Topologically-Aware Overlay Construction and Server Selection

Sylvia Ratnasamy, Mark Handley, Richard Karp, Scott Shenker

Abstract— **A number of large-scale distributed Internet applications could potentially benefit from some level of knowledge about the relative proximity between its participating host nodes. For example, the performance of large overlay networks could be improved if the application-level connectivity between the nodes in these networks is congruent with the underlying IP-level topology. Similarly, in the case of replicated web content, client nodes could use topological information in selecting one of multiple available servers. For such applications, one need not find the optimal solution in order to achieve significant practical benefits. Thus, these applications, and presumably others like them, do not require *exact* topological information and can instead use sufficiently informative hints about the relative positions of Internet hosts.**

In this paper, we present a *binning* scheme whereby nodes partition themselves into bins such that nodes that fall within a given bin are relatively close to one another in terms of network latency. Our binning strategy is simple (requiring minimal support from any measurement infrastructure), scalable (requiring no form of global knowledge, each node only needs knowledge of a small number of well-known landmark nodes) and completely distributed (requiring no communication or cooperation between the nodes being binned).

We apply this binning strategy to the two applications mentioned above: overlay network construction and server selection. We test our binning strategy and its application using simulation and Internet measurement traces. Our results indicate that the performance of these applications can be significantly improved by even the rather coarse-grained knowledge of topology offered by our binning scheme.

I. Introduction

Several ongoing projects make use of application-level or overlay networks [1], [2], [3], [4], [5], [6], [7], [8], [9]. In these applications, each participating end-host node is logically connected[1] to a small subset of the other participant nodes (we call this subset the node's neighbors) to form an overlay network. A path on the overlay network then consists of a series of application-level, not IP-level, hops between the source and destination nodes. However, in current applications, little effort is made to ensure that this application-level connectivity is congruent with the underlying IP-level network topology. This in turn can lead to inefficient routing where, for example, a node in Berkeley has its neighbor nodes in Europe and hence its path to a node in Stanford may traverse distant nodes in Europe. Ideally, one would like to improve routing performance by avoiding such unnecessary high latency hops. Thus, a fundamental challenge in using large-scale overlay networks is to incorporate IP-level topological information in the construction of the overlay to improve routing performance.

The utility of topological information is however, not restricted to overlay network construction. Content distribution over the Internet is another example where such information could improve performance. In recent years, the Web has moved from an architecture where data objects are located at a single origin server to one where objects are often replicated at multiple, geographically dispersed servers. Client requests for content are redirected to a close-by replica server rather than the origin server. The process of selecting a "good" server, *i.e.* one that is close to the client in terms of latency, might be significantly improved if both the client and servers could indicate their position on the Internet. Likewise, peer-to-peer file sharing applications such as Napster and Gnutella typically have the same file available at multiple peers. Topological information could be used to select a close-by peer for quicker down-loads.

The problem we explore in this paper is whether it is possible to gather topological information in a manner that is both practical and scalable and if so, how could this information be effectively incorporated into the design of distributed systems such as overlay networks and content distribution systems?

At this point, it is worth briefly discussing the desirable properties of a solution to the above problem. Administrator configured overlay networks, such as those used in CDNs [10], [11], [12], can be made to fit the underlying IP topology. However, such overlays are not generally applicable; the hand-crafted and centralized nature of the overlay construction process makes it untenable for large overlays (millions of nodes) and for overlays, such as peer-to-peer file sharing [1], [2], [13], where there is no single central administrator. One might also imagine centralized solutions [14] wherein a single site gathers network topology and routing information to infer the relative proximity of hosts. Such solutions however result in a single point of failure and potential bottleneck. Also, for completely decentralized applications such as peer-to-peer file sharing there is no clear incentive, economic or otherwise, for a third party to offer such a service. We also wanted to avoid more elaborate solutions where the overlay network structure is improved slowly over time [15]. This is because in many of the targeted applications [1], [2], [13] for overlay networks participant nodes join and leave the application on short time-scales. A solution that operates over long time-scales would be continually reacting to fluctuating node membership without stabilizing. Thus, we concluded that a desirable solution should be simple, fast, distributed, and should scale to millions of nodes.

Another question that arises is: what form of network measurements or data should we use to derive topological information? Network tools such as *traceroute* are primarily intended for network diagnostic purposes and are too heavy-weight and intrusive for use by large scale applications. Using *traceroute* in large scale applications would result in excessive load on the network. Additionally some edge sites disable ICMP for security reasons. The use of BGP routing table dumps [14] also faces certain problems. Such information is not directly available to end-user applications. One would thus either require privileged access to internal network information from ISPs, or would depend on third party monitoring services that do have such privi-

S.Ratnasamy, M.Handley,R.Karp and S.Shenker are with the ICSI Center for Internet Research, Berkeley, CA, USA. S.Ratnasamy and R.Karp are also with the Computer Science Division, University of California, Berkeley, CA, USA.

[1] For example, by a TCP connection.

leged access and which publish such information on the Web as a service to the community. Neither of the above is a reasonable option in general. We thus chose to use network latency because latency is often a direct indicator of the performance seen by end-host nodes and can be easily measured in a light-weight, end-to-end, non-intrusive manner.

Finally, one might question the required accuracy of the topological information. In both our targeted applications (overlay construction and server selection), incorporating topology awareness is really a performance optimization and is not *required* for correct operation. More importantly, our results show that both server selection and overlay construction show significant performance improvements with only approximate topological information. For these reasons, we view scalability and practicality as more important goals than accuracy. Hence, unlike many research projects [16], [17], [18], the focus of our work is not on highly accurate topology modeling, nor on building a general-purpose measurement service. While these are important problems in their own right, our focus is instead on the use of simple topological hints to solve certain application-level problems.

In this paper, we propose a distributed *binning* scheme whereby nodes partition themselves into *bins* such that nodes that fall within a given bin are relatively close to one another in terms of network latency. Our scheme requires a set of well-known *landmark* machines spread across the Internet. An application node measures its distance, *i.e.* round-trip time, to this set of well known landmarks and independently selects a particular bin based on these measurements. Our binning scheme is simple, requiring very little support from the infrastructure. The only infrastructure required is a small number (our results using Internet trace data indicates that 8-12 machines should suffice for the current scale of the Internet) of relatively stable landmark machines. Very little work is required of these landmark machines – they need only echo "ping" messages – and landmarks could in fact be unsuspecting participants in the binning! [2] Landmarks do not actively initiate measurements nor gather or disseminate measurement information. Binning is scalable because nodes independently discover their bins without communicating or coordinating with other application nodes.

Given the above binning strategy, we turn to the problem of how one might effectively incorporate such a scheme in distributed applications. We apply this binning strategy to two applications: overlay network construction and server selection. Results obtained through simulation and from Internet measurement traces indicate that even the rather coarse-grained topological information provided by our binning strategy can significantly improve the performance of systems such as overlay networks and CDNs.

The remainder of this paper is organized as follows: Section II describes and evaluates our binning scheme. In Sections III and IV we describe and evaluate the application of our binning strategy to overlay network construction and server selection respectively. Finally, we discuss related work in Section V and conclude in Section VI.

[2] For example, one might imagine using the DNS root name servers as the set of landmarks and using the DNS response times as latency measurements.

II. Distributed Binning

The goal of our distributed binning scheme is to have a set of nodes independently partition themselves into disjoint "bins" such that nodes within a single bin are relatively closer to one another than to nodes not in their bin.

Our scheme assumes the existence of a well known set of machines that act as landmarks on the Internet. Application nodes might discover the IP addresses of these machines using the DNS (for example, landmark machines could be named *lm1.bin.net, lm2.bin.net,* ... rather than hard-coding landmark IP addresses into the application).

We achieve a form of "distributed binning" of nodes based on their relative distances, *i.e.* latencies from this set of landmarks. A node measures its round-trip-time to each of these landmarks and orders the landmarks in order of increasing RTT. [3] Thus, based on its delay measurements to the different landmarks, every node has an associated ordering of landmarks. This ordering represents the "bin" the node belongs to. The rationale behind this scheme is that topologically close nodes are likely to have the same ordering and hence will belong to the same bin.

We can however do better than just using the ordering to define a bin. A node's RTT measurements to each landmark offers two kinds of information: the first is the *relative* distance of the different landmarks from the given node and the second is the *absolute* value of these distances. The ordering described above only makes uses of the relative distances of the landmarks from a node. The absolute values of the RTT measurements are indicated as follows: we divide the range of possible latency values into a number of *levels*. For example, we might divide the range of possible latency values into 3 levels; level 0 for latencies in the range [0,100]ms, level 1 for latencies between [100,200]ms and level 2 for latencies greater than 200ms. We then augment the landmark ordering of a node with a *level vector*; one level number corresponding to each landmark in the ordering. To illustrate, consider node A in Figure 1. Its distance to landmarks l_1, l_2 and l_3 are 232ms, 51ms and 117ms respectively. Hence its ordering of landmarks is $l_2 l_3 l_1$. Using the 3 levels defined above, node A's level vector corresponding to its ordering of landmarks is "0 1 2". Thus, node A's bin is "$l_2 l_3 l_1 : 012$".

Note that with the above binning scheme (and unlike schemes in [16], [17]), a node only needs to discover the distance between itself and the landmarks and can measure these distances itself. Nodes need not know the inter-landmarks distances or the distance of other nodes from the landmarks. Also, binning is robust to the failure of one or more landmark nodes. In the case of landmark failures, new nodes are binned using the surviving landmarks while previously binned nodes need only drop the failed landmark(s) from their bin identifier. Of course, performance degrades with fewer landmarks (the effect of the number of landmarks on binning is quantified later).

The purpose of this binning scheme is to be useful to applications. We explore this in Sections III and IV. However, we first address two questions. First, is our binning proposal scalable? From the point of view of the nodes being binned, our scheme is clearly scalable since nodes need only have knowledge of (and

[3] More precisely, if $L = \{l_0, l_1, ..l_{m-1}\}$ is the set of m landmarks, then a node A creates an ordering L_a on L, such that i appears before j in L_a if $rtt(a, l_i) < rtt(a, l_j)$ or $rtt(a, l_i) = rtt(a, l_j)$ and $l_i < l_j$.

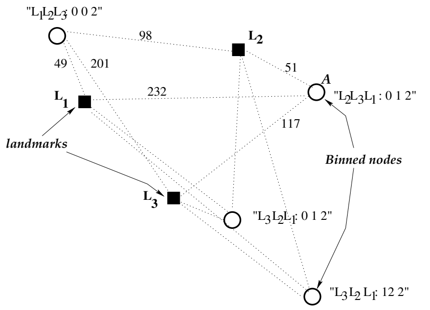

Fig. 1. *Distributed binning*

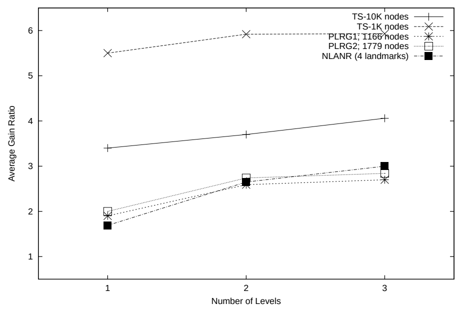

Fig. 2. *Effect of number of levels (#landmarks = 12)*

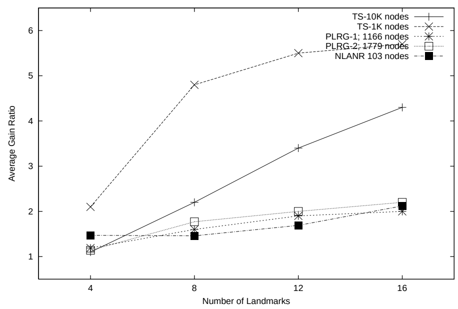

Fig. 3. *Effect of number of landmarks (#levels = 1)*

perform measurements to) a small set of landmarks. The only remaining issue is the load such measurement place on the landmarks; a quick back-of-the-envelope calculation indicates that this load is quite manageable for systems with a million nodes or more. For example, assume we have a million nodes and need 10 pings to obtain a good sample of the RTT to a landmark. If nodes refresh their bin information once per hour,[4] that would impose a load of approximately 2700 pings per second on each landmark.[5] To further improve scalability, one might have multiple close-by nodes act as a single logical landmark. For example, the Computer Science department at U.C.Berkeley might install 10 machines that all act as a single landmark, say *lm1.bin.net*. Since these machines are geographically and topologically co-located, application nodes can use any one of them as Landmark#1 (and DNS round-robin might be used to load balance amongst the 10 machines).

The second question we address is; does binning do a reasonable job of placing nearby nodes into the same bin? To do so, we calculate the following:

- average inter-bin latency: the average latency from a given node to all nodes not in its bin
- average intra-bin latency: the average latency from a given node to all nodes in its bin

For every bin containing two or more nodes, we compute the ratio of the inter-bin latency to the intra-bin latency for nodes within that bin. We call this the node's *gain* ratio. The metric we use to evaluate binning is the average over all nodes (that belong to bins of size greater than one node) of the gain ratio. Intuitively, what the gain ratio represents is that on the average if a node were to communicate with a random node from its own bin instead of a random node not in its bin, then the communication latency would be reduced by a factor equal to its gain ratio. A higher gain ratio indicates a bigger reduction in latency and is hence desirable. Of course, binning (and hence its performance) by itself is not particularly interesting; the utility really comes from its application to actual Internet systems. Our evaluation of binning is thus primarily a sanity check to confirm that binning does achieve its goal of clustering nearby nodes.

We tested our binning algorithm on both simulated topologies and Internet measurement data. The test topologies we use are as follows:

1. TS-10K and TS-1K: Transit-Stub topologies [19] with 10,000 and 1,000 nodes respectively. TS topologies model networks using a 2-level hierarchy of routing domains with transit domains that interconnect lower level stub domains. To these TS topologies, we assign link latencies of 20ms for intra-transit domain links, 5ms for stub-transit links and 2ms for intra-stub domain links (we also experimented with a delay distribution of 100, 10 and 1ms instead of 20, 5 and 2ms respectively with no real change in our results).
2. PLRG1 and PLRG2: Recent studies [20], [21] have indicated that the Internet's degree distribution follows a power-law. Motivated by these observations, degree-based generators have been proposed [22] which appear to better model the measured Internet topology. We make use of the same power-law random graph generator as used by [21], [22]. PLRG1 and PLRG2 are Power-Law Random Graphs with 1,166 and 1,779 nodes respectively. To each link in the topology, we assign a random delay between 5 and 100ms. [6]
3. NLANR: The Active Measurement Project (AMP) at the National Laboratory for Applied Network Research (NLANR) uses a distributed network of over 100 active monitors to systematically perform scheduled measurements between each other. Amongst other things, monitors measure the round trip times (RTT) between the different pairs of monitors. We use an NLANR data set with the round-trip-times between 103 such monitors. Our data set is from measurements taken in April 2001. The NLANR sites are primarily located at universities in North America. The details of the NLANR measurement methodology and sites is described in [23].

Recent work has focused on placement strategies for instrumentation boxes [24], [18]. In our work, we make minimal assumptions about the placement of our Landmark machines. For each of the above topologies, we place the required number of landmarks at random with the only condition that the landmarks be separated from each other by a certain number of hops. In our simulations, we use a separation distance of four hops. More so-

[4] A refresh rate of once/hour won't enable nodes to react immediately to events such as link failures; however, we deem this acceptable since stale binning information will only result in somewhat increased delays.

[5] Such loads are easily handled by modern midrange PCs. We experimented with an 800MHz Athlon-based machine and found that it could easily receive 2,700 pings per second with no loss. Moreover, as a reference point, a DNS trace from October 1999 shows that the root name server *f.root-servers.net* was handling (with some headroom) a 10 minute average of 1,600 requests per second. Since ICMP is presumably lower cost than serving DNS, the load on landmarks would not be a significant problem.

[6] These delay assignments are probably quite misleading, since the true Internet latencies are not random; at the very least, they usually obey the triangle inequality. However, we do not yet know how to realistically model the latencies on a PLRG.

phisticated placement schemes, as described in [24] would only serve to improve our results.

The key parameters affecting performance are

- type and scale of topology
- number of nodes being binned
- number of levels
- number of landmarks

For all results presented in this section, all nodes in the data set (other than landmark nodes) are binned.

In Figure 2, we fix the number of landmarks at 12 and plot the effect of increasing numbers of levels on the average gain ratio. For only the NLANR data plot, the number of landmarks was fixed at 4.[7] With a single level, all latencies belong to the same level and hence only the landmark ordering defines the bin. For higher levels, we use a demarcation latency value D_{dem} to divide the range of latency values into the desired number of levels. We experimented with different demarcation values and found that for most reasonably selected values, our results remain largely unchanged. In all the simulations presented here, we use the average latency of the underlying topology, denoted by D_{ip}, to demarcate the different levels. At two levels, we divide latencies into two levels as being less than D_{ip} (level 0) or greater than D_{ip}. We divide latencies into three levels as being less than $(0.75 \times D_{ip})$, or between $(0.75 \times D_{ip})$ and $(1.25 \times D_{ip})$ or greater than $(1.25 \times D_{ip})$.

In Figure 2, an average gain ratio of 4.06 for TS-10K with 3 levels indicates that for a given node, a node within its own bin is on an average, four times closer than one not in its bin. While increasing the number of levels improves the gain ratio, we see that the improvement appears to rapidly saturate, indicating that in practice, just 2-3 levels should suffice.

Figure 3 plots the average gain ratio for an increasing number of landmarks. The number of levels was fixed at one. For the reason mentioned above, the data points for the NLANR plot actually correspond to 2,3,4 and 5 landmarks (not 4,8,12 and 16; we plotted it on the same graph due to space limitations). As expected, increasing the number of landmarks results in more fine-grained binning, thereby improving the gain ratio.

By comparing the results for TS-10K and TS-1K, we see that the gain ratio is clearly affected by the size of the underlying topology. Our simulations showed that for a given topology, the gain ratio varies little with the number of nodes being binned, *i.e.* for a given topology, the density of nodes being binned does not affect the gain ratio.

We also measured the effect of the number of landmarks on the number of bins for the different topologies and found that with the exception of TS-10K, the number of bins saturates around 8 landmarks explaining why we see little improvement in the gain ratio beyond that point.

The above results tell us what kind of gain ratios our binning scheme provides and how it is affected by the number of levels and landmarks. But, we would like to know how well our binning scheme works relative to other binning techniques. *I.e.* are the above gain ratios good? We use the following algorithms to provide us with reasonable upper and lower bounds on the gain ratios one might expect from any binning technique:

- Random binning: Using the same number of bins as generated by our landmark-based binning scheme, each node selects a bin at random. Random binning thus makes no attempt to achieve locality and acts as a lower bound for the gain ratio.
- Nearest-neighbor clustering: Our binning problem is very similar to the clustering problem which has been extensively studied in the theory community. In clustering the input comprises a set of data points, each having a set of attributes and a similarity measure among them. The goal is to find clusters such that data points in one cluster are more similar to one another, and data points in separate clusters are less similar to one another. While optimal clustering is known to be NP-hard, a widely used clustering algorithm known to achieve good results for a variety of applications is Nearest Neighbor Clustering. In Nearest Neighbor clustering, each node is initially assigned to a cluster by itself. At each iteration, the two *closest* clusters are merged into a single cluster. The algorithm terminates when the required number of clusters are obtained. When applied to our problem, we repeatedly merge the two clusters with the minimum inter-cluster latency, where the inter-cluster latency between two clusters is the average latency between nodes from one cluster to nodes in the other. Nearest neighbor clustering requires global knowledge of the latencies between all the nodes and is clearly not practical for actual deployment on the Internet, but serves as a useful potential lower bound to any distributed binning scheme.

Figure 4 plots the average gain ratio for the different topologies using landmark-based binning, random binning and nearest-neighbor clustering. Not surprisingly, random binning yields a gain ratio of approximately 1.0 for all the test topologies. In all cases, the performance of landmark-based binning comes close to that of nearest-neighbor clustering and greatly outperforms random binning.

In conclusion, the binning scheme we've proposed does a reasonable job of placing nearby nodes into the same bin. We now address the more fundamental and important question of whether the binning scheme can be of use to applications.

III. Topologically-Aware Construction of Overlay Networks

In this section, we apply our binning scheme to the construction of overlay networks. We focus on two types of overlay networks.

The first are *structured* overlays wherein nodes are interconnected (at the application-level) in some well-defined manner. A number of designs for such structured overlays have been proposed recently [4], [3], [25], [26]. In our work, we focus on one such system, called a Content-Addressable Network (CAN). Our scheme for topologically sensitive CAN construction should however be applicable to similar systems such as Chord [3], PASTRY [26] and Tapestry [25].

The second type of overlay networks we consider are *unstructured* overlays. End-system multicast [9], [27] and Scattercast [28] both build such unstructured meshes over which multicast trees are constructed. We study the use of our binning scheme in a generic example of an unstructured overlay con-

[7] Because the NLANR data has only 103 nodes, we avoid using a larger number of landmarks because this would cause the nodes to be spread across a large number of bins, resulting in very few nodes per bin.

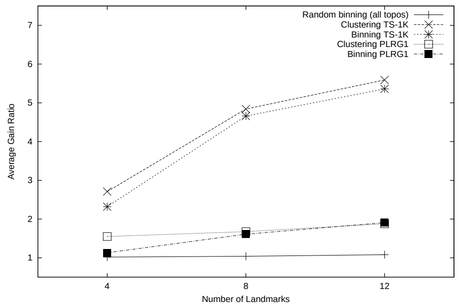

Fig. 4. *Comparison of different binning techniques (#levels=1)*

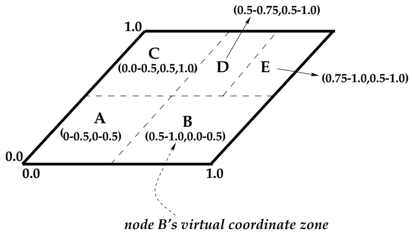

Fig. 5. *Example 2-d coordinate overlay with 5 nodes*

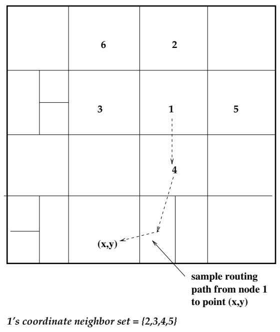

Fig. 6. *Example 2-d space*

struction.

The metric we use to evaluate our overlay construction algorithms is the ratio of the average inter-node latency on the overlay network to the average inter-node latency on the underlying IP-level network. We call this the *latency stretch*; lower values of stretch are thus desirable.

A. Topologically-sensitive CAN construction

A Content-Addressable Network is an application-level network whose constituent nodes can be thought of as forming a virtual d-dimensional Cartesian coordinate space. This coordinate space is completely logical and bears no relation to any physical coordinate system. At any point in time, the entire coordinate space is dynamically partitioned among all the nodes in the system such that every node "owns" its individual, distinct zone within the overall space. For example, Figure 5 shows a 2-dimensional $[0,1] \times [0,1]$ coordinate space partitioned between 5 CAN nodes. [8] Nodes in the CAN self-organize into an overlay network that represents this virtual coordinate space. A node learns and maintains as its set of neighbors the IP addresses of those nodes that hold coordinate zones adjoining its own zone. This set of immediate neighbors serves as a coordinate routing table that enables routing between arbitrary points in the coordinate space. Intuitively, routing on the CAN works by following the straight line path through the Cartesian space from source to destination coordinates. Figure 6 shows a sample routing path. For a d dimensional space partitioned amongst n nodes, the average routing path length is thus $O(d(n^{1/d}))$ and individual nodes maintain $O(d)$ neighbors.

The CAN construction mechanisms described in [4] allocate nodes to zones at random.[9] Thus, a node's neighbors on the CAN need not be topologically nearby on the underlying IP network. This can lead to inefficient routing because *every* application-level hop on the CAN could potentially be between two geographically (and topologically) distant nodes.

In this section, we apply our binning strategy to construct CAN topologies that are congruent with the underlying IP topology. Let us assume for the moment that only the ordering of landmarks is used for binning (the following ideas can be trivially extended to include level vectors). With m landmarks, $m!$ such orderings are possible. Accordingly we partition the coordinate space into $m!$ equal sized portions, each corresponding to a single ordering. Our current scheme to partition the space into $m!$ portions works as follows: assuming a fixed cyclical ordering of the dimensions (*e.g.* xyzxyzx...), we first divide the space, along the first dimension, into m portions, each portion is then sub-divided along the second dimension into $m-1$ portions each of which is further divided into $m-2$ portions and so on. Previously, a new node joined the CAN at a random point in the entire coordinate space. Now, at join time, a CAN node must first bin itself, *i.e.* based on its delay measurements to the set of landmarks, each node determines its associated bin. The new node then joins the CAN at a random point in that portion of the coordinate space associated with its landmark ordering.

A consequence of the above construction scheme is that the coordinate space is no longer uniformly populated. Because some bins are more highly populated than others their corresponding portions of the coordinate space are also more densely occupied than others leading to an uneven distribution of the size of zone spaces amongst the nodes. Thus some nodes hold much larger coordinate zones that others. We defer this problem of achieving While we believe the use of background load balancing techniques (described in [4]) where an overloaded node hands off a portion of its space to a more lightly loaded one might be used to alleviate this problem, we do not explore this question further in this paper and defer it to future work.

A subtle side-effect of the uneven partitioning of the space is that the average number of hops on the path between two points in the CAN space decreases. This is because a single node might own a a disproportionately large zone. Such a node thus has the ability to cross a large portion of the coordinate space in a single hop leading to shorter paths than would be the case if the space were uniformly partitioned. This reduced path length in turn leads to lower average CAN path latencies because (CAN path latency) = (#hops) $\times$ (latency of each hop). In order to not take advantage of this reduced CAN latency caused by an uneven partitioning of the space, we calculate the average CAN path latency using binning-based construction as follows: for a CAN with binning-based construction, we divide the path latency by the number of hops on the pathm to get the per-hop latency. We then multiply this per-hop latency by the average number of hops on the randomly constructed CAN (for which the space is evenly partitioned). This gives us the path latency for a CAN with binning based construction without taking advantage of the

[8]The description of how nodes create and maintain this structure is not required to follow the discussion in this paper. CAN operation is described in detail in [4].

[9]With the exception of Section 3.6 in [4] where the idea of binning is briefly introduced as work in progress.

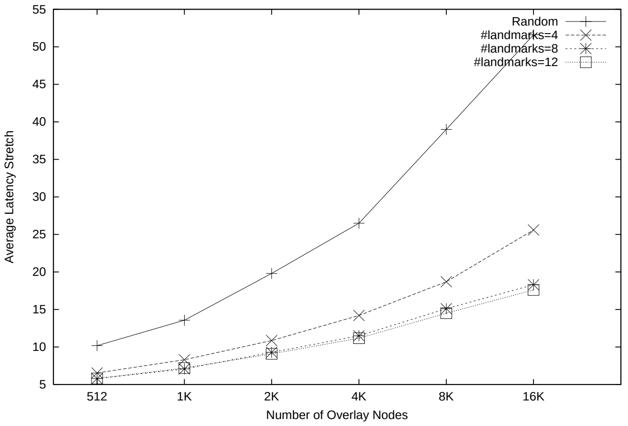

Fig. 7. *Stretch for a 2-dimensional CAN; topology TS-1K; #levels=1*

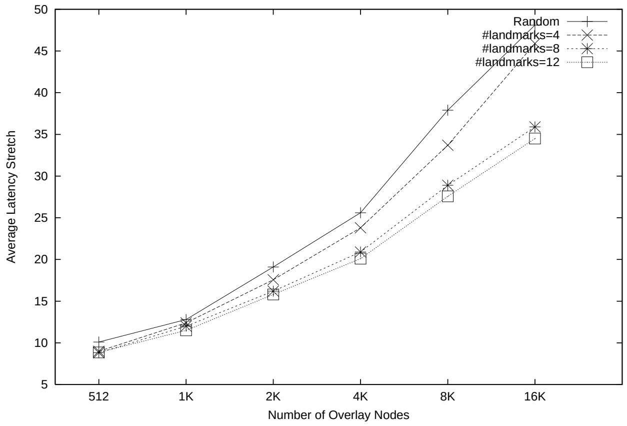

Fig. 8. *Stretch for a 2-dimensional CAN; topology PLRG2; #levels=1*

Construction	Latency Stretch
Random	4.44
Bin, #landmarks=2	4.33
Bin, #landmarks=3	3.9
Bin, #landmarks=4	3.2
Bin, #landmarks=5	3.1

TABLE I

Stretch on a 2-d CAN using NLANR; #levels=1

uneven space distribution.[10] Finally, we divide this by the path latency on the underlying IP-level network to obtain the latency stretch.

Figure 7 plots the CAN latency stretch (defined above) for increasing CAN sizes, *i.e.* for increasing numbers of CAN nodes. We use topology TS-1K and scale the CAN size by adding CAN nodes to the stub (leaf) nodes in the underlying topology. The delay of the link from the end-host node to the stub node is set to 1ms. Thus, in scaling the CAN size from 512 to 16K nodes, we're scaling the density of the graph without scaling the backbone (transit) domain. Figure 7 compares the latency stretch for randomly constructed CANs (where nodes join the CAN at a random point in the space, as described in [4]) to the stretch using the binning-based CAN construction scheme outline above. We see that binning-based construction greatly lowers the stretch. Also, as expected, with more landmarks, the binning is more accurate and the stretch decreases further.[11]

Figure 8 repeats the above test for topology PLRG2. As before, the CAN size is scaled by scaling the density of CAN nodes attached to underlying topology nodes. Again, we are not sure to what extent the random assignment of link delays affects our results for PLRGs. Table I lists the stretch for a 100 node CAN using the NLANR data set. Because the NLANR data set has only 103 nodes, we cannot experiment with increasing CAN sizes as we did for TS and PLRG.

[10] We wish to stress that this adjustment actually makes our results look worse.

[11] While the absolute value of the stretch appears high, this is primarily because we are using a CAN with only two dimensions. Increasing the dimensionality of the CAN space greatly reduces the stretch for all construction schemes. In [4], we also make use of a number of heuristic techniques to further lower the latency. These techniques however, are CAN-specific and not relevant to this paper. We thus do not make use of them in this paper so that we can more clearly expose the performance gains caused by binning alone.

B. Topologically-aware construction of unstructured overlays

The previous discussion applied to the class of "structured" overlays (such as CAN, Chord, Pastry, Tapestry). However, many deployed overlays (such as Gnutella, FreeNet) are much less structured. We now ask whether our topological hints could be of use in the construction of these unstructured overlays. To study this question we chose to not focus on any one particular existing overlay problem and instead consider the following more general one:

Given a set of n nodes on the Internet, have each node picks any k neighbor nodes from this set, so that the average routing latency on the resultant overlay is low (assuming shortest path routing).

Even under the assumption of global knowledge of the IP-level latencies between every possible pair of nodes (*i.e.* the n^2 distance matrix), the problem of constructing an optimal overlay is known to be NP-hard [29], [28]. Because we do not have an optimal construction algorithm, we experimented with a number of different heuristic algorithms and found one that appears to consistently perform well. Our heuristic algorithm works as follows: a node picks its k neighbors by picking the $k/2$ nodes in the system closest to itself (we call these connections *short links*) and then picks another $k/2$ nodes at random (we call these *long links*). Our intuition in devising the above algorithm was that the $k/2$ connections to closeby nodes will result in well-connected pockets of nearby nodes, while the random links serve to keep the graph connected and to interconnect these different pockets of nodes. Shortest path routing on the resultant overlay might then involve a series of short hops from the source node to a close-by node which has a long link to a node in the vicinity of the destination node and from there again taking a series of short hops to the destination node. We call this algorithm *Short-Long*.

Short-Long does not scale because a node would need global knowledge of all other nodes in the system in order to pick the $k/2$ closest to itself. We now ask how we can use our binning technique to make this Short-Long construction more scalable, but still retain its excellent performance properties. We do so

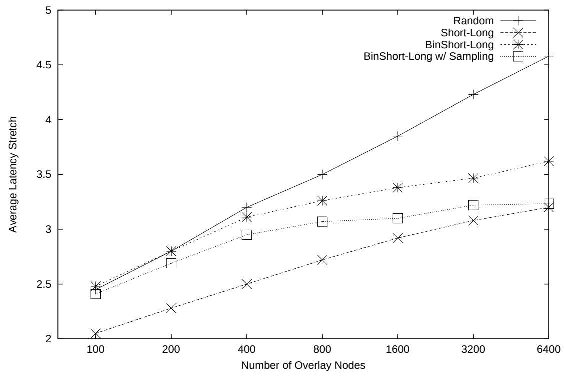

Fig. 9. *Unstructured Overlays; TS-10K; #levels=1; #landmarks=12*

by using our binning scheme to to approximate picking the $k/2$ closest nodes. Now, a node picks $k/2$ neighbors at random from its own bin and, as before, picks the remaining $k/2$ at random. We call this binning-based approximation algorithm *BinShort-Long*. If a node's bin is not large enough for it to pick $k/2$ neighbors, it picks the required number of neighbors from bins that are most similar to its own bin where the degree of similarity beteen two bins is the number of positions in their landmark orderings on which they match. Figure 9 plots the average latency stretch [12] for increasing system sizes for topology TS-10K. We compare Short-Long, BinShort-Long and randomly constructed overlays where each node picks k neighbors at random. Note the difference in scaling behavior between Random and Short-Long. We also see that BinShort-Long follows the scaling behavior of Short-Long. Thus our binning-based algorithm tracks the scaling behavior of the global-knowledge heuristic algorithm. Although the scaling behavior of Short-Long and BinShort-Long is similar, there appears to be a consistent performance gap between the two. To better understand this performance gap, we experimented with the following variant of BinShort-Long: rather than pick $k/2$ nodes at random from its bin, a node measures its RTT to a sample set of nodes in its own bin and picks the $k/2$ closest of the sampled nodes. Figure 9 shows once such case (labelled "BinShort-Long w/ sampling"). We see that with sampling, the performance gap between Short-Long and BinShort-Long decreases indicating that for a given node, the nodes in its own bin are indeed a good approximation of the nodes closest to it in the entire system.

Our aim in going through the discussion in this section is not to claim that either Short-Long or BinShort-Long is the ideal algorithm for constructing unstructured overlays. Rather, our point was to demonstrate that if someone were to develop a good overlay construction algorithm that required global relative proximity information, then binning (and probably any other similar, scalable, proximity inference scheme) offers a scalable and accurate way of approximating this global information.

IV. Topologically-Aware Server Selection

The replication of content over the Internet gives rise to the problem of server selection, *i.e.* from which of the multiple servers holding a given data object should a client attempt a retrieval ?

While many parameters might be used to select a good server, the most frequently mentioned parameters are server load and distance (*i.e.* network latency) from the client. In this paper, we focus only on the distance parameter and define a good server to be one that is close to the client. In this section, we describe a scheme for server selection based on distributed binning.

Given the client and server bins, the server selection process works as follows:

- If there exist one or more servers within the same bin as the client, then the client is redirected to a random server from its own bin.
- If no server exists within the same bin as the client, then an existing server is selected at random from the set of servers whose bin is most similar to the client's bin. We define the degree of similarity between two bins to be the number of positions in their landmark orderings on which they match.

In practice, the above server selection might be implemented by having the client include its bin information in a DNS query. DNS name servers could maintain the bin information for servers holding their content (for example, CNN's name server might maintain the bin information for Web servers holding CNN content). Name servers might then use the above scheme to select a server for the requesting client.

We compare the performance of our binning-based server selection to 3 schemes:

- Random: A client selects one of all available servers at random.
- Selection using the Hotz metric: Hotz [30], [31], like our binning scheme, uses RTT measurements from a node to a set of well known landmarks to estimate inter-node distances. The Hotz metric is computed as follows: Let ${d_a}^i$ represent the distance from a node A to landmark I. Then for any two nodes A and B and Landmark I, the distance between A and B is bounded below by $\left|{d_a}^i - {d_b}^i\right|$ and above by $\left|{d_a}^i + {d_b}^i\right|$ assuming triangle inequality. Extending this to m Landmarks, the distance between nodes A and B is bounded below by MAX($\left|{d_a}^1 - {d_b}^1\right|, \ldots, |{d_a}^m - {d_b}^m|$) and above by MIN($\left|{d_a}^1 + {d_b}^1\right|, \ldots, |{d_a}^m + {d_b}^m|$). Using Hotz's scheme, the distance between A and B is then the average of the lower and upper bounds as computed above. Applying Hotz's metric to server selection, a client selects the server to which its estimated distance is minimum. [13]
- Selection based on Cartesian distance: Here, we simply regard each landmark as defining an axis in a Cartesian space. We thus treat a node's vector of distances to m landmarks as its coordinates in m-dimensional Cartesian space and compute the distance between two nodes A and B as the Cartesian distance between their coordinates. For server selection, a client selects the server to which its estimated distance is minimum.

Note that one advantage in general (not necessarily for server selection) of using bins as a metric rather than the Hotz or Cartesian distance is that the latter metrics require more information.

[12] Calculated as the ratio of the path latency using shortest path routing on the overlay to the path latency on the underlying network topology.

[13] In [30], Hotz defines the metric we describe but does not apply his work to the server selection problem. This application of the Hotz metric is our interpretation of how it might be used for server selection.

Topology	Hotz	Crtsn	Bin	Rand
TS-10K	3.50	2.35	2.40	4.94
TS-1K	2.31	1.35	1.58	5.92
PLRG1	1.6	1.98	1.72	2.51
PLRG2	1.54	2.01	1.81	2.49
NLANR	1.32	1.51	1.39	2.27

TABLE II

Average Stretch

For example, with the latter metrics, for a node to locate close-by nodes (the Short-Long algorithm of Section III required this), it must learn of all available nodes in order to compare its distance to each one in turn. With binning on the other hand, a node need only learn about those nodes in its bin and can then pick one of them. Thus, binning serves as an implicit first-order screening process which makes binning-based schemes somewhat easier to apply scalably.

For server selection, we define the latency stretch from a client node as the ratio of its latency to the server selected by a selection scheme to its latency to the optimal server (*i.e.* the actual closest server). We then compute the average latency stretch as the ratio of the average latency to the selected server by the average latency to the closest server. We evaluate our server selection scheme using the test topologies described in Section II.

Table II compares the different server selection schemes over a range of different topologies. All the tests used 12 landmarks and 3 levels. The number of servers was set to 1000 servers for TS-10K, 100 for TS-1K, PLRG1 and PLRG2 and 10 for NLANR. We see that Cartesian-based selection performs the best for Transit-Stub topologies but not as well on PLRGs while Hotz-distance based selection shows exactly the opposite behavior. Binning-based selection works well across all the topologies. But again, the small performance differences between the different schemes is not particularly interesting. What is interesting however, is that the above results show that *all* the above simple topological hints work quite well. Our binning scheme is one such hint that has the added advantage of being easy to use in a scalable manner.

Figure 10 shows the effect of increasing numbers of landmarks on the performance of binning-based selection. We used 1,000 servers for TS-10k and 100 servers in all other cases. As expected, with increasing landmarks, performance improves but quite quickly offers diminishing returns. We suspect that the slight performance improvement for PLRGs is because most nodes are within a few (2-4) hops away from each other, and so even a randomly selected server is unlikely, in general, to be very far from the client.

Plotting only the average latency stretch hides the individual node performance. In the remaining graphs we plot the distribution of stretch over all nodes. Figure 11 plots the cumulative distribution of the latency stretch for TS-10K – transit-stub topology with 10,000 nodes – using 12 landmarks and 3 levels. Of the 10,000 total nodes, 1,000 nodes are selected at random as servers; the remaining act as clients. As can be seen, all three schemes perform significantly better than random server selection. Cartesian distance and binning-based selection yield better results than the use of Hotz's metric. The absolute value of the latency stretch (on the X-axis) appears high, ranging up-to a factor of twenty. These high values are largely due to our hierarchical link delay distribution (from Section II: 2ms inside Stub domains, 20ms inside Transit stubs and 5ms on Stub-Transit links). For example, consider a client whose closest server is within its own Stub domain. The closest server is then about 2ms away from the client. If the client incorrectly picks a server from across the Transit domain, then its distance to the selected server will be at least 34ms ($2 + 5 + 20 + 5 + 2$ as its travels from the source stub domain, across the transit domain and through the destination stub) yielding a stretch factor of approximately 17 even though a routing value of 34ms is quite good. To account for such inflated stretch values caused by having a server right next to a client, we recompute the stretch values in Figure 11 with the following adjustment: In calculating the latency stretch, we add a constant of 10.0 to both the numerator and denominator latencies. In other words,

$$stretch = \frac{latency - to - selected - server + 10.0}{latency - to - true - closest - server + 10.0}$$

Figure 12 repeats the results from Figure 11 with this adjustment. While the relative performance of the different schemes remains largely unchanged, we see that the absolute value of the stretch ranges falls dramatically showing that high stretch ratios are indeed for short latencies.

Figure 13 plots the cumulative distribution of stretch for the NLANR dataset (without the above adjustment since our NLANR trace did not have very short latencies to cause the above inflation problem). Of the 103 nodes in the dataset, we picked 10 servers at random, 12 landmarks at random, and the remaining nodes act as client nodes. The plot indicates that all three selection schemes perform significantly better than random selection. 50-60% of the nodes correctly select the closest server while around 90% of the nodes pick a server that is less than a factor of two away from the actual closest server.

Note that the Cartesian distance based selection has a longer tail than the other two selection schemes. Also, unlike our earlier simulation results, the performance of Hotz-distance based selection is competitive with the other two schemes. A conclusion one might draw from this plot is that we really do not need to work very hard to achieve good server selection. Hence in designing such topology inference systems one might argue that the simplicity, scalability and practicality of the system should be as important goals as prediction accuracy.

V. Related Work

The IDMaps [16], GNP [17] and WNMS [18] projects all describe architectures for a global distance estimation service. In constrast, our primary focus is on the applications and not on the infrastructure. We explore the problem of designing applications to be topologically aware while making minimal assumptions about any measurement infrastructure. As such, we view the above work as complementary to our own because any additional aid from the infrastructure, for example, in the form of more accurate topological information, can only improve our results. Compared to IDMaps, GNP and WNMS, binning requires less support from the infrastructure. The Geo-Ping algorithm in

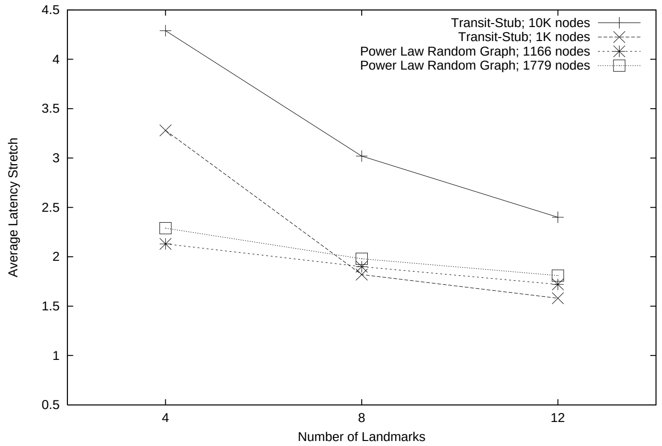

Fig. 10. *Effect of Number of Landmarks and Topology*

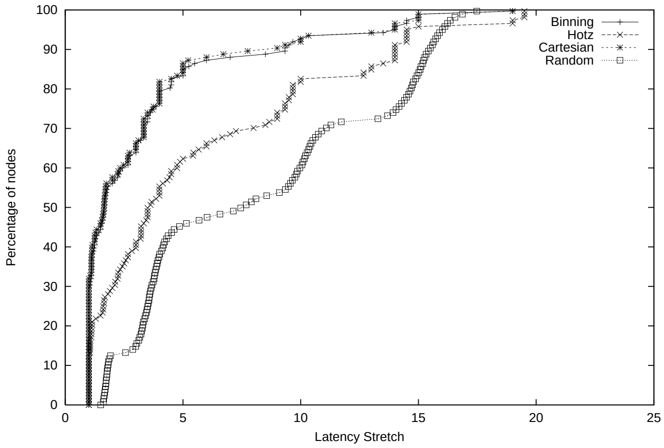

Fig. 11. *Cumulative distribution of Latency Stretch*

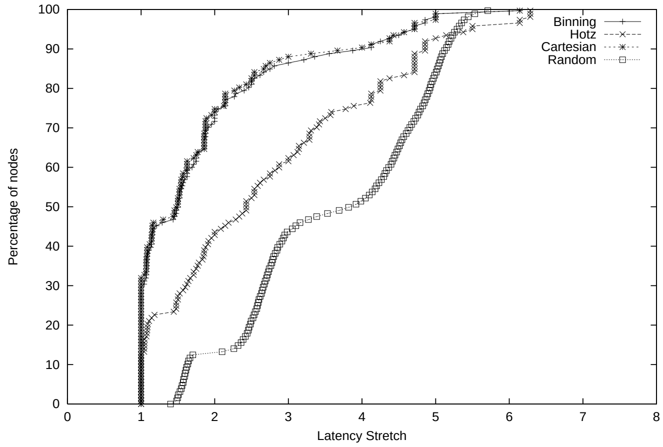

Fig. 12. *CDF of Latency Stretch adjusted to reduce inflated stretch values*

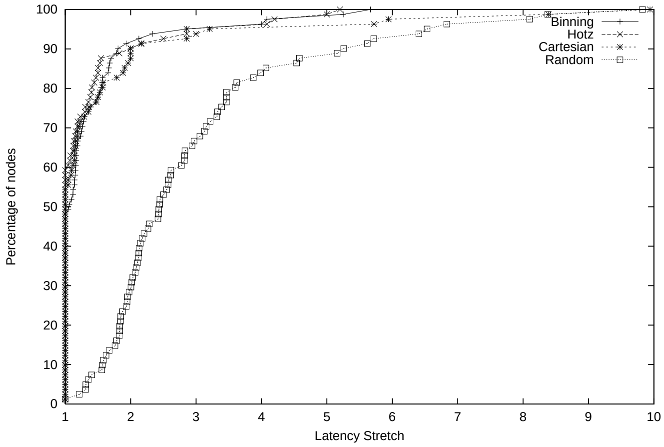

Fig. 13. *CDF of latency stretch for NLANR data*

[32] uses latency measurements to a set of well known landmarks to determine the *geographic* locality of Internet hosts. To determine geographic closeness between pairs of hosts, GeoPing uses a distance metric similar to the Cartesian metric described in Section IV. The authors in [14] describe a centralized clustering engine that uses BGP routing table dumps. For the reasons described in Section I we opted for a decentralized solution based only on end-to-end latency measurements.

In the context of application-level multicast, the authors in [28], [15], [27] propose heuristics for nodes in a multicast overlay to gradually improve the overlay structure to better map onto topology. Both schemes basically have nodes periodically probe other nodes to evaluate the usefulness of switching their neighbors in the overlay. In comparison, our binning scheme for topology awareness operates on short timescales and is, we believe, more lightweight.

Our initial work on CANs [4] explores a number of heuristics to lower the latency of CAN routing. These schemes however try to improve the selection of paths on an existing overlay. Our work in this paper, by contrast, tries to improve the structure of the overlay itself.

The authors in [33] propose a server selection technique where clients periodically perform measurements to all available mirrors. Binning-based server selection instead requires a one-time measurement (with possible periodic refreshes) by the client to a small, fixed set of landmarks. In [24], the accuracy of distance estimation in IDMaps is evaluated in the context of server selection. While the significant differences in the simulation environment make it is hard to draw any direct comparison, our server selection results appear at least comparable. The authors in [34] measure the the performance of certain commercial server selection schemes. Their results show that neither of the measured schemes achieve optimal (or even close to optimal) server selection but do improve significantly on random server selection. In view of the above, we find our server selection results very encouraging.

VI. Conclusion

In this paper, we describe a simple, scalable, binning scheme that can be used to infer network proximity information. We apply this scheme to the problem of topologically-aware overlay construction and server selection. Our results indicate that even rather coarse-grained topological information can significantly improve application performance. The behavior of all the schemes we tested is dependent on the nature of the underlying network topology. Interestingly, our results using actual Internet traces better match simulations using PLRGs than the hierarchical TS topologies possibly providing application-level corroboration to previous work showing that the network-level topology of the Internet is well modeled by a PLRG [21], [20]. While our results indicate that a small number of landmarks yield significant improvements which however, levels off quite quickly. Similar observations have been made by the authors in [24], [17]. An open problem would be to understand just how much further improvement in performance might be possible using sophisticated topology information.

Acknowledgments

We would like to thank Ion Stoica and Eugene Ng for useful discussions, Brad Karp for the DNS root name server statistics, Yan Chen and Morley Mao for help with the NLANR data and Hongsuda Tangmunarunkit for providing us with the Power-Law Random Graphs.

References

[1] Gnutella, "http://gnutella.wego.com," .

[2] FreeNet, "http://freenet.sourceforge.net," .

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of SIGCOMM 2001*, Aug. 2001.

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

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," available at http://research.microsoft.com/ antr/PAST/, 2001.

[6] J. Kubiatowicz et al., "Oceanstore: An Architecture for Global-scale Persistent Storage," in *Proceedings of ASPLOS 2000*, Cambridge, Massachusetts, Nov. 2000.

[7] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "The case for resilient overlay networks," in *HotOS VIII*, Oberbayern, May 2001.

[8] Infrasearch, "http://www.infrasearch.com," .

[9] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *SIGMETRICS 2000*, CA, June 2000.

[10] Inktomi, "http://www.inktomi.com," .

[11] Akamai, "http://www.akamai.com," .

[12] Cisco CDN, "http://www.cisco.com/go/cdn," .

[13] Napster, "http://www.napster.com," .

[14] Balachander Krishnamurthy and Jia Wang, "On network-aware clustering of web clients," in *Proceedings of SIGCOMM '00*, Stockholm, Sweden, Aug. 2000.

[15] Paul Francis, "Yoid: Extending the internet multicast architecture," Unpublished paper, available at http://www.aciri.org/yoid/docs/index.html, Apr. 2000.

[16] P. Francis et al., "An Architecture for a Global Internet Host Distance Estimation Service," in *Proceedings IEEE Infocom '99*, New York, NY, Mar. 1999.

[17] E.Ng and H.Zhang, "Predicting Internet network distance with coordiantes-based approaches," Under submission, 2001.

[18] Yan Chen and Randy Katz, "On the placement of network monitoring sites," http://www.cs.berkeley.edu/ yanchen/wnms/, 2001.

[19] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork," in *Proceedings IEEE Infocom '96*, CA, May 1996.

[20] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On Power-law Relationships of the Internet Topology," in *Proceedings of SIGCOMM '99*, Cambridge, MA, Sept. 1999, ACM.

[21] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topologies, power laws and hierarchy," Tech. Rep. TR01-746, Technical Report, University of Southern California, 2001.

[22] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," in *32nd Annual ACM Symposium on Theory of Computing*. ACM, 2000.

[23] T. Hansen, J. Otero, T. McGregor, and H. Braun, "Active measurement data analysis techniques," htpp://amp.nlanr.net.

[24] Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang, "On the placement of internet instrumentation," in *Proceedings IEEE Infocom '00*, Tel Aviv, Israel, Mar. 2000.

[25] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," UCB Technical Report, 2001.

[26] Antony Rowstron and Peter Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," in *Proceedings of the Eighteenth SOSP*. ACM, 2001.

[27] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay multicast architecture," in *Proceedings of SIGCOMM 2001*, Aug. 2001.

[28] Y. Chawathe, S. McCanne, and E. Brewer, "An architecture for internet content distribution as an infrastructure service," available at http://www.cs.berkeley.edu/ yatin/papers, 2000.

[29] M.R.Garrey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H.Freeman and Company, San Francisco, CA, 1979.

[30] S. Hotz, "Routing information organization to support scalable routing with heterogeneous path requirements," Tech. Rep. PhD thesis (draft), University of Southern California, 1994.

[31] James Guyton and Michael Schwartz, "Locating nearby copies of replicated internet servers," in *Proceedings of SIGCOMM '95*, Boston, MA, Sept. 1995, ACM.

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

[33] Z. Fei, S.Bhattacharjee, E. Zegura, and M. Ammar, "A novel server selection technique for improving the response time of a replicated service," in *Proceedings IEEE Infocom '98*, San Francisco, CA, Mar. 1998.

[34] K. Johnson, J. Carr, M. Day, and F. Kaashoek, "The measured performance of content distribution networks," in *Proceedings of the Fifth WCW*, Lisbon, Portugal, May 2000.