Structured Overlay For Heterogeneous Environments: Design and Evaluation of Oscar

2

ŠARŪNAS GIRDZIJAUSKAS
Ecole Polytechnique Fédérale de Lausanne (EPFL)
ANWITAMAN DATTA
Nanyang Technological University
and
KARL ABERER
Ecole Polytechnique Fédérale de Lausanne (EPFL)

Recent years have seen advances in building large Internet-scale index structures, generally known as *structured overlays*. Early structured overlays realized distributed hash tables (DHTs) which are ill suited for anything but exact queries. The need to support range queries necessitates systems that can handle uneven load distributions. However such systems suffer from practical problems—including poor latency, disproportionate bandwidth usage at participating peers, or unrealistic assumptions on peers' homogeneity, in terms of available storage or bandwidth resources. In this article we consider a system that is not only able to support uneven load distributions but also to operate in heterogeneous environments, where each peer can autonomously decide how much of its resources to contribute to the system. We provide the theoretical foundations of realizing such a network and present a newly proposed system Oscar based on these principles. Oscar can construct efficient overlays given arbitrary load distributions by employing a novel scalable network sampling technique. The simulations of our system validate the theory and evaluate Oscar's performance under typical challenges, encountered in real-life large-scale networked systems, including participant heterogeneity, faults, and skewed and dynamic load-distributions. Thus the Oscar distributed index fills in an important gap in the family of structured overlays, bringing into life a practical Internet-scale index, which can play a crucial role in enabling data-oriented applications distributed over wide-area networks.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks, network topology*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

The work presented in this article was supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322. Part of the work carried out was funded by A*Star SERC Grant No. 072 134 0055.
Author's address: S. Girdzijauskas, email: Sarunas.girdzijauskas@epfl.ch.

DOI 10.1145/1671948.1671950 http://doi.acm.org/10.1145/1671948.1671950

General Terms: Algorithms, Design

Additional Key Words and Phrases: Peer-to-peer systems, routing, structured overlays, small-world graphs, skewed key distributions

ACM Reference Format:

Girdzijauskas, Š., Datta, A., and Aberer, K. 2010. Structured overlay for heterogeneous environments: Design and evaluation of Oscar. ACM Trans. Autonom. Adapt. Syst. 5, 1, Article 2 (February 2010), 25 pages.
DOI = 10.1145/1671948.1671950 http://doi.acm.org/10.1145/1671948.1671950

1. INTRODUCTION

This decade has witnessed proliferation of the peer-to-peer paradigm in diverse application domains, where end users contribute and share various kinds of resources, including data and content. Ability to efficiently find resources in a large-scale networked and decentralized environment is an important ingredient enabling such systems. Structured overlays, for example, DHTs provide a distributed indexing mechanism, facilitating search in such environments. To support portability and modularity, it is also essential to achieve network data independence [Hellerstein 2003]. This requires ensuring transparency of network layer issues at the application layer, like node heterogeneity, system churn, load balancing issues, and lack of global knowledge at individual peers.

Numerous structured overlay topologies, often emulating interconnection networks, have been proposed in the literature. Many of these, despite numerous apparent differences, are essentially special instances of a family of small-world networks, where decentralized greedy search algorithms can work efficiently. Kleinberg [2000] identified the family of such small-world networks that have polylogarithmic search path lengths when individual nodes maintain a constant amount of network information. Aspnes et al. [2002] and Giakkoupis and Hadzilacos [2007] introduced tighter lower bounds and showed that such *Kleinbergian* construction is indeed optimal for decentralized greedy routing algorithms.

The Kleinberg model is based on a multidimensional lattice space that has a notion of distance between two points in the lattice. The summary conclusion of Kleinberg's seminal work is that the family of graphs on which greedy decentralized routing could be used efficiently were those where each node established a link to another node with a probability inversely proportional to the distance between these two nodes raised to the power equal to the dimension of the lattice space. This network construction model is extremely relevant in the context of peer-to-peer overlay index structures. Most structured overlays have an underlying identifier space and rely only on local decisions for navigating the overlay based on the distance metric of the identifier space, thus a greedy algorithm is feasible and easy to implement without global knowledge or coordination.

To ease the network construction task and to effectively balance the data load, most structured peer-to-peer systems use uniform hash functions (like

SHA-1[1]) for assigning identifers to peers and resources (e.g., Chord [Stoica et al. 2001], Symphony [Manku et al. 2003], etc). The usage of uniform hash functions ensures that load is uniformly randomly distributed over the key-space, and thus by dividing the key-space and the associated load uniformly, randomly among the peers, load balancing among the peers can be achieved. Furthermore, it enables the utilization of simple and unambiguous neighbor selection protocols, which guarantee a balanced node degree at all peers.

The use of uniform hash functions, however, limits the use of data-oriented overlays to simple exact identifer lookup capabilities. More complex queries like, for example, similarity or range queries become extensively ineffective since uniform hash functions disperse otherwise correlated data over many different peers, thus making the access highly inefficient if not infeasible. Therefore, semantic data processing cannot be successfully tackled by conventional uniform hash function-based peer-to-peer systems.

Furthermore, practical scalable peer-to-peer systems need to take heterogeneity into account explicitly in the system's design. For data-oriented overlays, heterogeneity is encountered both because of the peculiarities of the environment as well as the application characteristics. Measurement studies [Stutzbach et al. 2005] of deployed peer-to-peer systems show heterogeneity arising because of either diverse availability of resources like storage, bandwidth, computation, and content at peers, or variation in individual willingness to contribute resources to the system, as well as software artifacts like default configurations. Thus, it becomes evident that uniform hash functions are rendered to be ineffective, facing the consequences of heterogeneous environments since under such circumstances data-oriented applications are inevitably characterized by a nonuniform distribution of keys over the key-space as well as skewed query or access patterns.

This implies that overlay networks have to be able to handle hash functions that produce nonuniform key distributions (e.g., order-preserving hashing). The design of such overlay networks has to take into account the resulting skewed key distributions and adapt the construction mechanisms accordingly, to distribute the load among peers in a judicious manner. That is not a straightforward task, particularly in the absence of global knowledge and coordination. Most contemporary structured peer-to-peer approaches avoid addressing these issues, instead wrongly relying on unrealistic uniformity assumptions on peers' capacity in terms of bandwidth consumption and storage capacities, which limits the practicality for realistic peer-to-peer environments. In this article we suggest the algorithms for constructing a routing-efficient overlay network based on scalable sampling strategy. We call the resulting system *Oscar*, which stands for overlay network built using scalable sampling of realistic distributions. The novelty in our approach is both in the fact that we explicitly account for peers' heterogeneity, as well as a scalable sampling mechanism, which provides adequate information with low overheads to create an efficient structured overlay.

[1]http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html

We provide a full analysis for the Oscar algorithms with analytical guarantees for the performance of the resulting networks. The analytical results are validated with simulations with realistic skewed workloads, heterogeneous peers, and network dynamics (churn). To our knowledge this is the first scalable sampling approach that has been proposed for construction and maintenance of structured overlay networks for nonuniform key distributions while coping with and exploiting peer heterogeneity.

The rest of the article is organized as follows. In Section 2 we overview the main design principles of Oscar and its conceptual differences with the Mercury system [Bharambe et al. 2004]. In Section 3 we discuss the other related techniques and how Oscar compares to them. In Section 4 we recapitulate essential ideas from our previous work [Girdzijauskas et al. 2005]—namely the problem of dealing with skewed key spaces. We present the concept and the algorithms of the proposed Oscar system in Section 5. We evaluate the Oscar system in Section 6 based on rigorous simulations. We draw our conclusions in Section 7. We analyze the proposed mechanisms and algorithms in online Appendix A.

2. DEALING WITH A MULTITUDE OF HETEROGENEITY: THE DESIGN LANDSCAPE

Most structured overlay networks are instances of a Kleinbergian small-world graph. In such networks searches are usually performed by greedy routing, though other navigation techniques can also be implemented; for instance routing with "lookahead" [Manku et al. 2004] or "cautious greedy-routing" [Barbella et al. 2007]. In our previous work [Girdzijauskas et al. 2005] we showed how to build routing-efficient network topologies in the presence of nonuniform key distributions assuming the whole key distribution is known at every peer. Mercury [Bharambe et al. 2004] based its heuristic approach on a similar observation and proposed a structured overlay network that balances load among peers and attempts to construct a small-world graph-based overlay. For proper choice of long-range links Mercury needs to know about the key distribution. A simple sampling mechanism is employed in Mercury to approximately learn the global key distribution. However, the sampling technique that Mercury uses to determine the candidates for long-range links does not scale, given arbitrary load distributions that typically occur in practice.

Mercury can deal with simple monotonous skewed distributions but the sampling technique is inadequate for real-world distributions, which are typically arbitrary and nonmonotonous (see Section 4). In our initial work on the Oscar system [Girdzijauskas et al. 2006] we showed that under realistic workloads, Mercury nodes suffer a large imbalance in in-degree, which results in both poor search performance as well as load-imbalance and congestion.

The problem of in-degree imbalance in Mercury arises because it uses an approximation of the global key distribution for constructing the network from a limited set of samples done uniformly on the key-space. Since it is not possible to achieve a good approximation of an arbitrary distribution with a limited number of samples, the resulting P2P networks have poor performance.

Our proposed Oscar overlay, however, enjoys all the benefits of systems that support complex nonuniform key distributions (like Mercury or P-Grid [Aberer 2001]) and hence nonexact queries (e.g., range or similarity queries) but does not suffer from node in-degree imbalance, while exhibiting lookup performance comparable to traditional DHTs (which can only support exact queries).

Since Oscar builds the underlying network according to Kleinberg's small-world construction principles, it gives peers complete autonomy to determine the size of the key-space partition it would be responsible for, based on either its storage or bandwidth constraints. The in and out-degrees of a peer can vary depending on peers' local and autonomous decisions, while still providing guarantees of efficient search globally. Because the links are chosen randomly, there are multiple options to choose from, so that the in-degree of a peer can also be easily adjusted, where individual peers refuse further connections based on a local decision. Such features of small-world approaches enable accommodating and exploiting peer heterogeneity—storage as well as bandwidth, while also dealing with heterogeneous workloads—skew in key distribution over the key-space as well as query frequency for the keys. Peers are free to choose the maximum amount of outgoing and incoming links locally, depending on their bandwidth budget to maintain the links as well as to cater to the query traffic, based on their locally perceived bandwidth or other constraints. Similarly, peers are free to choose the key-space to be responsible for. This may be based on their storage capacity and bandwidth constraints to answer the corresponding queries. Thus, the Oscar overlay is capable of dealing with both the heterogeneity observed in the network, particularly bandwidth and storage resource heterogeneity at peers, as well as nonuniformity observed in the workloads in data-oriented applications, particularly skewed key distributions as well as skewed access loads.

3. RELATED WORK

Distributed Hash Tables (DHTs), for example Chord [Stoica et al. 2001] and Pastry [Rowstron and Druschel 2001] were originally proposed for efficient decentralized search and address-independent routing. Uniform hashing was used to generate keys, and a hash table distributed among the peers was used to store and efficiently locate these keys. One of the motivations to use uniform hashing in these early approaches was that load was distributed relatively uniformly on the key/identifier-space. Subsequent works [Manku et al. 2003; Hui et al. 2006] followed the same paradigm and provide only limited support for data-oriented applications, since uniform hashing prevented nonexact queries.

Such limitations of DHTs in supporting data-oriented applications spurred research on a next generation of order-preserving structured overlay networks. Preserving ordering relationships among keys is essential for data-oriented queries like approximate, range, similarity, and sky-line queries.

Preserving ordering relations can lead to skewed load distributions, in turn causing load imbalance at the peers. This led to a number of research efforts addressing this problem in various ways. Systems like CAN [Ratnasamy et al. 2001], Mercury [Bharambe et al. 2004], M-Chord [Novak and Zezula 2006],

P-Grid [Aberer et al. 2005], skip graphs [Aspnes and Shah 2003; Harvey et al. 2003] and derivatives [Aspnes et al. 2004; Ganesan et al. 2004; Guerraoui et al. 2006] looked into some subproblems, like addressing load-balance under nonuniform key distributions. These approaches usually assume uniformity of peers' capacity in terms of bandwidth consumption and storage capacity, which limits their practicality for realistic peer-to-peer environments. Most of these approaches also suffer from shortcomings with respect to essential properties of operation, for example, the search efficiency in terms of the number of overlay hops cannot be guaranteed in CAN for an arbitrary partitioning of the key-space (zones). Storage-load balanced P-Grid may have highly imbalanced peer degrees. Skip graphs need to have $O(\log N)$ level rings at each peer, where level ring neighbors are determined by a peer's membership vector and the existing skew in the system. Such a design omits the possibility of choosing routing table entries in a randomized manner. Therefore Skip graphs lack the flexibility provided by the truly randomized approaches (e.g., based on small-world construction principles like Mercury) and cannot address some of the heterogeneity issues, for example, different constraints on storage and bandwidth at each peer. M-Chord utilizes Chord [Stoica et al. 2001] structure as the underlying overlay and, similarly to Mercury, uses uniform sampling to discover the resource (object) distribution for the correct assignment of the peer keys. As we discussed, uniform sampling is inadequate in the presence of complex key distributions, and inevitable wrong estimates from such approaches lead to overlays with relatively poor performance. Oscar's strength is its scalable sampling technique even in the presence of arbitrary load skews, and using small-world principles to leverage the heterogeneity of available resources—leading to the design of a system with superior performance under realistic workloads compared to all other existing systems.

4. PRELIMINARIES

Basic Concepts for Structured P2P. A structured overlay network consists of set of peers $\mathcal{P}$ ($N = |\mathcal{P}|$) and set of resources $\mathcal{R}$. There exists an identifier space $\mathcal{I}$ (usually on the unit interval $\mathcal{I} \in [0..1)$, e.g., Chord [Stoica et al. 2001], Symphony [Manku et al. 2003]) and two mapping functions $F_{\mathcal{P}} : \mathcal{P} \to \mathcal{I}$ and $F_{\mathcal{R}} : \mathcal{R} \to \mathcal{I}$ (e.g., SHA-1). Thus, each peer $p \in \mathcal{P}$ and each resource $r \in \mathcal{R}$ is associated with some identifiers $F_{\mathcal{P}}(p) \in \mathcal{I}$ and $F_{\mathcal{R}}(r) \in \mathcal{I}$, respectively. There exists a distance function $d_{\mathcal{I}}(u, v)$, which indicates the distance between a peer u and a peer v in $\mathcal{I}$. Each peer p has some short-range links $\rho_s(p) \subset \mathcal{P}$ and long-range links $\rho_l(p) \subset \mathcal{P}$, which form a peer's routing table $\rho(p) = \rho_s(p) \cup \rho_l(p)$. There exists a global probability density function f characterizing how peer identifiers are distributed in $\mathcal{I}$. Any resource $r \in \mathcal{R}$ in the P2P system can be located by issuing a query for $F_{\mathcal{R}}(r)$. In structured P2P systems queries are usually routed in a greedy fashion—always choosing the link $\rho \in \rho(p)$ that minimizes the distance to the target's identifier.

Complex Distributions. Using a uniform hash function $F_{\mathcal{R}}$ (e.g., SHA-1) in data-oriented P2P applications is not adequate. It is necessary to deal with

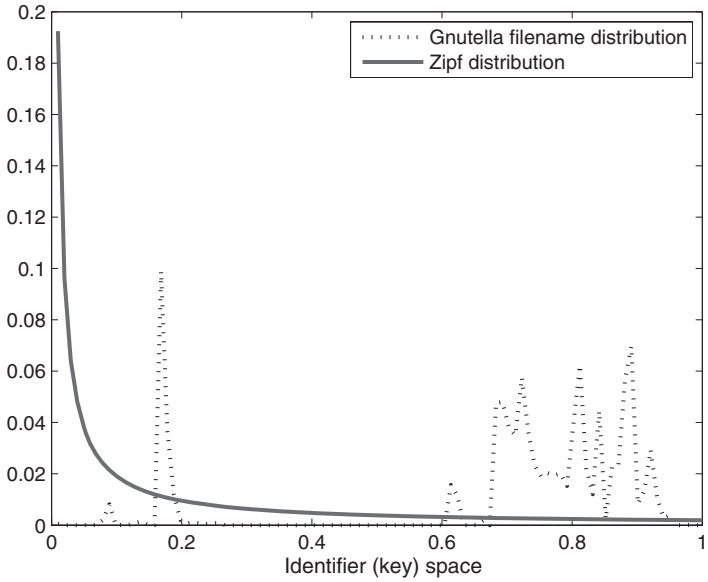


Fig. 1. Probability density functions: Monotonous Zipf with parameter $\alpha = 1$ (solid) vs Gnutella filename (dotted).

order-preserving hash functions, which produce skewed key distributions if the corresponding data distributions are skewed.

Let us assume a data-oriented P2P system, where the resources are identified and looked up by filenames. A widely used technique in P2P systems is the following: each peer p and each resource r has identifiers $F_{\mathcal{P}}(p)$ and $F_{\mathcal{R}}(r)$ on a 1-dimensional ring $\mathcal{I} \in [0..1)$. Each peer is responsible for all the resources that map to the identifier range $D(p) \in [F_{\mathcal{P}}(p), F_{\mathcal{P}}(p_{succ}))$, where the peer p_{succ} is the successor of the peer p on the identifier ring $\mathcal{I}$. We cannot use a uniform hash function such as SHA-1, since we want the function $F_{\mathcal{R}}$ to preserve ordering relationships among the resource keys (e.g., enabling the straightforward use of range queries), $F_{\mathcal{R}}(r_i) > F_{\mathcal{R}}(r_j)$ if and only if $r_i > r_j$. Such an order-preserving hash function will lead to a very skewed distribution of resource identifiers over the identifier ring $\mathcal{I}$. For example, in Figure 1 we can see a distribution function of filename identifiers in $\mathcal{I}$ extracted from a Gnutella trace (dotted line) of 20,000 filenames crawled in 2002 versus a monotonous Zipf distribution with parameter $\alpha = 1$ [Breslau et al. 1999; Zipf 1929]. Despite the complex skew of key distributions, we would like each peer p to be responsible for a fair (or equal) amount of resources and be storage-load balanced: $|\mathcal{R}_{p_i}| \approx |\mathcal{R}_{p_j}|$ for any i and j, where $\mathcal{R}_p \subset \mathcal{R}$ and $\forall r \in \mathcal{R}_p \; F_{\mathcal{R}}(r) \in D(p)$. In this case the peer identifiers will have to reflect the distribution of resource identifiers. Hence the peer identifier distribution will have a similar shape as the resource identifier distribution. Since in general the resource identifier distributions are usually nonuniform and exhibit complex skews, the resulting peer identifier distribution will have to have a complex skew as well.

Dealing with Skewed Spaces. The seminal work of Kleinberg [2000] proposes a "routing efficient" network on a uniform d-dimensional mesh. The follow-up works [Barrière et al. 2001; Manku et al. 2003] showed how to adapt the Kleinbergian network construction principles for P2P systems with uniform key distribution. In our previous work [Girdzijauskas et al. 2005] we showed that it is indeed possible to construct routing-efficient small-world networks in the 1-dimensional space even if the peers are nonuniformly distributed on the unit interval. For doing so, it was shown that a peer u has to choose a peer v as its long-range neighbor with a probability that is inversely proportional to the integral of the probability density function f between these two nodes,

$$P[v \in \rho_l(u)] \propto \frac{1}{\left| \int_{F_{\mathcal{P}}(u)}^{F_{\mathcal{P}}(v)} f(x) dx \right|}. \tag{1}$$

However, it is nontrivial to apply this technique in practice because it requires at each peer, global knowledge about the data load in the system, hence the key distribution f. A simple approach to obtain the distribution is to randomly sample the network and get an approximation of the key distribution, for example, Mercury [Bharambe et al. 2004]. However, the real-world distributions can be totally arbitrary and the only sufficient approximation of the distribution would be gathering in a sample set the complete set of values which, of course, does not scale. In this article we show that Mercury (which uses random sampling) fails to build routing-efficient networks given arbitrary distribution functions. Moreover, we also show that it is not necessary to know the distribution function over the entire identifier space with uniform resolution—it is sufficient to learn well the distribution for only some regions of the identifier space while leaving other regions vaguely explored, making this the base idea of Oscar algorithms.

5. OSCAR OVERLAY

The Insight. According to the continuous Kleinberg approach [Barrière et al. 2001; Manku et al. 2003; Girdzijauskas et al. 2005] for construction of a routing-efficient network in 1-dimensional space, each node u has to choose two short-range neighbors and one or more long-range neighbors. Short-range neighbors of u are its immediate successor and predecessor on the unit ring. A peer u chooses its long-range neighbor v in the unit interval with the pdf $g(x) = \frac{1}{x \ln N}$ on the range $[\frac{1}{N}, 1]$, where $x = d_{\mathcal{I}}(u, v)$. It has been proven that a network constructed in such a way is routing-efficient: a greedy routing algorithm on expectation requires $O(\frac{\log_2^2 N}{l})$ hops, where l is the number of long-range links at every peer. This means that a node u will tend to choose a long-range neighbor v from its close neighborhood, rether than from the farther away regions. The pdf $g(x)$ according to which the neighbors are chosen also has one nice property when partitioned into equally spaced segments on the logarithmic scale (logarithmic partitions). That is, if we partition the identifier space into $\log_2 N$ partitions $A_1, A_2, ..A_{\log N}$, such that the distance between the peer u and any other peer v in A_i is bounded by $2^{-i} \leq d_{\mathcal{I}}(u, v) < 2^{-i+1}$, the peer v will have equal probability to be chosen from each of the resulting partitions (Figure 2).

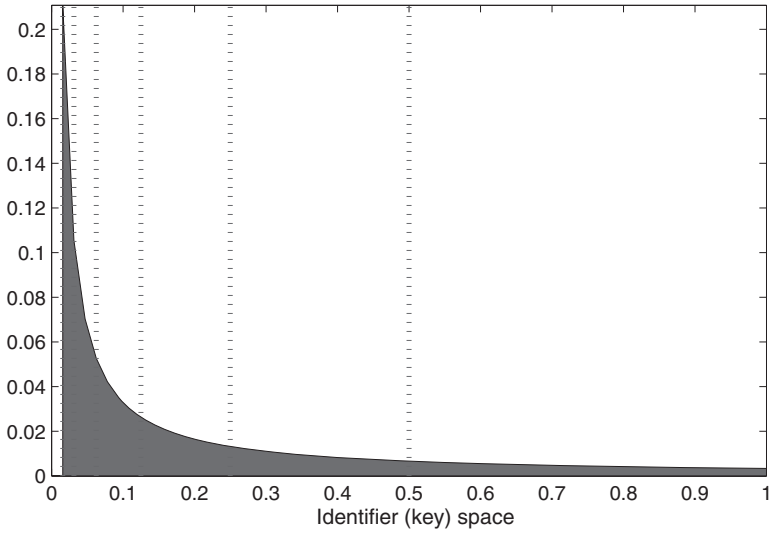


Fig. 2. pdf $g(x)$ (solid bars) and the $A_1, A_2, \ldots, A_{\log_2 N}$ partitions separated by the dotted lines.

Indeed, the probability that v will be chosen by u in some interval A_i is exactly $\frac{1}{\log N}$ and does not depend on i:

$$P(F_{\mathcal{P}}(v) \in A_i) = \int_{2^{i-\log N-1}}^{2^{i-\log N}} \frac{1}{x \ln N} dx = \frac{1}{\log N}. \tag{2}$$

In practice choosing nonuniformly at random, but according to some continuous pdf, is complicated. Thus, Equation (2) gives us an insight into how to modify a network construction algorithm, in which the neighbors will be chosen not directly by some continuous pdf $g(x)$, but uniformly at random in certain regions derived from $g(x)$. That is, if each peer u first chooses uniformly at random one logarithmic partition and then within that partition uniformly at random one peer v as a long-neighbor, then none of the pdf characteristics will be violated and all the desirable properties of the routing-efficient network will be preserved. Of course, this approach perfectly fits the case with uniform key-distributions. In such cases the partitions can be recalculated in advance at each peer. However, when assuming skewed key-spaces, it is not straightforward how to define logarithmic partitions, hence how to choose the long-range link.

5.1 Space Partitioning

In the case of uniformly distributed peer identifiers, the expected number of peers within some range of length d is actually equal to $d \cdot N$, assuming a unit length identifier space. Thus the division of such an identifier space into a logarithmic (base-2) number of partitions is nothing but recursively halving the peer population. That means a peer u with identifier 0 ($F_{\mathcal{P}}(u) = 0$) will define the partition A_1 that will contain half of the peer population: all the peers with

identifiers bigger than $\frac{1}{2}$, A_2 – all the peers with identifiers bigger than $\frac{1}{4}$ and smaller than $\frac{1}{2}$, and so on. This technique can be easily adapted to a case with any identifier skew, so Oscar uses this intuition in order to build its routing network in a simple and efficient manner. Instead of using the predefined borderlines between the logarithmic partitions we will use the median values of the exponentially decreasing peer populations. That is, an Oscar node u with an identifier u_{id} has to partition the identifier space into logarithmic partitions $A_1, A_2, ..A_{\log_2 N}$. Each border between neighboring partitions is determined by a median value of the peer identifiers in the exponentially decreasing subsets of peer population—the border between A_1 and A_2 will be the median m_1 of the peer identifiers from the whole peer population $\mathcal{P}$, the border between A_2 and A_3 will be the median m_2 of the identifiers from the subpopulation $\mathcal{P} \setminus A_1$ and so on. In general the border value between A_i and A_{i+1} will be the median m_i of peer identifiers from the subpopulation $\mathcal{P} \setminus B_i$, where $B_i = \cup_{j=1}^{i-1} A_j$. Ideally the first partition A_1 has to contain $\frac{1}{2}$ of the initial population, A_2 has to contain $\frac{1}{4}$ and so on. Since in practice it is not possible to know the precise members of all the partitions, an Oscar node has to approximate the key range for each partition. For finding the median values, an Oscar node has to uniformly sample each subpopulation B_i and determine the current median m_i from the acquired sample set. The random sampling technique proposed by Mercury for sampling the whole population is employed. To sample the subsets of the population B_i the Oscar nodes use random walkers that do not visit nodes with identifiers not belonging to the current population B_i. Our simulation experiments show that such a technique yields very good results in practice even with very low sample sizes.

5.2 Oscar Technique

Here we introduce the basis of Oscar's technique—the long-range link acquiring procedure: each peer u first chooses uniformly at random one logarithmic partition A_i and then within that partition uniformly at random one peer v. This peer v will become a long-range neighbor of u. Thus, for successful building of an overlay we only require each peer to have a snapshot of the current key distribution by acquiring the knowledge of the positions and sizes of the corresponding partitions $A_1, A_2, ..A_{\log_a N}$—a list of peer keys $F_P(p^{m_1}), F_P(p^{m_2}), \ldots, F_P(p^{m_{\log_a N}})$, where peers $p^{m_1}, p^{m_1}, \ldots, p^{\log_a N}$ represent the boundaries between the partitions. Such knowledge can be gained either by actively sampling the network (see Section 5.3) or by copying a snapshot of a global view from a ring neighbor. The sampling and long-range link creation process is illustrated in Figure 3. In case of copying the snapshot has to be adjusted accordingly by contacting the peers $p^{m_1}, p^{m_1}, \ldots, p^{\log_a N}$ and requesting the keys of their ring neighbors $F_P(p_{succ}^{m_1}), F_P(p_{succ}^{m_2}), \ldots, F_P(p_{succ}^{m_{\log_a N}})$, which are then included in the snapshot. Such copying incurs contacting only $O(\log N)$ peers and is very suitable for the propagation of the latest knowledge of the key distribution.

The network remains correctly wired if the global key distribution remains stable over time. However, even if the key distribution is changing, the peers

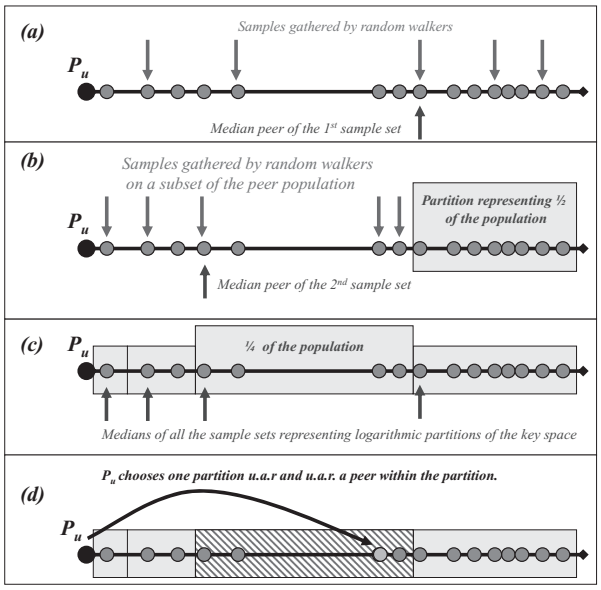


Fig. 3. Sampling and long-range link creation in Oscar. Peer P_u samples the entire population with the help of random walkers and chooses a median peer from the sample set (part (a)). The chosen median will represent the boundary in the key space by halving the space into two equal population regions. The 2nd wave of random walkers is issued on the subpopulation for which P_u belongs to, and the 2nd median peer is determined, representing a boundary of a new partition (part (b)). Recursively the boundaries of all $O(\log N)$ partitions are found (part (c)). To select a long-range neighbor peer, P_u first chooses u.a.r. one partition and u.a.r. a peer within that partition (part (d)).

can rewire only on demand when a network's performance starts to deteriorate. One of the indications that a network is not healthy is high in-degree Girdzijauskas et al. [2006] of some peers. This is an indication that a majority of the peers pointing to an overloaded peer have an outdated global view and wrong distribution estimation. Another indication for outdated knowledge is an increased average routing time. Therefore, the thresholds for the expected long-range link connectivity requests and the expected routing path lengths can be set for triggering the resampling and rewiring processes at every peer once these thresholds are exceeded. Such actions bring the connectivity of the network to the optimal state. Alternatively, if the network is required to be optimal at all times, a periodic resampling and rewiring strategy can be employed, which would always keep the network in a healthy condition. We show in our simulations that the Oscar sampling technique is not expensive and adapts the network well to the dynamic environments where peer-key distribution is not stable and changes continuously over time. Moreover, in our analysis (see online Appendix A) we prove that the error within the partitions can be relatively large without inflicting considerable damage on the search efficiency—a network can support quite high variation in the distribution without actively sampling the network.

Since Oscar is an instance of randomized small world networks—the number of long-range links in Oscar is not restricted and can be assigned individually according to the needs of a particular peer, as long as there exists at least one such link per peer. By allowing different in/out degree at each peer, Oscar can easily adapt to heterogeneous environments (workload of a peer or local available resources) still guaranteeing efficient global search as long as there is at least one long-range link maintained per peer. Our simulations show (Section 6) that Oscar performs in heterogeneous environments as good as in homogeneous.

As for the correctness of the system, we rely on the already devised self-stabilizing algorithms (e.g., Ghodsi [2006], Angluin et al. [2005], Li et al. [2004], Liben-Nowell et al. [2002], Shaker and Reeves [2005], and Stoica et al. [2001]), which maintain the virtual ring (short-range links) under churn. The establishment of short-range links ensures correctness of the greedy routing algorithm,[2] while long-range links are peculiar to different approaches and serve as routing-optimization links. Thus, we will focus mainly on the algorithms for establishing long-range links.

5.3 Oscar Algorithms

Here we will formally describe the *Oscar* network construction and maintenance algorithms.

The Join *Algorithm*. In *Oscar*, as in many other P2P approaches, to join the network a peer u has to know at least one peer already present in the system and to contact it. The joining peer is assigned some identifier $F_{\mathcal{P}}(u)$, which is usually based on the distribution of the global data in the peer-to-peer system (see the discussion in Section 4) and depends on load balancing algorithms which are orthogonal to our work, for example, in Ganesan et al. [2004], Karger and Ruhl [2004], Rao et al. [2003], Godfrey et al. [2004], and Giakkoupis and Hadzilacos [2005]. A naive approach for $F_{\mathcal{P}}(u)$ acquisition could be the following: a newly joined peer contacts several randomly selected peers, requests the information on their load, and joins as a ring neighbor (acquires $F_{\mathcal{P}}(u)$) next to the most overloaded peer. In such a way, the newcomer relieves the overloaded peer by taking over part of the identifier space it had to manage.

Upon joining the network, u issues a query with its identifer $F_{\mathcal{P}}(u)$ and it inserts itself into the unit ring between the responsible peer for peer u's key $F_{\mathcal{P}}(u)$ peer and its successor. Every peer keeps an estimated state of the global view as a set of pointers to the peers that mark the boundaries of the logarithmic partitions (see Subsection 5.1). A peer u learns about the current key distribution in the network from its immediate neighbor $u_{successor}$ by copying its snapshot set of the global-view pointers. Afterwards peer u establishes l long-range links using the *longRangeLink* algorithm (Algorithm 3).

[2]Establishment of short-range links results in a virtual ring topology, which ensures the correctness of the greedy routing algorithm (a message *always* can be forwarded closer to a target).

Algorithm 1. Scalable sampling algorithm for learning the key distribution *scalable Sampling*(u).

1: $\rho(u) = \emptyset; i = 0;$
2: $range = \left[F_{\mathcal{P}}(u_{successor}); F_{\mathcal{P}}(u)\right);$
3: $notEnoughPartitions = true;$
4: **while** *notEnoughPartitions* **do**
5: $\quad i = i + 1; P_{sample} = \emptyset;$
6: $\quad$ **for** j=1 to k **do**
7: $\quad\quad P_{sample} = P_{sample} \cup randomBoundedWalk(u, range, TTL)$
8: $\quad$ **end for**
9: $\quad m(i) = medianByF_{\mathcal{P}}(P_{sample})$
10: $\quad$ **if** $m(i) = F_{\mathcal{P}}(u_{successor})$ **then**
11: $\quad\quad notEnoughPartitions = false;$
12: $\quad$ **end if**
13: $\quad$ **if** i=1 **then**
14: $\quad\quad partitionsStart(u, i) = m(i); partitionsEnd(u, i) = u;$
15: $\quad$ **else**
16: $\quad\quad partitionsStart(u, i) = m(i); partitionsEnd(u, i) = m(i - 1)$
17: $\quad$ **end if**
18: $\quad range = \left[F_{\mathcal{P}}(u_{successor}); F_{\mathcal{P}}(m(i))\right);$
19: **end while**

The ScalableSampling *Algorithm.* In case a peer u has an outdated snapshot of the key distribution it can acquire an up-to-date one by using Oscar's scalable sampling technique. It has to determine $O(\log_2 N)$ logarithmic partitions (ranges) in the identifier space using the *scalableSampling* algorithm (Algorithm 1). To find the first partition the algorithm starts by issuing k random walkers within the defined *range* of the identifer space using the *randomBoundedWalk* algorithm (Algorithm 2). Initially the defined range spans the whole identifier space starting from the identifier of peer u's successor on the identifier ring up to the peer u's identifier itself (Algorithm 1, line 2). After the collection of the random samples in the set P_{sample} the peer u finds the median value of all the peer identifiers of the set P_{sample} (line 9). Having the median value the peer u can define the first, furthest, partition A_1, which will span the identifier space from the found median value up to the peer u's identifer value (line 14). The range value for performing the next random walk within the subgraph $\mathcal{P} \setminus A_1$ is reduced (line 18) and the algorithm continues by repeating the same steps (lines 4–19) to find the successive partitions A_2, A_3, ... and so on. The algorithm stops finding the partitions when the median value is equal to the identifier of the u's successor $F_{\mathcal{P}}(u_{successor})$ (line 10). In such a way the algorithm acquires on expectation, $\log_2 N$ partitions.

The RandomBoundedWalk *Algorithm*. For successful usage of the *scalableSampling* algorithm it is necessary to be able to sample not only the whole population of peers $\mathcal{P}$ but also some subpopulation of peers B. Therefore, a specific random walk algorithm is needed. The algorithm will produce random walkers that would be able to walk only within a subpopulation of peers $B \subset \mathcal{P}$

Algorithm 2. Bounded Random Walk Algorithm $[r] = randomBoundedWalk(u, range, TTL)$.

1: **if** $TTL > 0$ and $R \neq \emptyset$ **then**
2: $TTL = TTL - 1$
3: $R = \{p \in \rho(u) | F_{\mathcal{P}}(p) \in range\}$;
4: $next = chooseRandomly(R)$
5: $[r] = randomBoundedWalk(next, range, TTL)$
6: **else**
7: $r = u$
8: **end if**

Algorithm 3. Long range link construction algorithm $[longRangeNeighbors] = longRangeLink(u, outdegree)$.

1: **for** i=1 to outdegree **do**
2: $randPartition = \left[F_{\mathcal{P}}(partitionsStart(u,rand)); F_{\mathcal{P}}(partitionsEnd(u,rand))\right)$;
3: $[longRangeNeighbors(i)] = queryToRange(u,randPartition)$
4: **end for**

restricted by a predefined scope variable *range*, such that $p_B \in B$ if and only if $F_{\mathcal{P}}(p_B) \in range$. Such a *randomBoundedWalk* algorithm (Algorithm 2) is a modified random walker, where the message is forwarded not to any random link of the current message holder u, but to a randomly selected link p that satisfies the condition $F_{\mathcal{P}}(p) \in range$ (Algorithm 2, line 3). Such a link will always exist, assuming the underlying ring structure is in place.

The LongRangeLink *Algorithm*. To assign the long-range link, the *longRangeLink* algorithm is used (Algorithm 3), which chooses uniformly at random one of the partitions A_i (line 2) and then assigns the random peer v from that partition using the *queryToRange* algorithm (line 3). The *queryToRange* algorithm is a greedy routing algorithm, which minimizes distance to the given range A_i and terminates whenever the first peer v in that range is reached. Because of the randomized nature of the Oscar network, the probability of reaching peer v in range A_i is the same as for all the other peers in that range, which is required by the Oscar's long-range link acquisition procedure. Thus, peer v represents a random peer from that range. Since the algorithm requires k samples per each logarithmic partition, the expected number of needed samples per peer in total is $O(k \log N)$.

Note that the algorithm does not require knowledge or estimation of the total number of nodes in the network. The only place where in principle the estimation of N is needed is the *TTL* value of a random walk. As explained in Bharambe et al. [2004] the *TTL* should be set to a value of $\log_2 N$. However, the simulations show that it is sufficient to set the *TTL* value equal to the number of previously determined partitions. Likewise, newly joined peers acquire the information on the partition size from their ring neighbors (bootstrapping peers can set their initial *TTL* values to some minimal number, e.g., 20, which ensures that they will be larger than $log_2 N$ at the time of the network

bootstrap). In such a way the *Oscar* algorithms are designed to be independent of the estimation of the network size N.

Since churn exists in P2P networks and the peers join and leave the system dynamically each peer has to rewire its long range links from time to time. This can be done either periodically or adaptively. We used adaptive techniques for performing the sampling algorithms if the key distribution in the network changed considerably, and as a result some peers got overloaded and the average routing cost increased. In practice the sampling is rarely performed since, as we will show in the next section, Oscar partitioning is robust to imprecise measurements and distribution fluctuations. In such a way the Oscar system can self-optimize under dynamically changing network conditions.

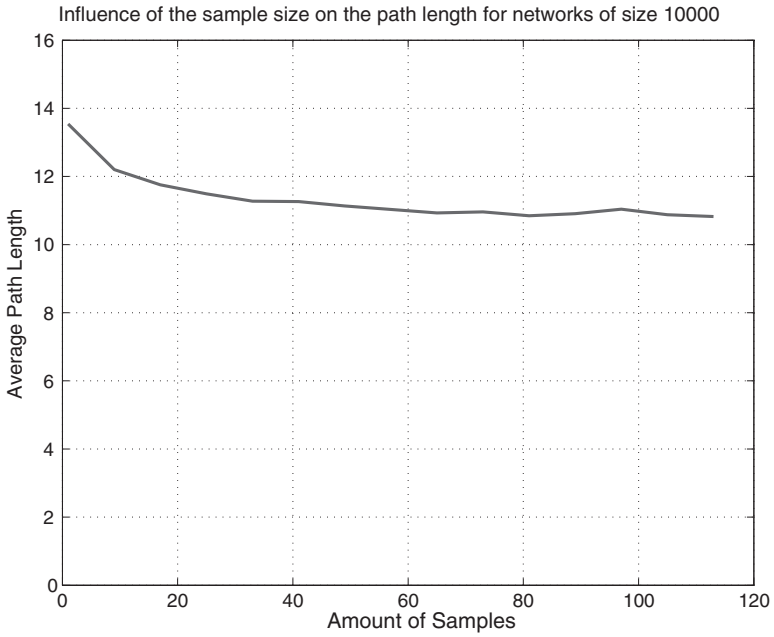
In online Appendix A we show that the Oscar overlay is robust to sampling errors and has logarithmic search performance given a logarithmic number of links per node.

6. SIMULATIONS

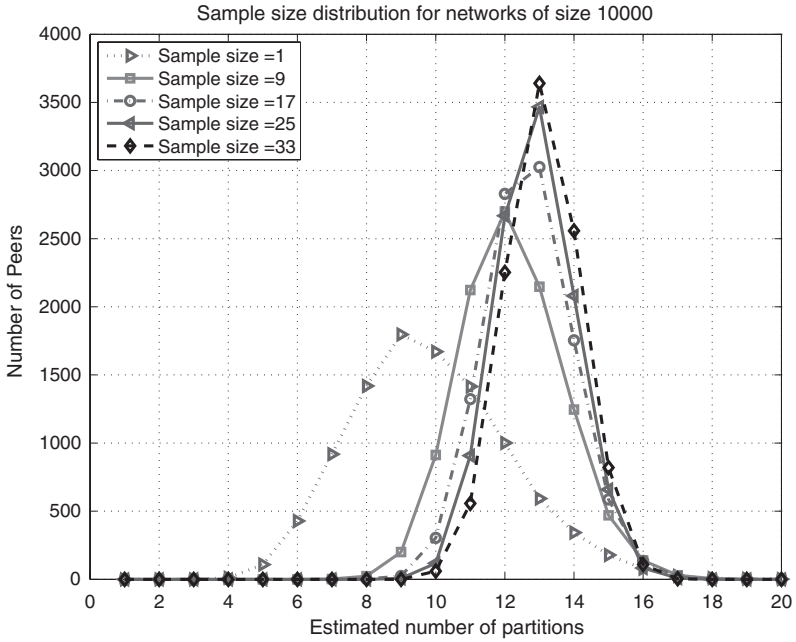
Here we show that the network built according to our proposed technique performs well and does not suffer the drawbacks of existing systems. Similarly as in Mercury [Bharambe et al. 2004], we created a discrete-event-based network simulator (in Java 1.6) where each application level hop is assigned a unit delay. Using the simulator, we simulate an Oscar network with bidirectional links starting from the network bootstrap of 2 nodes and simulating its growth until it reaches a peer population of 10000. Unless specified otherwise, we set the average node degree to 13 links per peer, Oscar sampling parameter k to 9, and use Gnutella filename trace (see Section 4) to model the key distribution in the network. We have performed the simulations under various settings, namely varying key and node degree distributions, and performed the simulations under churn, where the key distribution changes over time. In the following, we will describe the simulation settings in more detail.

6.1 Oscar's Performance with Low Sample Sizes

First we have investigated the optimal parameters for an Oscar overlay. As we show in online Appendix A the search performance of Oscar should be sufficiently efficient even with very inaccurate estimations of the medians—with very small sampling parameters k. Hence we measure the effect of the size of the sampling parameter k on Oscar's search performance. We have grown an Oscar network from scratch to 10000 peers with an average node degree of 7 and with different values of k. We compared the search performance (Figure 4(a)) together with the distribution of the estimated number of partitions by each peer (Figure 4(b)). The latter suggests how accurate the measurements are since the perfect estimation would result in exactly $\log N$ logarithmic partitions (for $N = 10000$, $\log N \approx 13$). From the experiments we can see that even with very low values of k, Oscar peers could estimate well the existing key distribution in the network (in terms of determining the logarithmic partitions), hence the search cost did not deteriorate much even with sampling values as low as $k = 1$. The average search costs given $k = 1$ and $k = 100$ differ by only 2.5 hops. This



(a) Search performance of Oscar network given different sampling parameters k



(b) The distribution of number of estimated logarithmic partitions at each peer

Fig. 4. Performance of the networks with various sample parameters k.

is a clear indication of the robustness of the small-world networks built using the Oscar technique.

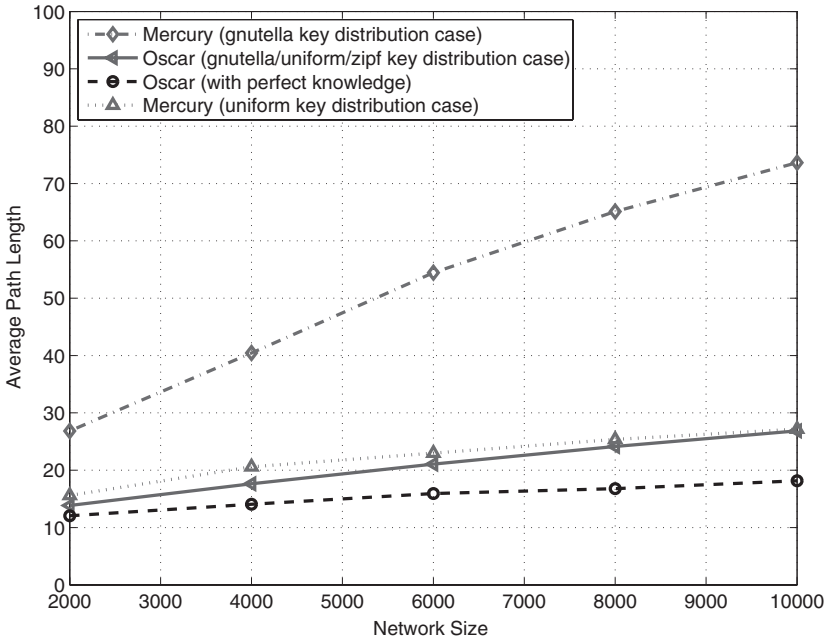
6.2 Oscar vs. Mercury

As suggested in Mercury [Bharambe et al. 2004], we have set Mercury's parameters k_1 and k_2 to $\log_2 N$ for constructing a Mercury network. Each peer in our Mercury simulation constructed a distribution approximation from the sample set of $k_1 \cdot k_2$ random walks. In this way we simulated the exchange of Mercury's distribution estimates in an epidemic manner where each peer issued k_1 random walks and each of the selected nodes reported back the k_2 most recent estimates. Thus, to sample the network, each Mercury peer had to issue $\log_2^2 N$ random walkers. We set the number of random walkers in Mercury to 169 per peer and Oscar's sampling parameter k to 9, which results in the average number of 108 samples per peer for networks of size 10000. With such a setting it is ensured that the sampling parameters in Oscar are not larger than in Mercury, which provides fair simulation conditions.

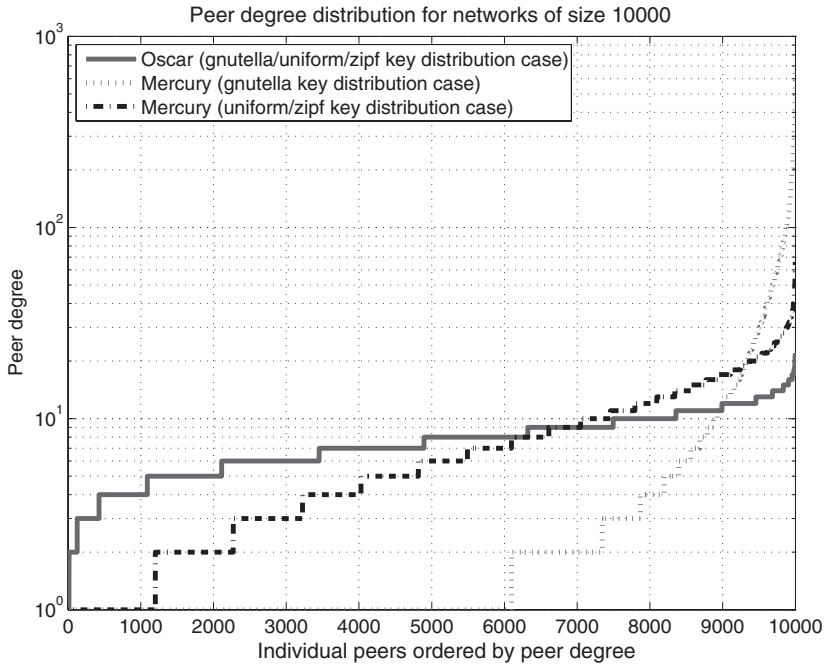
Each peer p in the network had values $\rho(p)$ and $\rho^{max}(p)$ as the preferred and maximal allowed degrees of a peer. During the network construction procedure each peer p was trying to establish $\rho(p)$ bidirectional connections to other peers using long-range links. However, only peers that had less than $\rho^{max}(p)$ degree acknowledged becoming peer p's neighbors. This allowed individual peers to autonomously determine their degree and thus the load incurred by them for network maintenance and query traffic. Since both Oscar and Mercury are randomized overlay networks we could employ the power-of-two choice technique [Mitzenmacher et al. 2001] to better load-balance the degree distribution among the peers. During the growth of the networks we were periodically rewiring long-range links of all the peers and measuring the performance of the current network.

Different Key Distributions. Since our goal is to show that our proposed technique results in routing-efficient networks, and dealing with churn is an orthogonal issue, we have simulated a fault-free environment—a system where peers do not crash. We simulated three cases of key distribution: uniform, monotonous Zipf (with parameter $\alpha = 1$) [Breslau et al. 1999; Zipf 1929], and Gnutella filename (as in Figure 1). In each case a peer joining the network was assigned an identifier randomly drawn from the corresponding key distribution. We compare the simulation results also with a perfect case, Kleinbergian small-world network [Girdzijauskas et al. 2005; Manku et al. 2003] built based on perfect knowledge of the global key-distribution. The preferred degree $\rho(p)$ was set to 7 links for every peer. To show the actual capacity of every system to construct routing-efficient overlay networks we did not limit the maximum degree value $\rho^{max}(p)$ for any peer. We have measured the performance of the resulting networks; specifically, the average routing cost and the average node degree.

As expected, the simulations showed that Mercury performed well given uniform and monotonous skews, but poorly given a complex Gnutella distribution (Figure 5(a)). In contrast the Oscar network resulted in a much more efficient



(a) Search performance of Mercury and Oscar given various key distributions



(b) Distribution of node degree with Gnutella key distribution

Fig. 5. Simulation results.

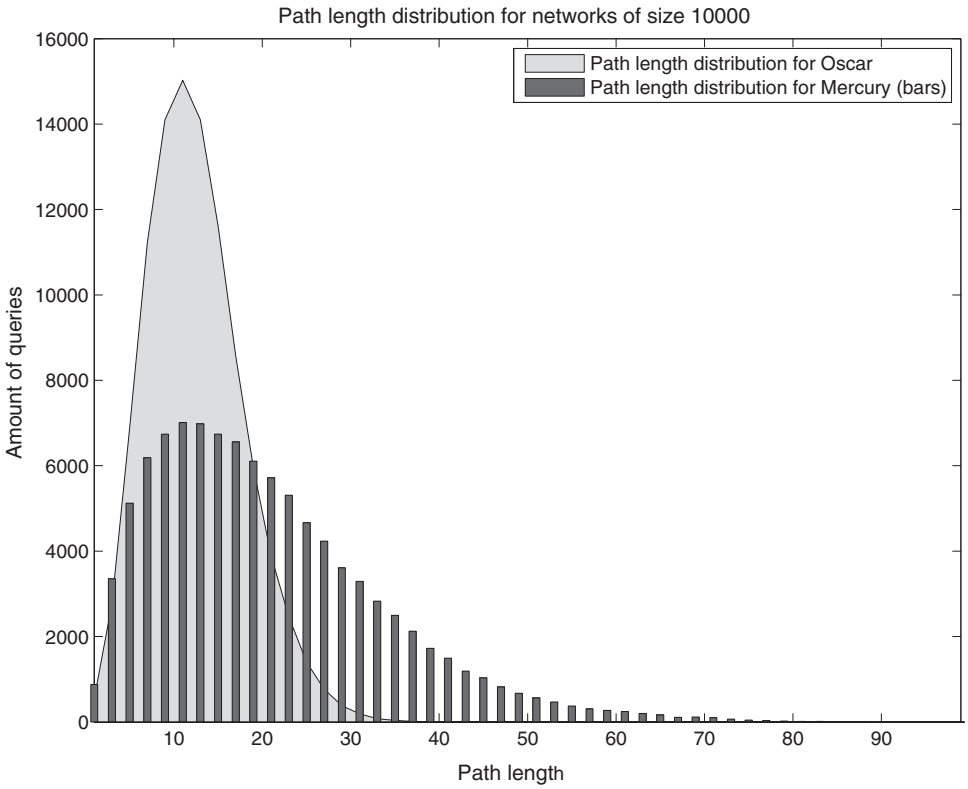


Fig. 6. Distributions of search costs with Gnutella key-distribution.

network for highly complex key distributions. Figures 5(b) and 6 indicate that the Oscar network had a much better distribution of node degree and lower message cost for the case of the Gnutella key distribution. In contrast, the Mercury overlay could not cope with the complex distribution of the keys and had significantly higher node degree imbalance, which in turn resulted in poorer lookup performance. As expected the results have shown that Oscar is robust for realistic skews in the key distribution.

Different Node Degree Distributions. Heterogenous Peers. We have also shown by simulation that the Oscar technique results in routing-efficient networks not only given homogeneous peers but also assuming node-degree heterogeneity. We performed simulations of Oscar given three different node degree distributions: realistic, linear, and constant. In the realistic node degree distribution case the maximum degree value ρ^{max} of each peer was drawn from a predefined synthetic spiky distribution (Figure 7) to emulate the behavior of real P2P systems [Stutzbach et al. 2005]. To match the data from Stutzbach et al. [2005], we chose 13 bidirectional links as the mean degree. In the linear node degree distribution case, for each peer the ρ^{max} value was drawn uniformly at random from the range of 6 to 20. In the constant degree distribution case, for all peers, ρ^{max} was set to 13. Note that for all the aforementioned cases the average node

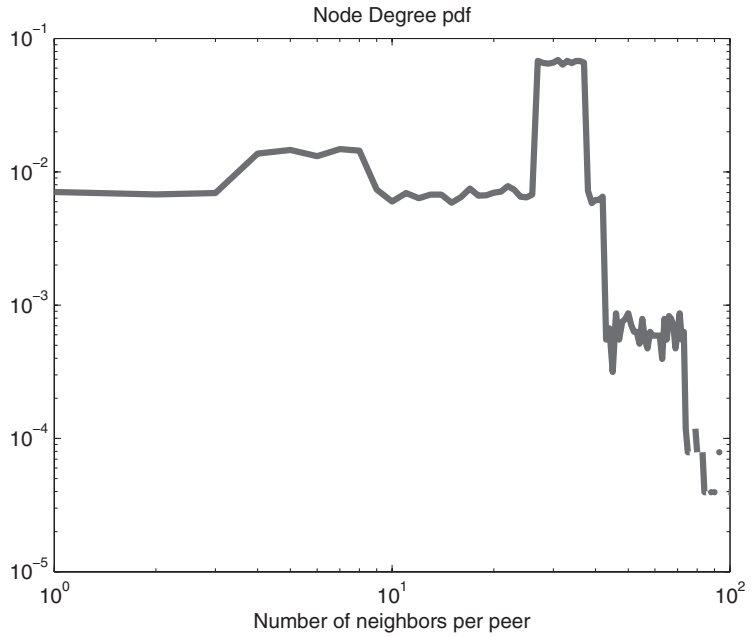
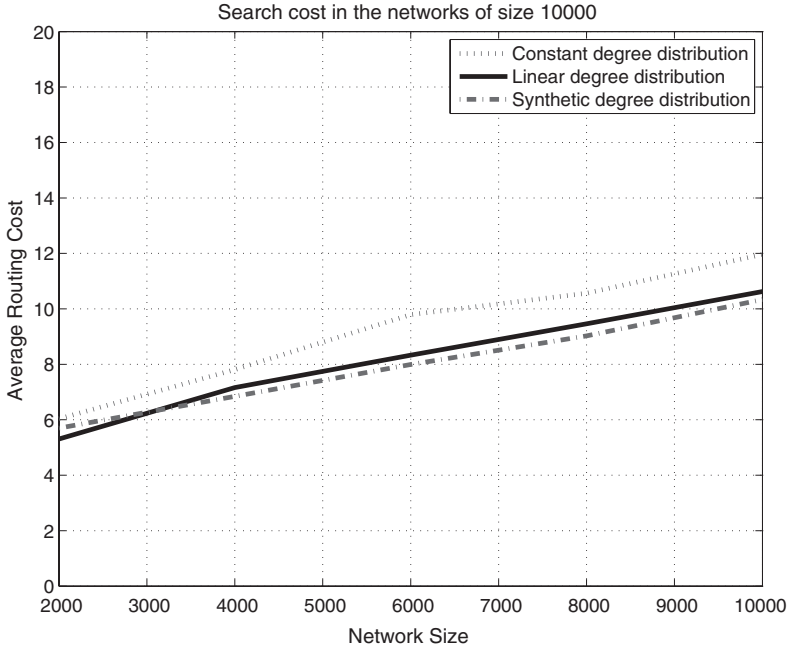


Fig. 7. Synthetic spiky node degree distribution.

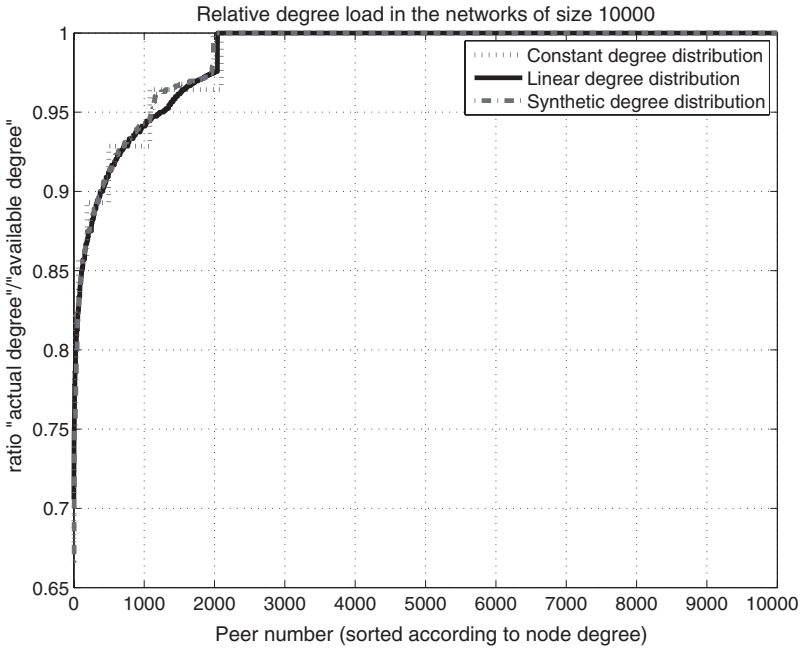
degree remained 13. The keys for the peers were drawn from the Gnutella filename distribution.

After performing network construction, the results showed that Oscar performed almost identically for all the degree distribution cases (Figure 8(a)). This shows that Oscar can easily adapt to various degree distributions without any loss in search performance. To measure how well the potential network connectivity is exploited we calculate for each peer p_i, the ratio $\frac{\rho(p_i)}{\rho^{max}(p_i)}$ between the actual peer degree and the available (maximal) peer degree. In Figure 8(b) we can see that the node degree distribution ratio was very similar in all three cases and exploited around 98% of available degree volume ($\frac{100}{N}\sum_i \frac{\rho(p_i)}{\rho^{max}(p_i)}$) in the system of 10000 peers. We also observed in our experiments that in the Mercury network with the same setting and constant node degree distribution, only 61% of available degree volume was exploited and the Mercury network had an average search cost of 27.3 routing hops per query. The inability of Mercury to exploit the full capacity of the heterogeneous environment is due to the fact that many Mercury peers are forced to drop some of their links because of the node-degree overload (when Mercury protocol requires peer p to acquire more than $\rho^{max}(p)$ links), caused by wrongly estimated peer key distribution. Since Mercury could acquire fewer links than Oscar, naturally the search cost in Mercury was higher compared to Oscar.

Oscar Under Churn. Since the data stored on the peers is not static but dynamic, it is expected that because of the storage load-balancing, the peer key distribution will be changing as well. To investigate the robustness of the Oscar network under churn we performed simulations of our system in a dynamic



(a) Search performance of Oscar given various node degree distributions



(b) Relative degree load in Oscar peers

Fig. 8. Oscar's performance given various key and node degree distributions.

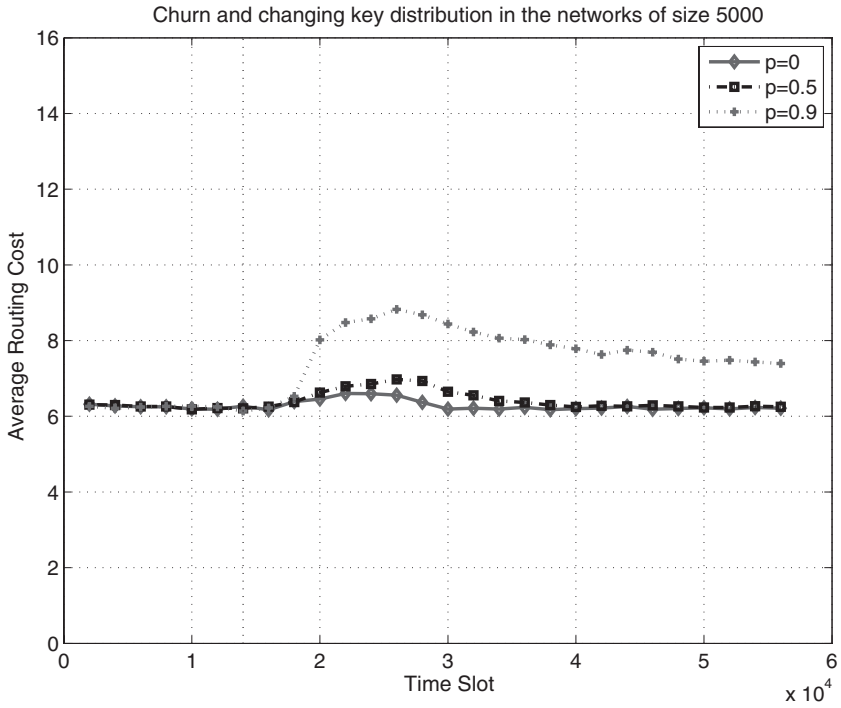


Fig. 9. Routing cost of Oscar under churn.

environment where the key distribution changes over time. We have modeled a volatile transition from a simple uniform key distribution to a Gnutella filename distribution. Our simulation starts with the Oscar overlay of 5000 peers with the uniform key distribution. We model a stable churn rate where in every time slot a peer joins or leaves the network with 50% probability. The simulation has two phases. In the first, in the arriving peers acquire a key drawn from a uniform distribution. After some time the second phase starts and the new peers start acquiring the keys drawn from the Gnutella filename distribution. We perform the measurements at every time step and measure the average lookup length in the network. Upon joining the network, a peer has two options for acquiring the global view of the distribution function: (1) by copying the estimated boundaries of the logarithmic partitions from a ring neighbor; (2) estimating by using sampling. In our simulations a peer chooses with the probability p option (1) and with the probability $(1-p)$ – option (2). We run our simulations with three different settings, where $p = 0$, $p = 0.5$, and $p = 0.9$. In Figure 9 the first dotted vertical line marks the starting time of the 2nd phase (the arriving peers start acquiring keys from the Gnutella filename distribution). As expected, the network adapts very quickly to the dynamic churn when $p = 0$, but even with $p = 0.9$ (the network is sampled only by 10% of the peers), the search cost is still relatively low. This shows that Oscar overlay can sustain efficient routing properties under volatile and dynamic network conditions, and changing load-skews.

7. CONCLUSIONS

In this article, we have addressed the problem of dealing with skewed key distributions as encountered in data-oriented applications, in a realistic P2P environment characterized by churn and heterogeneity. We have shown that current approaches cannot cope successfully with such complex workload distributions. Furthermore most existing structured overlay designs do not deal with peer heterogeneity, and instead assume homogeneity and aim at achieving load-balancing. We take a more pragmatic look at the problem. In deployed (unstructured) overlays, it has been observed that contributions made by participating peers have large variations, and are decided autonomously by peers subject to their own physical constraints. The ability to use load-balancing schemes in the presence of peer heterogeneity and autonomy is an impractical ideal. What is pragmatic is instead to respect peers' autonomy to exploit whatever is available from each of these peers to fulfill the system's needs adequately and efficiently. External mechanisms based on incentives or punishments to achieve load-balancing in a manner where peers are individually deciding to contribute to the system equally is an orthogonal issue, and will work well in our settings. The system design is founded in fundamental understanding of the small-world networks, and applicability of the principles are validated with simulation experiments. In the actual implementation, our system is able to reuse a lot from fine-tuned implementations of ring-based overlay networks like Chord, particularly using existing self-stabilization algorithms for ring maintenance and content replication. The only required changes are the choice of the long range link based on the sampling mechanism to make these choices proposed here.

REFERENCES

Aberer, K. 2001. P-Grid: A self-organizing access structure for P2P information systems. In *Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS)*.

Aberer, K., Datta, A., Hauswirth, M., and Schmidt, R. 2005. Indexing data-oriented overlay networks. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*. VLDB Endowment, 685–696.

Angluin, D., Aspnes, J., Chen, J., Wu, Y., and Yin, Y. 2005. Fast construction of overlay networks. In *Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'05)*.

Aspnes, J., Diamadi, Z., and Shah, G. 2002. Fault-tolerant routing in peer-to-peer systems. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC)*. 223–232.

Aspnes, J., Kirsch, J., and Krishnamurthy, A. 2004. Load balancing and locality in range-queriable data structures. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 115–124.

Aspnes, J. and Shah, G. 2003. Skip graphs. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 384–393.

Barbella, D., Kachergis, G., Liben-Nowell, D., Sallstrom, A., and Sowell, B. 2007. Depth of field and cautious-greedy routing in social networks. In *Proceedings of the 18th International Symposium on Algorithms and Computation (ISAAC'07)*. 574–586.

Barrière, L., Fraigniaud, P., Kranakis, E., and Krizanc, D. 2001. Efficient routing in networks with long range contacts. In *Proceedings of the 15th International Conference on Distributed Computing (DISC'01)*. Springer-Verlag, London, UK, 270–284.

BHARAMBE, A. R., AGRAWAL, M., AND SESHAN, S. 2004. Mercury: supporting scalable multiattribute range queries. *SIGCOMM Comput. Comm. Rev. 34,* 4, 353–366.

BRESLAU, L., CAO, P., FAN, L., PHILLIPS, G., AND SHENKER, S. 1999. Web caching and Zipf-like distributions: evidence and implications. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM),* vol. 1, 126–134.

GANESAN, P., BAWA, M., AND GARCIA-MOLINA, H. 2004. Online balancing of range-partitioned data with applications to peer-to-peer systems. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB)*. VLDB Endowment, 444–455.

GHODSI, A. 2006. Distributed k-ary System: Algorithms for distributed hash tables. Ph.D. thesis, KTH—Royal Institute of Technology.

GIAKKOUPIS, G. AND HADZILACOS, V. 2005. A scheme for load balancing in heterogenous distributed hash tables. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 302–311.

GIAKKOUPIS, G. AND HADZILACOS, V. 2007. On the complexity of greedy routing in ring-based peer-to-peer networks. In *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing (PODC)*. 99–108.

GIRDZIJAUSKAS, S., DATTA, A., AND ABERER, K. 2005. On small world graphs in non-uniformly distributed key spaces. In *Proceedings of the 21st International Conference on Data Engineering Workshops (ICDEW)*. 1187.

GIRDZIJAUSKAS, S., DATTA, A., AND ABERER, K. 2006. Oscar: Small-world overlay for realistic key distributions. In *Proceedings of the 4th International Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*.

GODFREY, B., LAKSHMINARAYANAN, K., SURANA, S., KARP, R., AND STOICA, I. 2004. Load balancing in dynamic structured P2P systems. In *Proceedings of IEEE INFOCOM*.

GUERRAOUI, R., HANDURUKANDE, S. B., HUGUENIN, K., KERMARREC, A.-M., FESSANT, F. L., AND RIVIERE, E. 2006. Gosskip, an efficient, fault-tolerant and self organizing overlay using gossip-based construction and skip-lists principles. In *Proceedings of the IEEE International Conference on Peer-to-Peer Computing* 12–22.

HARVEY, N. J. A., JONES, B., M., SAROIU, S., THEIMER, M., AND WOLMAN, A. 2003. Skipnet: A scalable overlay network with practical locality properties. In *Proceedings of the USENIX Symposium on Internet Technologies*.

HELLERSTEIN, J. M. 2003. Toward network data independence. *SIGMOD Rec. 32,* 3, 34–40.

HOEFFDING, W. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Statis. Assoc. 58*, 13–30.

HUI, K. Y., LUI, J. C., AND YAU, D. K. October, 2006. Small-world overlay p2p networks: Construction and handling dynamic flash crowd. *Comput. Netw. J. 50*, 15.

KARGER, D. R. AND RUHL, M. 2004. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proceedings of the 16th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA'04)*. 36–43.

KLEINBERG, J. 2000. The small-world phenomenon: an algorithm perspective. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC'00)*. 163–170.

LI, X., MISRA, J., AND PLAXTON, C. G. 2004. Active and concurrent topology maintenance. In *Proceedings of the 18th Annual Conference on Distributed Computing (DISC)*. Springer, 320–334.

LIBEN-NOWELL, D., BALAKRISHNAN, H., AND KARGER, D. 2002. Analysis of the evolution of peer-to-peer systems. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing (PODC)*. 233–242.

MANKU, G. S., BAWA, M., RAGHAVAN, P., AND INC, V. 2003. Symphony: Distributed hashing in a small world. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*. 127–140.

MANKU, G. S., NAOR, M., AND WIEDER, U. 2004. Know thy neighbor's neighbor: the power of lookahead in randomized p2p networks. In *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC'04)*. 54–63.

MITZENMACHER, M., RICHA, A. W., AND SITARAMAN, R. 2001. The power of two random choices: a survey of techniques and results. In *Handbook of Randomized Computing*, Kluwer, 255–312.

NOVAK, D. AND ZEZULA, P. 2006. M-chord: a scalable distributed similarity search structure. In *Proceedings of the 1st International Conference on Scalable Information Systems (InfoScale)*. 19.

RAO, A., LAKSHMINARAYANAN, K., SURANA, S., KARP, R., AND STOICA, I. 2003. Load balancing in structured p2p systems. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SCHENKER, S. 2001. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev. 31,* 4, 161–172.

ROWSTRON, A. AND DRUSCHEL, P. 2001. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*.

SHAKER, A. AND REEVES, D. S. 2005. Self-stabilizing structured ring topology p2p systems. In *Proceedings of the IEEE International Conference on Peer-to-Peer Computing*. IEEE Computer Society, 39–46.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. 149–160.

STUTZBACH, D., REJAIE, R., AND SEN, S. 2005. Characterizing unstructured overlay topologies in modern p2p file-sharing systems. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC)*. USENIX Association, Berkeley, CA, 5–5.

ZIPF, G. K. 1929. Relative frequency as a determinant of phonetic change. *Harvard Studies Classical Philology 40*, 1–95.

Received June 2008; revised March 2009; accepted July 2009