USENIX Association

Proceedings of USITS '03: 4th USENIX Symposium on Internet Technologies and Systems

Seattle, WA, USA
March 26–28, 2003

A Study of the Performance Potential of DHT-based Overlays

Sushant Jain Ratul Mahajan David Wetherall

{sushjain,ratul,djw}@cs.washington.edu
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350

Abstract

We use simulation to study whether overlays based on the recent distributed hash tables (DHTs) have the potential to deliver performance comparable to that of overlays based on measurements. Our work is motivated by the use of DHTs for services such as multicast, which is already targeted by measurement-based overlays; there is currently little understanding of how the two approaches compare at scales where both are viable.

We compare three DHT-based overlays (CAN, Chord and Pastry) with two measurement-based overlays (Narada and NICE), as well as power-law random graphs (PLRGs) that represent Gnutella. To enable comparisons, we configure the overlays with the same average out-degree and focus on moderate scale. To gauge potential, we look at current and idealized DHT algorithms. We find that basic versions of DHTs have a latency stretch that is at least twice that of NICE and Narada, but similar performance in terms of bandwidth hotspots. However, DHT performance can be improved considerably with routing heuristics and topology-aware overlay construction, which have the potential to bring DHT performance at par with NICE. We also report on performance of overlays with power-law structure and the impact of hierarchy on performance.

1 Introduction

Overlays have become the preferred vehicle for providing new Internet services, e.g., CDNs [1, 12], application-level multicast [15, 4, 17, 16, 20, 25, 7, 39, 20, 34], and P2P file sharing [18, 13, 10, 22, 11]. As such, overlay construction protocols that provide good levels of performance, scalability, and robustness are of considerable importance. There has been a surge of interest in the area, along with rapid advances that have led to two main approaches to overlay construction: protocols based on measurements, and protocols based on the recently developed distributed hash tables (DHTs).

Measurement-based protocols use estimates of network properties such as latency between overlay nodes to make an informed choice of overlay structure. They were initially motivated by application-level multicast. Examples include Narada [15], RON [2], NICE [4], Kudos [16] and TAG [20]. These algorithms provide good performance via high quality paths. Recent extensions have allowed them to scale up to tens of thousands of nodes by using hierarchy [4, 19].

On the other hand, DHT-based protocols begin with structure in mind, and may fold measurement information into that structure. They map the overlay nodes to a virtual space commonly known as the node identifier space, typically pseudo-randomly to achieve a balanced distribution. The overlay topology, which determines how the nodes connect to each other, is governed mainly by these node identifiers. Examples include CAN [27], Chord [30], Pastry [28] and Tapestry [37]. These algorithms are extremely scalable and were motivated by peer-to-peer (P2P) applications such as distributed file sharing, where millions of nodes may be involved.

While originally developed for different purposes, DHT-based overlays are now being targeted at some of the same applications as measurement-based overlays, most visibly application-level multicast [25, 7, 31, 39]. Here, efficient use of the network is a key concern, more so than for earlier DHT applications such as distributed indexing. Much recent work thus aims to better the performance of DHT-based overlays with improved construction heuristics [5, 36, 38, 24, 32]. However, despite this convergence of purpose and plenitude of work, there is no real understanding of how the two approaches compare.

In this paper, we study the performance of DHT-based overlays at moderate scales (1000s of nodes) where they represent an alternative to measurement-based overlays for multicast and other services. In contrast, most other work on DHTs studies scales up to hundreds of thousands or mil-

lions of nodes. We seek to determine whether DHT-based overlays have the potential to deliver performance that is comparable to measurement-based overlays at our scales. If so, then research on improved heuristics is more likely to be fruitful.

Our approach is to side-step ongoing improvements to DHTs. Rather than report on the many proposed versions of DHT-based overlays that may quickly become outdated, we study both basic and idealized DHT variants that use global knowledge. This allows us to bound the extent to which performance can be improved as better heuristics are discovered. We use simulation to compare CAN, Chord and Pastry with Narada and NICE when run on the same topologies, at the same scale, and with the same metrics. We also report on power-law random graphs (PLRGs) with flooding based routing, representing Gnutella, to provide another point of comparison. To our knowledge, this is the first "apples to apples" comparison of these approaches.

We find that when configured with the same average out-degree, basic versions of CAN, Chord and Pastry have a latency stretch longer than NICE and Narada by a factor of two or more, depending on the scale. Somewhat surprisingly, all these algorithms showed similar performance in terms of bandwidth hotspots. PLRGs performed better than the basic versions of DHTs in terms of latency stretch due to the use of flooding as the routing mechanism, but poorly in terms of bandwidth hotspots due to highly variable node out-degrees and the same use of flooding. We also find that considerable latency performance gain can be achieved in DHTs with better routing heuristics and topology awareness, though it is more difficult to simultaneously achieve both good latency and good bandwidth performance. Together, these techniques have the potential to bring DHT performance at par with NICE, and thus are a promising direction for future research. As others [4, 19, 16] we find that the hierarchy used to help NICE scale does not significantly degrade its performance as compared to Narada.

The paper proceeds as follows. We describe the relevant overlay algorithms in Section 2. Section 3 discusses the metrics of interest when evaluating overlays, and Section 4 describes our experimental methodology. We present our results in Section 5, discuss related work in Section 6, and conclude in Section 7.

2 Background

This section provides an overview of the overlay construction algorithms that we study. An overlay is built by forming virtual links or tunnels between the participating nodes; a tunnel between a pair of nodes usually traverses multiple links in the underlying network. Given a set of nodes, the goal of an overlay construction algorithm is to select the virtual links and to determine how to route over that topology.

2.1 Measurement-based Overlays

Measurement-based overlays are constructed primarily using active measurements of network properties such as latency between overlay nodes. Several algorithms for building these kind of overlays exist, most of which target multicast services [15, 4, 34, 16, 23, 20, 17, 9]. We study Narada [15] and NICE [4], both of which are optimized for latency.

Narada creates a flat i.e no hierarchal overlay topology that minimizes the latency between nodes while keeping a small number of tunnels per node. This is accomplished by choosing an initial set of tunnels, and periodically adding new tunnels and dropping existing ones. The tunnel addition and deletion process is governed by the utility of the tunnel. The utility metric is computed using the reduction in distance to other nodes the tunnels brings about. This computation requires periodic exchange of routing tables between neighbors and every node probing every other node. This poses a barrier to scalability, but provides all nodes with near global knowledge and leads to an optimized overlay.

To address the scalability problem with a flat measurement based overlay, NICE creates a hierarchy of node clusters. At the bottom of hierarchy, nodes are partitioned into clusters of fixed size. Each cluster has a representative node that lies roughly at the topological center of the cluster. It is determined by having nodes in the cluster probe each other. The representative node of a lower-level cluster is a member in the next level of the hierarchy. This process is recursively repeated, yielding a tree topology. The per node network bandwidth required for overlay maintenance is $O(log(n))$, compared to $O(n)$ in Narada, where n is the number of nodes.

For the purpose of our study, Narada provides a baseline for an overlay with high quality paths at small scale. NICE provides an indication of the performance that can be maintained when additional structure is imposed to scale to larger sizes.

2.2 DHT-based Overlays

DHT-based overlays are constructed using algorithms for distributed hash tables (DHTs) [27, 30,

28, 37, 32]. These algorithms were originally developed to provide highly scalable and fully distributed indexing services for peer-to-peer file sharing. They have since been applied to other services such as multicast [25, 31, 7, 39]. We study CAN [27], Chord [30] and Pastry [28].

In CAN, nodes are mapped pseudo-randomly to a virtual d-dimensional Cartesian space, which wraps around at the edges and has no resemblance to the underlying physical topology. Every node has $2 \times d$ neighbors, corresponding to the adjacent nodes in each dimension. Routing is achieved by following a path through the Cartesian space that increasingly progresses from source to destination. The average path length is $O(dn^{1/d})$, where n is the number of nodes in the overlay and the per node network bandwidth required for protocol maintenance is $O(d)$.

In Chord, every node is pseudo-randomly assigned an m bit identifier. For simplicity of explanation, assume $n = 2^m$, where n is the number of nodes in the overlay. Every node is connected to m neighbors (i.e. $log(n)$ neighbors) with identifiers that are spaced at distances of $2^0, 2^1, 2^2 \ldots 2^{m-1}$ from its identifier in identifier space using modulo arithmetic. A packet from node n_s to node n_d is forwarded to the neighbor of n_s that is arithmetically closest to n_d but less than or equal to n_d. The above process ensures that route between any two nodes has at most $log(n)$ hops. The per node network bandwidth required for protocol maintenance for Chord is $O(log(n))$.

In Pastry, nodes are pseudo-randomly mapped to an m-bit identifier space in base 2^b. The routing table of a Pastry node is a matrix with m/b rows, and 2^b columns. The node in cell (r, c) shares the first r digits with the local node and has the last digit equal to c. Routing is accomplished by each node forwarding a message for key k to the node in its routing table with the longest matching prefix; ties are broken using arithmetic closeness to k. Both the average number of hops required to route a message and per node bandwidth required for overlay maintenance in Pastry is $O(log(n))$.

What we have summarized above are the basic versions of CAN, Chord and Pastry. Definitive descriptions are provided in the papers that we reference. Several modifications have also been proposed [27, 30, 24, 11, 5]. Some of these target performance, while others target application-specific aspects such as data availability, resiliency and hotspot management. Since the primary focus of our work is performance, we study only the performance enhancing heuristics, in Section 4.1, and ignore the rest.

For the purpose of our study, CAN, Chord and Pastry represent the different, major, families of algorithms, all of which are being actively developed [25, 24, 31, 11]. Tapestry [37] is similar to Pastry, and we believe our results are relevant for Tapestry as well.

2.3 Random Power-Law Overlays

We study one more class of overlays to provide another point of comparison: power-law random graphs (PLRGs). It has been observed that the topology of "naturally emerging overlays" such as Gnutella, which are formed when users create links to other nodes, are characterized by a node out-degree distribution that obeys a power law. Routing in such overlays is accomplished through flooding over all links. These overlays are attractive for their simplicity and high tolerance to random failures [29].

3 Overlay Performance Metrics

In this section, we describe the metrics used in our study. Choosing appropriate metrics for comparing overlays is not straightforward because performance depends on the applications. However, the latency overhead of an overlay and its use of network bandwidth is always a concern. For this reason two criteria have been widely used in overlay evaluation: relative delay penalty (RDP) and link stress [15, 4, 25, 31, 17, 5, 7]. We use both of these, along with a third, *Load Balance Ratio*, that measures the distribution of the routing responsibility in the overlay.

Ideally, an overlay would deliver latencies and bandwidths between nodes that match those of the underlying network. However, there is a penalty for routing between overlay nodes. If there are multiple virtual links between two overlay nodes, the path between them through the overlay will be longer than the path through the underlying network. If several virtual links pass over an underlying physical link, the link will experience higher load and in case of multicast the same message may travel over it multiple times. This latency overhead is measured using RDP, and the bandwidth penalty using link stress.

3.1 Relative Delay Penalty

Relative delay penalty (RDP) is a measure of the additional packet delay introduced by the overlay on the delivery of a message between two nodes. It is defined as the ratio of the latency experienced when sending data using the overlay to the latency experienced when sending data directly using the underlying network.

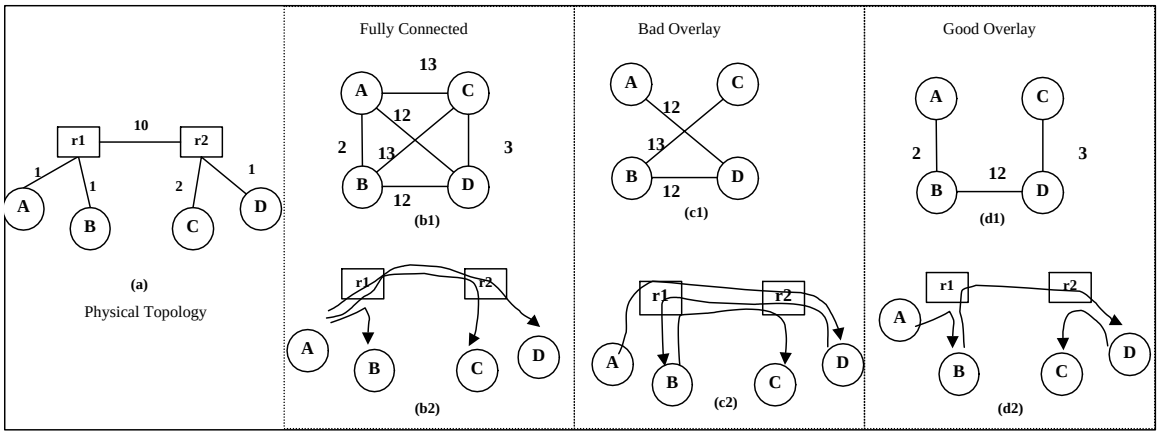

Figure 1: Example overlay topologies. The physical network is shown in the first pane followed by three possible overlays. The logical topology in shown on the top, and the paths taken when A sends a message to B, C and D is shown on the bottom.

As defined, RDP is measured between a pair of nodes and thus provides a set of values for the overlay. To gauge the level of performance of the entire overlay, we use the 90th percentile from the RDP distribution. Various authors have used the same criteria for several reasons [15, 31, 25]. First, the 90th percentile serves to bound the delay multiplier that will be seen by most nodes in practice, whereas the average by itself does not convey any sense of the variation. Second, choosing the 90th percentile rather than the worst case RDP hides sensitivity to simulation parameters. Finally, RDP can easily be very high when the physical latency is small, and using the 90th percentile instead of the worst case RDP filters out these outliers [15].

3.2 Link Stress

Link stress is defined as the number of tunnels that send traffic over a physical link. Links with high stress are potential bandwidth hotspots in the system. In case of unicast, it is a measure of load experienced by the link; more tunnels means higher load. In case of multicast, it becomes a measure of excess bandwidth consumption induced by the overlay; more tunnels means more duplicate messages. For multicast, stress is a function of both the topology and the multicast tree. Two approaches to implement multicast exist – flooding [25] and tree-based [31, 7, 39]. For DHTs and measurement-based overlays, we consider only the latter as it is believed to be more efficient [8]. Here the multicast tree is the union of the unicast routes from the source to all destinations. For PLRGs we use flooding, the default mechanism.

Stress as defined above is a distribution over all physical links for each multicast tree, i.e, per source. To gauge the level of performance of the entire overlay, we use the 90th percentile of the distribution given by the worst stress for each multicast tree. Using worst case distribution conveys a bound on the stress seen by any link, and using the 90th percentile reduces the sensitivity to simulation parameters as before.

An ideal overlay should have both low RDP and low stress. Unfortunately, it may be difficult to simultaneously achieve both objectives. For instance, consider the physical network and three possible overlays topologies shown in Figure 1. Figure 1(b1) shows a fully connected overlay topology, in which the RDP between all pairs is 1, but the stress on links close to the end hosts is high. To communicate with B, C, and D simultaneously, A must send three packets over its physical access link, leading to a stress of 3. Next consider the overlay topology in Figure 1(c1), which has low stress on links close to the end hosts. In this case, all overlay tunnels go over the high latency physical link joining **r1** and **r2**. This leads to high RDP between most pairs. Figure 1(d) shows a good overlay with both acceptably low RDPs and low stresses.

3.3 Load Balance

In an overlay, nodes also act as routers and forward packets between other nodes. Ideally an overlay would not place a much higher forwarding load on one node compared to other nodes[1]. Otherwise, with increasing workload the overloaded node would soon become a hotspot for the system, leading to

[1]Here we are assuming homogeneous nodes. Although this may not be a realistic assumption in practice, existing protocols are not designed to take into account heterogeneity of members [26]. Any variation in forwarding load is thus unintended.

degraded performance. For example, logical topologies such as a star or a tree are not well balanced compared to a ring topology because they have key nodes that are used for most point-to-point communications.

To measure the distribution of the forwarding load, we define a metric called the *Load Balance Ratio*, which is computed as follows. For each node in the overlay, the routing load is the number of source-destination pairs between which it forwards messages. Load balance ratio is the ratio of the maximum routing load to the median routing load. This measures how much worse the maximally loaded node is compared to the halfway loaded node. This metric is less relevant for multicast where all nodes forward one message for a given source. It is important when overlays are used more generally for multiple point-to-point communications.

For the purpose of our study, load balance ratio exposes how performance-based routing concentrates traffic in non-uniform ways. One of the favorable arguments behind DHT-based approaches is that they are better load balanced because of their regular geometric structure and use of randomization. But heuristics that improve performance can interfere with this. We also note that load balance distinguishes topologies and routing protocols that are only suited for multicast from those that are more widely applicable. For example, protocols such as NICE that use hierarchy are good for multicast but would place extremely high load on nodes high in the tree if used for general unicast communications.

3.4 Other Considerations

There are several other performance measures that we do not explore in this paper. We do not measure the overhead of the overlay protocols, either in terms of the amount of state or traffic that is needed to maintain the overlay. It is well known that pure DHT algorithms make only local measurements and scale extremely well, while schemes such as Narada perform global measurements and scale relatively poorly. Our emphasis instead is on the performance levels that can be achieved at a given scale for which the overhead of the algorithms under study has been deemed acceptable. For example, NICE is able to scale to 1000s of nodes (because its overhead is logarithmic with overlay size), while Narada is not. Thus, for overlays this large, NICE is our only option among measurement-based overlays for comparison with DHTs.

We also do not measure protocol dynamics, such as maintaining connectivity in face of failures. Overlays are expected to operate with members leaving dynamically, and different overlay construction algorithms may be disrupted in different ways; in the extreme, partitions are possible. While these dynamic properties are important, our focus is to first understand the static performance potentials of the different algorithms.

4 Methodology

In this section, we describe our experimental methodology. We first describe the variations of the DHT-based algorithms that we compare. We then describe our simulation set-up.

Recall that our goal is to understand whether DHT-based overlays will be able to match the level of performance of measurement-based overlays, which are constructed specifically to provide good quality paths. One difficulty is that different heuristics are continually being proposed to enhance the performance of DHT-based overlays [24, 27, 38, 31, 32, 5], and we do not wish our results to quickly become irrelevant by being tied too closely to specific heuristics. Instead, we side-step this race by taking advantage of simulation as a tool to report on the performance of both the current versions and idealized versions that enhance performance by using global knowledge.

4.1 DHT Heuristics

We study performance-enhancing variations to DHTs along two dimensions – those that attempt to find better paths over a given overlay, and those that construct the overlay itself in a topology-aware manner. We describe each in turn.

4.1.1 Routing Heuristics

In the simplest version of DHTs, routing proceeds using only the geometric properties of the overlay algorithm. We refer to this as the *Base* variant. It is possible to improve performance by routing across the overlay in a manner that takes latency into account. The structure of the overlay itself remains unaffected. We study the heuristic specified in [27] for CAN and [11] for Chord. Here, the chosen next hop is the one out of all possible neighbors that results in the maximum progress towards the destination, where progress is defined as the ratio of the distance in identifier space to the physical latency. We refer to these versions of CAN and Chord as the *Proximity* variants. In Pastry, the next hop is unique and therefore it does not have a corresponding proximity routing variant. Comparison between *Base* and *Proximity* shows the value of the current routing heuristics.

Other heuristics have been proposed, e.g., smallest physical latency for CAN as described in [8]. However, rather than simulate all the different heuristics we can find, we stick with the above as a yardstick (since it applies to both CAN and Chord and is the subject of most published results) and define a new variant intended to provide an upper bound on how well any future heuristic can perform. This variant, *Shortest-Path,* comes from the observation that the maximum gain achievable by any routing heuristic is that of shortest path routing (implemented using a distance vector or link state algorithm). This algorithm requires global knowledge, unlike proposed heuristics, and so may not be a good choice at large scales. However, it can readily be computed in our simulation setting to provide a bound on the performance of better heuristics that will inevitably be proposed. That is, comparison between *Proximity* and *Shortest-Path* shows how much room routing heuristics have for improvement. Another advantage is that the *Shortest-Path* variant is independent of the actual overlay algorithm and therefore it applies to all three DHTs.

4.1.2 Topology Awareness

In the simplest version of DHTs, both the mapping of nodes to identifiers and the choice of tunnels (when multiple choices are possible as in Pastry) is pseudo-random to provide a well-balanced structure. This overlay construction process, which we refer to as the *Random* variant, is unaware of the underlying topology.

In a topology aware overlay, heuristics are used to create an overlay that reflects the underlying network topology. The intuition is that if the overlay and the underlying network topology closely mirror one another, then the overlay paths will closely follow network paths and good performance will result. We now describe how to achieve topology awareness in Pastry, CAN and Chord.

In Pastry topology awareness can be achieved through neighbor selection [5]. A routing table entry can be filled by any node that matches the criteria for that cell. Topology aware construction chooses the closest such node. Achieving perfect awareness using this method requires global knowledge, though reasonable approximations that rely only on limited information exchange are possible [5]. In this paper, we only consider the global knowledge approach.

The above neighbor-selection approach is not applicable to CAN and Chord because in both these algorithms topology is completely defined once the identifier assignment is done. Instead topology awareness is achieved through intelligent identifier assignment. For instance, identifier assignment based on proximity to landmarks has been proposed as a heuristic [24] to achieve a topology aware mapping in CAN.

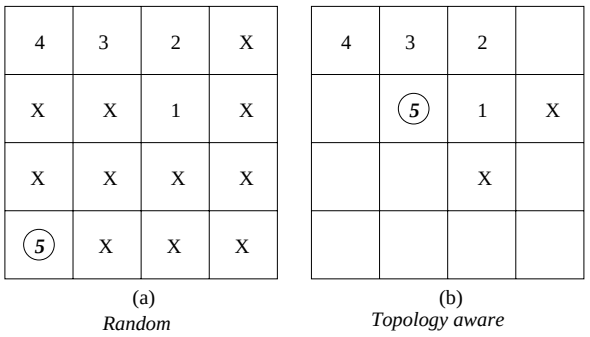

Figure 2: Topology awareness in a 2-d CAN. A new node 5, which is physically closest to 1, joins the overlay. *X* denotes the possible assignments for 5 in case of *Random* and *Topology-Aware* overlay construction. The *Topology-Aware* case yields an assignment in which 5 is geometrically close to 1.

However, it is generally the case that heuristics for topology-aware identifier assignment are not as well defined nor studied as heuristics for improving routing. To understand how much topology awareness itself can improve performance, assuming that good heuristics will be found, we would like to use an analogue to our *Shortest-Path* variation above, rather than simulate many possible heuristics. However no such globally optimal assignment has been defined for either CAN or Chord. So, we use a greedy assignment based on global knowledge to construct a good assignment. Our greedy assignment[2] rule works as follows: the identifier of a new node joining the overlay is chosen so that it is a neighbor of the node that is closest to it in the underlying network. That is, the rule makes the overlay neighbors of a node similar to the neighbors of the node in the underlying topology. Figure 2 illustrates the concept for a 2-dimensional CAN. This also requires global knowledge (or at least a knowledge of distances to all nodes in the vicinity) and so may not be a good implementation choice at large scales. Interestingly, a similar heuristic has been proposed in parallel with our work [32].

4.2 Simulation Setup

Since our primary interest is in understanding performance under static environments, we simulate

[2]We believe that our approach based on global knowledge would yield a better mapping than using a fixed number of landmarks. The performance of the latter is also sensitive to the choice and number of landmarks.

overlay construction using centralized algorithms. The only exception is Narada, which we simulated using an event-driven simulator because Narada does dynamic evaluations to improve the overlay over time. To generate PLRGs overlays, we first generated power-law topologies using Brite [21], and then randomly mapped the nodes in this graph to the nodes in the overlay. For DHTs, pseudo-random hash functions are used to provide a well-balanced structure. However, since we are using simulation we created well-balanced structures by uniformly partitioning the space directly. This is a conservative simplification that is consistent with our goal of presenting the DHT-based overlays in their best light.

An important parameter that affects the performance is the average out-degree (the number of neighbors of a node). It determines the number of links in the overlay and therefore directly affects performance. For example, an overlay with more links will have lower RDPs because of shorter paths but higher stress because more overlay links traverse a physical link. We study the effect of out-degree in Section 5.4. The average out-degree is a configurable parameter in all overlays except Chord. In Chord the outdegree is $log_2(overlay\_size)$. In Pastry the average degree depends on b and varies as $blog_{2^b}(overlay\_size)$. We fixed b as 1 to make Pastry's average degree comparable to that of Chord. The values of degree we study range from 4 to 12.

We used the transit-stub model of GT-ITM [35] to generate the physical network topology. GT-ITM also assigns latencies to links in the physical topology. Additional nodes were attached to the stub nodes to represent hosts connected to lower level routers. Overlay nodes were picked randomly from these hosts.

The overlay algorithms that we compare are summarized in Table 1. We study one version of each of Narada [15], NICE [4], and PLRG. Narada and NICE build their topologies using latency measurements as described earlier, and they use shortest path routing using latency as the metric. PLRG forms a random topology where the node outdegree distribution follows a power law, and nodes flood over this topology; this represents Gnutella-like overlays.

For each parameter setting we ran 9 simulations, three different physical network topologies, each with three random seeds. Each topology had 4,040 backbone nodes and 20,200 stub nodes. Simulated overlay sizes were varied from 64 to 4096.

	Topology Awareness	Routing	Degree
Narada	-	*Shortest-Path*	$4-12$
NICE	-	*Shortest-Path*	$4-12$
CAN	*Topology-Aware, Random*	*Shortest-Path, Proximity, Base*	$4-12$
Chord	*Topology-Aware, Random*	*Shortest-Path, Proximity, Base*	–
Pastry	*Topology-Aware, Random*	*Shortest-Path, Base*	–
PLRG	-	*Flooding*	$4-12$

Table 1: Summary of simulated algorithms.

5 Results

We now present the results of comparing different protocols and their variants on each of our three performance metrics. We first consider a comparison using an out-degree of 10 for all overlays except Chord and Pastry. This out-degree was found to be a good representative by experimenting with different out-degrees. The effect of out-degree on performance is described in Section 5.4.

5.1 RDP

This section investigates the latency stretch of various overlays. We start by studying the impact of heuristics in DHT-based overlays, and then compare all the overlays.

5.1.1 Effect of Heuristics

Figures 3 and 5 show the impact of routing heuristics in CAN on top of an overlay with *Random* and with *Topology-Aware* overlay construction, respectively. They plot the 90th percentile RDP for the *Base* version of CAN, CAN with *Proximity* routing, and CAN with *Shortest-Path* routing (the best possible performance for a given overlay). Similar results were obtained using 50th percentile and 95th percentile RDP and are not shown. Figures 4 and 6 show the same results for Chord. Results from all 9 simulations are presented to show the variance and the line is drawn through the average. This representation is used in all plots unless otherwise specified.

For both CAN and Chord, improved routing brings about significant improvement in RDP. *Proximity* routing does quite well; it reduces the RDPs to roughly halfway between *Base* and *Shortest-Path*

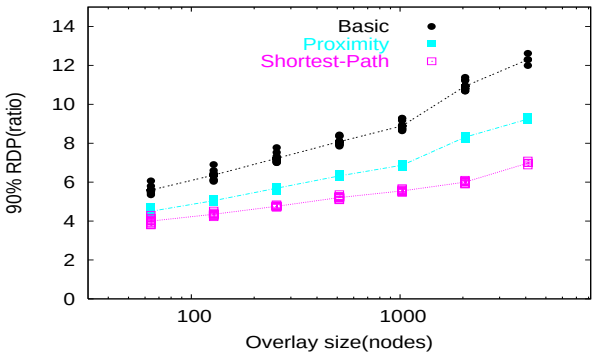

Figure 3: Effect of routing heuristics on RDP in CAN with *Random* overlay construction.

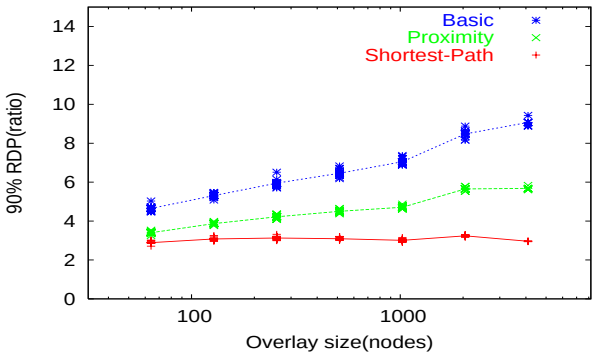

Figure 5: Effect of routing heuristics on RDP in CAN with *Topology-Aware* overlay construction.

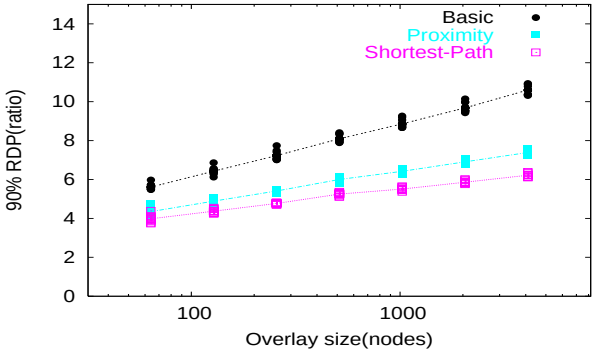

Figure 4: Effect of routing heuristics on RDP in Chord with *Random* overlay construction.

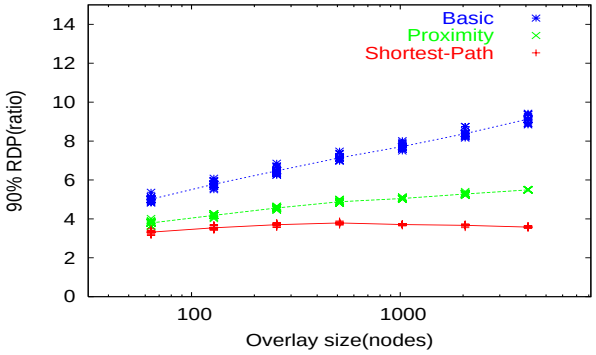

Figure 6: Effect of routing heuristics on RDP in Chord with *Topology-Aware* overlay construction.

Routing	Topology Awareness	
	Random	*Topology-Aware*
Base	**8.89**	7.05
Proximity	6.87	4.71
Shortest-Path	5.55	**3.02**

Table 2: Effect of heuristics on 90th percentile RDP for CAN with 1024 nodes

routing. However, it does not match the performance of *Shortest-Path* routing because while the former is a greedy decision at each hop, the latter computes globally optimal paths. The improvement in Chord using *Proximity* routing is slightly greater because as we increase the overlay size, the number of choices for the next hop increases, leading to better paths. As a result, *Proximity* routing comes closer to *Shortest-Path* routing in Chord than in CAN.

To understand the effect of combining the different heuristics, we tabulate the 90th percentile RDP for CAN with 1024 nodes in Table 2. Observe that the effect of heuristics compose and the best performance is achieved by enabling both *Shortest-Path* routing and *Topology-Aware* overlay construction. This improvement is substantial, from 8.89 to 3.02 (almost a 70% reduction), and indicates the potential for improvement through more practical heuristics. By looking across columns in the table, we deduce that being topology aware by itself brings about a significant performance gain.

Figure 7 shows the effect of both topology awareness and routing heuristics on RDP for Pastry. As before the heuristics lead to substantial improvement.

5.1.2 Comparing All Protocols

We try to answer two questions now:

1. How do DHTs with only scalable heuristics (such as *Proximity* routing) compare to measurement-based overlays, especially NICE since it has similar scalability?

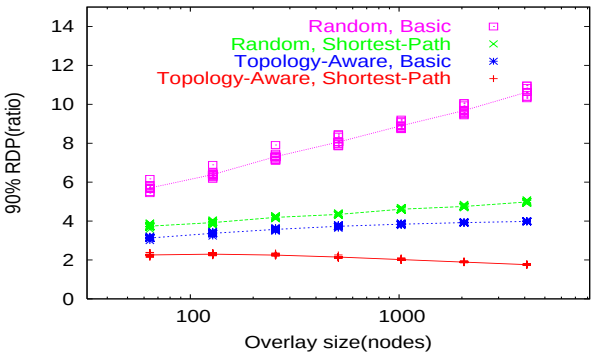

Figure 7: Effect of heuristics on RDP in Pastry.

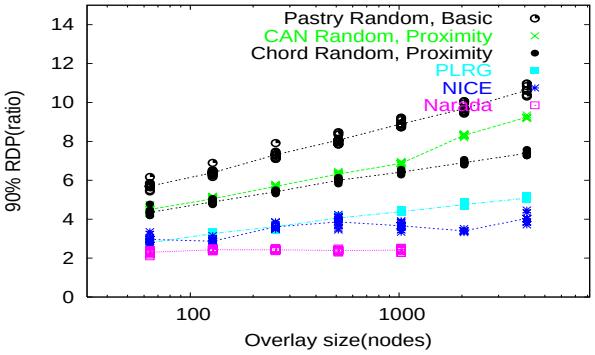

Figure 8: Variation of RDP with size. Simple practical versions of DHTs are shown – CAN and Chord with *Random* overlay construction and *Proximity* routing, and Pastry with *Random* overlay construction and *Base* routing.

2. How do DHTs compare to the other approaches when *Shortest-Path* routing along with *Topology-Aware* overlay construction is implemented?

Figure 8 shows the 90th percentile RDP as a function of overlay size for different protocols[3]. *Proximity* routing is used for CAN and Chord with *Random* overlay construction. For Pastry we show *Random* overlay construction with *Base* routing. These represent simple, efficient, and practical versions of these overlays. A distributed implementation of topology awareness is shown to be a reasonable approximation in [6]. We believe that by the same token, similar implementations are possible for CAN/Chord because a new node being able to find the closest live node is a key assumption in both scenarios. Instead of assuming this to be a fact and for fairness of comparison, we chose to use the *Random* overlay construction for all DHTs.

[3]For Narada, results are not shown for overlay sizes greater than 1024 nodes because simulation times were on the order of days.

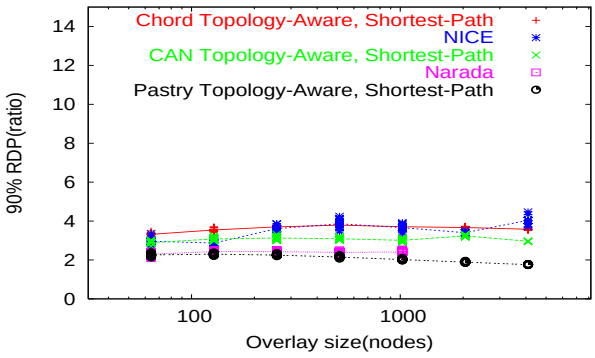

Figure 9: Variation of RDP with size for various overlays. Idealized versions of DHTs – with *Topology-Aware* overlay construction and *Shortest-Path* routing – are shown.

We make the following observations:

- The difference in RDP between NICE and the DHTs is large and grows with the overlay size. It is a factor of two for 1024 nodes.
- Flooding over PLRGs performs well, because with flooding the shortest path between nodes is taken. This also indicates that without heuristics DHT overlays are not better than random overlays from a latency perspective.
- All DHTs have similar RDPs. The slightly worse performance for Pastry stems from the absence of a *Proximity* routing equivalent, and the slightly better performance of Chord for overlay sizes greater than 1024 stems from a higher average out-degree in Chord compared to CAN beyond this size,[4] which leads to shorter paths.
- The performance of NICE does not deteriorate with increasing overlay size even though the levels of hierarchy increase. Further, it is not much worse than that of Narada. Our findings agree with those of the authors of NICE [4].

Figure 9 compares DHTs with *Topology-Aware* overlay construction and *Shortest-Path* routing with all other protocols. This represents the maximum performance one can achieve from a latency perspective for these overlays. There is no real qualitative difference between performance in this case. This is encouraging because it shows that RDPs comparable to measurement-based overlays can be achieved using improved versions of DHT-based overlays.

[4]Average degree for Chord is $log(n)$ – for 1024 nodes average degree is 10 which is the same as the average degree configured for CAN.

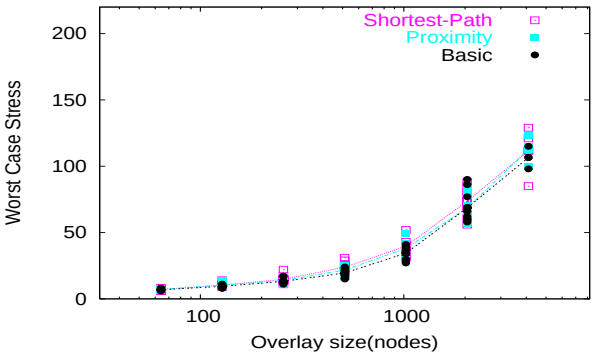

Figure 10: Effect of routing heuristics on link stress in CAN with *Random* overlay construction.

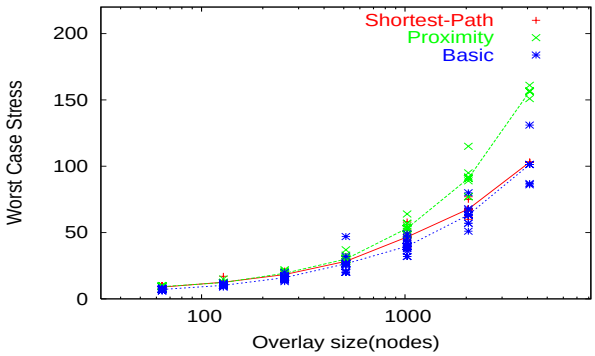

Figure 11: Effect of routing heuristics on link stress in CAN with *Topology-Aware* overlay construction.

5.2 Stress

We now investigate how the various protocols compare on the metric of the worst case stress, and the impact on stress of RDP-enhancing heuristics in DHT-based overlays.

5.2.1 Effect of Heuristics

Figures 10 and 11 show stress as a function of overlay size for CAN with *Random* and *Topology-Aware* overlay construction, respectively. The three lines correspond to the three routing variants – *Base*, *Proximity* and *Shortest-Path*. The results for Chord were similar, and have been omitted due to space constraints.

Somewhat surprisingly, improved routing does not have much negative impact on stress. With improved routing, we expected stress near a few well-placed nodes to go up. This is evident to some degree in the fact that the worst case stress does go up slightly. But at the same time, this increase is reasonably countered because improved routing leads to shorter paths, which means that fewer links are traversed. By comparing the results across *Random* (Figure 10) and *Topology-Aware* (Figure 11), we can also see that topology awareness has little impact on stress.

Figure 12 shows the effect of heuristics on link stress for Pastry. As for CAN, both routing heuristics had no significant impact on stress. However, topology awareness with *Base* routing had much worse (almost factor of 2) stress values. A key difference between the topology aware overlay construction mechanisms of Pastry and CAN/Chord is that in Pastry a few well placed nodes can be the neighbors of many nodes, whereas in CAN/Chord a node can be a neighbor of only a small number of other nodes. This has a direct consequence on link stress and load balancing. Hence, our results should be interpreted as highlighting the difference between the two methods to achieve topology-awareness – choosing node identifier assignment and choosing neighbors.

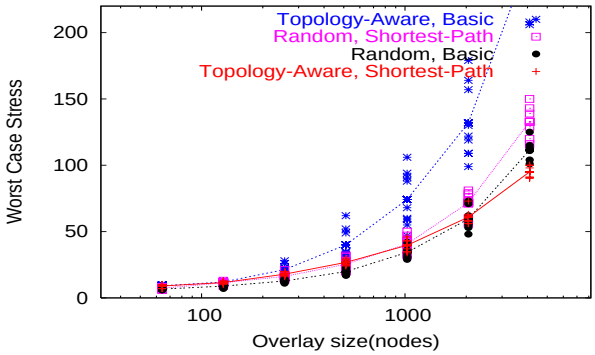

Figure 12: Effect of heuristics on link stress in Pastry.

5.2.2 Comparing All Protocols

Figure 13 shows how stress varies with size for different algorithms. Since the performance of various variants of DHTs was largely similar, we show the stress only for the best version, *Topology-Aware* overlay construction with *Shortest-Path* routing. The striking artifact in the graph is that PLRG has very high stress, more than five times worse than the other protocols for 1024 nodes. This is due to its use of flooding as the routing mechanism, and an uneven out-degree distribution among nodes. All other protocols exhibit similar stress values, although for large overlay sizes (over 2048 nodes) NICE has slightly smaller values. Large group sizes have higher density for a fixed topology size, which has the tendency to increase the stress on the backbone links in DHTs. In this situation, the clustering

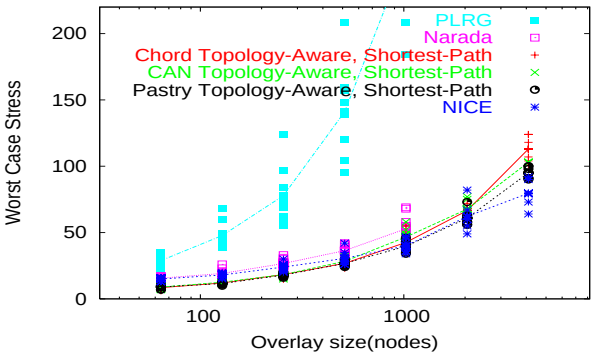

Figure 13: Variation of stress with size for various overlays. Variants of DHT with *Topology-Aware* overlay construction and *Shortest-Path* routing are shown.

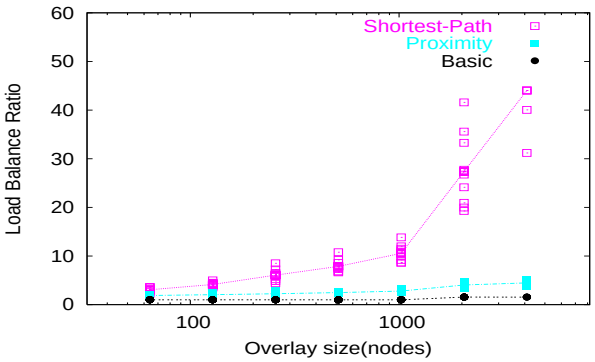

Figure 14: Effect of routing heuristics on load balancing in CAN with *Random* overlay construction.

of close nodes in NICE helps to reduce the stress on the backbone links.

In summary, for similar average out-degrees all protocols exhibited similar worst case stress properties. In case of CAN and Chord heuristics have no significant impact on stress properties. For Pastry, however, *Topology-Awareness* with *Base* routing had significantly higher stress than other variants.

5.3 Load Balancing

We now study the load balancing properties.

5.3.1 Effect of Heuristics

Figure 14 shows the load balancing ratio for different overlay sizes for CAN with *random* overlay construction. As before, the three lines correspond to the three routing variants. With *Base* routing, the load is balanced very evenly, with the ratio between one and two. This is a direct consequence of the regular structure and the random overlay construction

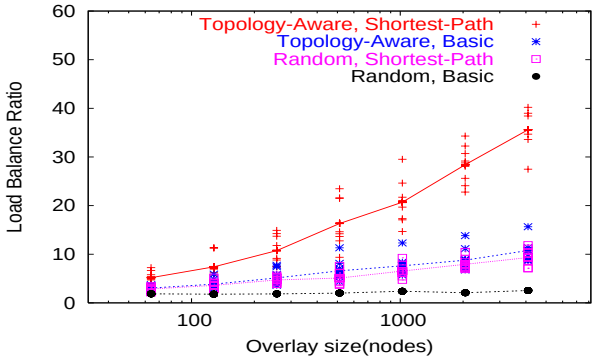

Figure 15: Effect of heuristics on load balancing in Pastry.

in the DHT algorithms. With the *Proximity* routing, load balancing deteriorates slightly but grows very slowly with the overlay size. However, there is a significant degradation with *Shortest-Path* routing because there exist well-placed nodes in the topology that have a low latency to many nodes, and therefore are used very often independent of their node-ID. This raises the issue of the value of a routing heuristic that mimics *Shortest-Path* routing if balanced load is desired.

We observed similar behavior for CAN with *Topology-Aware* overlay construction because being topology aware only changes the relative assignments and does not modify the DHT structure itself. The results for Chord were also similar; we omit them due to space constraints.

Figure 15 shows the load balancing ratio for different heuristics. As in CAN we find significant degradation with *Shortest-Path* routing. We can also see that in the case of Pastry *Topology-Awareness* caused the load balance to deteriorate. This is for same reasons, as for higher link stress, mentioned in Section 5.2.1.

5.3.2 Comparing All Protocols

Figure 16 shows the load balancing ratio for all the protocols. The *Shortest-Path* routing variant for DHTs is shown, as it had the highest load. Note that the scale of y-axis differs from that in Figure 14. We can see that NICE has an extremely high load balancing ratio (two orders of magnitude for overlays bigger than 1024). This is because in the NICE hierarchy, the root of hierarchy is responsible for forwarding all packets whose source and destination lie in different sub-trees. On the other hand, Narada, a measurement-based overlay with no hierarchy, performs similarly to CAN and Chord with *Shortest-Path* routing. PLRG too has a high load balancing

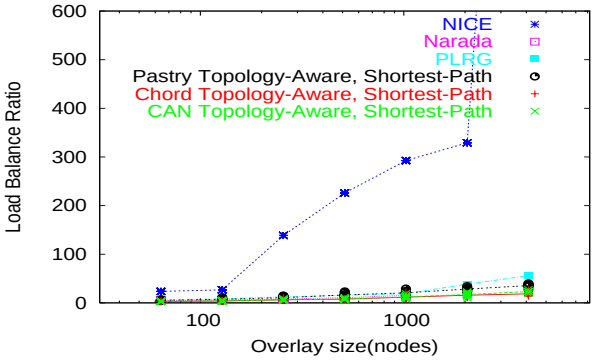

Figure 16: Variation of load balancing ratio with size for various overlays. Variants of DHTs with *Topology-Aware* overlay construction and *Shortest-Path* routing are shown.

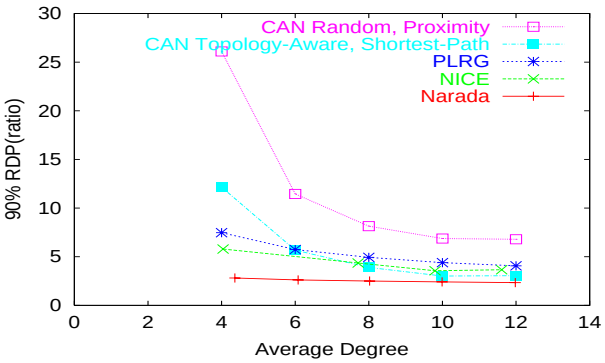

Figure 17: Effect of out-degree in various overlays. Two variants of CAN are shown – *Random* overlay construction with *Proximity* routing and *Topology-Aware* overlay construction with *Shortest-Path* routing.

ratio because of a highly uneven degree distribution. Note that the relatively sharp increase in NICE at some points (256 and 4096) is a quantization effect due to formation of unbalanced trees.

In summary, our results point at the limitations of hierarchy for applications involving general unicast communications. DHT routing heuristics that mimic *Shortest-Path* routing can have significant negative impact on load balancing and thus may not be a suitable choice for some applications. Although *Topology-Aware* overlay construction by itself does not degrade load balancing in CAN/Chord, it degrades it in the case of Pastry.

5.4 Effect of Out-Degree

In this section we study the effect of increasing the average out-degree of nodes in various overlays. Intuitively, increasing out-degree will decrease RDP

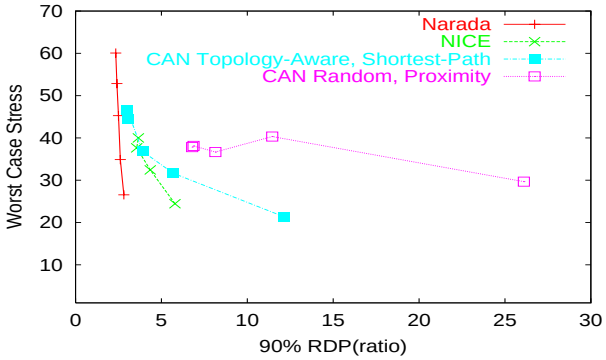

Figure 18: Variation of link stress with RDP. Two variants of CAN, same as those in Figure 17, are shown.

because it reduces the number of overlay hops. But at the same time, stress could increase because the number of overlay links per physical link increases.

Node out-degree is controlled in CAN by varying the number of dimensions. We consider two versions of CAN: *i*) *Proximity* routing with *Random* overlay construction; and *ii*) *Shortest-Path* routing with *Topology-Aware* overlay construction. Narada and PLRG algorithms can be configured with an average out-degree, and the out-degree in NICE is controlled through cluster size. Chord and Pastry are not shown because the out-degree of nodes can not be varied independent of the overlay size.

The results in this section are presented for a fixed overlay size of 1024 nodes and only averages over simulations are plotted. Figure 17 shows the effect of increasing average out-degree on RDP for different protocols. Two observations can be made. First, there is a sharp reduction in RDP for CAN variants as the average degree is increased. For other protocols also, the RDP decreases with increasing degree but the gain is much less. Second, NICE was able to achieve good RDPs with a much lower average out-degree compared to CAN with *Proximity* routing over *Random* overlay construction.

We now explore the trade-off between RDP and link stress. Figure 18 shows the relationship between stress and RDP. It plots the measured stress for a given RDP obtained by varying the node out-degree (Figure 17). The worst case stress for PLRGs was extremely high, and hence has not been shown in this graph. All overlays exhibit the basic tradeoff between RDP and link stress, though to varying degrees. By increasing average out-degree, RDP can be reduced, but that reduction comes at the cost of higher stress. However, note that the CAN

variant with *Proximity* routing over *random* overlay construction lies further away from the origin than NICE. This means that it is harder to simultaneously achieve both low stress and low RDP in this variant of CAN than in NICE. At the same time CAN with *Shortest-Path* routing and *Topology-Aware* overlay construction performs comparably to NICE, which again points favorably towards the potential of optimization in DHT-based approaches.

6 Related Work

Many overlay construction schemes have been proposed, both measurement-based [15, 4, 20, 2, 23, 9, 34, 17] and DHT-based [27, 30, 28, 37]. Much ongoing work aims to improve the performance of DHT-based approaches [5, 36, 38, 24, 33, 31, 11] and the scalability of measurement-based approaches [4, 20], as well as look at how DHT-based approaches can provide multicast and other services [25, 7, 39, 31]. However, there has been very little work on studying how these different overlay algorithms compare to each other. Our work and that of a few other researchers are first steps in this direction.

Castro et.al compare the performance of tree building and flooding on top of CAN and Pastry [8]. They find that flooding has high overhead compared to tree-based approaches. This is consistent with our results, where we found that flooding on top of PLRGs has much higher overhead compared to tree-based protocols. A qualitative comparison of various overlay protocols is provided in [3]. Our focus, however, is on empirical evaluation under identical environments.

Finally, Ratnasamy et. al outline the challenges facing DHT-based overlays [26]. They identify performance with respect to latency as a key open issue. Our work indicates that with good routing heuristics and topology-awareness, DHT-based overlays can match the performance of measurement-based overlays.

7 Conclusions and Future Work

In this paper we studied the performance potential of DHT-based overlays at moderate scale (1000s of nodes) where they represent an alternative to measurement-based overlays for multicast and other services. We used simulation to compare basic and ideal DHT-based overlay protocols with measurement-based protocols, at the same scale, using the same topology, and with the same performance metrics. Specifically, we compared CAN, Chord and Pastry with Narada and NICE, as well as power-law random graphs.

Our key findings are:

• For the same average out-degree, basic versions of DHTs have a latency stretch that is longer than NICE and Narada by a factor of two or more, depending on the size of the overlay. The performance in terms of bandwidth hotspots is similar however.

• Considerable performance gains in latency can be achieved in DHTs with better routing heuristics and with topology-aware overlay construction. These heuristics had no substantial adverse effect on link stress, except when the choice of tunnels was directly sensitive to latency, as in Pastry. However, the heuristics (and especially the routing heuristics) do lead to relatively higher load on some nodes.

• As others [4, 16, 19], we found that the hierarchy in NICE does not significantly degrade performance compared to Narada from a latency perspective. The effect of hierarchy on bandwidth hotspots was minimal too. We also note that power-law random graphs had a latency stretch that is competitive with that of NICE, but performed poorly in terms of bandwidth hotspots.

In summary our findings indicate that DHT-based overlays are a promising direction, with the potential to achieve not only scale but good levels of performance. Better heuristics that come closer to achieving this potential without sacrificing scalability are the subject of other, ongoing research.

There are also directions in which we hope to extend our study. The optimizations we considered in this paper are biased towards latency. We do not yet know if bandwidth heuristics (we are not aware of any at present) can deliver levels of bandwidth that are comparable to that of bandwidth optimizing protocols such as Overcast [17] and the enhanced version of Narada [14]. Performance in dynamic environments is another area worthy of exploration.

Acknowledgements

We would like to thank Gaetano Borriello and Steve Gribble for discussions during early stages of this work.

This work was supported by DARPA grant F30602-98-1-0205.

References

[1] Akamai. http://www.akamai.com.

[2] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *ACM SOSP*, 2001.

[3] S. Banerjee and B. Bhattacharjee. A comparative study of application layer multicast protocols. In *Submitted for review*, 2002.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *ACM SIGCOMM*, 2002.

[5] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in peer-to-peer overlay networks. In *FuDiCo*, 2002.

[6] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in peer-to-peer overlay networks. In *Technical Report MSR-TR-2002-82*, 2002.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralised application-level multicast infrastructure. In *IEEE JSAC*, 2002.

[8] M. Castro, *et al.* An evaluation of scalable application-level multicast built using peer-to-peer overlay networks. In *IEEE INFOCOM*, 2003.

[9] Y. Chawathe, S. McCanne, and E. Brewer. An architecture for internet content distribution as an infrastructure service, Unpublished work, 2000.

[10] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Design Issues in Anonymity and Unobservability*, 2000.

[11] F. Dabek, *et al.* Wide-area cooperative storage with cfs. In *ACM SOSP*, 2001.

[12] Digital Island. http://www.digitalisland.com.

[13] Gnutella. http://www.gnutella.com.

[14] Y. hua Chu, S. G.Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the Internet using an overlay multicast architecture. In *ACM SIGCOMM*, 2001.

[15] Y. hua Chu, S. Rao, and H. Zhang. A case for end system multicast. In *ACM SIGMETRICS*, 2000.

[16] S. Jain, R. Mahajan, D. Wetherall, and G. Borriello. Scalable self organizing overlays. Tech. Rep. UW-CSE 02-06-04, University of Washington, 2002.

[17] J. Jannoti, *et al.* Overcast: Reliable multicasting with an overlay network. In *OSDI*, 2000.

[18] Kazaa. http://www.kazaa.com.

[19] D. Kostic and A. Vahdat. Latency versus cost optimizations in hierarchical overlay networks. Tech. Rep. CS-2001-04, Duke University, 2001.

[20] M. Kwon and S. Fahmy. Topology-aware overlay networks for group communication. In *ACM NOSSDAV*, 2002.

[21] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. Tech. Rep. 2000-004, Boston University, 2000.

[22] Napster. http://www.napster.com.

[23] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *3rd USNIX Symposium on Internet Technologies and Systems (USITS '01)*, pp. 49–60. San Francisco, CA, USA, 2001.

[24] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *INFOCOM*, 2002.

[25] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker. Application-level multicast using content-addressable network. In *NGC*, 2001.

[26] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. In *IPTPS*, 2002.

[27] S. Ratnasamy, *et al.* A scalable content addressable network. In *ACM SIGCOMM*, 2001.

[28] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[29] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *MMCN*, 2002.

[30] I. Stoica, *et al.* Chord: A peer-to-peer lookup service for Internet applications. In *ACM SIGCOMM*, 2001.

[31] I. Stoica, *et al.* Internet indirection infrastructure. In *ACM SIGCOMM*, 2002.

[32] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. In *Proceedings of HotNets-I*, 2002.

[33] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. Research Report RZ-3436, IBM, 2002.

[34] Yoid: Your own Internet distribution. http://www.isi.edu/yoid/.

[35] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *IEEE Infocom*, 1996.

[36] B. Y. Zhao, A. Joseph, and J. Kubiatowicz. Locality-aware mechanisms for large-scale networks. In *FuDiCo*, 2002.

[37] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, UCB, 2001.

[38] B. Y. Zhao, *et al.* Brocade: Landmark routing on overlay networks. In *IPTPS*, 2002.

[39] S. Q. Zhuang, *et al.* Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV*, 2001.