CDF

Figure 9.5: Node degree for different overlay techniques

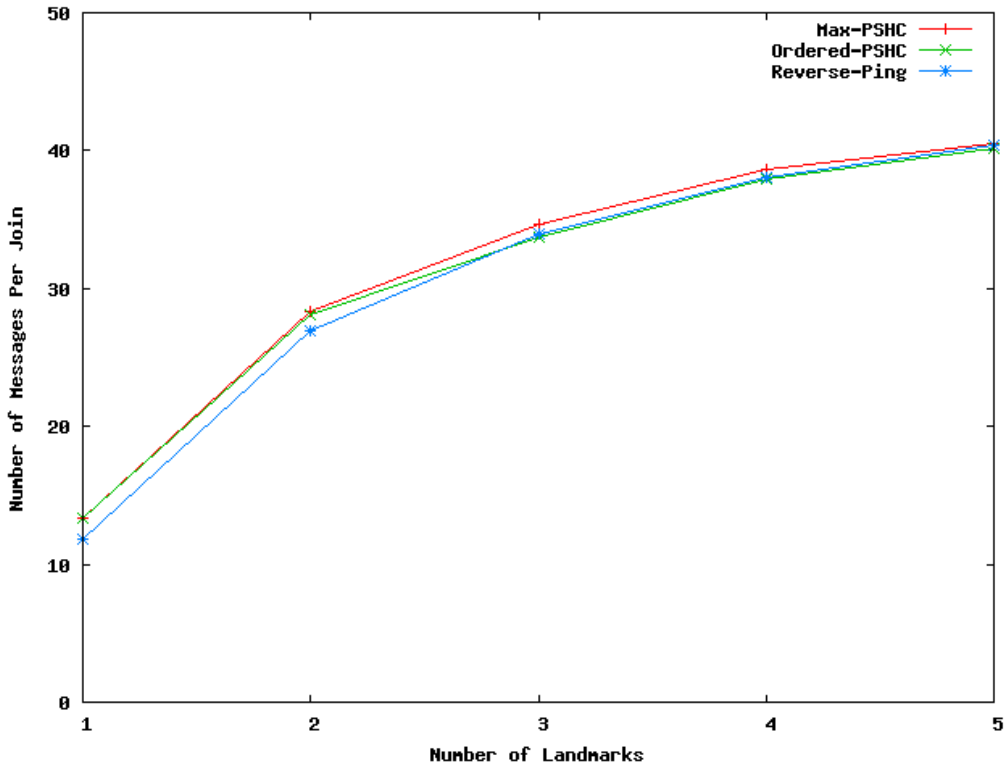
The similar average node degrees in Figure 9.5(a) and node degree distributions in Figure 9.5(b) indicate that the shape of the overlay probably does not vary greatly with the metric for a given set of landmarks. This is logical, since, a node's position in the overlay is dependant not only on its PSHC with its neighboring nodes, but its latency to the landmark as well.

9.3. Cost

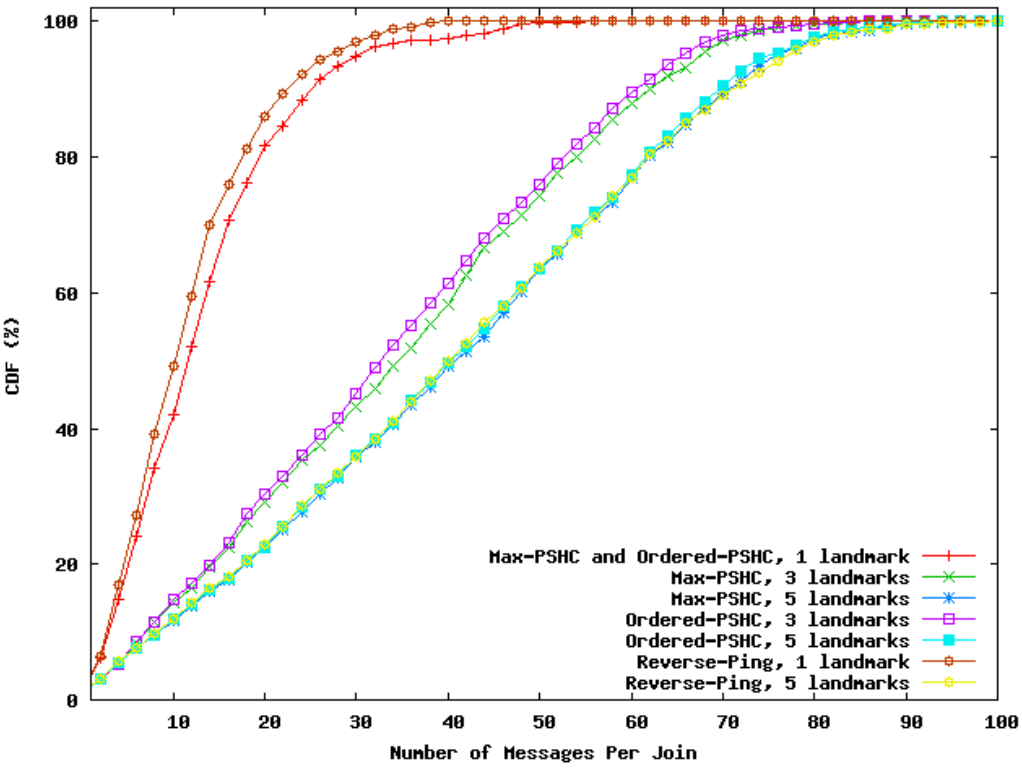
We measure cost in terms of number of pings, messages exchanged and connections required to maintain a consistent overlay as well as to discover the closest node.

Assuming a traceroute can be represented in 120 bytes on average, as indicated by our analysis in Section 5.1, each query message has a size of approximately 120*L bytes, where L is the number of landmarks. However, in our current implementation, we rely only on the shared hop IP addresses and last-hop latency, hence the average size of a query message reduces to 60*L bytes. Hence for 5 landmarks the average message size is 300 bytes, and an average of 40 messages incurs of total network load of 12KB per query, which is negligible given the state-of-the-art networking infrastructure deployed in the Internet today.

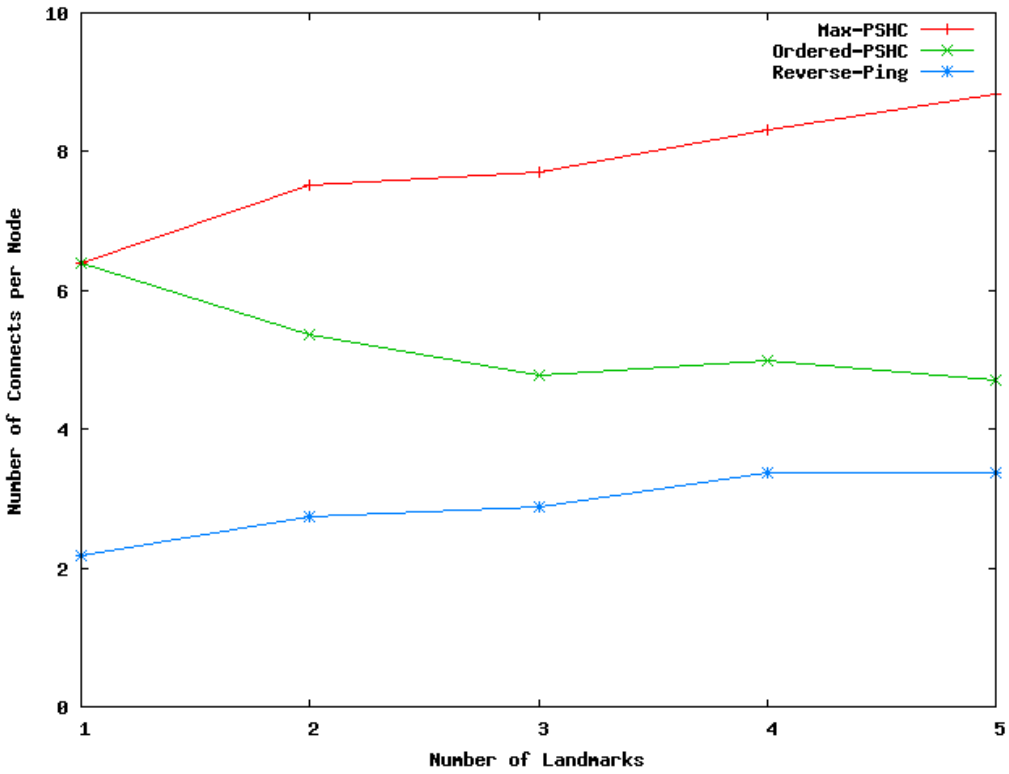
Figure 9.6: Cost of the TAO Algorithm



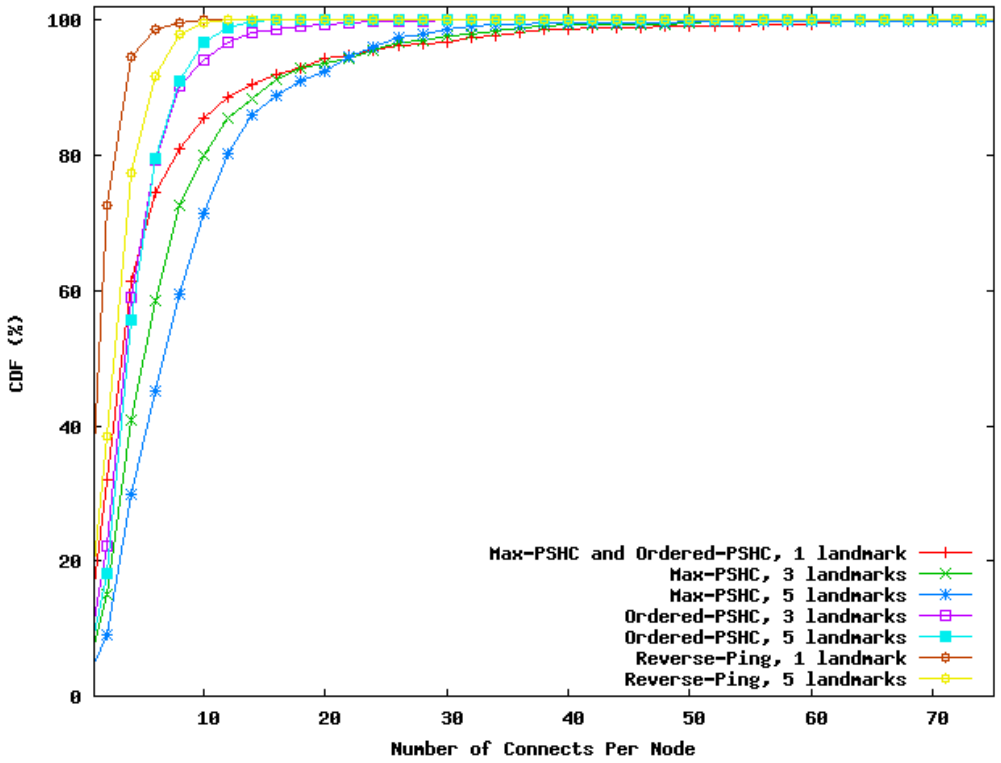
(a) Average Number of Messages per Join vs. Landmarks.



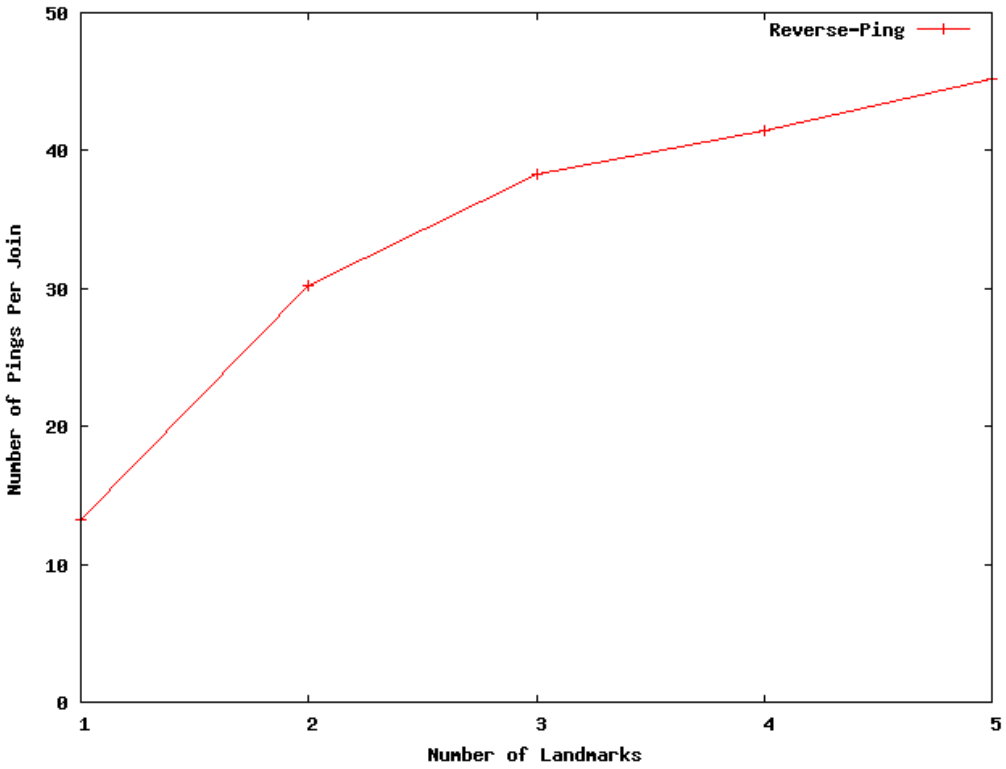
(b) CDF of Average Number of Messages per Join.



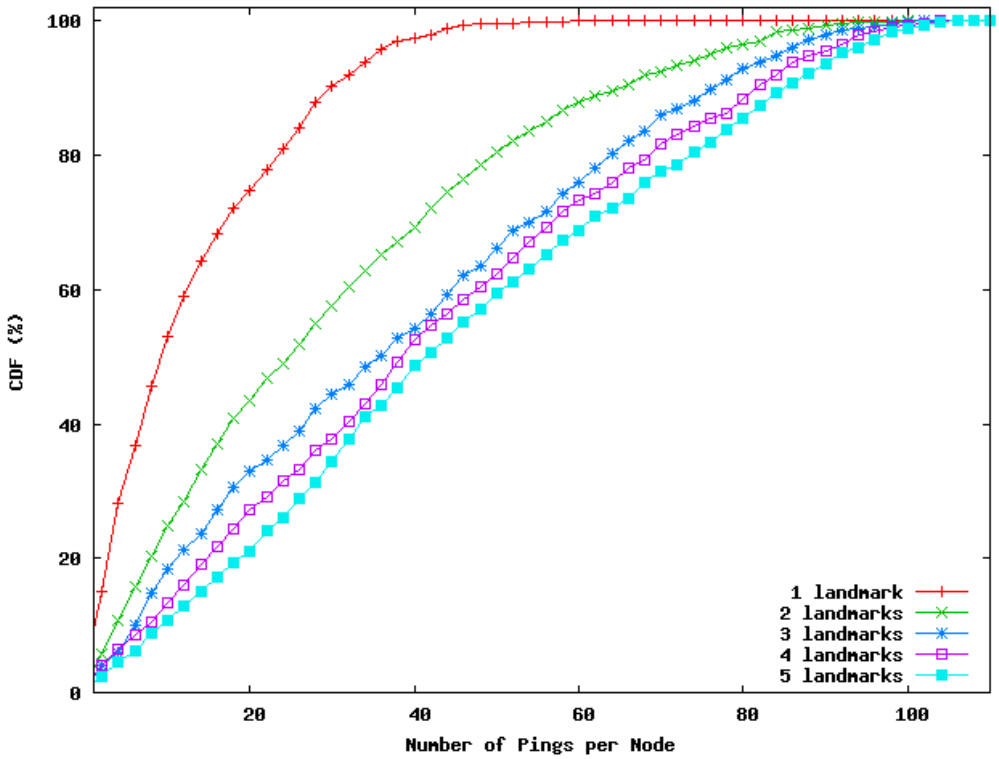
(c) Number of Connects per Node vs. Number of Landmarks.



(d) CDF of Number of Connects per Node (includes connects during overlay restructuring.)



(e) Number of Pings per Node vs. Number of Landmarks for Reverse-Ping.



(f) CDF of Pings per Node for Reverse-Ping.

A majority of the messages incurred is due to *repeat messages*, which are unavoidable. This is because the query is traced over different trees in the virtual topology, and it is likely that these trees may converge to the same result node in the graph over different paths. However, we have devised simple techniques to reduce the occurrence of repeat messages by having a node suppress tracing over a landmark tree in further hops once the branch point for that tree is found.

The average number of messages per join increases with the number of landmarks, as seen in Figure 9.6(a). This increase is because there are more branch points for each landmark, and hence more nodes representing these branch points which must be routed along for each query. However, the increase per additional landmark reduces towards 5 landmarks, and this is because there are more nodes representing branch points for multiple landmark trees in the virtual topology. Hence the cost of tracing the query along each landmark tree in the virtual topology is reduced.

A connection is assumed to provide RTT information for nodes that it does not already know the latencies to. However, these RTT measurements on connecting are part of a separate cost, and hence are not included in the ping measurements. The connect operation also reflects the number of connections changed during overlay restructuring. The average number and distribution of connects per node in Figure 9.6(c) illustrates the reason behind the difference in results in Max-PSHC and Ordered-PSHC. The number of connects per node actually decreases with more landmarks in the case of Ordered-PSHC because of its restrictive nature, whereas the number increases for Max-PSHC. The number of connects is lowest for Reverse-

Ping, since it makes the most accurate selections, avoiding connecting to non-closest nodes. The CDF in Figure 9.6(d) shows that for all metrics, more than 70% of the nodes need to make less than 10 connects. The number of connects also includes the overlay connections required to maintain a consistent overlay.

The overlay node degree distribution is much lower for TAO than other overlays, as can be seen in Figures 9.5(a) and (b), with approximately 70% nodes having 5 connections or less. Only the Random overlay has comparable node degree, since we expressly constructed random overlays with similar degree distribution to provide a fair comparison of other metrics. This indicates that the cost of *maintaining* the TAO network would be very low compared to other overlay networks.

In the case of Reverse-Ping, the number of pings per node varies similarly to the number of messages per join, increasing with the number of landmarks, and a smaller increase in pings for each additional landmark.

The cost analysis suggests that Max-PSHC is the most optimal metric to use, as it provides the best tradeoff of high accuracy and low cost.

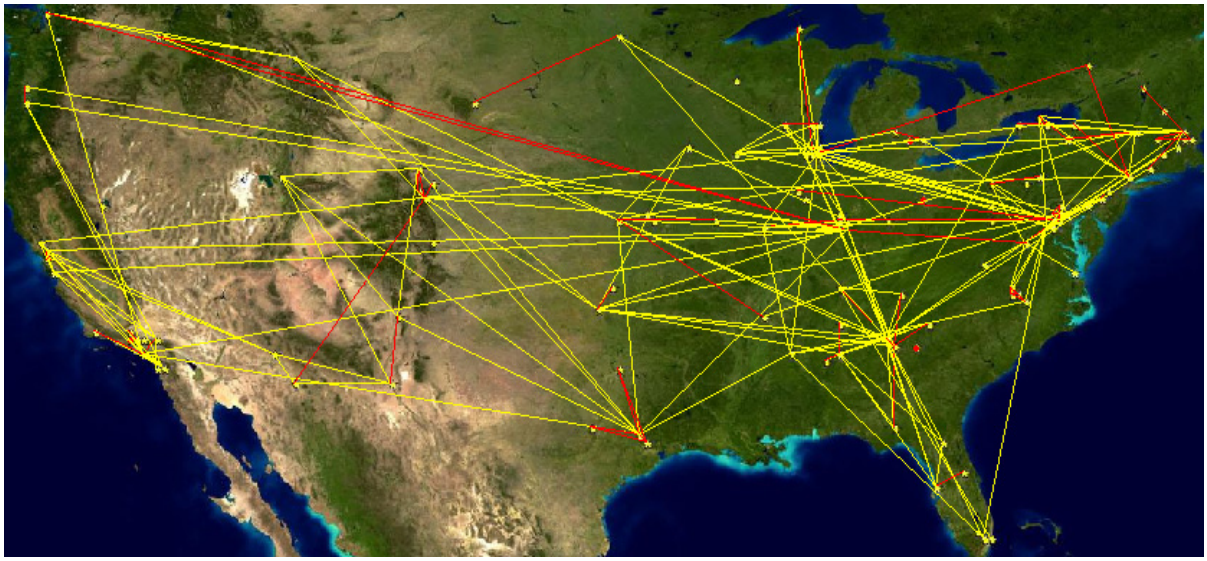


Figure 9.7: A Map of the TAO network (using Max-PSHC and five landmarks for NLANR nodes in North America.)

9.4. Overview

TAO gives much higher node selection accuracy and lower stretch than other techniques with relatively very low cost. Using more landmarks results in more accuracy but follows the law of diminishing returns. Performance is not affected by the selection of landmarks, giving similar results for random landmarks. TAO also has an interesting property that, for a given set of landmarks, the overlay would dynamically adopt a similar shape regardless of the order of the nodes joining, since the position of each node in the overlay tree for each landmark is dictated by its distance to that landmark. The reason behind the low stretch of TAO networks is the greater percentage of fractional stretch paths that it discovers.

Figure 9.7 shows the overlay graph for NLANR nodes in North America. The red dots point out the landmarks and the red lines indicate a connection to a closest node. The map does not depict nodes outside the North American continent and hence it does not reveal the fact that almost all international nodes are connected to the North American section of the overlay via

their closest nodes, which is already apparent from the low values of the maximum and average RDP in the results. Also, it can be seen that the overlay consists of interconnected star-like clusters, which is very similar to the topology of the Internet.

10. COMPARISON WITH PREVIOUS WORK

TAO is most similar in approach to Topology Aware Grouping [16]. However, Kwon, Fahmy et al designed TAG specifically for single source overlay multicast, and was not oriented towards closest node selection or general overlay organization. Hence they did not solve the long hop problem, although they do identify it as high-delay last hops, such as satellite links, and discuss it briefly. Furthermore, nodes organized themselves using traceroutes to the single source node, effectively using only one landmark, and as our results show, a single landmark performs poorly compared to the case of multiple landmarks.

In the case of closest node selection, TAO apparently performs better than Meridian, although both techniques have not been evaluated on a common Internet topology dataset as yet. However, Meridian cannot be compared with TAO in terms of stretch, since the former constructs an overlay such that nodes maintain neighbors as topologically diverse as possible to improve the efficiency of its closest node discovery, whereas TAO does precisely the opposite. Thus TAO has greater functionality than Meridian, since besides discovering closest nodes, it can directly be used as a low network overhead substrate for distributed applications.

The TAO framework has several inherent advantages over previous work, especially recursive probing techniques. As mentioned before, traceroutes need only be collected on the first join. Periodic traceroutes and regular probing is not required to discover newly joined nodes that could be closer, because such nodes would themselves be routed directly to their closest node in the overlay when they join. Existing nodes can then adaptively drop connections that are unnecessary for correctness of TAO.

Another major advantage of TAO over recursive probing is that if a node happens to randomly connect to its closest node, the query would not travel very far, since the closest node would be amongst the highest or second highest shared path cluster. Recursive probing techniques have no indication if they have found the closest node, and hence can continue recursively probing without end if some heuristic limit is not employed to stop them.

11. FUTURE WORK

Despite its impressive preliminary performance TAO, the framework has several open problems and great potential for enhancements. There is a lot of scope for research on how to efficiently use shared path semantics to infer more topological information with greater accuracy, and use this to meet a wider variety of application requirements, such as self-organizing resilient overlay networks, for instance.

11.1. Scalability and Performance Under Churn

The most important work would be to study how TAO scales for larger networks, to test its performance under churn, and devise effective techniques for these. Having a more loosely structured overlay, one that does not require TAO nodes to maintain a consistent virtual topology might be a feasible approach. In this case, nodes are aware of only some of their router-level peer representatives, and random walks or limited flooding may be used for routing. This trades off higher network overhead at joins for lower maintenance overhead.

11.2. Comparison With More Recent Techniques

We would need to compare the performance of TAO for node selection with more current techniques like Meridian. Meridian selects nodes with an average of 40% accuracy and a Relative Delay Penalty CDF curve similar to that of TAO, but it has been evaluated on different datasets. Hence our current work focuses on testing TAO on other topologies, and evaluating Meridian on the NLANR dataset.

11.3. Handling Routing Anomalies

A major open problem is how to deal with severe routing anomalies. Due to router mis-configurations in the Internet, it is possible that routes between two nodes could follow a highly suboptimal path, and TAO discovers overlay paths that are shorter. Routing asymmetry where the reverse path is different from the forward path, also affects TAO, since traceroute only returns forward paths. TAO is resilient to smaller anomalies, but since we could not find any large ones in the NLANR dataset, we could not analyze its impact on TAO or devise ways to circumvent it. However, using multiple landmarks reduces the

probability of having all collected traceroutes being corrupt, and hence the effect, of routing anomalies.

11.4. Graph Theoretic Analysis

The simplest, most obvious solution to construct a globally optimal overlay would be to measure latencies between all pairs of nodes and apply some algorithm on this data. The obvious problem with this is that it is not scalable. The less obvious, but more fundamental, problem is that even if such comprehensive measurements were available, to date there is no clear definition or description of a "globally optimal overlay", especially when considering the requirements of a specific application that would use this overlay. For instance, what proportion of closer and distant connections at each node would result in a globally minimum average network latencies between all pairs for a given application and topology? Is this a sufficient metric of optimality? Even if an optimal organization technique is discovered, can it be applied in a distributed and scalable fashion?

A Mathematical Foundation for Topology Awareness [26] is the only work to date that attempts to quantify topology matching and design a globally optimal overlay. It mathematically models a given overlay and uses this model to quantify topology matching, formulate an optimization problem to minimize the average inter-node latency and theoretically analyze the characteristics of a globally optimal overlay. The authors use the connectivity graph of the given overlay as the problem space and model the minimization as a Minimum Linear Arrangement (MINLA) problem, which is known to be NP-complete. To this end, they propose a simple technique whereby nodes recursively probe the neighbors of

their neighbors and swap positions in the overlay with their neighbors based on these measurements to converge to a globally optimal overlay. As such, this solution can be classified as a Recursive Probing approach. They prove the effectiveness of this method analytically based on their model of the overlay organization problem. However, this paper suffers from a critical limitation that it uses the given overlay graph as a starting point, and maintains it throughout the optimization problem. Even when nodes swap positions in the overlay graph, the layout does not change. It does not explore or discuss the possibility of changing the topology of the overlay graph itself.

To this end, we briefly propose a graph theoretic approach to construct an overlay network globally optimized for latency that is more generalized, where the overlay graph topology is only constrained by degree bounds. We identify the average delay between all pairs in the overlay as the most significant metric, and the optimal overlay would be one that minimizes this metric. We represent the overlay as a complete graph K_n of n vertices representing the n overlay nodes, with edges weighted with the Internet latencies between the nodes represented by the vertices adjacent to each edge. We define the optimum overlay as the *degree-constrained shortest-path subgraph G* that gives the minimum average distance between all pairs. The degree-constraint imposed on the subgraph has the same distribution as the node degree distribution in the overlay. It is possible that the degree-constraint is not constant, for instance, some high-capacity nodes can have a degree higher than other nodes, which is often the case in real-world overlays.

This is a non-trivial problem, since constructing a single-source degree-bounded shortest path tree (db-SPT) itself is an NP-complete problem, and shown to be a reduction of the Traveling Salesman Problem in [11]. However, [14] presents an algorithm to efficiently compute a single-source degree-constrained shortest path tree with complexity $O(|E|\ |V|)$. In our case, since the initial graph is complete, $|V|$ is n and $|E|$ is $n(n\text{-}1)/2$, which gives a complexity of $O(n^2(n\text{-}1)/2)$. Extending this to compute a degree-constrained sub-graph for all n vertices is not a trivial problem, and we propose exploring that avenue as future work. However, for preliminary exploration, a single-source db-SPT should be sufficient to provide some intuition as to the optimal selection of edges for shortest paths with bounded degree. Specifically, it would provide the weight distribution of edges adjacent to each vertex, which could indicate the distribution of nearer and farther neighbors an overlay node would have to maintain so that the resultant overlay would have the minimum average latency between all pairs of nodes.

12. CONCLUSIONS

In this thesis we present TAO, a Topology-Adaptive Overlay framework which uses traceroute information to adapt itself to the underlying network and shared-path semantics to search for closest nodes. TAO is distributed, self-organizing, lightweight and accurate. It does not require any services from the landmarks or the network, relying only on existing network techniques. TAO is lightweight, requiring minimal resources per node and state maintenance at each node. Besides topology mismatch, we identified and addressed two more problems, the Asymmetric Node Selection and the Long Hop problems. Hence TAO provides very precise network localization with as few as a three to five traceroutes and few

messages per join. Traceroutes are very concise, and can be piggybacked on application-specific messages to exchange topology information. These features make it easy to incorporate into existing distributed applications. It is one of the few schemes directly applicable to unstructured as well as structured overlays, and we describe a few such applications. We evaluated TAO on a real Internet dataset for three heuristic node selection strategies and identified the best of these for efficient closest node discovery. TAO locates closest nodes with 90% accuracy and incurs a sub-millisecond average relative delay penalty. TAO also discovers some overlay paths that have a lower end-to-end delay than the IP latency between the corresponding nodes, and hence results in an overlay with a very low stretch of about 1.2 with five landmarks. TAO outperforms all known topology-aware and network positioning schemes in terms of efficiency and accuracy.

TAO represents a paradigm shift in topology aware overlays in that it efficiently uses traceroutes and guarantees topology matching. We have taken an unconventional approach and used simple techniques to provide a very powerful and flexible framework to construct overlay networks that can be optimized for a wide range of applications. How to use this framework to optimize an overlay for a given application and topology is, however, an open research problem.

BIBLIOGRAPHY

[1.] D. Andersen, H. Balakrishnan, F. Kaashoek, R. Morris, "Resilient Overlay Networks," in the 18th ACM SOSP, 2001.

[2.] S. Banerjee, C. Kommareddy and B. Bhattacharjee, "Efficient Peer Location on the Internet," in Computer Networks, Volume 24, 2004.

[3.] BitTorrent, Inc. BitTorrent web site, 2001–2005, http://www.bittorrent.com

[4.] M. Castro, P. Druschel, Y. C. Hu and A. Rowstron, "Exploiting network proximity in peer-to-peer overlay networks," in the International Workshop on Future Directions in Distributed Computing, 2002.

[5.] M. Costa, M. Castro, A. Rowstron and P. Key, "PIC: Practical Internet Coordinates for Distance Estimation," in the 24th IEEE International Conference on Distributed Computing Systems (ICDCS), 2004.

[6.] F. Dabek, R. Cox, F. Kaashoek and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," in ACM SIGCOMM, 2004.

[7.] P. Francis, S. Jamin, V. Paxson, L. Zhang, D.F. Gryniewicz and Y. Jin, “An architecture for a global Internet host distance estimation service,” in IEEE INFOCOM, 1999.

[8.] “Gnutella,” http://www.gnutella.com

[9.] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload", in the 19th ACM SOSP, 2003

[10.] K. Gummadi, S. Saroiu and S. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts," in SIGCOMM IMW, 2002.

[11.] P. Jurcík and Z. Hanzálek, “Construction of the Bounded Application layer Multicast Tree in the Overlay Network Model by the Integer Linear Programming,” in the 10th IEEE International Conference on Emerging Technologies and Factory Automation, 2005.

[12.] D. Kaminsky, “Paratrace: Parasitic Traceroute via Established TCP Flows & IP-ID Hopcount,” http://www.doxpara.com/read.php/docs/paratrace.html

[13.] “Kazaa,” http://www.kazaa.com

[14.] S. Khuller, K. Lee and M. Shayman, “On Degree Constrained Shortest Paths,” in ESA, 2005.

[15.] B. Krishnamurthy and J. Wang, "On Network-Aware Clustering of Web Clients", in ACM SIGCOMM, 2000.

[16.] M. Kwon and S. Fahmy “Topology-aware Overlay Networks for Grouping Communication,” in NOSSDAV, 2001.

[17.] Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, “Location-aware topology matching in P2P systems", in IEEE INFOCOM, 2004.

[18.] “Napster,” http://www.napster.com

[19.] T. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," In IEEE INFOCOM, 2002.

[20.] The NLANR project, http://amp.nlanr.net/AMP/

[21.] M. Pias, J. Crowcroft, S. Wilbur, S. Bhatti and T. Harris, "Lighthouses for scalable distributed location," in the 2nd Intl. Workshop on Peer-to-Peer Systems, 2003.

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

[23.] S. Ratnasamy, M. Handley, R. Karp and S. Shenker, “Topologically-aware Overlay Construction and Server Selection,” in IEEE INFOCOM, 2002.

[24.] M. Ripeanu, A. Iamnitchi and I. Foster, “Mapping the Gnutella Network,” in IEEE Internet Computing, 2002.

[25.] Jordan Ritter, "Why Gnutella Can't Scale. No, Really.", http://www.darkridge.com/~jpr5/doc/gnutella.html

[26.] H. Rostami and Jafar Habibi, "A Mathematical Foundation for Topology Awareness of P2P Overlay Networks", in Grid and Cooperative Computing (GCC), 4th International Conference, 2005.

[27.] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems,” in International Conference on Distributed Systems Platforms (Middleware), 2001.

[28.] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," in ACM SIGCOMM Internet Measurement Workshop, 2002.

[29.] Y. Shavitt and T. Tankel, "On the Curvature of the Internet and its usage for Overlay Construction and Distance Estimation," in IEEE INFOCOM, 2004.

[30.] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in ACM SIGCOMM, 2001.

[31.] M. C. Toren, “tcptraceroute,” http://michael.toren.net/code/tcptraceroute/

[32.] http://www.exit109.com/~jeremy/news/providers/traceroute.html

[33.] M. Waldvogel and R. Rinaldi, "Efficient Topology-Aware Overlay Network", in Hot Topics in Networks I (HotNets-I) 2002.

[34.] X. Wang, Y. Zhang, X. Li and D. Loguinov, "On Zone-Balancing of Peer-to-Peer Networks: Analysis of Random Node Join," in ACM SIGMETRICS, 2004.

[35.] "What's the average wingspan of a packet?" http://www.nlanr.net/NA/Learn/wingspan.html

[36.] R. Winter, T. Zahn and J. Schiller, "Topology Aware Overlay Construction in Dynamic Networks," In Proc. of 3rd International Conference on Networking, March 2004

[37.] B. Wong, A. Slivkins and E. G. Sirer, "Meridian: A Lightweight Network Location Service without Virtual Coordinates," in ACM SIGCOMM, 2005.

[38.] H. Zheng, E. K. Lua, M. Pias and T. Griffin, "Internet Routing Policies and Round-Trip-Times," in PAM, 2005.