Distributed Hashtable on Pre-structured Overlay Networks

Kai Shen
kshen@cs.rochester.edu
Department of Computer Science, University of Rochester

Technical Report #831

Abstract

Internet overlay services must adapt to the substrate network topology and link properties to achieve high performance. A common overlay structure management layer is desirable for enhancing the architectural modularity of service design and deployment. For instance, new link probing techniques can be incorporated into the common structure layer such that a large number of overlay services can benefit transparently. Additionally, a shared substrate-aware overlay structure can potentially reduce redundant per-service link-selection probing when overlay nodes participate in multiple services. The concept of building services on a common structure management layer fits well with *unstructured services*, those that do not place specific requirements on the overlay connectivity structure (*e.g.*, Gnutella).

Despite the benefits, it is unclear how the distributed hashtable (DHT) service can take advantage of a service-independent structure management layer, considering recently proposed scalable DHT protocols all employ protocol-specific overlay structures. In this paper, we present the design of a self-organizing DHT protocol based on the Landmark Hierarchy. Coupled with a simple low-latency overlay structure management protocol, this approach can support low-latency DHT lookup without any service-specific requirement on the overlay structure. Compared with Chord, a well-known DHT protocol, simulations and experimentation on 51 PlanetLab sites find that the proposed scheme can deliver better lookup performance (reducing the lookup latency by almost half) under the same link density. This benefit is achieved at the cost of less balanced lookup routing overhead. Our evaluation also demonstrates that the balance of key placement and fault tolerance for the proposed scheme are close to those of Chord. However, our approach produces more key reassignments after overlay membership changes, due to its structure-sensitive DHT mapping scheme.

1 Introduction

Internet overlays are successfully bringing large-scale wide-area distributed services to the masses. A key factor to this success is that an overlay service can be quickly constructed and easily upgraded because it only requires engineering at Internet end hosts. However, Internet overlay services may suffer poor performance when they ignore the topology and link properties of the substrate network. Various service-specific techniques have been proposed to adapt to Internet properties by selecting overlay routes with low latency or high bandwidth. Notable examples include the unicast overlay path selection [1, 25] and measurement-based end-system multicast protocols [2, 5, 10, 33].

Under such a context, it is natural to conceive a common layer that maintains overlay connectivity structure, which can greatly ease the design and deployment of overlay services. For instance, more effective link probing techniques or a new partition repair protocol can be incorporated into the common structure layer such that a large number of overlay services can benefit transparently. Another powerful supporting argument for this concept is that a common substrate-aware overlay structure layer can reduce redundant service-specific link-selection probing when overlay nodes participate in multiple services. Early experience on PlanetLab [19], an overlay hosting platform, indicates that link-selection probing can consume significant network resources. A common structure management layer can maintain an overlay connectivity structure with *good* links of low latency or high bandwidth. Upper-level services can use Internet transport services along overlay links selected by this layer, which allows these services to achieve high performance without incurring any additional link probing overhead.

The key for overlay services to take advantage of a common structure management layer is that they must be able to function on pre-structured overlay networks. In other words, these services must not dictate how overlay links are structured in any service-specific way. This requirement fits well with services such as unstructured

peer-to-peer search [7, 14]. This layer can also benefit unicast or multicast overlay path selection services (*e.g.*, RON [1] and Narada [5]). For instance, Narada employs a DVMRP-style multicast routing protocol running on top of a low-latency pre-structured overlay network. It achieves high performance without any additional link-selection probing beyond the structure management layer.

Despite these benefits, it is unclear how a given substrate-aware overlay structure can assist the construction of the distributed hashtable (DHT) service. A DHT is a self-organizing overlay network of hosts that supports the insertion, deletion, and lookup with hash keys. A distributed hashtable can serve as a powerful utility supporting many Internet applications including cooperative mirroring, time-shared storage, and distributed indexes [28]. The heart of a DHT protocol is a distributed lookup scheme that maps each hash key into a deterministic location with well balanced object placement and runtime overhead. Recently proposed scalable DHT protocols such as Chord [28], CAN [20], and Pastry [24] are all *strongly structured*. In other words, they place protocol-specific requirements on overlay connectivity structures. As a result, substrate-aware link-selection enhancements built for these DHT protocols [3, 21, 35, 34] cannot benefit other services.

In this paper, we set out to answer the following questions: *Can a scalable DHT service be built to take advantage of a service-independent structure management layer?* And if so, *can such a service perform competitively compared with strongly structured DHT protocols?* More specifically, we are interested in the DHT performance in the following aspects: the lookup latency, fault tolerance, the balance of key placement, the load balance of runtime query routing overhead, and the amount of key reassignment due to overlay membership changes.

The rest of this paper is organized as follows. Section 2 describes a low-latency structure management layer that our DHT construction can be built on. Section 3 then presents the design of a hierarchical DHT protocol that operates on pre-structured overlays. Section 4 describes our evaluation results based on discrete-event simulations. Section 5 reports our implementation and experimentation on the PlanetLab testbed. Section 6 discusses related work to our research and Section 7 concludes the paper.

2 Substrate-aware Overlay Structure Construction

Our DHT construction is based on our earlier design of a common structure management layer, called *Saxons*, that provides *S*ubstrate-*A*ware *C*onnectivity *S*upport for *O*verlay *N*etwork *S*ervices [27]. The Saxons overlay structure management layer contains six components. The bootstrap process determines how new nodes join the overlay structure. The structure quality maintenance component maintains a high quality overlay mesh while the connectivity support component actively detects and repairs overlay partitions. They run periodically to accommodate dynamic changes in the system. The above Saxons components are all supported by the membership management component that tracks a random subset of overlay members. The structure quality maintenance is further supported by two other components responsible for acquiring performance measurement data for overlay links and finding nearby overlay hosts. Figure 1 illustrates

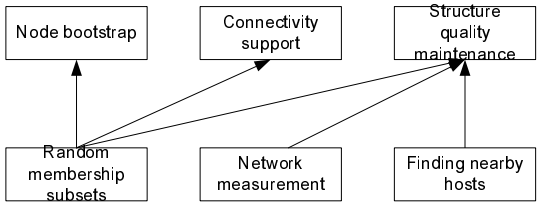

Figure 1: Saxons components.

the six Saxons components and their relationship. Below we briefly describe the low-latency structure quality maintenance that is most related to our DHT construction in this paper. A more complete description of the Saxons structure management layer can be found in [27].

A critical problem for low-latency overlay services is how to acquire latency performance data to support efficient overlay service construction. Measurement accuracy, scalability, and ease of deployment are some important issues for such network measurement. Previous studies have proposed various techniques for estimating Internet host distances [6, 16] or locating nearby hosts [8, 21]. In principle, Saxons can utilize any of the existing techniques for latency estimation. In particular, we point out a *landmark-based Cartesian distance* approach for its simplicity. This approach requires a set of l well-known landmark hosts spread across the network and each landmark defines an axis in an l-dimensional Cartesian space. Each group member measures its latencies to these landmarks and the l-element latency vector represents its coordinates in the Cartesian space. For nearby host selection, a node chooses the one to which its Cartesian distance is minimum. This approach has been shown to be competitive to a number of other landmark-based schemes [21].

A main functionality of the Saxons structure management layer is to maintain a high-quality overlay structure connecting member nodes. The protocol runs at a certain link density, specified by a node degree range $<d_a - d_t>$. Each node makes d_a active overlay links and each node can also accept a number of passive links as long as the total degree does not exceed d_t. The degree upper-bound is maintained to control the stress on each node's physical access link and limit the impact of a node failure on

the Saxons structure. Note that the average node degree is $2d_a$ under such a scheme.

The heart of the Saxons structure quality maintenance is a periodic routine that continuously adjusts for potentially better structure quality. More specifically, it checks the distance to hosts in the local random membership subset and replace the longest existing links if new hosts are closer. The host distance comparison is determined based on landmark-based Cartesian distance or other means if available. In addition to quality-oriented link adjustments, each node also periodically pings its Saxons neighbors. A neighbor link will be replaced when several consecutive pings fail.

The overlay structure stability is important for the performance of overlay services that maintain link-related state, including the DHT service we describe in the next section. In order to avoid frequent structure changes, we require that a link adjustment occurs only when a new link is shorter than the existing overlay link for more than a specified threshold. If needed, it is also possible to disable the quality-oriented structure changes sometime after a node bootstrap to further enhance the structure stability. It should be noted that runtime structure changes cannot be completely avoided at the presence of overlay membership changes.

If we use landmark-based latency estimation for structure quality maintenance, link probing is only necessary at node startup to measure latencies to a few designated landmark nodes. No additional runtime network overhead would be needed for the structure quality maintenance. Other Saxons components incur a runtime network overhead of around 1.3Kbps [27][1].

3 Distributed Hashtable on Pre-structured Overlay Networks

Distributed hashtable on pre-structured networks resembles network routing in subtle ways. A DHT lookup can be considered as a network routing request with its destination labeled with a hash key instead of a host ID. Our DHT design can draw upon many earlier efforts in designing scalable network routing protocols. In particular, we choose to base our design on the Landmark Hierarchy [29][2], due to its potential of self-organization and automatic adaptation to overlay membership changes. Our Landmark Hierarchy-based DHT protocol is designed to achieve the following goals:

Self-organization and automatic adaptation: In order to achieve quick deployment with low maintenance overhead, this protocol must be able to self-organize at startup and it must also automatically adapt to dynamic overlay membership changes due to node joins, leaves, and failures.

Scalability: To support large overlay groups, the per-node protocol maintenance overhead and storage consumption must increase *slowly* with the growth of the overlay size. For instance, a linear increase in such overhead would cripple the protocol scalability.

High-performance DHT lookup: We are interested in the DHT lookup performance in terms of both lookup latencies and hop-counts. Without counting any processing time at each intermediate overlay node, the lookup latencies represent the best-case DHT lookup performance. The hop-counts, on the other hand, indicate the amount of potential performance penalty in the presence of transient congestion or faults at intermediate nodes.

Balanced key placement and lookup routing overhead: Load balance is also an essential goal for a distributed hashtable and it is especially difficult to achieve for hierarchical schemes. We consider protocol load balance in terms of both key placement and lookup routing overhead.

The rest of this section describes a Landmark Hierarchy on pre-structured overlays (Section 3.1), followed by its automatic construction and adaptation scheme (Section 3.2). We then present our DHT design based on this Landmark Hierarchy (Section 3.3).

3.1 Landmark Hierarchy on Pre-structured Overlays

Our concept of Landmark Hierarchy mostly follows that of the original Landmark Hierarchy [29], with necessary changes to support the DHT service. A *Landmark* is a node whose neighbors within a certain number of hops (called *routing radius*) contain routing entries for it. This is usually achieved by having each landmark periodically flood a route advertisement message along the overlay structure for up to a hop-count bound of its routing radius. For our DHT protocol, all overlay nodes form a hierarchy of landmarks, with level 0 being the lowest level, and level H being the highest level. Every overlay node is at least a level 0 landmark. All landmarks in the same level have the same routing radius. A higher-level routing radius is always larger than a lower-level routing radius. And the level H landmarks flood their route advertisements to the complete overlay. Let rr_l be the level l routing radius. Therefore we have $rr_{l+1} > rr_l$ for each $0 \leq l < H$; and $rr_H = \infty$.

[1] Higher runtime network overhead may be incurred for bandwidth-oriented structure optimizations in Saxons. However, bandwidth-oriented structure optimizations are not needed for the DHT construction targeted in this paper.

[2] Do not confuse the *Landmark* Hierarchy with the *landmark*-based Cartesian distance approach we use for latency estimation.

We call a child-parent relationship exist between node A and B when A is one-level lower than B in the Landmark Hierarchy and they are within each other's routing radii, *i.e.*, they have routing entries for each other. We require that all nodes except those at the top level have at least a parent. Note that a node may have multiple parents in the hierarchy. We also require each node carry IDs of all its children in its route advertisements and subsequently they are stored as part of the routing entry at nodes within the routing radius. This information is critical for our DHT construction, though it is not needed for network routing in the Landmark Hierarchy [29]. Figure 2 illustrates a Landmark Hierarchy and its routing table layout.

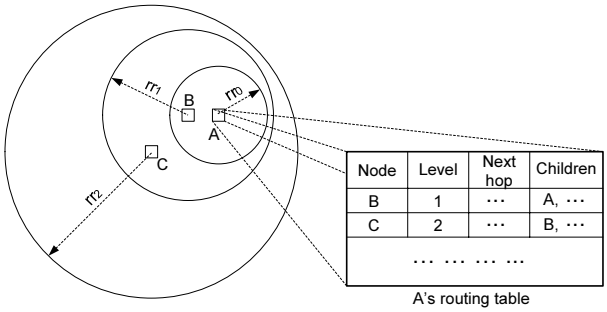

Figure 2: A Landmark Hierarchy and its routing table layout. Nodes A, B, and C are level 0, 1, and 2 landmarks respectively.

Having small routing tables is a key benefit for hierarchical routing schemes. Kleinrock and Kamoun found that the average number of routing table entries in an H-level hierarchy with a single top-level node is at best $H \times N^{\frac{1}{H}}$, where N is the total number of nodes in the hierarchy [12]. This may only be achieved when every landmark (except level 0 nodes) has the same number of children and no landmark has more than one parent. In practice, the routing table sizes are somewhat larger and we will examine that in the performance study.

3.2 Hierarchy Construction and Adaptation

We now describe an automatic hierarchy construction and adaption scheme. The goal is to dynamically maintain a balanced hierarchy with exponentially smaller population at higher levels. All overlay nodes start at level 0 with a routing radius of rr_0 hops. Each node sends out periodic routing advertisements at interval t_{int} to other nodes within its routing radius. Routing entries are recorded or refreshed upon the receipt of these advertisements. Routing entries are kept as soft state such that they expire after not being refreshed for several rounds. Periodically, every node checks the existence of an unexpired routing entry for a parent. If a parent routing entry does not exist and the hierarchy level-bound has not be reached, it schedules a *promotional* procedure at a random delay. The node increases its landmark level by one at the execution of the promotional procedure. The random delay for scheduling promotional procedures is chosen uniformly from $[t_{int} + t_{delay}, (1 + \alpha N_{peer})t_{int} + t_{delay}]$, where t_{delay} is the estimated message propagation delay upper-bound in the overlay network, α is a constant, and N_{peer} is the number of same-level peers in the local routing table. The linear back-off component on N_{peer} is employed to prevent many nodes in a densely connected area to promote themselves simultaneously. The scheduled promotional procedure is canceled when a routing advertisement from a parent is later received.

When the hierarchy level-bound is large, it is desirable to stop the hierarchy buildup when there is only a single top-level landmark node. Following an idea presented in [13], a node without any same-level peer in its routing table never promotes itself. This scheme works when each node sees at least one same-level peer if any exists, which can be ensured by having $rr_{l+1} > 2 \times rr_l$ for all $l \geq 0$. However, one drawback with this approach is that the routing radii increase too fast for high levels, resulting in large number of children nodes at high levels. We fix this problem by introducing a peer notification radius (denoted by pr_l at level l) independent of the routing radius. Each level l landmark floods the overlay network with a peer notification announcement for up to pr_l hops. We require $pr_{l+1} > 2 * rr_l$ such that each node sees at least one same-level peer if any exists.

A node may want to lower its hierarchy level after some other nodes depart from the overlay or if an earlier promotion has been pre-mature. We employ two demotion rules in the automatic hierarchy adaptation.

Rule 1: Each node periodically checks the existence of an unexpired routing entry for a child. When discovering no child is present, it schedules a *demotional* procedure at the delay of $t_{int} + t_{delay}$. We use a constant scheduling delay because no back-off is necessary in this case.

Rule 2: Each node also checks its routing table for whether a hierarchy peer can serve as a parent if it demotes itself. This is the case when the hop count distance to one of the peers is within the routing radius of the hierarchy level after the demotion. If so, a demotional procedure is scheduled at a random delay between $[t_{int} + t_{delay}, (1 + \beta N_{peer})t_{int} + t_{delay}]$. The linear back-off component on N_{peer} is employed to prevent all peers to demote themselves simultaneously. This demotion rule ensures that no two level l landmarks are within the distance of rr_{l-1} from each other.

3.3 Distributed Hashtable on the Landmark Hierarchy

In this section, we first describe the mapping scheme between each hash key and a deterministic host in our DHT protocol. We then present a distributed algorithm for any node to find such location with a given hash key.

One of the building blocks in our DHT mapping scheme is the Chord identifier circle [28]. It is necessary to provide a brief description of it. In Chord, each overlay node and key is assigned an *identifier* in $<0 - ID_{max}>$ using an ID assignment function such as SHA-1 [26] or MD5 [22]. A node's identifier is chosen by hashing the node's IP address, while a key identifier is generated by hashing the key. All identifiers are ordered in an identifier circle modulo $ID_{max} + 1$. Key k is assigned to the first node whose identifier is equal to or follows k's in the identifier space, called $owner(k.id)$. If identifiers are represented as a circle of numbers from 0 to ID_{max}, then $owner(k.id)$ is the first node clockwise from k. Figure 3 shows one such identifier circle, following an example given in [28].

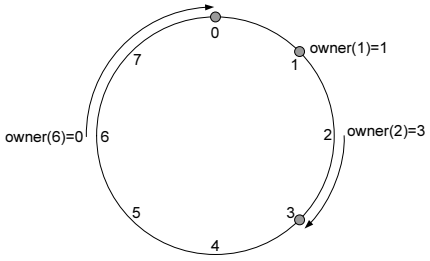

Figure 3: An ID circle with three nodes 0, 1, 3. In this example, key 1 maps to node 1, key 2 to node 3, and key 6 to node 0.

Instead of a single identifier circle, our protocol employs a hierarchy of identifier circles to map hash keys to overlay nodes. First, all top level (*e.g.*, level H) landmarks form a level H identifier circle (denoted by idc_H). In addition to the top level identifier circle, all children of each level $l+1$ landmark X ($0 \leq l < H$) form a level l identifier circle (denoted by $idc_l(X)$). Note that there are typically multiple identifier circles in each level below level H. Each hash key k is first mapped to a level H landmark node (denoted by $n_H(k)$) in the top level identifier circle. Then it is subsequently mapped to a level $H-1$ landmark in $n_H(k)$'s children identifier circle. This process continues until the hash key is eventually hashed into a level 0 landmark $n_0(k)$, which is considered as the key's final owner. One problem with this scheme is that hash keys mapped to a particular landmark are close to each other in the identifier circle. Therefore these keys would always map into the same region in subsequent lower-lever identifier circles, causing imbalance in key placement. In order to fix this problem, we use different key-identifier assignment functions in each level such that keys with close identifiers in one level are spread out in the identifier circles for all other levels. We use $\mathrm{MD5}_l()$ to denote the ID assignment function at level l. Equation (1) and Figure 4 illustrate our DHT mapping scheme at each level. Note again that the level 0 DHT owner $n_0(k)$ is considered as k's final owner.

$$n_l(k) = \begin{cases} idc_H.owner(MD5_H(k)) & \text{if } l = H, \\ idc_l(n_{l+1}(k)).owner(MD5_l(k)) & \text{if } 0 \leq l < H. \end{cases} \quad (1)$$

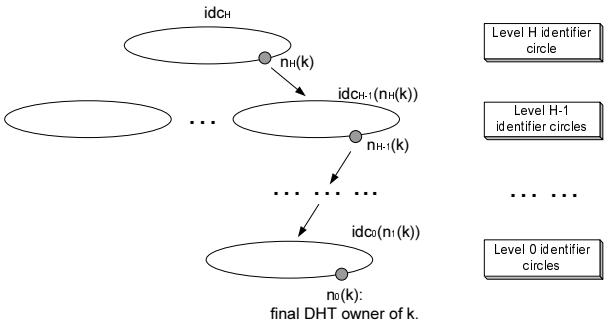

Figure 4: Illustration of the proposed DHT mapping scheme.

We now describe a distributed lookup algorithm to implement the DHT mapping described above. For each given hash key k, the lookup initiator node A first finds k's level H owner (denoted by $n_H(k)$) in the top level identifier circle. This can be performed locally at every node since the identifiers of all top level landmarks are known to all through their route advertisements. Because all children identifiers are carried in each route advertisement, A is also able to locally find k's level $H-1$ owner (denoted by $n_{H-1}(k)$) in n_H's children identifier circle. If A has $n_{H-1}(k)$'s routing entry (and therefore the identifiers of all its children), A would continue to perform lookup for lower-level DHT owners. When A's local lookup stops because it does not have the routing entry for k's level $l-1$ DHT owner $n_{l-1}(k)$, A forwards the lookup query to its next hop node toward $n_l(k)$, which it has a routing entry for. This process continues until $n_0(k)$ is located. Figure 5 illustrates this algorithm in recursive form. This algorithm is invoked at the lookup initiator node with DHT_LOOKUP(key, H, $n_H(key)$).

Note that lookup queries normally do not go through high-level DHT owners before finding the final level 0 owner. This is because a lookup query aiming at the level l DHT owner shifts toward the level $l-1$ owner as soon as it moves within its routing radius. This is more so when rr_i is much larger than rr_{i-1}. This behavior is essential

Algorithm 3.1: DHT_LOOKUP(key, l, n_l)

Input: key: the hash key.
Input: l: the current lookup level.
Input: n_l: the DHT owner in the current lookup level.

// Local lookup for lower-level DHT owners.
while $l > 0$ **do**
  $n_{l-1} \leftarrow idc_{l-1}(n_l).owner(MD5_{l-1}(key))$;
  if n_{l-1} is not in the local routing table **then break**;
  $l \leftarrow l - 1$;
enddo;

// Finding n_0 – global termination.
if $l = 0$ **then return** (n_0);

// Proceed to the next hop and perform recursive lookup.
$m \leftarrow$ the next hop node toward n_l;
return $(m$.DHT_LOOKUP$(key, l, n_l))$;

Figure 5: The distributed lookup algorithm in recursive form.

for offloading higher-level landmarks in terms of lookup routing overhead.

The stability of the DHT mapping scheme is important for applications that require data migration when key assignments change, such as distributed storage systems. Our hierarchy generation and DHT mapping scheme are sensitive to changes in the overlay structure. For instance, a link change may alter a parent-child relationship, which results in a change of the DHT mapping. As being mentioned in Section 2, the Saxons structure quality maintenance protocol can be configured to provide high stability. However, structure changes are necessary as the result of overlay membership changes. In comparison, key/node mapping schemes in structured DHT protocols are independent of overlay structures, therefore they are likely to produce fewer key reassignments due to membership changes. Section 4.5 provides a quantitative comparison on this.

4 Simulation Results

Our performance evaluation consists of simulations and Internet experiments. The goal of simulation studies is to assess the effectiveness of proposed techniques for large-scale overlays while Internet experiments illustrate the system performance under a small but practical real-world environment.

In this section, we evaluate the performance of our Saxons-based DHT protocol using discrete-event simulations. We first describe our evaluation methodology and setup, followed by results on the quality of the Saxons overlay structure management layer. We then demonstrate our DHT performance compared with a strongly-structured scalable DHT protocol. We show the performance results in terms of lookup latency, load balance, and performance under frequent overlay membership changes. Section 5 presents experimental results on 51 PlanetLab sites.

4.1 Simulation Methodology and Setup

We use a locally-developed discrete-event simulator in our evaluations. We simulate all packet-level events at overlay nodes. We do not simulate the packet routing at the substrate network routers. Instead, we assume shortest-path routing in the substrate network and use that to determine the overlay link latency. We acknowledge that this model does not capture packet queuing delays or packet losses at routers and physical links. However, such a tradeoff is important to allow us achieve reasonable simulation speed for large networks.

Backbone	Node count	Link latency
ASmap	3,104	$1-40$ms
Inet	3,050	$1-40$ms
TransitStub	3,040	$1-20$ms for stub links $1-40$ms for other links
AMP-all	118	measurement
AMP-domestic	108	measurement

Table 1: Backbone networks.

The substrate networks we use in the simulations are based on four sets of backbone networks including a measurement-based one. First, we use Internet *Autonomous Systems* maps extracted from BGP routing table dumps, available at NLANR [17] and at the University of Oregon Route Views Archive [23]. Second, we include some transit-stub topologies generated using the GT-ITM toolkit [32]. We also use topologies generated by the Michigan *Inet*-3.0 Internet Topology Generator [31]. For ASmap and Inet topologies, we assign a random link latency of $1-40$ms. For TransitStub topologies, we assign a random link latency of $1-20$ms for stub links and $1-40$ms for other links. The last set of backbone network is based on end-to-end latency measurement data among 118 Internet nodes, reported by the Active Measurement Project (AMP) at NLANR [18]. Table 1 lists some specific backbone networks we used in our evaluations. The AMP-domestic network excludes 10 foreign hosts from the full AMP dataset. These 10 hosts have substantially larger latencies to other hosts than others. With a given backbone network, each overlay node in our simulations is randomly attached to a backbone node through an edge link. We assign a random latency of $1-4$ms for all edge links.

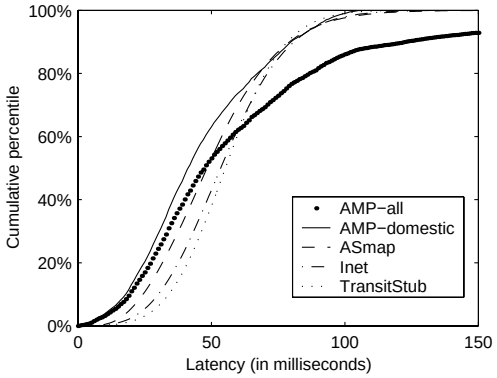

Figure 6: All-pair latency CDFs of backbone networks.

Our link latency assignments for non-measurement networks are quite arbitrary. As an interesting exercise, we calculated the cumulative percentile of all-pair shortest path latencies for each backbone network (shown in Figure 6). It appears that the latency CDFs of the three non-measurement networks are quite close to that of AMP-domestic. The latency CDF for AMP-all has a lower tail than others due to the substantially larger latencies for paths associated with the 10 foreign hosts.

The evaluation results are affected by many factors, including the backbone networks, protocol parameters, and the combination of different schemes for various protocol components. Our strategy is to demonstrate the protocol effectiveness at the most typical settings and then evaluate the impact of additional factors when necessary. Except a few experiments explicitly evaluating the impact of backbone networks, most results shown in this paper are based on the ASmap backbone network. The periodic Saxons structure quality maintenance routine runs at 30-second intervals.

4.2 Overlay Structure Quality

We show the overlay structure quality in two metrics: 1) *overlay path latency*, defined as the end-to-end latency along the shortest overlay path for each pair of nodes; and 2) *relative delay penalty* (or *RDP*), defined as the ratio of the overlay path latency to the direct Internet latency. We compare three different overlay structure construction schemes. First, we consider the Saxons protocol with the landmark-based Cartesian distance approach for latency estimation (denoted by *Saxons (Landmark)*). Second, we examine Saxons with an accurate latency estimation between any two nodes (denoted by *Saxons (Accurate)*). Even though we are unaware of any accurate latency estimation scheme that scales to a large number of nodes, the results for Saxons (Accurate) are useful in indicating the performance potential of a Saxons-like structure management protocol. We finally consider the degree-bounded random structure construction used in protocols like Gnutella [7] (denoted by *Random*).

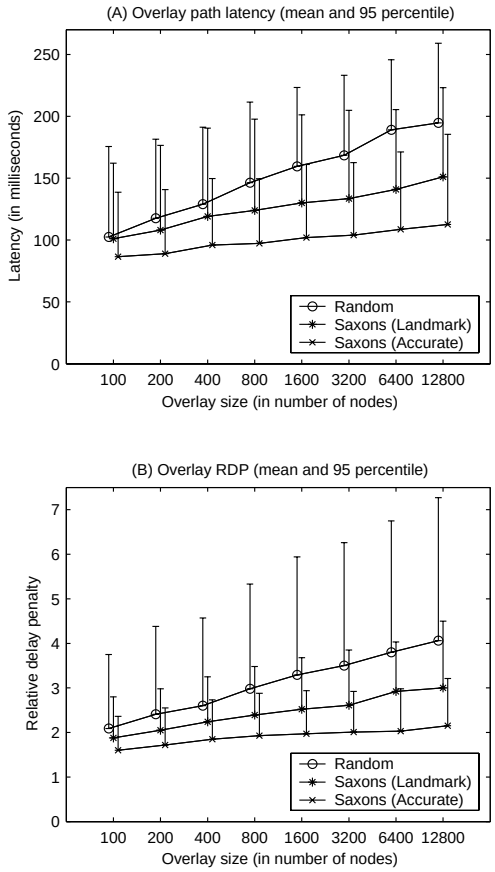

Figure 7: Overlay structure quality at various overlay sizes.

Figure 7 illustrates the overlay path latency and RDP at various overlay sizes. For each overlay size, nodes join the network at the average rate of 10 joins/second with exponentially distributed inter-arrival time. Node joins stop when the desired overlay size is reached and the measurement results are taken after the system stabilizes. All three schemes are configured with a node degree range of $<4-16>$. In other words, each node is allowed to make 4 active links. Each node can also accept a number of passive links as long as the total degree does not exceed 16. For 12800-node overlays, results show that Saxons (Accurate) achieves 42% lower overlay path latency and 47% lower RDP compared with Random. The saving for Saxons (Landmark) is 22% on overlay path latency and 26% on RDP.

Figure 8 shows such comparison over different backbone networks. 3200-node overlays are used for the three large backbones and 200-node overlays are used for the two measurement-based backbones (AMP-all and AMP-domestic). Results show that the performance difference

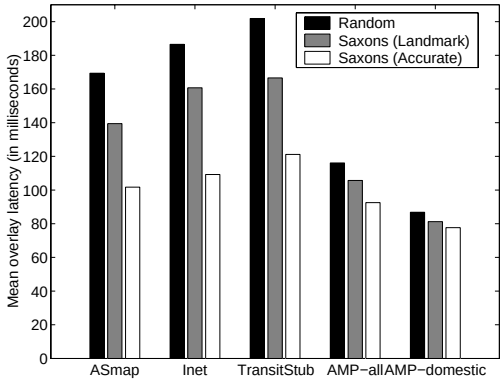

Figure 8: Overlay path latency over different backbone networks.

is not significantly affected by the choice of backbone networks. The savings are smaller for the two measurement-based backbones due to their small sizes.

The performance results of the Saxons structure management layer indicate how much it can benefit overlay services built on top of it. The rest of the performance evaluation focuses on our proposed DHT protocol running on top of the Saxons layer.

4.3 Statistics on the Landmark Hierarchy

We show some statistics on the self-organizing Landmark Hierarchy construction over the Saxons overlay structure. The Saxons overlay structure is configured with a node degree range of $<4-16>$. The routing radii (starting from level 0) for the Landmark Hierarchy are set as 2, 4, 8, 16, 32, 64, ... The peer notification radii (starting from level 0) are 2, 5, 9, 17, 33, 65, ... Note that we require $pr_{l+1} > 2 * rr_l$ such that each node sees at least one other same-level peer if any exists.

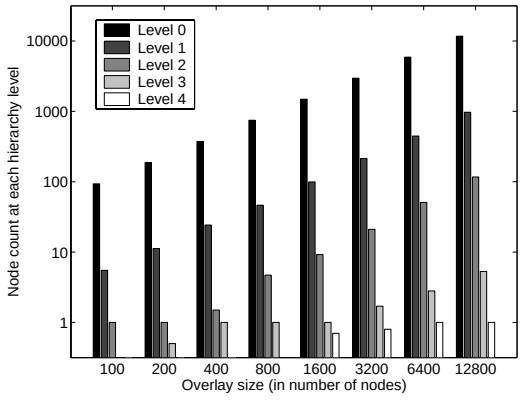

Figure 9: Node count at each hierarchy level. Note that the Y-axis is on the logarithmic scale.

Figure 9 shows the overlay node count in each hierarchy level for up to 12800 overlay nodes. Note that the Y axis is on the logarithmic scale. The results are assigned using the average value of five runs, which explains some non-integer values in the figure. The results indicate an exponentially larger population at lower hierarchy levels with around ten times more nodes at each immediate lower level.

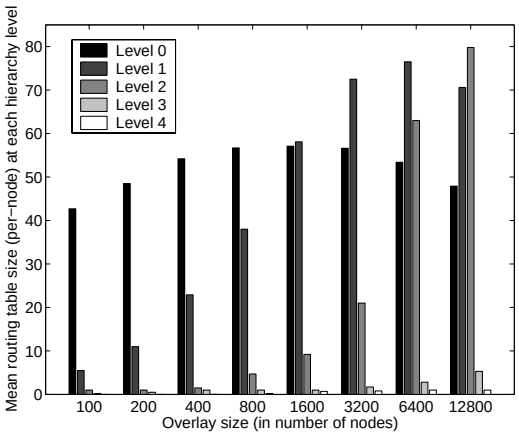

Figure 10: Mean routing table size at each hierarchy level.

Figure 10 illustrates the mean routing table size in each hierarchy level. The level-l routing table at a node contains all received route advertisements from level-l landmarks. We observe that the number of routing entries at each level remains around or below 80 for up to 12800 overlay nodes. The reason is that the large advertisement flooding hops of high-level landmarks are compensated by their sparse presence in the overlay structure. Note that the routing table sizes can be controlled by adjusting the routing radii in the hierarchy generation. Our adaptive promotion and demotion schemes result in the automatic construction of balanced hierarchies.

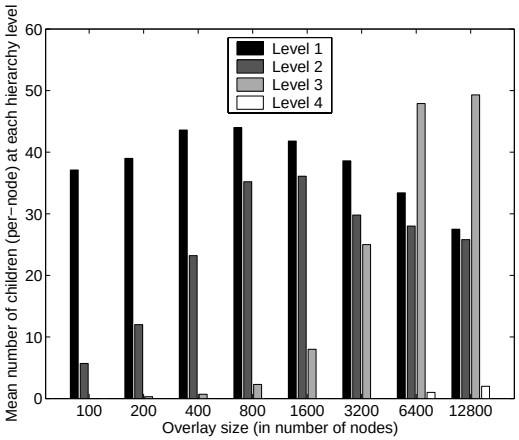

Figure 11: Mean number of children at each hierarchy level.

Since the list of children is included in the landmark route advertisement, a large children population at overlay nodes may result in excessive route advertisement overhead. Figure 11 shows the average number of children for nodes in each hierarchy level. There are no level-0 results because level-0 landmarks have no child. We observe that the number of children at each node remains around or below 50 for up to 12800 overlay nodes. Again, the large advertisement flooding hops of high-level landmarks are compensated by their sparse presence in the overlay structure.

4.4 DHT Lookup Performance

We examine the performance of our proposed DHT protocol that does not employ service-specific overlay structures. We compare our protocol built on top of the Saxons structure management layer (denoted by *Saxons-DHT*) with Chord [28], a well-known DHT protocol. A recent study [9] shows that Chord performs competitively against other strongly-structured DHT protocols such as CAN [20] and Pastry [24] in terms of lookup latency and load balance. We implemented the SaxonsDHT and Chord protocols in our overlay simulator. In our comparison, both schemes are configured at the same link density. For SaxonsDHT, the Saxons overlay structure is configured with a node degree range of $<4-16>$ and therefore an average degree of 8. For Chord, each node maintains an 8-entry finger table supporting DHT lookups [3]. Higher-level finger entries in Chord point to nodes with exponentially larger distances in the identifier circle.

Figure 12(A) illustrates the DHT lookup hop-count for SaxonsDHT and Chord at various overlay sizes. The results are based on the average of five runs, each with 100,000 DHT lookups on randomly chosen initiator nodes and hash keys. A quick analysis finds that the mean lookup hop-count for a Chord protocol with d finger entries is $d(\sqrt[d]{N}-1)/2$. Results in Figure 12(A) show that SaxonsDHT achieves slightly better performance (around 12% fewer lookup hops for 12800-node overlays) due to its hierarchical lookup routing scheme. Figure 12(B) shows the DHT lookup latency at various overlay sizes. We introduce variations of the SaxonsDHT and Chord protocols that may have significant impact on the lookup latency. For SaxonsDHT, we examine two variations: one with the landmark-based Cartesian distance approach for latency estimation and another with an accurate latency estimation. As we discussed earlier, accurate latency estimation may not be practical for large-scale overlays, but it is useful in indicating the performance potential of the

[3] Strictly speaking, the Chord structure with an 8-entry finger table at each node has an average node degree of 16. It might appear that Chord has some unfair advantage in this comparison. Our rationale for such a comparison setting is that Saxons links are bidirectional while Chord finger links are not.

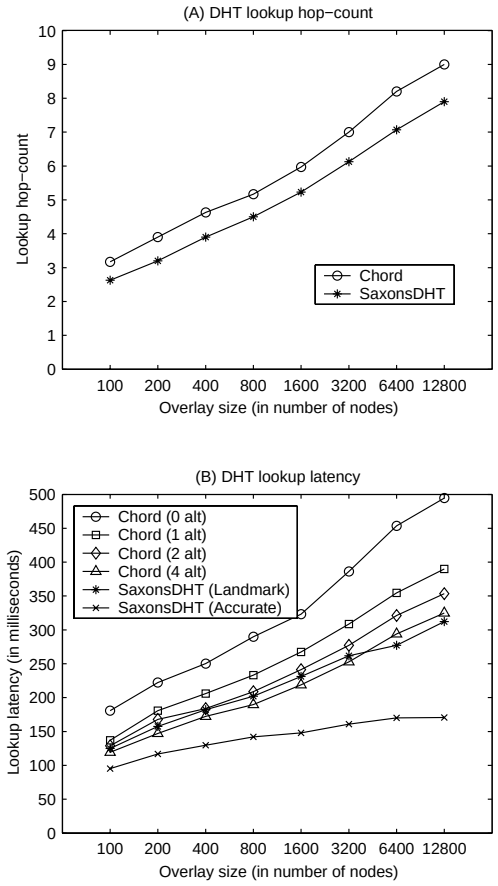

Figure 12: DHT lookup performance at various overlay sizes.

SaxonsDHT protocol if more accurate latency estimation schemes become available. We also examine variations of the Chord protocol with a substrate-aware link-selection enhancement, as indicated in Section 7 of [28]. In this enhancement, instead of simply picking the first node in each finger table entry's interval in the identifier ring, a few alternative nodes in each interval are probed and then the closest node is chosen to fill the finger table entry. We use *Chord (n alt)* to denote the Chord protocol with n alternative link probings for each finger table entry. In particular, Chord (0 alt) stands for the basic Chord protocol without the link-selection enhancement. For 12800-node overlays, results in Figure 12(B) show that SaxonsDHT (Landmark) achieves 37% less lookup latency than the basic Chord protocol and its performance is close to that of Chord (4 alt). Results also show that SaxonsDHT (Accurate) outperforms Chord (0 alt) and Chord (4 alt) at 65% and 31% respectively, indicating the vast performance potential of SaxonsDHT.

Figure 13 shows the DHT lookup latency over different backbone networks. Results indicate that the performance difference is not significantly affected by the

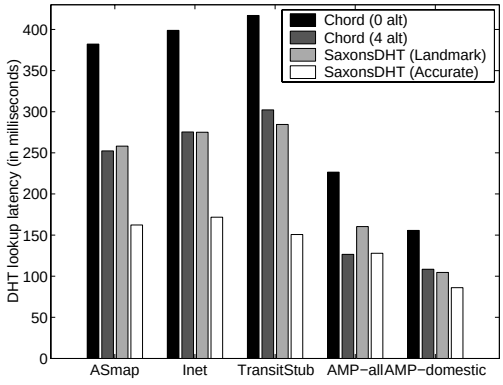

Figure 13: DHT lookup latency over different backbone networks.

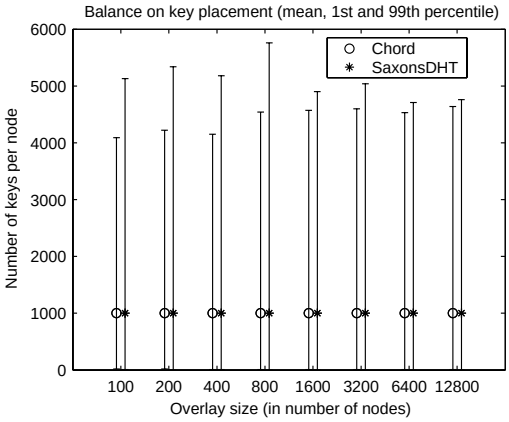

Figure 14: DHT load balance on key placement.

choice of backbone networks. Again, savings are smaller for the two measurement-based backbones due to their small sizes.

4.5 DHT Load Balance

Load balance is another essential goal for a distributed hashtable and it is particularly challenging for hierarchical schemes. In our DHT protocol, we employ different key-identifier assignment functions at each hierarchy level to achieve balanced key placement. The balance on lookup routing overhead is supported by the property that queries often shift toward lower-level DHT owners before actually reaching any high-level DHT owner.

Figure 14 illustrates the DHT load balance on key placement over various overlay sizes. $1000 \times N$ (N is the overlay size) random keys are generated and mapped into overlay nodes. The mean, 1st, and 99th percentile values are shown for each configuration. Results show that the balance of key placement for SaxonsDHT is close to that of Chord. We do not show results for different variations of the SaxonsDHT and Chord protocols because they do not have significant impact on the balance of key placement.

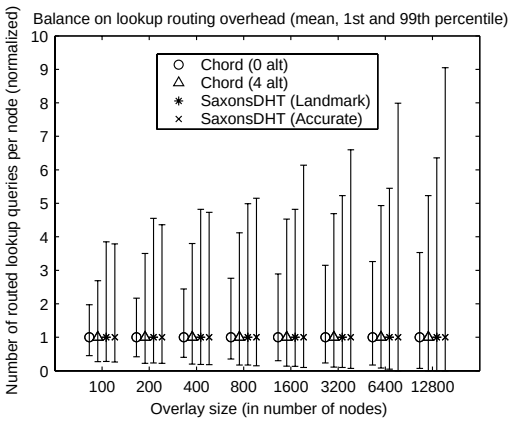

Figure 15: DHT load balance on lookup routing overhead.

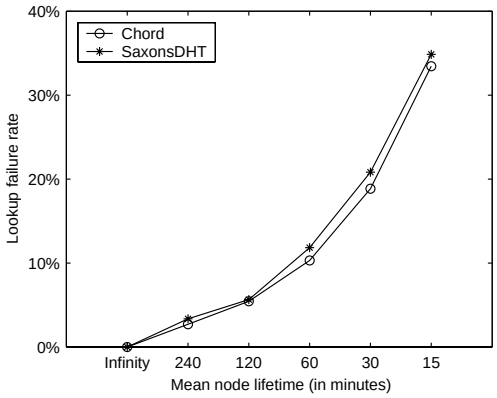

Figure 16: Lookup failure rate during overlay membership changes.

Figure 15 shows the DHT load balance in terms of the lookup routing overhead. Note that the results in Figure 15 are normalized to the mean values. We observe that protocols with lower lookup latency typically exhibit less desirable load balance. For 12800-node overlays, the normalized 99-percentile lookup routing overhead for Chord (0 alt) is 44% and 61% less than those of SaxonsDHT (Landmark) and SaxonsDHT (Accurate) respectively. The difference between the substrate-aware Chord (4 alt) and the two SaxonsDHT schemes is much less. The inferior load balance of substrate-aware DHT protocols can be explained by their tendency of avoiding nodes that have long latency to other overlay nodes. We will revisit this issue in Section 5 using Internet experimentation.

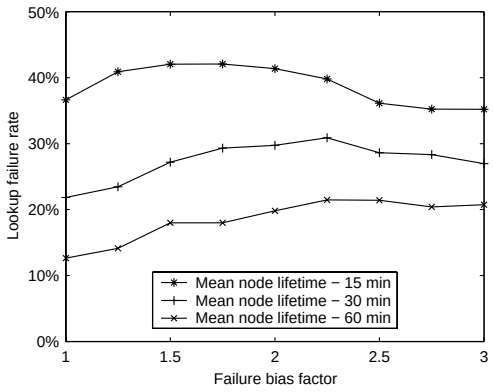

Figure 17: SaxonsDHT lookup failure rate under biased node failures.

4.6 DHT Performance under Unstable Environments

Figure 16 shows the DHT lookup failure rate of SaxonsDHT and Chord under frequent node joins and departures at various average node lifetimes. The results are for 3200-node overlays. Individual node lifetimes are picked following an exponential distribution with the proper mean. In these experiments, a lookup is considered a success if it reaches the current level 0 DHT owner of the desired key. In a real system, however, there might be delays in which the current owner has not yet acquired the data associated with the key from the prior owner. We do not consider this factor since it is highly dependent on higher-level service implementation while we are primarily concerned with the DHT protocol. Results in Figure 16 show that SaxonsDHT and Chord deliver similar lookup success rate during overlay membership changes. Following an argument in [28], the lookup success rate under a certain frequency of membership changes mainly depends on the average lookup hop-counts. The similar success rate between SaxonsDHT and Chord can be explained by their similar lookup hop-counts.

A hierarchical scheme like SaxonsDHT may suffer poor performance under targeted attacks against nodes of high importance. We examine the lookup failure rate when nodes at higher hierarchy levels have shorter lifetime. To quantify such scenarios, we introduce the concept of *failure bias factor*, defined as the ratio of the average node lifetime at each hierarchy level to that of the immediate higher level. In other words, for a failure bias factor of 2, level 1 nodes are twice likely to fail than level 0 nodes while level 4 nodes are 16 times more likely to fail than level 0 nodes. Figure 17 illustrates the SaxonsDHT lookup failure rate under biased node failure rates for 3200-node overlays. The results show that the lookup failure rate increases initially at the increase of the failure bias factor. However, this increase tapers off quickly and it may even decrease as the failure bias factor continues to grow. This is because the SaxonsDHT lookups do not critically rely on the constant aliveness of high-level landmark nodes. Lookups are mainly based on local routing entries and high-level nodes are often not visited. This result shows that SaxonsDHT lookups are not particularly susceptible to targeted attacks despite the nature of its hierarchical design.

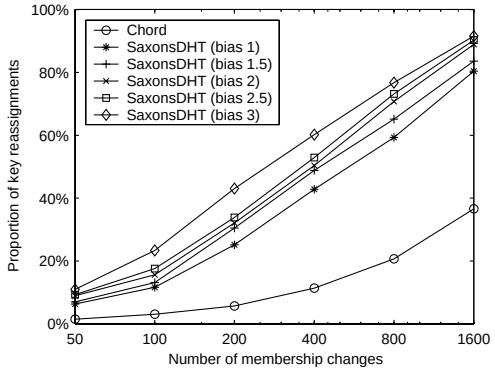

Figure 18: Key reassignments due to overlay membership changes. Note that the X-axis is on the logarithmic scale.

Due to its structure-sensitive DHT mapping scheme, SaxonsDHT tends to produce more key reassignments after overlay membership changes. Figure 18 shows such performance of SaxonsDHT and Chord. The results are for 3200-node overlays. We measure the proportion of key assignment changes after certain number of random node joins and leaves. We consider the SaxonsDHT performance under several failure bias factors to model the impact of targeted failures of high-level landmarks. The results show that SaxonsDHT produce two to three times more key reassignments (186% more in average) after overlay membership changes. Targeted failures of high-level landmarks may result in even more key reassignments. Such a performance difference is significant. This suggests that our DHT design may be better suited for applications that do not require large data migration after the DHT mapping changes. We should also point out that the DHT mapping in Chord is based on *consistent hashing*, a technique with provable minimal key reassignments after membership changes [11].

5 Implementation and Internet Experimentation

We have made a prototype implementation of the proposed DHT design on the Saxons structure management layer. Saxons components are implemented in a single event-driven daemon. The Saxons prototype can run as

a standalone process communicating through UNIX domain sockets with hosted overlay applications linked with a Saxons stub library. Alternatively, the Saxons runtime can be dynamically linked and run inside the application process space. A standalone Saxons process allows possible runtime overhead sharing when overlay nodes host multiple services.

In our DHT implementation, every node maintains a TCP connection with each of its overlay neighbors. The route advertisement flooding and lookup query routing flow through these TCP connections on the overlay structure. All the DHT components are implemented in a single event-driven daemon (separate from the Saxons daemon). The DHT service periodically queries the Saxons kernel for up-to-date overlay structure information. Link-related state such as the TCP connections to neighbors and some routing table entries may have to be adjusted when directly attached overlay links change. For the purpose of comparison, we also made a prototype implementation of Chord. Our prototype can correctly form the Chord finger tables at the absence of node departures. We did not implement the full Chord stabilization protocol for simplicity. Each node in our Chord prototype maintains a TCP connection with each of the nodes listed in its finger table. Lookup queries flow through these pre-established TCP connections. For both DHT implementations, node IDs are assigned using MD5 [22] hashed IP addresses.

We conducted experiments on the PlanetLab testbed [19] to evaluate the performance of the proposed DHT service in a real-world environment. Our experiments involve 51 PlanetLab nodes, all from unique wide-area sites. Among these 51 sites, 43 are in the United States, 5 are in Europe. The other three sites are in Australia, Brazil, and Taiwan respectively. The round-trip latency between a U.S. site and a non-U.S. site is often much higher than that between two U.S. sites. Due to the small number of sites in the experiments, we are able to employ direct runtime latency measurements for the Saxons structure quality maintenance. This scheme is close to *Saxons (Accurate)* examined in the simulation studies. For runtime latency measurement, a node pings the other N times and measure the round-trip time. It then takes the average of the median 60% of the measurement results. We choose N=10 in practice. To demonstrate its accuracy, we compare measurement results under this scheme with those using 100 pings in each test (shown in Figure 19). Results are ranked in the ascending order of the 100-ping measurements.

Though a 51-node overlay is too small for a practical DHT deployment, we believe it still serves our purpose of verifying the simulation results. We had the option of experimenting with larger overlays by running multiple DHT and Saxons daemons inside each node. However, this unrealistic setting would result in many artificial clusters in the Saxons structure, which would put the measurement results in doubt. In order to compensate the small size of our testbed, we use a relatively sparse overlay structure in our experimentation. For SaxonsDHT, the Saxons overlay structure is configured with a node degree range of $<2-8>$. In other words, each node is allowed to make 2 active links and therefore the average node degree is 4. The settings for routing radii and peer notification radii are the same as those in the simulation study. A typical run shows that the Landmark Hierarchy contains one level 3 landmark, three level 2 landmarks, and eight level 1 landmarks. The remaining nodes are at level 0.

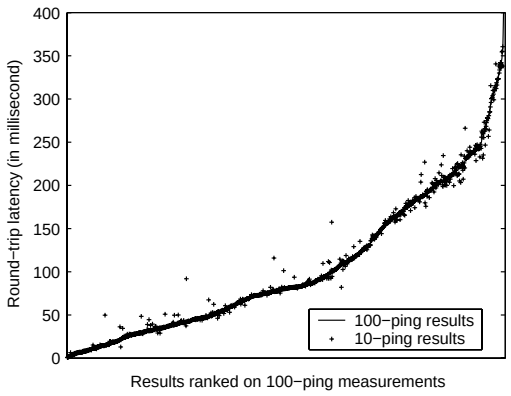

Figure 19: All-to-all round-trip latency measurements between 51 PlanetLab sites.

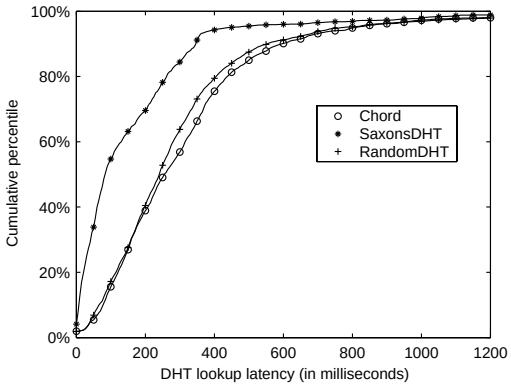

Figure 20: DHT lookup latency CDFs on 51 PlanetLab sites.

We compare the performance of SaxonsDHT against Chord with a 4-entry finger table at each node. For the purpose of comparison, we also consider the performance of our proposed DHT service running on a degree-bounded random overlay structure (denoted by *RandomDHT*). In each run of our experiments, 1000 DHT lookups are initiated at each participating node with random hash keys. Figure 20 illustrates the cumulative distribution functions of the 51,000 DHT lookup latency mea-

surements taken out of a typical test run. We observe that the performance of RandomDHT is close to that of Chord while SaxonsDHT significantly outperforms them in terms of DHT lookup latency. In average, SaxonsDHT achieves about 48% latency reduction compared Chord (335.5ms *vs.* 643.0ms).

We also examine the DHT load balance on lookup routing overhead for SaxonsDHT and Chord. Figure 21 shows the number (normalized to the mean value over all nodes) of routed lookup queries over each node. Results are increasingly ranked on the lookup routing overhead for each scheme. Larger markers in the figure represent 8 non-U.S. sites in the testbed. The results show that Chord exhibits better load balance than SaxonsDHT. We also observe that SaxonsDHT tends to avoid the non-U.S. sites in query routing while Chord is oblivious to network distances between participating sites. Such a behavior helps SaxonsDHT to achieve better lookup performance at the expense of load balance.

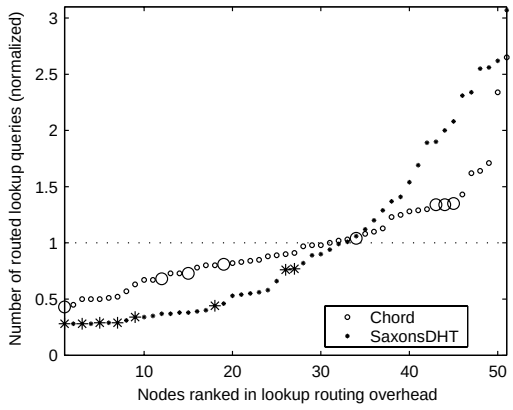

Figure 21: Load balance of DHT lookup routing overhead on 51 PlanetLab sites. Larger markers represent non-U.S. sites.

Overall, our Internet experiments on DHT lookup performance and runtime load balance match the simulation results presented in Section 4.

6 Related Work

Substrate-aware techniques have been incorporated into the construction of many Internet overlay services, including unicast overlay path selection (*e.g.*, RON [1]) and measurement-based end-system multicast protocols (*e.g.*, Narada [5], Overcast [10], HMTP [33], and NICE [2]). Substrate-aware techniques in these protocols are optimized for specific service needs. In comparison, we explore the model of providing a common overlay structure management layer that can benefit the construction of a wide range of services.

Previously proposed scalable DHT protocols such as Chord [28], CAN [20], and Pastry [24] all function on protocol-specific overlay structures to support DHT lookups. A recent work [9] suggests that measurement-based overlay structures often have much lower latency than structures provided by Chord, CAN, or Pastry. However, it did not explain how a DHT service can be built on top of a low-latency measurement-based overlay structure. A number of recent studies have proposed substrate-aware techniques to enhance particular DHT protocols. Zhao *et al.* proposed to construct a secondary overlay (called Brocade) on top of existing DHT structures to exploit unique network resources available at each overlay node [35]. Ratnasamy *et al.* introduced a distributed binning scheme for CAN such that the overlay topology resembles the underlying IP network [21]. In Mithos [30], Waldvogel and Rinaldi proposed an efficient overlay routing scheme based on an energy-minimizing node ID assignment in a multi-dimensional ID space. Zhang *et al.* suggested a random sampling technique is effective for incrementally reducing lookup latency in DHT systems [34]. These approaches are valuable in improving the performance of specifically targeted protocols. However, substrate-aware techniques built for particular DHT structures cannot benefit other services.

Kleinrock and Kamoun proposed hierarchical routing protocols to achieve low routing latency with small routing table sizes [12]. Landmark Hierarchy was later introduced by Tsuchiya to allow minimal administration overhead and automatic adaptation to dynamic network environments [29]. Two recent studies (SCOUT [13] and L^+ [4]) have employed the Landmark Hierarchy-based routing and location schemes for sensor and wireless networks. Our design draws upon results and experience of these work. New techniques are introduced in our design to construct a distributed hashtable service and satisfy its performance requirements. For instance, balanced key placement is a unique performance objective for DHT and it has not been addressed in previous studies on hierarchical routing.

Nakao *et al.* have recently proposed an Internet topology probing kernel as a general utility for overlay service construction [15]. They constructed several end-system routing services on top of the topology probing kernel, which acquires AS-level Internet topology and routing information from nearby BGP routers. By separating the service construction from overlay structure generation (as we proposed in this paper for the DHT service), such topology probing utilities can be incorporated into the common structure layer such that a large number of overlay services can benefit transparently.

7 Conclusion

In this paper, we construct a distributed hashtable protocol that operates on pre-structured overlays, and thus is able to take advantage of a common structure management layer such as Saxons [27]. Compared with Chord [28], simulations and Internet experimentation find that the proposed scheme can deliver better lookup performance at the cost of less load balance on query routing overhead. Evaluation results also show that the balance of key placement and fault tolerance for our approach are close to those of Chord. In addition, we find that the proposed scheme is not particularly susceptible to targeted attacks despite the nature of its hierarchical design. However, the proposed approach produces more key reassignments after overlay membership changes, due to its structure-sensitive DHT mapping scheme. We also recognize that structured DHT protocols such as Chord, CAN, and Pastry are able to provide provable performance guarantee and overhead bound. In comparison, performance and overhead of a DHT protocol operating on pre-structured overlays are highly dependent on the given overlay structure. For instance, it is not yet clear how to find optimal landmark routing radii in a pre-structured overlay such that routing table sizes at all levels are tightly controlled.

In conclusion, our design of a DHT service on pre-structured overlays provides a promising alternative to previously proposed DHT protocols. More importantly, this effort supports the broader goal of providing a common overlay structure management layer that can benefit the construction of a wide range of overlay services. While it is well understood that this model works well with services like unstructured peer-to-peer search [7, 14] and unicast/multicast path selections (*e.g.*, RON [1] and Narada [5]), our work is the first to examine its applicability on the important distributed hashtable service.

Acknowledgment

This work was supported in part by NSF grants CCR-0306473 and ITR/IIS-0312925.

References

[1] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. of SOSP*, pages 131–145, Banff, Canada, October 2001.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proc. of SIGCOMM*, pages 205–217, Pittsburgh, PA, August 2002.

[3] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting Network Proximity in Peer-to-Peer Overlay Networks. In *Proc. of the FuDiCo Workshop*, Bertinoro, Italy, June 2002.

[4] B. Chen and R. Morris. L+: Scalable Landmark Routing and Address Lookup for Multi-hop Wireless Networks. Technical Report MIT-LCS-TR-837, Laboratory for Computer Science, MIT, 2002.

[5] Y.-H. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In *Proc. of SIGMETRICS*, pages 1–12, Santa Clara, CA, June 2000.

[6] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz, and Y. Jin. An Architecture for a Global Internet Host Distance Estimation Service. In *Proc. of the IEEE INFOCOM*, New York, NY, March 1999.

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

[8] J. Guyton and M. Schwartz. Locating Nearby Copies of Replicated Internet Servers. In *Proc. of SIGCOMM*, pages 288–298, Boston, MA, September 1995.

[9] S. Jain, R. Mahajan, and D. Wetherall. A Study of the Performance Potential of DHT-based Overlays. In *Proc. of USENIX Symp. on Internet Technologies and Systems*, Seattle, WA, March 2003.

[10] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable Multicasting with an Overlay Network. In *Proc. of USENIX Symp. on Operating Systems Design and Implementation*, San Diego, CA, October 2000.

[11] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistency hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proc. of the ACM Symp. on Theory of Computing*, pages 654–663, El Paso, TX, May 1997.

[12] L. Kleinrock and F. Kamoun. Hierarchical Routing for Large Networks. *Computer Networks*, 1:155–174, 1977.

[13] S. Kumar, C. Alaettinoglu, and D. Estrin. SCalable Object-tracking Through Unattended Techniques (SCOUT). In *Proc. of the Intl. Conf. on Network Protocols*, Osaka, Japan, November 2000.

[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of the ACM International Conference on Supercomputing*, pages 84–95, New York, NY, June 2002.

[15] A. Nakao, L. Peterson, and A. Bavier. A Routing Underlay for Overlay Networks. In *Proc. of SIGCOMM*, Karlsruhe, Germany, August 2003.

[16] E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-based Approaches. In *Proc. of the IEEE INFOCOM*, New York, NY, June 2002.

[17] National Laboratory for Applied Network Research. http://moat.nlanr.net/Routing/rawdata.

[18] Active Measurement Project at the National Laboratory for Applied Network Research. http://amp.nlanr.net.

[19] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. of the HotNets Workshop*, Princeton, NJ, October 2002.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. of SIGCOMM*, pages 161–172, San Diego, CA, August 2001.

[21] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *Proc. of the IEEE INFOCOM*, New York, NY, June 2002.

[22] R. Rivest. The MD5 Message-Digest Algorithm. Rfc-1321, Internet Engineering Task Force, April 1992.

[23] University of Oregon Route Views Archive Project. http://archive.routeviews.org.

[24] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems. In *Proc. of the IFIP/ACM Middleware*, Heidelberg, Germany, November 2001.

[25] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The End-to-End Effects of Internet Path Selection. In *Proc. of SIGCOMM*, pages 289–299, Cambridge, MA, August 1999.

[26] FIPS PUB 180-1: Secure Hash Standard. National Institute of Standards and Technology, U.S. Dept. of Commerce, April 1995.

[27] K. Shen. Structure Management for Scalable Overlay Service Construction. In *Proc. of USENIX/ACM Symp. on Networked Systems Design and Implementation (to appear)*, San Francisco, CA, March 2004.

[28] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of SIGCOMM*, pages 149–160, San Diego, CA, August 2001.

[29] P. F. Tsuchiya. The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks. In *Proc. of SIGCOMM*, pages 35–42, Stanford, CA, August 1988.

[30] M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. In *Proc. of the HotNets Workshop*, Princeton, NJ, October 2002.

[31] J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, Dept. of EECS, University of Michigan, 2002.

[32] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. of the IEEE INFOCOM*, San Francisco, CA, March 1996.

[33] B. Zhang, S. Jamin, and L. Zhang. Host Multicast: A Framework for Delivering Multicast To End Users. In *Proc. of the IEEE INFOCOM*, New York, NY, June 2002.

[34] H. Zhang, A. Goel, and R. Govindan. Incrementally Improving Lookup Latency in Distributed Hash Table Systems. In *Proc. of SIGMETRICS*, San Diego, CA, June 2003.

[35] B. Zhao, Y. Duan, L. Huang, A. D. Joseph, and J. D. Kubiatowicz. Brocade: Landmark Routing on Overlay Networks. In *Proc. of the Workshop on Peer-to-Peer Systems*, Cambridge, MA, March 2002.