Structure Management for Scalable Overlay Service Construction

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

Abstract

This paper explores the model of providing a *common* overlay structure management layer to assist the construction of large-scale wide-area Internet services. To this end, we propose *Saxons*, a distributed software layer that dynamically maintains a selected set of overlay links for a group of nodes. Saxons maintains high-quality overlay structures with three performance objectives: low path latency, low hop-count distance, and high path bandwidth. Additionally, it provides partition repair support for the overlay structure. Saxons targets large self-organizing services with high scalability and stability requirements. Services can directly utilize the Saxons structure for overlay communication. Saxons can also benefit unicast or multicast overlay path selection services by providing them a small link selection base without hurting their performance potential.

Our simulations and experiments on 55 PlanetLab sites demonstrate Saxons's structure quality and the performance of Saxons-based service construction. In particular, a simple overlay multicast service built on Saxons provides near-loss-free data delivery to 4 times more multicast receivers compared with the same multicast service running on random overlay structures. Our experiments also show that this performance is close to that of direct Internet unicast without simultaneous traffic.

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, overlay services may suffer poor performance when their designs 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, 29], measurement-based end-system multicast protocols [2, 7, 17], and recent efforts to add substrate-awareness into the scalable distributed hash table (DHT) protocols [6, 27, 35].

In this paper, we present the design and implementation of a distributed software layer providing *S*ubstrate-*A*ware *C*onnectivity *S*upport for *O*verlay *N*etwork *S*ervices, or *Saxons*. Saxons constructs and maintains overlay connectivity structures with qualities such as low overlay latency, low hop-count distance, and high overlay bandwidth. Through a very simple API, service instances at overlay nodes can query the locally attached overlay links in the structure. Overlay services can directly use Saxons structure links for communication. They may also further select overlay links from the given structure based on application-specific quality requirements or service semantics.

Many popular overlay services, such as Gnutella, are *unstructured* in that they are designed to operate on any overlay network structures. Saxons works naturally with these services by providing them a high quality connectivity structure for overlay communication. Saxons can also benefit unicast [1] or multicast overlay path selection services [7] by providing them a small link selection base. These services would normally incur much higher overhead if they directly run on the completely connected overlay, which often limits their scalability [1, 7]. The Saxons overlay structure can be configured with different link density to offer tradeoff between overlay complexity and path redundancy or quality. A more connected Saxons structure would provide better *achievable* overlay quality in terms of path latency, hop-count distance, or bandwidth. However, it typically consumes more resources to find the optimal path in a structure with higher link density.

It should be noted that a general-purpose overlay structure layer cannot be easily integrated with strongly structured protocols where the protocol semantics dictates how overlay nodes should be connected. Prominent examples are the recently proposed scalable DHT protocols [26, 31]. However, we believe that the "strongly structured" nature of these protocols are not inherent to the DHT service itself. For instance, we have demonstrated that a scalable DHT service can be constructed on pre-structured overlay networks [30].

We should also emphasize that the primary goal of this work is to provide a general easy-to-use software layer with wide applicability. Achieving optimal performance for individual services is not our focus. The rest of this paper is organized as follows. Section 2 describes our design objectives and theirs rationale. Section 3 presents the design and analysis of the Saxons overlay structure management layer. Section 4 illustrates our simulation-based evaluation results. Section 5 and 6 describe a prototype Saxons implementation and service constructions on the PlanetLab testbed. Section 7 discusses related work and Section 8 concludes the paper.

2 Design Objectives

Saxons is designed to provide efficient overlay connectivity support that can assist the construction of large-scale Internet overlay services. In order to assemble a comprehensive set of performance goals for the Saxons structure management layer, we first examine existing overlay services and categorize them based on similar communication patterns. Then we describe our objectives on overlay structure qualities and how they support our targeted services. We will also describe other Saxons design objectives.

2.1 Targeted Services

Below we list five categories of overlay communication patterns and some example services in each category. We do not make claims on the completeness of this list but we believe it should cover the communication patterns of a large number of overlay services.

Short unicast: Services of this type involve single-source singe-destination short messages delivered along the overlay structure. An example of such services is the index-assisted object lookup [8], where the object lookup query messages are routed in the overlay network based on heuristics maintained at each node.

Short multicast/broadcast: Services of this type involve single-source multiple-destination short messages delivered along the overlay structure. An example of such services is the query flooding-style object lookup, such as Gnutella.

Long unicast: Services of this type involve a large amount of data delivered between two nodes in the overlay network. The large data volume makes it important to find efficient data delivery paths. Since the underlying substrate network (*i.e.*, the Internet) provides native unicast transport service, an overlay unicast delivery optimized for short latency only makes sense when the triangular inequality on Internet latencies does not hold, *e.g.*, due to the policy-influenced BGP routing or transient outages [1, 29].

Long multicast/broadcast: Due to the slow adoption of native multicast support in the Internet, there have been a large number of recently proposed overlay multicast protocols. In these protocols, large amount of streaming data is delivered between a single source and multiple destinations in the overlay network. These protocols focus on constructing multicast data distribution trees optimized for low latency [7], high bandwidth [17], or both.

Periodic pairwise neighbor communication: This communication pattern is frequently used for soft state maintenance in the overlay network, such as membership management [7], routing table maintenance, and Web proxy cache index maintenance. In the case of Web proxy caching, a large number of cooperative caches can form an overlay network connected by an efficient structure. Proxy caches maintain the information about each other's content through periodic pairwise neighbor communication. Such a scheme may consume less network bandwidth than broadcast-based cache sharing protocols [11] do.

2.2 Objectives on Overlay Structure Qualities

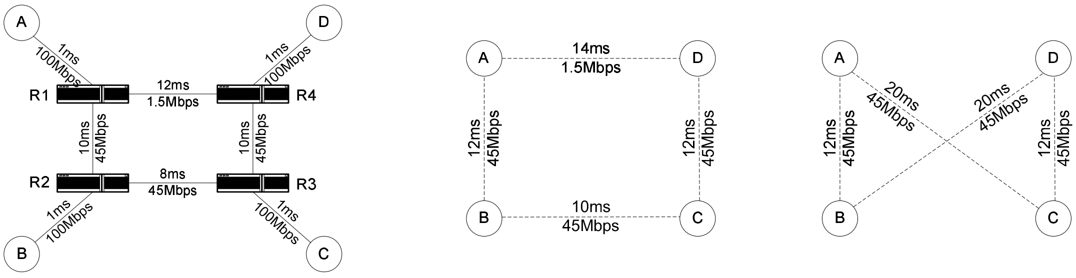
Saxons assists the construction of large-scale Internet overlay services by providing a high quality connectivity structure. The Saxons structure can be configured with different link density to offer tradeoff between overlay complexity and path redundancy. The link density also affects the achievable overlay quality. Such a density is determined through a configurable node degree range[1] in Saxons. Below we discuss performance objectives for the Saxons structure along three lines: the overlay latency, hop-count distance, and the overlay bandwidth.

The first performance objective for the Saxons structure construction is to achieve low *overlay path latency*, defined as the end-to-end latency along the shortest overlay path for each pair of nodes. The *relative delay penalty* (or *RDP*) is another common metric for overlay latency, which is defined as the ratio of the overlay path latency to the direct Internet latency. Consider a physical network illustrated in Figure 1(a). A, B, C, and D are edge nodes while R1, R2, R3, and R4 are internal routers. The latency and bandwidth for each physical link are indicated. Figures 1(b) and 1(c) illustrate two overlay connectivity structures both of which have a per-node degree of two. The latency and bandwidth of each virtual overlay link are also indicated with the assumption that the underlying substrate network employs shortest-path routing. Both connectivity structures provide the same overlay latency for A-B and C-D. Structure II allows slightly less latency for A-C and B-D while structure I provides much shorter paths for A-D and B-C. In average, Structure I is superior to Structure II in terms of the overlay path latency (15.33ms *vs.* 21.33ms) and RDP (1.03 *vs.* 1.58). The overlay latency is very important for the first four categories of services described in Section 2.1. For services with the periodic pairwise neighbor communication pattern, the data propagation delay between two nodes depends mostly on the overlay hop-count distance since the neighbor communication frequency is often independent of the link latency.

Our second performance objective is to achieve low *overlay hop-count distance*, defined as the hop-count distance along the overlay structure for each pair of nodes. The two structures illustrated in Figure 1 have the same average overlay hop-count distance of 1.33. As mentioned earlier, the hop-count distance in an overlay structure is important for services with the periodic pairwise neighbor communication pattern. We further argue that this metric also affects the performance of other services utilizing overlay structures. This is because an overlay route with larger hop-count may suffer more performance penalty in the presence of transient congestion or faults at intermediate nodes.

In terms of the overlay bandwidth, we are interested in two specific metrics: *bandwidth along the shortest overlay path* and *bandwidth along the widest path*. Bandwidth along the shortest overlay path is important when the path selection desires both low latency and high bandwidth. Band-

[1] The node degree is defined as the number of links attached to the node in question.



(a) Physical network (b) Overlay structure I (c) Overlay structure II

Figure 1: An example to illustrate overlay structure qualities.

Service types	Overlay latency	Hop-count distance	Overlay bandwidth
Short unicast	✓	✓$^{\times}$	×
Short multi/broadcast	✓	✓$^{\times}$	✓$^{\times}$
Long unicast	✓	✓$^{\times}$	✓
Long multi/broadcast	✓	✓$^{\times}$	✓
Neighbor comm.	×	✓	✓

Table 1: Importance of structure qualities for overlay services. A check mark "✓" means the metric is important to this type of services while a "×" represents the opposite. A mark "✓$^{\times}$" means the metric may be important under certain circumstances.

width along the widest path is the essential metric when the upper-level service is predominantly interested in finding high-bandwidth overlay routes. For the two overlay structures illustrated in Figure 1, structure II provides high bandwidth (45Mbps) routes for all pairs of nodes while structure I has low bandwidth (1.5Mbps) along the shortest path for A-D. High overlay bandwidth is desired for long unicast/multicast/broadcast communications as well as periodic pairwise neighbor communications. Services with short-message communications generally do not place much demand on overlay bandwidth. However, short multicast/broadcast services like query flooding may incur high bandwidth consumption when a large number of queries flood the system simultaneously.

In summary, Table 1 shows these performance metrics and their importance to each of the five types of services we listed in Section 2.1. We can see that services with different communication patterns place different quality demand on the overlay structure construction. We also believe that the above three lines of performance objectives cover the quality demand of a wide range of overlay services.

2.3 Other Design Objectives

In addition to constructing high quality overlay structures, we describe below several additional design objectives for Saxons. 1) **Scalability**: Recent measurements show that the main Gnutella network contains more than 100,000 nodes [19]. In order to support large-scale services, it may be infeasible to maintain the complete system view at any single node. 2) **Connectivity**: The fault or departure of a few bridging nodes may cause the partition of remaining overlay nodes, which could degrade or even paralyze the overlay service. Saxons provides partition detection and repair support such that upper-level services do not need to worry about the overlay partitioning. 3) **Stability**: Frequent node joins and leaves in a self-organizing system may cause instability in the overlay structure. Saxons is designed to hide such instability from upper-level services by maintaining the overlay structure quality with infrequent link adjustments. Structure stability is especially important for the performance of services that maintain link-related state, such as many routing services.

3 System Design

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 2 illustrates the six Saxons components and their relationship. Note that the connectivity support is an optional component in Saxons, which could be turned off for structures that are unlikely to be partitioned (*e.g.*, those configured with a high link density).

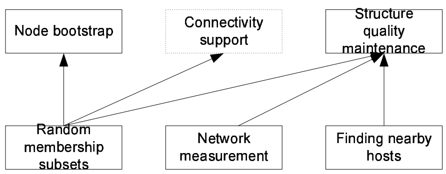


Figure 2: Saxons components.

The scalability of Saxons is achieved by controlling the system management overhead when the overlay scales up. The key guideline of our design is that the per-node management cost should only depend on the number of directly attached overlay links, not on the overlay size. In order to maintain a high level of robustness, Saxons employs a non-hierarchical or functionally symmetric architecture (with the exception of the optional Saxons connectivity support). Non-hierarchical designs are inherently free of scaling bottlenecks and they exhibit strong robustness in the face of random failures or even intentional attacks.

Our design and analysis in this paper assumes a fail-stop node departure model in which the departing node abruptly stops all actions at the exit time. We also assume there is no malicious participant in the overlay.

3.1 Random Membership Subsets

Accurately tracking the complete list of group members often requires the dissemination of such complete list among overlay nodes [7]. Per-node bandwidth consumption and storage requirement for such dissemination grow linearly with the increase of the overlay size, which can be problematic for large-scale services. In comparison, the Saxons structure management layer maintains periodically changing random membership subsets, which can satisfy many of the membership service needs with controllable bandwidth consumption and storage requirement.

In Saxons, each node maintains a dynamically changing *random-subset* data structure containing a number of other overlay members. The size limit of the random-subset (denoted by s) is determined according to the allowed per-node storage consumption. Membership subset queries are fulfilled through random selection from the local random-subset. Each node periodically disseminates a certain number of overlay members to each of its neighbors for updating the recipient's random-subset. The membership update size (denoted by k) and frequency are determined such that the bandwidth consumption is properly controlled. When random-subsets have reached their size limit s, randomly chosen old members are replaced by new members from membership updates.

The key for providing a random membership subset service with uniform representation over all overlay participants is to ensure such uniform representation in membership update sets. In our scheme, the update set from a node (called A) contains a selected portion of A's random-subset with uniform selection probability. A itself may also be included in each update set at probability k/n (n is the overlay size) to ensure its own equal representation. Since the overlay size n is often unknown at runtime, we use an approximate value $\tilde{n}_A = s/p$, where p is the proportion of all disseminated members received at A that are already in A's local random-subset at the time of receipt. Such an approximation is accurate when disseminated members received at A uniformly represent all overlay participants.

We provide simulation results to demonstrate the level of uniform randomness achieved by the Saxons random membership subset component. Simulations were run on a 6400-node randomly connected overlay structure with an average node degree of 8 and a degree upper-bound of 16. In the simulations, all the random-subsets are empty at the beginning of round 0. The length of a round is the average update interval between each pair of overlay neighbors. Each membership update contains 20 members in the simulations. Figure 3 illustrates the growth of the accumulated number of overlay members learned through membership updates (averaged over all 6400 nodes) since the overlay startup. Note that this metric is not equivalent to the number of members in each node's random-subset. It is a hypothetic metric whose growth over time illustrates the uniform randomness of membership updates. Results for several random-subset sizes are compared with the result for the uniformly random updates, which is produced when all membership updates contain uniformly random overlay nodes. The intuition for this comparison is that a membership update scheme that contains larger positive correlation among its update sets would generate slower growth in the number of learned overlay members.

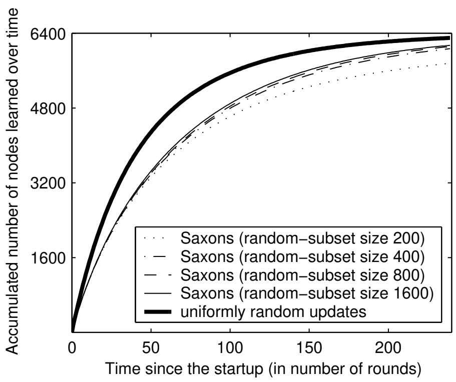


Figure 3: Growth of the accumulated number of overlay members learned from membership updates (averaged over all nodes) in a 6400-node overlay.

Results in Figure 3 show that the Saxons random membership subset component performs very close to the ideal uniform randomness. We also observe that the result is not very sensitive to the random-subset size, though larger random-subsets exhibit slightly higher uniform randomness. This is because each random-subset serves as a buffer to screen out non-uniformity in incoming membership updates and larger random-subsets are more effective on this.

Kostić *et al.* recently proposed a random membership subset protocol for tree-shaped overlay structures [18]. In comparison, the Saxons membership component supports more general mesh-like overlay structures. Our membership management also differs from gossip-based dissemination protocols (*e.g.*, *lpbcast* [10]) in that disseminations in Saxons follow the overlay structure links. This restriction allows dissemination messages to flow through pre-established TCP connections along Saxons structure links.

3.2 Network Measurement

A fundamental problem for substrate-aware overlay service construction is how to acquire network latency and bandwidth data to support efficient overlay service construction. For latency measurement between two nodes, we simply let one node ping the other N_l times and measure the round-trip times. We remove the top 20% and bottom 20% of the measurement results and take the average of the remaining values. Pings could be conducted using ICMP

ECHO messages or by employing a user-level measurement daemon responding to ping requests at each host.

Bandwidth measurement requires more consideration because it is harder to get stable results and it consumes much more network resources. Since the measurements would be conducted repeatedly in the runtime due to system dynamics, our goal is to acquire sufficiently accurate measurement data at a moderate overhead. The bandwidth measurement scheme we use is derived from the *packet bunch* technique proposed by Carter and Crovella [5] as well as Paxson [24]. Specifically, when node A wants to measure the bandwidth from node B, it sends out a UDP request to the measurement daemon at B, which replies back N_b UDP messages at the size of S_b each. A then records the receipt times of the first and the last messages (denoted by t_{first} and t_{last}). Note that A may not receive all messages due to congestion and drops at buffer queues. Assume A actually received $\tilde{N}_b$ messages, we determine the link bandwidth as $(\tilde{N}_b - 1) * S_b/(t_{last} - t_{first})$. In order to avoid transient network congestions, we repeat the tests three times with a random interval between 2 and 6 seconds and take the median value from the three rounds as the final result.

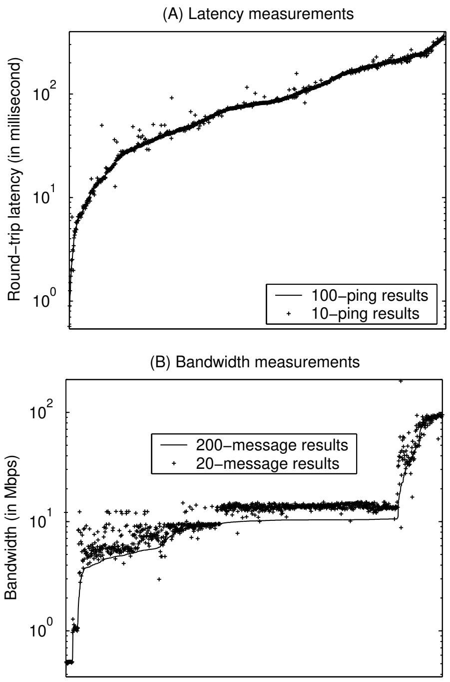


Figure 4: Network measurement results for all-to-all site pairs on 61 PlanetLab sites. Y-axises are in the log scale.

In practice, we use 10 pings for latency measurements (*i.e.*, N_l=10) and we use 20 UDP messages of 8KB for each of the three bandwidth measurement rounds (*i.e.*, N_b=20 and S_b=8KB). Each bandwidth test costs 480KB at this setting. We conducted experiments on the PlanetLab testbed [25] to assess the effectiveness of our network measurement schemes. In the experiments, we compare our measurement results at the above mentioned setting with results using 10 times more messages. Figure 4 illustrates the measurement results for all-to-all node pairs between 61 PlanetLab nodes, all from unique wide-area sites. For both latency and bandwidth measurements, the results are ranked in ascending order for the more accurate measurements that use 10 times more messages. Figure 4(A) shows that the latency measurement with 10 pings are already very accurate. From Figure 4(B), we notice that a large number of site pairs have 10Mbps bandwidth between them. It turns out that many PlanetLab nodes are equipped with the *Hierarchical Token Bucket* filter [9] that limits the per-user outgoing bandwidth at 10Mbps [4]. Our measurements give slightly higher bandwidth estimates for these links. It appears the reason is that these filters let go about 64KB data before the rate control kicks in.

We should point out that almost any network measurement techniques can be used in Saxons. And different schemes may be better suited for different network environments. For instance, the behavior of the Hierarchical Token Bucket filter requires the bandwidth measurement to use much larger than 64KB data for being effective.

3.3 Finding Nearby Hosts

In addition to network performance measurement, finding nearby hosts is also needed by the Saxons overlay structure management. Accuracy, scalability, and ease of deployment are some important issues for this component. Previous studies have proposed various techniques for locating nearby hosts [14, 15, 21, 27], most of which require infrastructure support or established landmark hosts. Saxons can utilize any of the existing techniques in principle. For ease of deployment, we introduce a random sampling approach that does not require any infrastructure support or landmark hosts.

The basic idea of random sampling, or *Rsampling*, is to randomly test the network latency to f_{rs} (or the random sampling factor) nodes from the overlay group and picks the one with shortest latency. The overhead of this approach can be controlled by choosing a small f_{rs}. The performance of Rsampling is not directly competitive to more sophisticated landmark-based schemes. However, in addition to its advantage of ease of deployment, it has the property of converging to the closest host when running repeatedly. We compare Rsampling with the *landmark-based Cartesian distance* approach for locating nearby hosts. 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 other landmark-based schemes [27].

Figure 5 illustrates the simulation performance of latency estimation schemes. The backbone substrate network used in this experiment is based on a 3104-node Internet *Autonomous Systems* map available at NLANR [22]. More details for this network map and the simulation setup will be described in Section 4.1. The metric *latency stretch* in Figure 5 is defined as the ratio of the latency to the selected

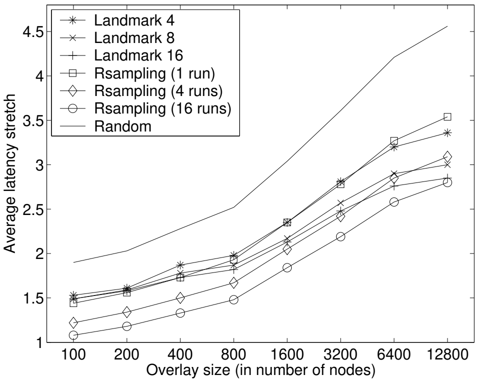


Figure 5: Performance of schemes for finding nearby hosts.

host to the latency to the optimal (*i.e.*, the closest) host. The Rsampling approach tests the network latency to four randomly selected nodes at each run and the old selection is replaced if a newly tested node is closer. The performance for Rsampling after 1, 4, and 16 runs are shown, compared with the performance of the landmark approach with 4, 8, and 16 landmark nodes. We observe that the Rsampling performance after 4 runs is competitive to the landmark approach even for 12800-node overlays. This performance is achieved with only a total of $4 \times 4 = 16$ latency tests.

Note that the random sampling technique could also be used for finding hosts with high bandwidth connections. The main difference is that bandwidth measurements cannot be conducted frequently because they are much more expensive than latency tests.

3.4 Node Bootstrap

When a node joins the Saxons connectivity structure, it must first know at least one active bootstrap node through out-of-band means. In principle, every active overlay member can serve as a bootstrap node. However, employing a small number of bootstrap nodes for each overlay group allows the use of a DNS-like naming system to locate them. Since joining nodes do not establish direct links to bootstrap nodes in our scheme (described later), it is feasible to employ a small number of bootstrap nodes as long as they are not overloaded with processing bootstrap requests and a desired level of availability can be provided.

During bootstrap, the joining node first contacts an bootstrap node to acquire a list of nodes randomly selected from the bootstrap node's local random-subset. The joining node then attempts to establish links with d_a nodes in the list. Link establishment attempts may fail because a target node may have departed from the system or have reached its degree bound. Further attempts may be needed to complete link establishments. As soon as the initial links are established, the joining node starts the random membership subset component to learn the existence of other nodes and also makes itself known to others. Periodic structure quality maintenance and connectivity management routines are also scheduled after the bootstrap.

3.5 Structure Quality Maintenance

A main goal of the Saxons structure management is to continuously maintain a high-quality overlay structure connecting member nodes. The structure quality is determined along three lines: low overlay latency, low hop-count distance, and high overlay bandwidth. The structure management component runs at a certain link density, specified by a node degree range $<d_a - d_t>$. Each node can initiate the establishment of d_a overlay links (called *active* links) and each node also passively accepts a number of link establishments (called *passive* links) as long as the total degree does not exceeds 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. Below we consider several different approaches to maintain the structure quality. In all cases, a routine is run periodically to adjust active links for potentially better structure quality.

The overlay structure quality maintenance has been studied in the context of end-system multicast. The Narada protocol maintains a low-latency structure through greedily maximizing a latency-oriented utility value [7]. This approach requires a complete membership view at each node and a full-scale shortest-path routing protocol running on the overlay network, which may not be feasible for large-scale services. We include a more scalable latency-only approach in our study. This approach, called *AllShort*, continuously adjusts active links to connect to closest hosts it could find. More specifically, it utilizes the random sampling policy to measure the latency to a few randomly selected hosts and replace the longest existing active links if new hosts are closer. Note that such link adjustments must not violate rules in the Saxons connectivity support described in the next section.

Latency-only protocols like AllShort tend to create mesh structures with large hop-count distances. Consider a two-dimensional space with uniform node density and assume the network latency is proportional to the Cartesian distance. Let n be the total number of nodes and assume the node degree is bounded by a constant. Latency-only protocols would create grid-like structures with the hop-count diameter of $O(\sqrt{n})$, much larger than the $O(ln(n))$ diameter for randomly connected structures [3]. A straightforward idea is to add some random links into the structure to reduce the overlay hop-count distance. The second approach we consider, called *ShortLong*, was proposed by Ratnasamy *et al* [27]. In this approach, each node picks $d_a/2$ neighbors closest to itself and chooses the other $d_a/2$ neighbors at random (called *long links*).

However, neither of the above schemes considers the overlay bandwidth. To this end, we propose the third approach, called *ShortWide*, that also optimizes the overlay structure for high bandwidth. In this approach, half of the active links are still maintained for connecting to closest hosts each node could find. The other half are connected to randomly chosen hosts with high bandwidth (we call these *wide links*). The wide links are also maintained using random sampling, although at a much lower adjustment frequency and higher adjustment threshold than latency-oriented link adjustments. These are made necessary by the high overhead and inaccuracy of bandwidth measurements. Additionally, the high adjustment threshold preserves a large amount of *randomness* in the overlay structure, which is important for achieving low overlay hop-count distance.

It should be noted that latency and bandwidth measurements may not be always accurate. In order to avoid link oscillations, we require that a link adjustment occurs only when the new link is shorter or wider than the existing overlay link for more than a specified threshold.

3.6 Connectivity Support

In large-scale self-organizing overlay services, the fault or departure of a few *bridging* nodes may cause the partition of remaining nodes. Without careful consideration, the structure quality maintenance may also create overlay partition by cutting some *bridging* links. Saxons provides overlay connectivity support that actively checks the overlay connectivity and repairs partitions when they occur.

The Saxons connectivity support is based on periodic broadcasts of sequenced connectivity messages from a core node C. Let t_{int} be the interval between consecutive broadcasts. The connectivity messages flood the network along the overlay links. Node A detects a possible partition when the connectivity message is not heard for $t_{int} + t_{A,C}$, where $t_{A,C}$ is A's estimate of the message propagation delay upper-bound from C. When a partition is detected, A schedules a repair procedure at a random delay chosen uniformly from $[D_l t_{int},\ D_u t_{int}]$. In the repair procedure, the node randomly picks another node from its local random-subset and attempts to establish a partition repair link. This procedure may fail because the contacted node may have reached its degree bound, be disconnected too, or have departed from the system. The partition repair procedure is continuously rescheduled at random delays until the connectivity messages are heard again. While it is always possible to reconnect to the network by directly establishing a link to the core node, this should be avoided since the core could be inundated with such requests. Nondeterministic delays in scheduling repair procedures are important to avoid all nodes in the partitioned network initiate such repair simultaneously. In many cases, successful repair at a single node can bring the network completely connected again. This is similar to the prevention of response implosion in the SRM reliable multicast protocol [13].

In addition to partition repairs, Saxons also tries to avoid partitioning caused by link adjustments of the structure quality maintenance. This is achieved by having each node remember its upstream link to the core, defined as the link through which the connectivity message bearing the highest sequence number first arrived. The structure quality maintenance can avoid causing overlay partition by preserving the upstream link to the core.

The availability of the core node is critical to the Saxons connectivity support. In the case of the core node failure or physical disconnection from the network, no overlay nodes would succeed in its regular repair procedure. At several repeated failures, a node will ping the core node to determine whether a physical disconnection occurs. If so, the node then waits for a random delay before trying to broadcast connectivity messages as the new core node. Simultaneous broadcasts are arbitrated based on a deterministic total order among all nodes, *e.g.*, ordering by IP addresses. The same arbitration applies when multiple disconnected partitions rejoin with a core in each partition.

Saxons component	Overhead per interval	
	Active overhead	Passive overhead
Membership	$\le d_t$ msgs	$\le d_t$ msgs
Connectivity	$\le d_t$ msgs	$\le d_t$ msgs
Latency meas.	$f_{rs} * N_l$ pings	$\sim f_{rs} * N_l$ pings
Bandwidth meas.	$3 * N_b * S_b$	$\sim 3 * N_b * S_b$

Table 2: Saxons overhead. d_t is the node degree bound and f_{rs} is the random sampling factor. N_l and N_b are message counts for the latency and bandwidth measurements respectively. S_b denotes the message size for bandwidth measurements.

3.7 System Overhead

Table 2 illustrates the per-interval overhead of the Saxons components under stable conditions. Note that different Saxons components can run at different intervals. An overhead is counted as *active* when the node in question initiates the network transmission. Both the latency and bandwidth measurements are part of the ShortWide structure quality maintenance policy. Below we attempt to quantify the system overhead in a typical setting. We separate the overhead of bandwidth measurements from other overhead to highlight its dominance in resource consumption. The connectivity messages and the latency measurement messages are small (8 bytes each) in our implementation. A membership message with 20 member records at 8 bytes each[2] has a size of 160 bytes. Assume all these components run at 30-second intervals, the node degree bound d_t=16, the random sampling factor f_{rs}=4, and the latency measurement message count N_l=10. Accounting for the 28-byte IP and UDP headers, the Saxons runtime overhead excluding bandwidth measurements is about 1.3Kbps.

Bandwidth measurements are typically run at a low frequency, *e.g.*, 120-second intervals. Assume the message count N_b=20 and message size S_b=8KB, the bandwidth measurement overhead is about 32Kbps. The total Saxons management cost under this protocol setting is similar to the RON probing overhead for a 50-node overlay [1]. Note that the Saxons overhead does not directly depend on the overlay size, and thus it is able to achieve high scalability. Since most of the network overhead is caused by the bandwidth measurement, nodes that cannot afford such overhead can reduce the frequency of bandwidth measurements or even disable it. This would simply result in lower performance in overlay bandwidth. Bandwidth-oriented structure main-

[2]Each membership record contains a 4-byte IPv4 address and a 4-byte timestamp.

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

Table 3: Backbone networks. A random bandwidth between 1.5−45Mbps and 45−155Mbps (with 50% probability for each range) is assigned for each backbone link.

tenance can also be made more efficient by adjusting the interval between consecutive runs depending on the system stability. For instance, it can run less often when prior runs result in no link adjustments, an indication that the overlay structure has stabilized.

During service growth or frequent membership changes, additional overhead of link adjustments is incurred for structure quality maintenance and partition repairs. We will evaluate such link adjustment overhead in Section 4.3.

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 particular real-world environments. We describe simulation results in this section. Section 5 presents the Saxons performance on 55 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 and bandwidth. 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.

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 [22] and at the Route Views Archive [28]. Second, we include some transit-stub topologies generated using the GT-ITM toolkit [34]. We also use topologies generated by the Michigan *Inet*-3.0 Internet Topology Generator [33]. For ASmap and Inet topologies, we assign a random link latency of 1−40ms. For Transit-Stub topologies, we assign a random link latency of 1−20ms for stub links and 1−40ms for other links. Our final set of backbone network is based on end-to-end latency measurement data among 118 Internet nodes, reported by the NLANR Active Measurement Project [23]. Table 3 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 the average host-to-host latency. 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 1−4ms for all edge links. In terms of link bandwidth, a random bandwidth between 1.5−45Mbps and 45−155Mbps (with 50% probability for each range) is assigned for each backbone link. Edge links are assigned 100Mbps. These reflect commonly used T1, T3, OC-3, and Ethernet links.

The evaluation results are affected by many factors, including the substrate network topologies, protocol parameters, and the combination of different schemes for various Saxons components. Our strategy is to first demonstrate the effectiveness of proposed techniques at a typical setting and then explicitly evaluate the impact of various factors. Unless stated otherwise, results in Sections 4.2 and 4.3 are all based on the ASmap backbone topology, the Rsampling scheme for finding nearby hosts, and a node degree range of <4−16> (*i.e.*, d_a=4 and d_t=16). All periodic Saxons routines run at 30-second intervals except for bandwidth-oriented structure adjustments, which run at 120-second intervals. The link adjustment threshold for the structure quality maintenance is max$\{$4ms, 10%$\}$ for short links (*i.e.*, a new link replaces an old link when it is at least 4ms and 10% shorter in latency) and max$\{$1.0Mbps, 20%$\}$ for wide links. All other protocol parameters default to those described in Section 3.7.

4.2 Structure Quality

We compare different quality maintenance approaches in constructing high-quality overlay structure. AllShort represents protocols optimized solely for low overlay latency. ShortLong introduces a certain degree of randomness to the overlay structure. ShortWide is designed for achieving low overlay latency, low hop-count distance, and high overlay bandwidth at the same time. We also include a *Random* approach in our comparison. This approach makes no quality-oriented link adjustment after randomly establishing links during bootstrap.

Overlay latency. Figure 6 illustrates the structure quality on overlay latency at different 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, *i.e.*, when the average link adjustment rate falls below one per hour per node. Figure 6(A) and 6(B) show the results in overlay path latency and the relative delay penalty respectively. We show both the average values and the 95 percentile values for the overlay path latency results. 95 percentile values do not exhibit different patterns from the average values and we do not show them in other figures for clarity. Overall, we observe all three schemes perform significantly better than the random overlay construction, especially for large networks.

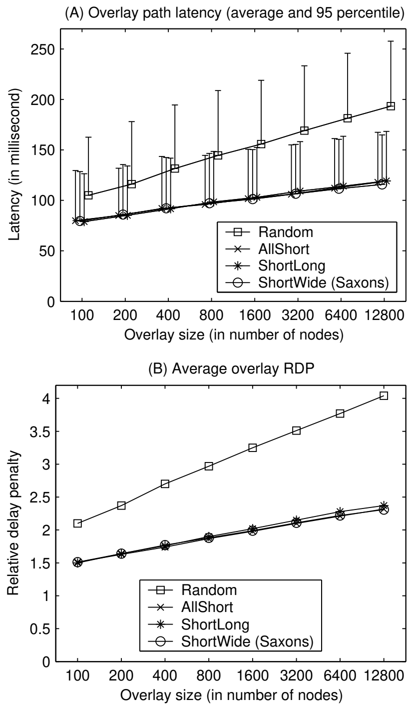


Figure 6: Overlay latency.

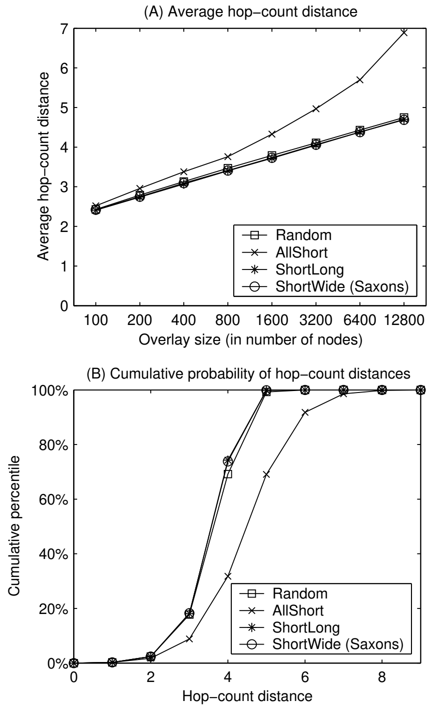


Figure 7: Hop-count distance.

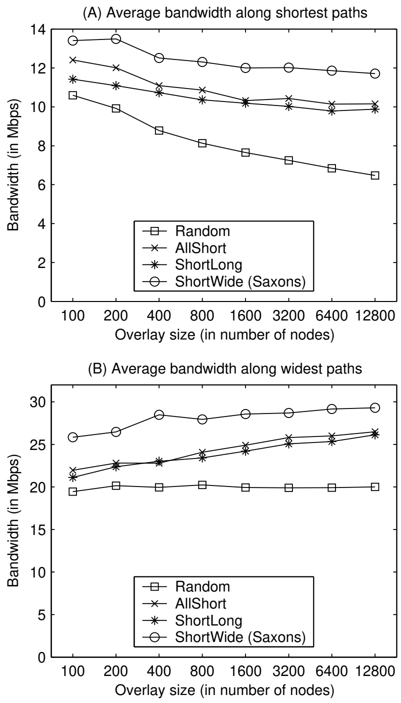


Figure 8: Overlay bandwidth.

Hop-count distance. Figure 7(A) shows the results on structure hop-count distance averaged over all node pairs at different overlay sizes. Figure 7(B) illustrates the cumulative probability of all-pair hop-count distances for 3200-node overlays. We observe that AllShort performs much worse than other schemes, with around 47% larger average overlay hop-count distance for 12800-node overlays. All other three approaches perform very close in the hop-count distance due to the employment of randomness in the overlay structure construction [3].

Overlay bandwidth. Figure 8 shows the average overlay bandwidth along shortest paths and widest paths. We observe that ShortWide outperforms other approaches in terms of overlay bandwidth. For 12800-node overlays, the improvement over its nearest competitor is 15% for the bandwidth along shortest paths and 12% for the bandwidth along widest paths. We also observe that both AllShort and ShortLong significantly outperform Random in overlay bandwidth. This is because short overlay paths often have high bandwidth. Such an inverse correlation between latency and bandwidth is even stronger for TCP-based data transfer due to its various control mechanisms.

In summary, the ShortWide structure management policy outperforms other policies in terms of overlay path bandwidth while achieving competitive performance in terms of overlay path latency and hop-count distance. The results also confirm our conjecture that the AllShort policy could produce overlay structures with high hop-count distances.

4.3 Stability and Connectivity

In this section, we first study the system stabilization at the startup, *i.e.*, right after a large number of nodes have joined the overlay. We then examine Saxons's stability and connectivity support under frequent node joins and departures.

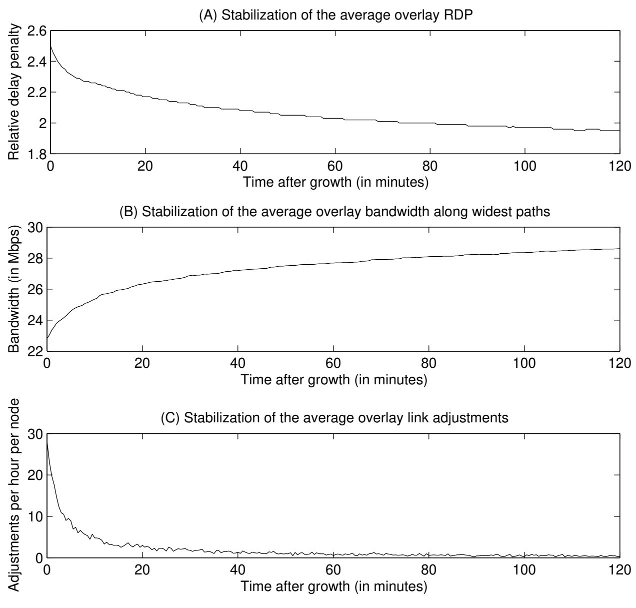


Figure 9: Stabilization after growth (3200-node overlay).

Figure 9 shows the system stabilization at the overlay startup. In this simulation, 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 we measure network samples after this point to assess the system stabilization. Figures 9(A) and 9(B) illustrate the stabilization of average overlay RDP and

Average node lifetime	Connectivity	Relative delay penalty	Hop-count distance	Bandwidth along widest paths	Average per-node link adjustments	
					Partition repair	Quality maintenance
7.5 minutes	96.0%	4.13	6.13	18.08 Mbps	0.24 links/hour	17.04 links/hour
15 minutes	99.1%	3.93	6.03	19.57 Mbps	0.07 links/hour	11.13 links/hour
30 minutes	100.0%	3.76	6.02	21.50 Mbps	0.02 links/hour	6.85 links/hour
1 hour	100.0%	3.55	5.99	22.60 Mbps	0.01 links/hour	4.08 links/hour
2 hours	100.0%	3.49	5.99	24.09 Mbps	0.00 links/hour	2.34 links/hour
Infinity	100.0%	3.41	5.97	26.73 Mbps	0.00 links/hour	0.00 links/hour

Table 4: Stability and connectivity under frequent node joins and departures (3200-node overlay).

average overlay bandwidth respectively. We also show the average link adjustment rate in Figures 9(C). The RDP and overlay bandwidth values are sampled at 30-second intervals and the link adjustment counts are accumulated at the same frequency. We observe that most of the link adjustments occur in the first 20 minutes. Further adjustments occur at very low rate (less than 2 links/hour per node), but they are very important in continuously optimizing the structure quality in terms of overlay latency and bandwidth. We do not show the hop-count distance stabilization which stays mostly the same over the whole period. This is because the random structure generated by node bootstraps already has a low overlay hop-count distance.

Table 4 illustrates the Saxons stability and connectivity support under frequent node joins and departures at various membership change rates for 3200-node overlays. The partition repair scheduling delay parameters are set as D_l=0.5 and D_u=4.0 (defined in Section 3.6). Individual node life times are picked following the exponential distribution with the proper mean. We include results at some unrealistically high rates of membership changes (*e.g.*, average node lifetime of 7.5 minuets) to assess the worst case performance. For the same reason, we use a relatively sparse overlay structure with the node degree range <$2-8$>. Therefore, performance values here is not directly comparable with results shown earlier. The connectivity values in Table 4 are the percentage of fully connected network snapshots out of 5,000 samples. Other values are the average of samples taken at 30-second intervals. We observe that the Saxons connectivity support keeps the overlay structure mostly connected even under highly frequent overlay membership changes. Furthermore, this connectivity support is provided at a very low rate of link adjustments (up to 0.24 links/hour per node). We also observe that the structure quality degrades gracefully as the overlay membership changes become more frequent. Such structure quality maintenance incurs a moderate link adjustment rate of up to 6.85 links/hour for average node lifetime of 30 minutes or longer.

4.4 Impact of Factors

We study the performance impact of backbone topologies, node degree ranges, and the scheme for finding nearby hosts. All results shown in this section are for 3200-node overlays. Figure 10 illustrates the impact of different backbone topologies on the average and 95 percentile overlay RDP. We observe that the performance results are largely stable with different backbone topologies. Overlay RDPs are lower for AMP topologies due to their small size. Very similar results are found for overlay bandwidth and hop-count distance. We do not show them here due to the space limitation.

Figure 11 shows the impact of node degree ranges on the overlay RDP. We observe that overlays with higher link densities tend to make the existence of high-quality paths more likely. However, the relative performance difference among various overlay structure construction approaches remain mostly unchanged. Again, this conclusion is also true for overlay bandwidth and hop-count distance.

Figure 12 shows the overlay RDP for different structure construction schemes under Rsampling and the landmark-based Cartesian distance approach. The Landmark approach uses 8 landmarks in this experiment. We also show the results for an *Ideal* case where the actually closest host is always chosen. The result for Random is only shown for the Landmark approach since its performance is not affected by the policy for finding nearby hosts. We observe that Rsampling constructs structures with less overlay latency compared with the Landmark approach. In particular, it achieves 24% less overlay RDP for AllShort. This is because Rsampling can gradually converge to the optimal selection when running repeatedly. Though such performance is achieved at the cost of substantially more link adjustments during stabilization, we believe the benefit of higher-quality connectivity structure would offset such cost in the long run. Rsampling still generates around 10% higher RDP than Ideal since link adjustments stop when new links are no better than existing links over the specified threshold.

5 Implementation and Experimentation on PlanetLab

We have made a prototype implementation of the Saxons overlay structure management layer. Our prototype assumes the availability of a DNS-like naming system that maps each overlay group name to a small number of bootstrap nodes in a round-robin fashion. Most of the Saxons components are implemented in a single event-driven daemon while the network measurement daemon runs separately due to its time-sensitive nature. Our 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 whole Saxons runtime can be dynamically linked and run inside the application process space. A standalone Saxons process allows possible

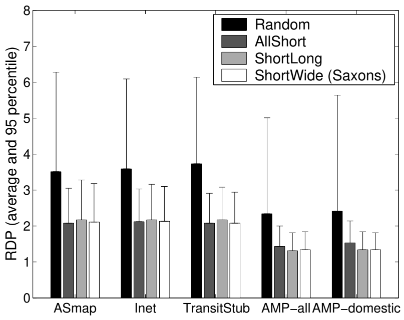


Figure 10: Backbone topologies.

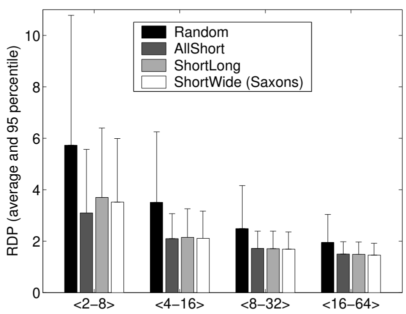


Figure 11: Node degree ranges.

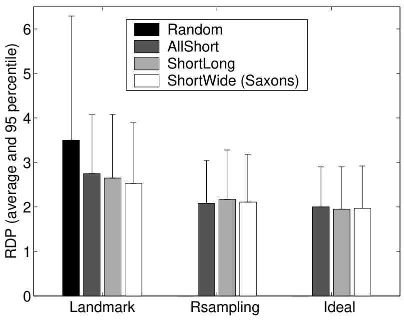


Figure 12: Finding nearby hosts.

runtime overhead sharing when overlay nodes host multiple services.

```
/* type def. of link adjustment callback functions */
typedef int (*SX_CALLBACK)(int linkcnt, link_t *links);

/* join an overlay group */
int sx_join(char *olgroup, int activedegree,
            int maxdegree, SX_CALLBACK cb_adjustment);

/* leave an overlay group */
int sx_leave(char *olgroup);

/* get directly attached overlay links */
int sx_getlinks(int *ptr_linkcnt, link_t *links);
```

Figure 13: The core C/C++ API for Saxons.

Figure 13 shows the core C/C++ interface for developing overlay applications on Saxons. In particular, we provide two ways for overlay applications to access the structure information in Saxons. First, they can directly query the Saxons layer to acquire information about attached overlay links (sx_getlinks). They can also provide a nonblocking callback function[3] (cb_adjustment) at the startup. If so, Saxons will invoke the application-supplied callback function each time an overlay link adjustment occurs. Link adjustment callbacks are useful for applications that maintain link-related state, such as various overlay routing services. Without the callback mechanism, they would have to poll the Saxons layer continuously to keep their link-related state up-to-date.

We conducted experiments on the PlanetLab testbed [25] to evaluate the Saxons performance in a real-world environment. We compare the overlay structure quality achieved by various quality maintenance policies: Random, AllShort, ShortLong, and ShortWide. In the experiments, nodes join the overlay network at the average rate of one per 3 seconds. Measurement results are taken when the average link adjustment rate falls below one per hour per node. Overlay structures are configured with the node degree range of <4−16>. Figure 14 illustrates the Saxons overlay latency and bandwidth CDFs for all node pairs among 55 PlanetLab nodes, all from unique wide-area sites. We provide round-trip latency results because they are easier to measure than one-way latencies. Note that the simulation results shown earlier are for one-way latencies. We do not show the performance on the overlay hop-count distance because the overlay size is too small to make this metric meaningful.

[3]Callback functions are not allowed to block on I/Os and they must return in a bounded amount of time.

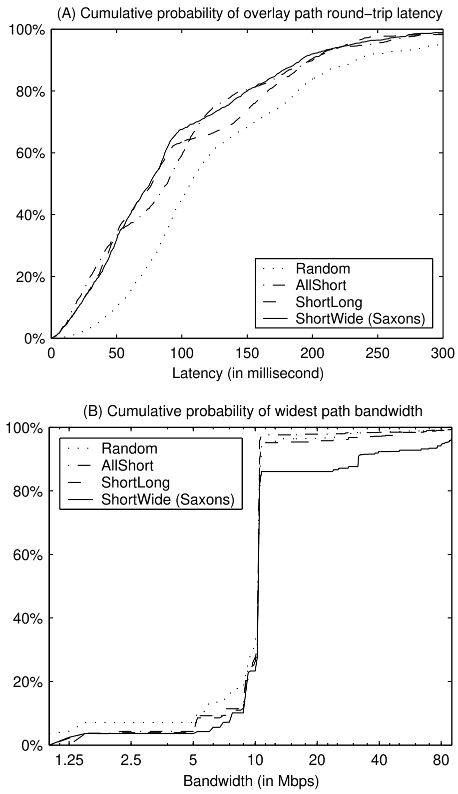


Figure 14: Saxons overlay latency and bandwidth CDFs for all site pairs among 55 PlanetLab sites. The X-axis for Figure (B) is in the log scale.

In terms of overlay latency, all three quality maintenance policies outperform the random overlay structure with over 18% less overlay path latency in average. This performance difference is close to the simulation result for small overlays in Figure 6(A). As for the bandwidth, we observe that most of the node pairs have 10Mbps overlay bandwidth between them. As discussed in Section 3.2, this is because most of the PlanetLab nodes are equipped with a packet filter [9] that limits the per-user outgoing bandwidth at 10Mbps [4]. With only 8 out of 55 nodes that are not subject to the bandwidth limit, ShortWide is able to provide high-speed overlay path

(>10Mbps) for three times as many node pairs as its nearest competitor. This quantitative result is not typical due to the particular bandwidth control mechanism equipped on many of the PlanetLab nodes. Nonetheless, it provides an example of Saxons's ability to discover high bandwidth paths when they exist.

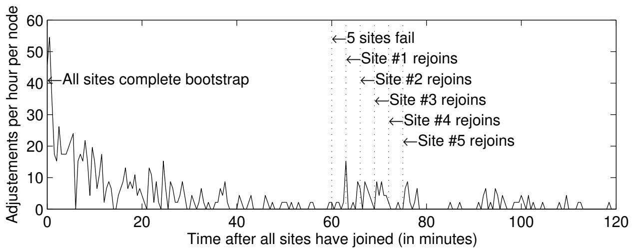


Figure 15: Saxons link adjustments on 55 PlanetLab sites.

Figure 15 shows the Saxons structure stability during membership changes. Again, nodes join the overlay network at the average rate of one per 3 seconds. We start tracking the link adjustment counts right after the last node has joined. Results are accumulated at 30-second intervals. We observe that the link adjustment rate mostly fall below 5 per hour per node after the 30th minute. We then inject a simultaneous 5-node failure at the 60th minute and we let them rejoin the overlay one by one at 3-minute intervals. We observe that the link adjustment activity is moderate (mostly under 10 per hour per node) during the membership changes. After the 100th minute, the average link adjustment count falls around 2 per hour per node, which indicates around one link adjustment at a single node during each 30-second interval.

6 Service Construction

Saxons actively maintains a stable and high quality overlay structure with partition repair support. Services can directly utilize the Saxons structure for overlay communication. Saxons can also benefit unicast or multicast overlay path selection services [1, 7, 17] by providing them a small link selection base, thus making them scalable without hurting their performance potential. In this section, we describe the construction of two services (query flooding and overlay multicast) that utilize the Saxons overlay structure in different ways. We have also implemented a Saxons-based distributed hash table and compared its performance against a well-known DHT protocol. Results of that work are reported in [30].

Saxons-based query flooding. We implemented the Gnutella query flooding protocol on top of the Saxons structure management layer. The default Gnutella protocol uses a bounded-degree random structure, which is similar to the *Random* structure we used in our evaluations. Our service directly uses the Saxons structure for query flooding without any further link selection. It uses the Saxons direct link query interface instead of the callback interface because no link-related state is actively maintained at the service-level. The low overlay hop-count distance in the Saxons structure allows query flooding to reach more nodes at a particular TTL. The low overlay latency allows fast query response while the high overlay bandwidth alleviates the high network load of query flooding. We do not provide performance results in this paper because they closely match the raw Saxons performance shown earlier.

Saxons-based overlay multicast. We also implemented an overlay multicast routing service on Saxons. Similar to DVMRP [32], our multicast service is based on *Reverse Path Forwarding* over a distance vector unicast routing protocol. The service actively monitors link latency and bandwidth between Saxons neighbors while the DV unicast routing protocol maintains path latency and bandwidth by aggregating the link properties. We employ a simple path cost function in the DV protocol that considers path latency, bandwidth, and hop-count distance: $cost = \frac{latency}{L_{unit}} + \frac{hop}{H_{unit}} - \frac{bandwidth}{B_{unit}}$.

We empirically choose L_{unit}=20ms, H_{unit}=1; and B_{unit}=1.0Mbps in our implementation. Note that our main purpose is to demonstrate the effectiveness of Saxons in supporting overlay service construction. Therefore we do not pursue optimized service implementation in this paper.

We evaluated the performance of the Saxons-based overlay multicast routing service on the PlanetLab testbed. We choose a PlanetLab node that does not have outgoing bandwidth control as the multicast source: `planetlab2.cs.duke.edu`. The overlay structure is configured at the node degree range of <4−12>. We compare the performance of Saxons-based overlay multicast with the same multicast service running on top of a random overlay structure. For additional comparison purposes, we approximate the *ideal* multicast performance using the independent direct unicast, which measures the latency and bandwidth along the direct Internet path from the source to each receiver in the absence of simultaneous traffic. Figure 16 illustrates the round-trip latency from the source to all receivers, ranked increasingly on the latency. We observe that Saxons-based overlay multicast achieves 24% less latency in average than multicast over random structure. Compared with independent direct unicast, overlay multicast produces longer latency due to its multi-hop forwarding nature.

We also measured the multicast bandwidth on the PlanetLab testbed. We conducted two experiments, one with a 1.2Mbps multicast stream and another with a more aggressive 2.4Mbps stream. For unicast transport along overlay links, we use the default UDP service without lost recovery or congestion control. Congestion-controlled transport protocols such that TFRC [12] may improve the performance of our service implementation. However, it is not crucial to our purpose of evaluating the Saxons overlay structure management. Figure 17 shows observed bandwidth at all receivers, ranked increasingly on the bandwidth. We observe that the bandwidth performance of Saxons-based overlay multicast is quite close to that of the independent direct unicast for both 1.2Mbps and 2.4Mbps streams. Compared with multicast over random structure, the Saxons-based multicast provides near-loss-free ($< 5\%$ loss) data delivery to more than 4 times as many multicast receivers.

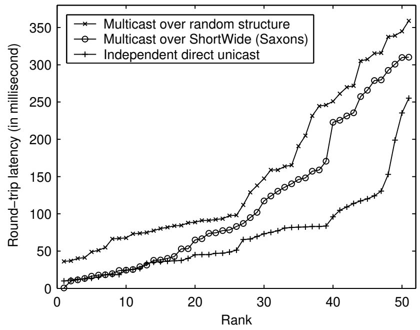


Figure 16: Multicast latency over 52 PlanetLab sites.

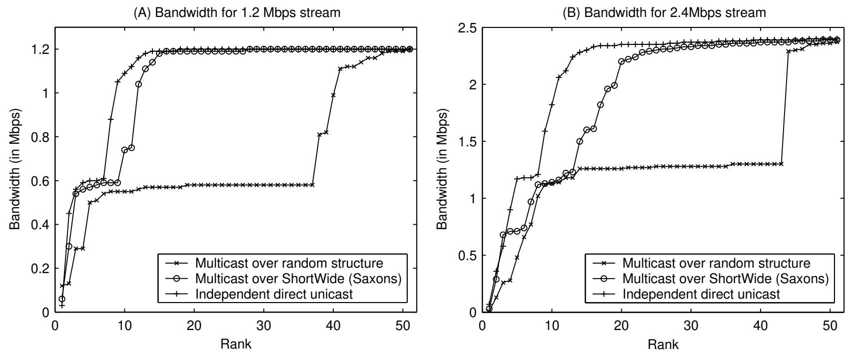


Figure 17: Multicast bandwidth on 52 PlanetLab sites.

In addition to help the multicast service to achieve high performance, Saxons is also crucial for the scalability of this service. While it is conceivable for the multicast routing to run directly on the completely connected overlay, the overhead of tracking link properties and maintaining routing indexes along overlay links would become prohibitively expensive when the overlay grows to a large size. It should be noted that choosing an appropriate node degree range for the Saxons structure often has a significant impact on service performance. A highly connected Saxons structure makes the existence of high-performance overlay routes more likely. However, discovering them would consume more overhead at the service-level.

7 Related Work

The concept of structure-first overlay construction has been studied before in the Narada end-system multicast protocol [7]. Narada maintains a low latency mesh structure on top of which a DVMRP-style multicast routing protocol handles the data delivery. However, Narada is not designed for large-scale systems. For instance, its group management protocol requires each node to maintain the complete list of other overlay members, which would require excessive maintenance overhead for large overlays.

Prior studies have examined substrate-aware techniques in the construction of many Internet overlay services, including unicast overlay path selection (*e.g.*, RON [1]), end-system multicast protocols (*e.g.*, Overcast [17] and NICE [2]) and scalable DHT protocols (*e.g.*, Binning [27], Brocade [35], and Pastry [6]). These studies focused on specific services and substrate-aware techniques were often tightly integrated with the service construction. Saxons supports a comprehensive set of performance objectives in a separate overlay structure management layer and therefore it can be used as a building block to benefit the construction of a wide range of services. Additionally, we are unaware of any prior work on scalable overlay structure management that explicitly considers the overlay path latency, hop-count distance, and the overlay bandwidth at the same time.

Nakao *et al.* recently proposed a multi-tier overlay routing scheme [20]. In their approach, several overlay routing services are constructed on top of a topology probing kernel, which acquires AS-level Internet topology and routing information from nearby BGP routers. Saxons differs from their work by constructing the overlay structure using end-to-end network measurements. More importantly, their work does not explicitly address the multiple structure quality metrics that are investigated in this paper.

Many prior studies provided ideas that are related to the design of various Saxons components. For instance, a number of studies have examined scalable estimation schemes for finding nearby Internet hosts, including Hotz [16], IDMaps [14], GNP [21], and Binning [27]. While Saxons can utilize any of the existing techniques, we also introduce a light-weight random sampling approach to locate nearby hosts without the need of infrastructural support or established landmark hosts. Additionally, Kostić *et al.* recently proposed a random membership subset service for tree-shaped overlay structures [18]. This approach ensures that membership in the subset changes periodically and with uniform representation of all overlay nodes. The key difference with our random membership subset component is that Saxons is designed to support mesh-like overlay structures.

8 Concluding Remarks

In this paper, we propose Saxons, a substrate-aware overlay structure management layer that assists the construction of scalable Internet overlay services. Saxons dynamically maintains a high quality structure with low overlay latency, low hop-count distance, and high overlay bandwidth. At the same time, Saxons provides connectivity support to actively repair overlay partitions in the presence of highly frequent membership changes. Simulations and experiments on 55 PlanetLab sites demonstrate the performance, stability, and connectivity support of our proposed design. Additionally, this paper also describes the construction of two Saxons-based overlay services.

It is conceivable for a common overlay structure management layer to allow runtime overhead sharing when overlay nodes host multiple services. However, different overlay services often desire different link density, structure qualities, and stability support. Additionally, although overlay groups may overlap, they often contain a substantially large number of non-overlapping nodes. These factors make it difficult for multiple services to share the same overlay structure. As a result, we believe it is more feasible for sharing low-level activities such as link property measurements. For instance, the discovery of a high bandwidth link to a particular node may interest multiple hosted services. Further investigation on this issue is needed in the future.

Acknowledgment: This work was supported in part by NSF grants CCR-0306473 and ITR/IIS-0312925. We would like to thank Yuan Sun, the URCS systems group, the anonymous referees, and our shepherd David Culler for their valuable comments. We are also indebted to Liudvikas Bukys and the PlanetLab support for making possible the wide-area experimentation in this study.

References

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

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

[3] B. Bollobas. *Random Graphs*. Academic Press, London, UK, 1985.

[4] P. Brett. The PlanetLab support team, August 2003. Personal communication.

[5] R. L. Carter and M. E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. Technical Report BU-CS-96-006, Computer Science Department, Boston University, March 1996.

[6] 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.

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

[8] I. Clarke, T. W. Hong, S. G. Miller, O. Sandberg, and B. Wiley. Protecting Free Expression Online with Freenet. *IEEE Internet Computing*, 6(1):40–49, 2002.

[9] M. Devera. Hierarchical token bucket. http://luxik.cdi.cz/∼devik/qos/htb/.

[10] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight Probabilistic Broadcast. *ACM Trans. on Computer Systems*, 21(4):341–374, November 2003.

[11] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *Proc. of the ACM SIGCOMM*, pages 254–265, Vancouver, BC, September 1998.

[12] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based Congestion Control for Unicast Applications. In *Proc. of the ACM SIGCOMM*, pages 43–56, Stockholm, Sweden, August 2000.

[13] S. Floyd, V. Jacobson, and S. McCanne. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. In *Proc. of the ACM SIGCOMM*, pages 342–356, Cambridge, MA, August 1995.

[14] 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.

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

[16] S. Hotz. *Routing Information Organization to Support Scalable Routing with Heterogeneous Path Requirements*. PhD thesis, Dept. of Computer Science, University of Southern California, 1994.

[17] 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 the USENIX OSDI*, San Diego, CA, October 2000.

[18] D. Kostić, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat. Using Random Subsets to Build Scalable Network Services. In *Proc. of the USENIX Symp. on Internet Technologies and Systems*, Seattle, WA, March 2003.

[19] Limeware. http://www.limeware.com.

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

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

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

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

[24] V. Paxson. End-to-End Internet Packet Dynamics. In *Proc. of the ACM SIGCOMM*, pages 139–152, Cannes, France, September 1997.

[25] 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.

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

[27] 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.

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

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

[30] K. Shen. Distributed Hashtable on Pre-structured Overlay Networks. Technical Report TR831, Dept. of Computer Science, University of Rochester, January 2004. http://www.cs.rochester.edu/trs/systems-trs.html.

[31] 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 the ACM SIGCOMM*, pages 149–160, San Diego, CA, August 2001.

[32] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. IETF RFC-1075, November 1988.

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

[34] 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.

[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.