Distributed Hashtable on Pre-structured Overlay Networks*

Kai Shen[1] and Yuan Sun[2]

[1] Department of Computer Science, University of Rochester, Rochester, NY 14627, USA
kshen@cs.rochester.edu
[2] Department of Computer Science, University of California, Santa Barbara, CA 93106, USA
suny@cs.ucsb.edu

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. A shared substrate-aware overlay structure can also save redundant per-service link-selection probing when overlay nodes participate in multiple services. Despite the benefits, the concept of building services on a common structure management layer does not work well with recently proposed scalable distributed hashtable (DHT) protocols that 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. Using simulations and experimentation on 51 PlanetLab sites, we measure the performance of the proposed scheme in terms of lookup latency, load balance, and stability during node churns.

1 Introduction

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] and measurement-based end-system multicast protocols [5]. The employment of a common software layer that maintains overlay connectivity structure can greatly ease the design and deployment of overlay services. For instance, a more effective link probing technique 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. Furthermore, 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 [2] indicates that link-selection probing can consume significant network resources. With a common structure management layer, upper-level services built on top of it can share the cost of structure maintenance.

* This work was supported in part by the National Science Foundation grants CCR-0306473 and ITR/IIS-0312925.

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 (*e.g.*, Gnutella and random walks [12]). 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 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 insertion, deletion, and lookup with hash keys. 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 [21], CAN [16], and Pastry [19] are all *strongly structured*, *i.e.*, they place protocol-specific requirements on overlay connectivity structures. As a result, substrate-aware link-selection enhancements designed for specific DHT protocols [3, 17, 26, 27] cannot benefit other services.

This paper examines the construction of a scalable DHT service on top of a service-independent structure management layer. The rest of this paper is organized as follows. Section 2 describes a low-latency structure management layer that our DHT construction can rely on. We then present the design of a hierarchical DHT protocol that operates on pre-structured overlays in section 3. Section 4 describes evaluation results based on simulations. Our implementation and experimentation on the PlanetLab testbed is reported in section 5. Section 6 discusses related work and section 7 concludes the paper.

2 Service-independent Overlay Structure Management

The proposed DHT construction is based on our earlier design of a common structure management layer, called *Saxons* [20]. The Saxons overlay structure management layer contains six components, as illustrated in Figure 1. 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. Below we briefly describe the low-latency structure quality maintenance component 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 [20].

The quality maintenance component 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 it 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

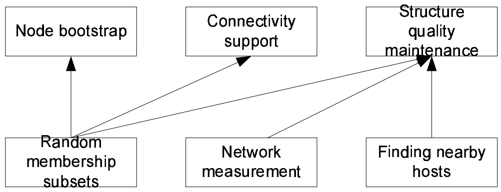

Fig. 1. Saxons components.

links and limit the impact of a node failure on the Saxons structure. The heart of the Saxons structure quality maintenance is a periodic routine that continuously adjusts for potentially better structure quality. Specifically, it checks the distance to hosts in the local random membership subset and replaces the longest existing links if new hosts are closer. 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. To avoid frequent structure changes, we require that a link adjustment occurs only when a new link is shorter than the existing one for more than a specified threshold. It is also possible to disable the quality-oriented structure changes 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.

A critical problem for latency-oriented structure optimization is to efficiently acquire the latency performance data. Previous studies have proposed various techniques for estimating Internet host distances [7, 13] or locating nearby hosts [17]; Saxons can utilize any of them 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 [17]. Using landmark-based latency estimation for structure quality maintenance, link probing is only necessary at node startup to measure latency to a few designated landmark nodes. No additional runtime network overhead would be needed for the structure quality maintenance. Other Saxons components incur a low per-node runtime network overhead of around 1.3Kbps [20].

3 DHT on Pre-structured Overlay Networks

A common structure management layer, such as Saxons, can greatly ease the construction of overlay services with better design modularity and shared structure maintenance cost. In this section, we investigate how distributed hashtable can be constructed based on this layer. Our design considers common DHT service objectives as in other DHT protocols [16, 19, 21]: self-organization and automatic adaptation; scalability in supporting large overlay groups; high-performance DHT lookup; balanced key placement and lookup routing overhead. Note that our key contribution is that the proposed DHT can function on pre-structured overlay networks while the previous solutions require protocol-specific structures.

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 earlier efforts in designing scalable network routing protocols. In particular, we choose to base our design on the Landmark Hierarchy [22][3], due to its potential of self-organization and automatic adaptation to overlay membership changes. Under this context, the load balance objective is especially difficult to achieve because of the hierarchical nature of such scheme.

3.1 Landmark Hierarchy on Overlay Networks

Our concept of Landmark Hierarchy mostly follows that of the original Landmark Hierarchy [22], 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 a routing entry 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 one 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 $rr_H = \infty$.

We call a child-parent relationship exists between nodes 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 original Landmark Hierarchy [22]. Figure 2 illustrates an example of our Landmark Hierarchy and its routing table layout.

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 [10]. 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 larger and we will examine this in the performance evaluation in section 4.3.

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

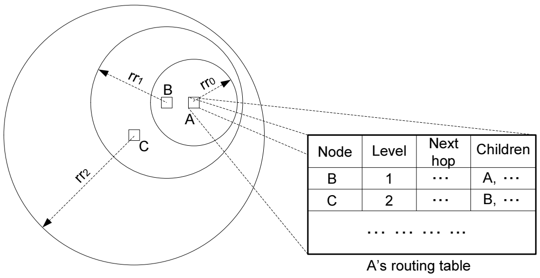

Fig. 2. An exemplar Landmark Hierarchy and its routing table. Nodes A, B, and C are level 0, 1, and 2 landmarks respectively.

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 kept as soft state and are refreshed upon the reception of these advertisements; 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 and increases its landmark level by one. This random delay 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 [11], 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 attempt to avoid 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 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 DHT on Landmark Hierarchy

Based on the constructed landmark hierarchy, we present the design of the proposed DHT protocol 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 is the Chord identifier circle [21]. In Chord, each overlay node and key is assigned an *identifier* in <$0 - ID_{max}$> using an ID assignment function such as SHA-1 or MD5. 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.

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. It is then 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. A disadvantage of 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, resulting in unbalanced key placement. To avoid this problem, we use different key-identifier assignment functions at each level such that keys with close identifiers in one level are spread out in the identifier circles for all other levels. We use MD5$_l()$ to denote the ID assignment function at level l. Equation (1) and Figure 3 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.

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

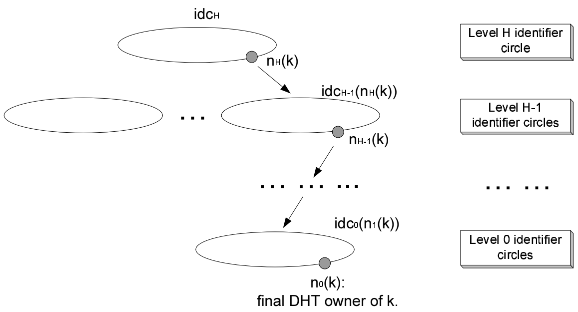

Fig. 3. Illustration of the proposed DHT mapping scheme.

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 $n_H(k)$ in the top level identifier circle. This can be performed locally at every node since the identifiers of 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 $n_{H-1}(k)$ in n_H's children identifier circle. If A has a routing entry for $n_{H-1}(k)$ (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 4 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 for offloading higher-level landmarks in terms of lookup routing overhead.

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 simulations. Section 5 presents experimental results on 51 PlanetLab sites.

4.1 Simulation Methodology and Setup

We use a locally-developed discrete-event simulator that simulates all packet-level events at overlay nodes in our evaluations. 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 overlay link latency. This model does not capture packet

Algorithm 3.1: DHT_LOOKUP(key, l, n_l)

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

// 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;

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

// 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));

Fig. 4. The distributed lookup algorithm in recursive form.

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-US	108	measurement

Table 1. Backbone networks.

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.

The substrate networks we use in the simulations are based on four sets of backbone networks listed in Table 1. First, we use Internet Autonomous Systems maps extracted from BGP routing table dumps, available at NLANR [14] and at the University of Oregon Route Views Archive [18]. Second, we include topologies generated by the Michigan *Inet*-3.0 [24]. We also use some transit-stub topologies generated using the GT-ITM toolkit [25]. 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 NLANR Active Measurement Project (AMP) [15]. The AMP-US network excludes 10 non-U.S. hosts from the full AMP dataset. These 10 hosts have substantially larger latencies to other hosts than others. We use both AMP-all and AMP-US in the evaluation. 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.

In all simulations, the Saxons overlay structure is configured with a node degree range of <$4-16$> and the periodic structure quality maintenance routine runs at 30-second intervals. Except explicitly evaluating the impact of different backbone net-

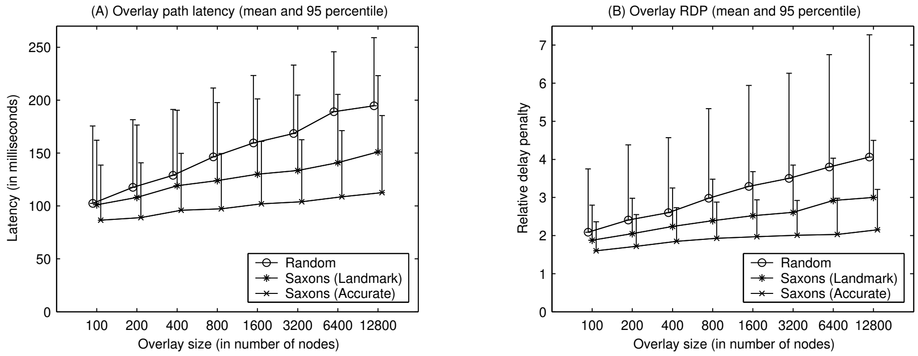

Fig. 5. Structure quality at various overlay sizes.

works, most results shown here are based on the ASmap network. Each data point represents the average value of five runs.

4.2 Overlay Structure Quality

This set of simulations examine the quality of overlay structure constructed using Saxons, upon which our proposed DHT protocol is built. 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)*). Although it might not be practical, 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 (denoted by *Random*).

Figure 5 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. 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. The performance results of the Saxons indicate that it can benefit overlay services built on top of it by providing low-latency overlay structures.

4.3 Statistics on the Landmark Hierarchy

Our proposed DHT is based on a self-organizing landmark hierarchy. In this set of simulations, we further explore the statistics on the landmark hierarchy construction over the Saxons overlay structure. The routing radii (starting from level 0) for the landmark

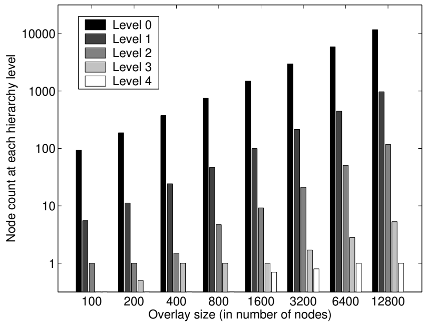

Fig. 6. Node count at each hierarchy level. Y-axis is on the logarithmic scale.

hierarchy are set as 2, 4, 8, 16, 32, 64, $\cdots$ The peer notification radii (starting from level 0) are 2, 5, 9, 17, 33, 65, $\cdots$ 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 6 shows the overlay node count at each hierarchy level for up to 12800 overlay nodes. The results indicate an exponentially larger population at lower hierarchy levels with around ten times more nodes at each immediate lower level.

Figure 7 illustrates the mean routing table size at each hierarchy level. A level-l routing entry at a node corresponds to a level-l landmark whose advertisements are received. 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.

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 8 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 Performance

This section investigates the performance of our proposed DHT protocol on the Saxons structure management layer (denoted by *SaxonsDHT*). We examine the DHT performance in terms of lookup latency, fault tolerance, the balance of key placement and lookup routing overhead. We assess SaxonsDHT performance in relation to that of Chord [21], a well-known DHT protocol. A previous study [8] shows that Chord performs competitively against other strongly-structured DHT protocols such as CAN [16] and Pastry [19] in terms of lookup latency and load balance. We implemented the SaxonsDHT and Chord protocols in our simulator and both schemes are configured at the same link density in our evaluation. For SaxonsDHT, the node degree range of <$4-16$>

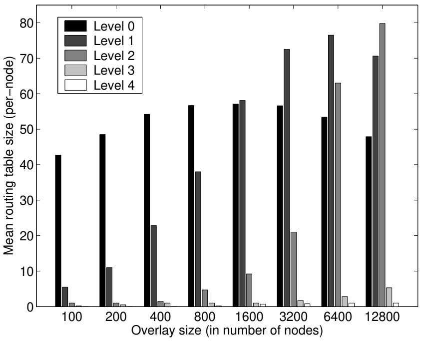

Fig. 7. Mean routing table size at each hierarchy level.

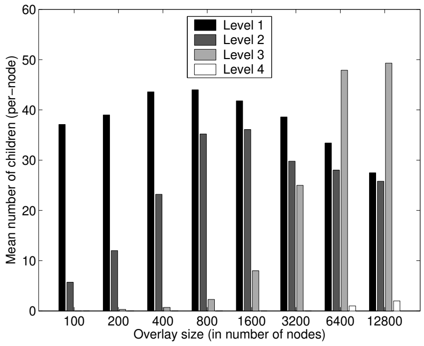

Fig. 8. Mean number of children at each hierarchy level.

results in an average degree of 8. For Chord, each node maintains an 8-entry finger table supporting DHT lookups. Higher-level finger entries in Chord point to nodes with exponentially larger distances in the identifier circle.

It should be noted that the purpose of our evaluation is to assess the performance of SaxonsDHT. We do not intend to make claim on its performance superiority over strongly structured DHT protocols. In particular, the comparison between SaxonsDHT and Chord at the same link density is not strictly fair for at least two reasons. First, Chord can freely structure the overlay network in order to achieve the best performance while SaxonsDHT has to function on top of a service-independent structure. On the other hand, SaxonsDHT requires a fairly large routing table at each node while Chord's finger table size is the same as the number of outgoing links. Further, recent enhancements on Chord have considered the latency of fetching DHT data in addition to the lookup latency [6]. We do not consider the data fetch latency in this paper.

DHT Lookup Latency. The performance metrics for the DHT lookup include both lookup latency and hop-counts. Figure 9(A) illustrates the DHT lookup hop-count for SaxonsDHT and Chord at various overlay sizes. For each configuration, we measure the average performance of 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 9(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 9(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 SaxonsDHT protocol. We also examine variations of the Chord protocol with a substrate-aware link-selection enhancement, called *proximity neighbor selection* [6]. In this enhancement, instead of simply picking the first node in each finger table entry's

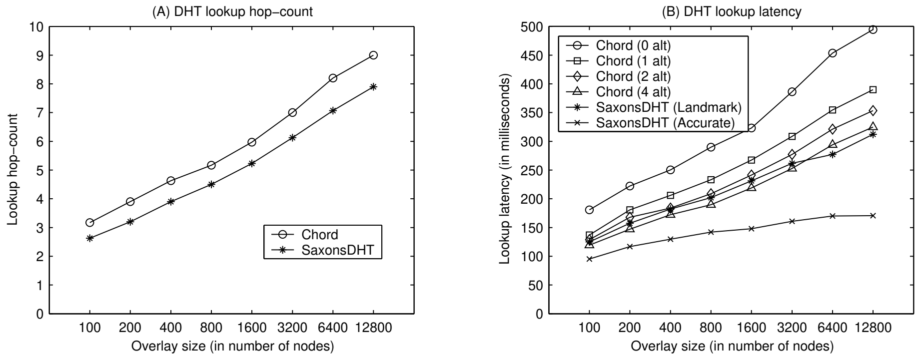

Fig. 9. DHT lookup performance at various overlay sizes.

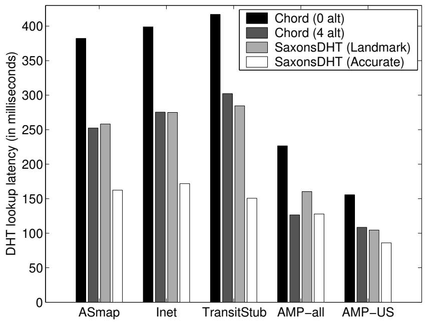

Fig. 10. DHT lookup latency over different backbone networks.

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 9(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 10 shows the DHT lookup latency over different backbone networks. Results indicate that the performance difference is not significantly affected by the choice of backbone networks. Savings are smaller for the two measurement-based backbones due to their small sizes.

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.

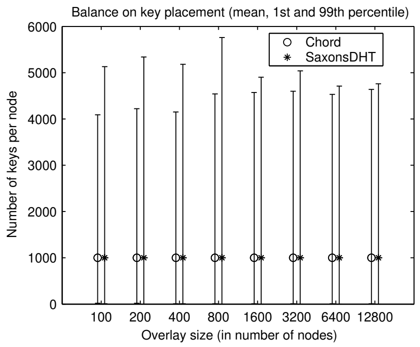

Fig. 11. DHT load balance on key placement.

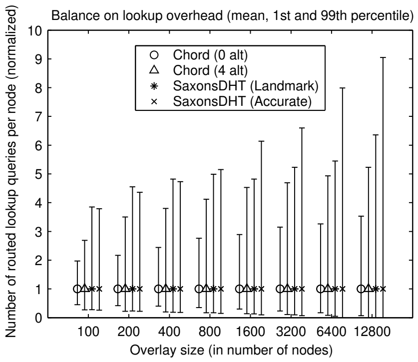

Fig. 12. DHT load balance on lookup routing overhead.

Figure 11 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. Results show that the balance of key placement for SaxonsDHT is close to that of Chord. We do not show results for variations of the SaxonsDHT and Chord protocols because they do not have significant impact on the balance of key placement.

Figure 12 shows the DHT load balance in terms of the lookup routing overhead. Note that the results in Figure 12 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.

DHT Performance under Unstable Environments. Figure 13 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 13 show that SaxonsDHT and Chord deliver similar lookup success rate during overlay membership changes. Following an argument in [21], 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

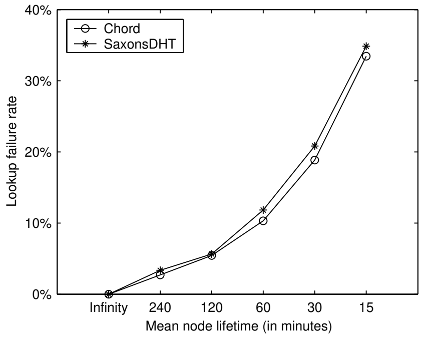

Fig. 13. SaxonsDHT lookup failure rate with biased node failures.

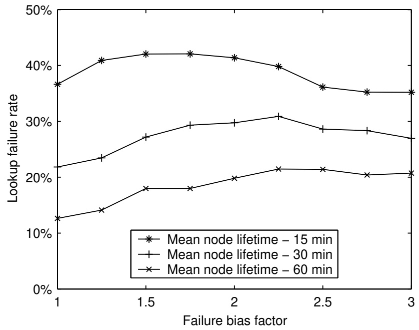

Fig. 14. SaxonsDHT lookup failure rate with biased node failures.

nodes at higher hierarchy levels have shorter lifetime. To quantify such scenarios, we introduce the *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 14 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 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.

Due to its structure-sensitive DHT mapping scheme, SaxonsDHT tends to produce more key reassignments after overlay membership changes. Figure 15 shows the proportion of key assignment changes of SaxonsDHT and Chord with certain number of random node joins and leaves in 3200-node overlays. We consider the SaxonsDHT performance with varying failure bias factors to model the impact of targeted failures of high-level landmarks. Comparing to Chord, SaxonsDHT produces two to three times key reassignments (186% more in average) after overlay membership changes. Targeted failures of high-level landmarks may result in more key reassignments. Such a performance difference suggests that our DHT design may be better suited for applications that do not require large data migration after the DHT mapping changes, such as those that involve time-sensitive information or require repetitive refreshment. We should point out that the DHT mapping in Chord is based on *consistent hashing*, a technique with provable minimal key reassignments after membership changes [9].

5 Internet Experimentation

The implementation of our proposed DHT design is based on a Saxons prototype [20]. In our DHT implementation, every node maintains a TCP connection with each of its overlay neighbors. The route advertisements and lookup queries flow through these TCP connections on the overlay structure. The DHT service periodically queries the

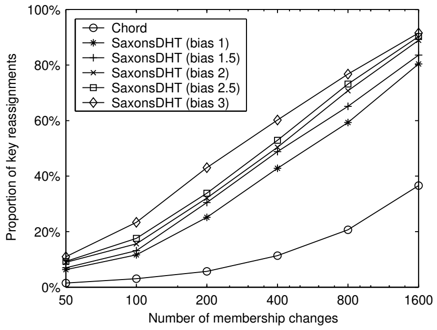

Fig. 15. Key reassignments due to overlay membership changes.

Saxons kernel for up-to-date overlay structure information. Link-related state such as the TCP connections to neighbors and 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 and lookup queries flow through these connections. For both DHT implementations, node IDs are assigned using MD5-hashed IP addresses.

We conducted experiments on the PlanetLab testbed [2] to evaluate the performance of the proposed DHT service. Our experiments involve 51 PlanetLab nodes, all from unique wide-area sites. Among them, 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. Specifically, a node pings the other 10 times and measure the round-trip time. It then takes the average of the median 6 measurement results. This scheme is close to *Saxons (Accurate)* examined in the simulation studies.

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 $<2-8>$, and consequently an average node degree of 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.

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 16 illustrates the cumulative distribution functions of the 51,000 DHT lookup latency measurements taken out of a typical test run. We observe that the lookup latency of RandomDHT is close to that

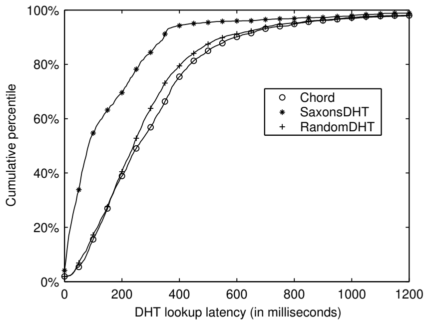

Fig. 16. DHT lookup latency CDFs on 51 PlanetLab sites.

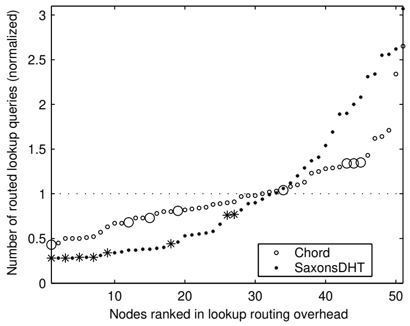

Fig. 17. Load balance of DHT lookup routing overhead on 51 PlanetLab sites. Larger markers represent non-U.S. sites.

of Chord while SaxonsDHT significantly outperforms them. In average, SaxonsDHT achieves about 48% latency reduction compared with Chord (335.5ms *vs.* 643.0ms).

We also examine the DHT load balance on lookup routing overhead for SaxonsDHT and Chord. Figure 17 shows the number (normalized to the mean value over all nodes) of routed lookup queries over each node. Results in the figure are increasingly ranked and larger markers 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.

6 Related Work

Previously proposed scalable DHT protocols such as Chord [21], CAN [16], and Pastry [19] all function on protocol-specific overlay structures to support DHT lookups. A recent work [8] 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. Several 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 [27]. Ratnasamy *et al.* introduced a distributed binning scheme for CAN such that the overlay topology resembles the underlying IP network [17]. In Mithos [23], 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 [26]. 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 [10]. Landmark Hierarchy was later introduced by Tsuchiya to allow minimal administration overhead and automatic adaptation to dynamic networks [22]. Recent studies (SCOUT [11] and L^+ [4]) 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.

7 Conclusion

This paper presents 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 [20]. Compared with Chord [21] at the same overlay link density[4], simulations and Internet experiments 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 its hierarchical nature. Due to the structure-sensitive DHT mapping scheme, however, the proposed approach may produce significantly more key reassignments at high node churn rates.

Overall, our 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 and unicast/multicast path selections, our work is the first to examine its applicability on the distributed hashtable service.

Acknowledgment: We would like to thank anonymous referees for their valuable comments.

References

1. D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. of SOSP*, pages 131–145, Banff, Canada, Oct. 2001.
2. A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *Proc. of NSDI*, pages 253–266, San Francisco, CA, Mar. 2004.
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.

[4] Note that at the same link density, our DHT protocol requires a larger routing table compared with Chord's finger table.

6. F. Dabek, J. Li, E. Sit, J. Robertson, M. F. Kaashoek, and R. Morris. Designing a DHT for Low Latency and High Throughput. In *Proc. of NSDI*, pages 85–98, San Francisco, CA, Mar. 2004.
7. P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz, and Y. Jin. An Architecture for a Global Internet Host Distance Estimation Service. In *Proc. of INFOCOM*, New York, NY, Mar. 1999.
8. S. Jain, R. Mahajan, and D. Wetherall. A Study of the Performance Potential of DHT-based Overlays. In *Proc. of USITS*, Seattle, WA, Mar. 2003.
9. 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.
10. L. Kleinrock and F. Kamoun. Hierarchical Routing for Large Networks. *Computer Networks*, 1:155–174, 1977.
11. S. Kumar, C. Alaettinoglu, and D. Estrin. SCalable Object-tracking Through Unattended Techniques (SCOUT). In *Proc. of ICNP*, Osaka, Japan, Nov. 2000.
12. 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.
13. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-based Approaches. In *Proc. of INFOCOM*, New York, NY, June 2002.
14. BGP Routing Data at the National Laboratory for Applied Network Research. http://moat.nlanr.net/Routing/rawdata.
15. Active Measurement Project at the National Laboratory for Applied Network Research. http://amp.nlanr.net.
16. 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, Aug. 2001.
17. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-Aware Overlay Construction and Server Selection. In *Proc. of INFOCOM*, New York, NY, June 2002.
18. University of Oregon Route Views Archive Project. http://archive.routeviews.org.
19. 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, Nov. 2001.
20. K. Shen. Structure Management for Scalable Overlay Service Construction. In *Proc. of NSDI*, pages 281–294, San Francisco, CA, Mar. 2004.
21. I. Stoica, R. Morris, D. Karger, M. F. 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, Aug. 2001.
22. P. F. Tsuchiya. The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks. In *Proc. of SIGCOMM*, pages 35–42, Stanford, CA, Aug. 1988.
23. M. Waldvogel and R. Rinaldi. Efficient Topology-Aware Overlay Network. In *Proc. of the HotNets Workshop*, Princeton, NJ, Oct. 2002.
24. J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, Dept. of EECS, University of Michigan, 2002.
25. E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. of INFOCOM*, San Francisco, CA, Mar. 1996.
26. 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.
27. 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, Mar. 2002.