Computer Science department, Boston University
Tech. Rep. BUCS-TR-2007-006
Superceded by BUCS-TR-2007-013; "EGOIST : Overlay Routing using Selfish Neighbor Selection"

Improving the Performance of Overlay Routing and P2P File Sharing using Selfish Neighbor Selection

Georgios Smaragdakis
BOSTON UNIVERSITY
gsmaragd@cs.bu.edu

Nikolaos Laoutaris
HARVARD UNIVERSITY
nlaout@eecs.harvard.edu

Azer Bestavros
BOSTON UNIVERSITY
best@cs.bu.edu

John Byers
BOSTON UNIVERSITY
byers@cs.bu.edu

Mema Roussopoulos
HARVARD UNIVERSITY
mema@eecs.harvard.edu

ABSTRACT

A foundational issue underlying many overlay network applications ranging from routing to P2P file sharing is that of connectivity management, *i.e.*, folding new arrivals into the existing mesh and re-wiring to cope with changing network conditions. Previous work has considered the problem from two perspectives: devising practical heuristics for specific applications designed to work well in real deployments, and providing abstractions for the underlying problem that are tractable to address via theoretical analyses, especially game-theoretic analysis. Our work unifies these two thrusts first by distilling insights gleaned from clean theoretical models, notably that under natural resource constraints, selfish players can select neighbors so as to efficiently reach near-equilibria that also provide high global performance. Using EGOIST, a prototype overlay routing system we implemented on PlanetLab, we demonstrate that our neighbor selection primitives significantly outperform existing heuristics on a variety of performance metrics; that EGOIST is competitive with an optimal, but unscalable full-mesh approach; and that it remains highly effective under significant churn. We also describe variants of EGOIST's current design that would enable it to scale to overlays of much larger scale and allow it to cater effectively to applications, such as P2P file sharing in unstructured overlays, based on the use of primitives such as scoped-flooding rather than routing.

[1]This work was supported in part by a number of NSF awards, including CISE/CSR Award #0720604, ENG/EFRI Award #0735974, CISE/CNS Award #0524477, CNS/NeTS Award #0520166, CNS/ITR Award #0205294, CISE/EIA RI Award #0202067, CAREER Grant #0446522, and the European Commission (Contract No. RIDS-011923).

1. INTRODUCTION

Motivation: Overlay networks are used for a variety of applications ranging from routing [4], to content distribution [47], to peer-to-peer file sharing [1, 2]. A foundational issue underlying many such overlay network applications is that of connectivity management. Connectivity management manifests itself in many ways, including how to wire a newcomer into the existing mesh of nodes (bootstrapping) and how to rewire the links between overlay nodes to deal with churn and changing network conditions. Connectivity management is particularly challenging for overlay networks because overlays often consist of nodes that are distributed across multiple administrative trust domains where auditing or enforcing global behavior can be difficult or impossible. As such these nodes may act selfishly to maximize the benefit they receive from the network. Such selfish behaviour has been well studied in the context of selfish routing [32] and free riding [14].

Selfish behavior has many implications also for connectivity management. In particular, it creates additional incentives for nodes to rewire, not only for operational purposes (bootstrapping and rewiring), but also for the purpose of incrementally maximizing the utility that nodes derive from the overlay. While much attention has been paid to the harmful downsides of selfish behavior, the impact of adopting selfish connectivity management techniques in *real* overlay networks has received very little attention. In our work, we dwell not on the negatives, but instead focus on the potential benefits from selfish behavior, which include the obvious benefits to selfish nodes, but more surprisingly, to the network as a whole.[1] Indeed, we confirm that selfishness is not the problem, so much as inaction, indifference, or naive reaction: all of which incur high social costs. Our paper addresses these issues by providing a methodical evaluation of the design space for connectivity management in overlay networks, including the demonstration of the implications and promise from adopting a selfish approach to neighbor selection in real network overlays.

Selfish Neighbor Selection: In a typical overlay network, a node must select a fixed number (k) of immediate overlay neighbors for routing traffic or queries for files. Previous work has considered this problem from two perspectives: (1) devising *practical heuristics* for specific applications in real deployments, such as bootstrapping by choosing the k closest links, or by choosing k random links in a P2P file-sharing system; and (2) providing abstractions of the underlying fundamental neighbor selection problem, which are amenable to *theoretical formulation and analysis* as exemplified in the recent work on Selfish Neighbor Selection (SNS) [20, 19]. This SNS formulation focused on characterizing the emergent overlay topology when overlay nodes behave selfishly and employ "Best-Response" neighbor selection strategies tailored to optimizing the local performance of a node. (Here, nodes choose k neighbors to optimize the connections of a node to all other nodes in a static overlay). This prior work demonstrates that selfish players can select neighbors so as to efficiently reach near-equilibria in the Nash sense, while also providing good global performance. Indeed, one implication from that prior work is that SNS may result in improved routing performance, with positive implications for many overlay applications. Left unanswered by this work is whether it is practical to build SNS-inspired overlays, whether the benefits from doing so would extend to objectives that go beyond routing, and whether such overlays would scale.

Paper Scope and Contributions: In this paper we tackle the questions mentioned above and describe the design, implementation, and evaluation of EGOIST: an SNS-inspired protoype *overlay routing network* for PlanetLab. EGOIST serves as a building block for the construction of efficient and scalable overlay applications consisting of (potentially) selfish nodes. EGOIST delivers the high global performance promised by analysis, while at the same time incurring reasonable compuational and information exchange overheads. We also describe a number of variants of our design that allow us to tackle practical issues, including churn, large scale, and the consideration of objectives that are not limited to unicast overlay routing performance.

[1]Rational egoists believe that it is rational to act in one's own self-interest, and for the applications we consider, we demonstrate that this philosophy is not far off.

Our contributions are best summarized based on the type of applications that EGOIST would support. For applications that utilize shortest-path based routing services, we first demonstrate through real measurements on PlanetLab that overlay routing atop EGOIST is significantly more efficient than that by systems utilizing common heuristic neighbor selection strategies under multiple performance metrics, including delay and available bandwidth. Second, we demonstrate that the performance of EGOIST approaches that of a (theoretically-optimal) full-mesh topology, while achieving superior scalability, requiring $O(nk)$ link announcements compared with the $O(n^2)$ required in a full mesh. Third, to accomodate high-churn environments, we introduce a hybrid extension of the "Best-Response" neighbor selection strategy, in which nodes "donate" a portion of their k links to the system to assure connectivity, leaving the remaining links to be chosen selfishly by the node. Our experiments show that such an extension is warranted, especially when the churn rate is high relative to the size of the network.

While many traffic routing overlays aim to maximize performance based on finding "shortest paths" between nodes, unstructured file sharing overlays typically route queries using scoped-flooding or random walks. Thus, for applications that do not utilize shortest-path based routing services, and to address the question of whether Best-Response approaches can successfully be applied in an *unstructured peer-to-peer file sharing* setting, we reformulate the SNS objective function in the following way: instead of minimizing delay or maximizing the effective bandwidth to achieve high-quality paths, we aim to maximize the number of distinct nodes reachable through scoped flooding. We show through detailed trace-driven simulations that our new design returns significantly improved search performance compared to existing peer-to-peer file sharing systems.

2. BACKGROUND

2.1 Basic Definitions

Let $V = \{v_1, v_2, \ldots, v_n\}$ denote a set of nodes. Node v_i establishes a *wiring* $s_i = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ by creating links to k other nodes (we will use the terms link, wire, and edge interchangeably). Edges are *directed* and *weighted*, thus $e = (v_i, v_j)$ can only be crossed in the direction from v_i to v_j, and has cost d_{ij}. Going in the opposite direction requires crossing edge (v_j, v_i) and incurring cost d_{ji} ($d_{ji} \neq d_{ij}$ in the general case). Let $S = \{s_1, s_2, \ldots, s_n\}$ denote a *global wiring* between the nodes of V and let $d_S(v_i, v_j)$ denote the cost of a shortest directed path between v_i and v_j over this global wiring; $d_S(v_i, v_j) = M \gg n$ if there is no directed path connecting the two nodes. For the overlay networks discussed here, the above definition of cost amounts to the incurred end-to-end delay when performing shortest-path routing along the overlay topology S, whose direct links have weights that capture the delay of crossing the underlying IP layer path that goes from the one end of the overlay link to the other. Let $C_i(S)$ denote the cost of v_i under the global wiring S, defined as a weighted summation of its distances to all other nodes, *i.e.*, $C_i(S) = \sum_{j=1,j\neq i}^{n} p_{ij} \cdot d_S(v_i, v_j)$, where the weight p_{ij} denotes "preference" *e.g.*, the percentage of v_i's traffic that is destined for node v_j.

DEFINITION 1. *Best-Response (BR) Given a residual wiring $S_{-i} = S - \{s_i\}$, a best response for node v_i is a wiring $s_i \in S_i$ such that $C_i(S_{-i}+\{s_i\}) \leq C_i(S_{-i}+\{s_i'\})$, $\forall s_i' \neq s_i$, where S_i is the set of all possible wirings for v_i.*

The *Selfish Neighbor Selection* (SNS) game was introduced in [20] as a strategic game where nodes are the players, wirings are the strategies, and C_i's are the cost functions. It was shown that under hop-count distance, the BR of v_i can be obtained by solving an asymmetric k-median problem on the residual wiring S_{-i}. In [19] it was proved that every instance of the SNS game with uniform preference and link weights has pure Nash equilibria whose social cost is within a constant factor away from the social cost of a socially-optimal solution. It was also shown that non-uniform instances of the game may have no equilibria at all.

2.2 Related Work

Our work is largely inspired from recent work on the SNS game [20, 19]. These works introduced the SNS game and presented basic theoretic and experimental results but did not touch on any of the practical systems issues that are covered in our work, such as dealing with churn in realistic network conditions or achieving high global performance without the computational and control message overhead required by the theory. Previous works on Network Creation Games [13, 3, 10, 30, 8] consider games in which the nodes can buy as many links (neighbors) as they like and thus differ fundamentally from ours, in which the fixed constraints on the number of neighbors have a central role. Fundamentally different is also the work on Selfish Routing [32, 37] in which the network topology is fixed, and the players aim at routing selfishly so as to avoid overloaded links.

Overlay networks have been realized in the context of two, fundamentally different settings: in *Overlay Routing Systems* (ORS) and in *Peer-to-Peer (P2P) Applications*. ORS's perform "real" routing, in the usual sense (link probing, information exchange protocols, routing algorithms), with the only difference that instead of routers, they are implemented at (typically dedicated) internet end-systems. P2P systems on the other hand, offer "poor man's" routing of requests, mostly in the context of applications for file exchange. Since P2P applications may run on under-provisioned devices, typically they cannot (or do not want to) afford the overhead of running a full routing protocol offering shortest-path or other optimized routes, and instead, they rely on simpler alternatives such as scoped-flooding and Distributed Hash Tables [43, 33, 38, 51], described below. More fundamentally, in their current form, ORSs operate under a single authority (whoever deploys them), whereas P2P systems tend to operate under multiple authorities, with potentially different interests.

Overlay Routing Systems: A number of routing overlay systems have been recently proposed [39, 4, 25, 24, 50, 26, 18, 52, 40, 41, 45, 12] Most of these works have been proposed as ways of coping with some of the inefficiencies of native IP routing. The basic design pattern is more or less the same: overlay nodes monitor the characteristics of the overlay links between them (overlay topology may differ among systems) and employ a full-fledged or simpler [18] routing protocol to route at the overlay layer. The work by Nakao et al. [31] is an exception, as it is not a full overlay routing system but rather a routing underlay intended to be used by different overlay routing systems on top of it. Some overlay routing systems optimize route hop count [24, 40, 41], others optimize for application delay [39, 4, 32, 25, 50, 26, 18], and others optimize for available bandwidth [52]. In our work, we provide mechanisms to support optimization of all three metrics and leave it up to the application designers to choose the most suitable metric.

P2P Applications: P2P overlays have been built to support a variety of applications including file-sharing, content distribution, multicast, event notification, continuous data stream querying, network monitoring, and digital libraries. In this paper, we focus on file-sharing. Previous work in this area has focused predominantly on making file search queries more efficient. As a result, we have seen a steady progression of designs from the original flooding-based unstructured networks such as Gnutella and KaZaA [27], to systems using random walks [28] followed by biased random walks [7] and structured DHTs [43, 33, 38, 51]. Random walks and biased random walks aim to eliminate the overhead of flooding messages without changing unstructured systems, while DHTs constrain the topology such that search queries follow well-defined paths with bounded lengths. These basic approaches have been extended further to increase efficiency. For example, Chawathe et al. [7] propose mechanisms for unstructured networks that make higher capacity nodes have higher degrees and thus receive more search queries leading to faster query lookups. They do not focus on selfish neighbor selection, although they do propose mechanisms for dealing with selfish nodes that lie about their capacities to avoid receiving queries. They also do not impose any constraints on node degree. In structured DHTs, proximity neighbor selection has been proposed to make the overlay topology match the underlying IP topology as much as possible [34, 17] to achieve faster lookups. Nodes can choose the physically closest nodes from a set of candidate nodes. While this give nodes some flexibility to choose neighbors selfishly, the set of nodes from which they can choose is constrained by node ID. In our work, we focus on unstructured networks and do not constrain a node's choices except to limit its degree, a practice in line with all currently deployed unstructured networks.

Recent work has focused on minimizing the effect of network churn, *i.e.*, the continuous arrivals and departures of nodes in the network [16, 35]. These studies do not consider selfish neighbor selection.

Finally, recent work has proposed mechanisms to encourage cooperative behavior in peer-to-peer file sharing networks. For example, Blanc et al. [6] propose a reputation system that provides incentives for nodes to route packets on behalf of other nodes in a peer-to-peer overlay. A number of systems have proposed file-trading strategies that discourage nodes from "free-riding behavior" (*e.g.*, [15, 9, 46]). Our work complements these studies.

3. THE EGOIST OVERLAY SYSTEM

In this section we overview the basic design of our EGOIST overlay routing system.

3.1 Basic Design of EGOIST

EGOIST is a prototype system that allows the creation and maintenance of an overlay network on PlanetLab in which every node selects and continuously updates its k overlay neighbors in a selfish manner—namely to minimize the node's routing cost. For ease of presentation, we will assume that *delay* is used to reflect the cost of a path, noting that other metrics, which we will discuss later in the paper, could well be used to account for cost, including bandwidth and node utilization, for example.

In EGOIST, a *newcomer* overlay node v_i connects to the system by querying a *bootstrap* node, from which it receives a list of *potential* overlay neighbors. The newcomer connects to at least one of these nodes, enabling it to participate in the link-state routing protocol running at the overlay layer. As a result, after some time, v_i will obtain the full residual graph G_{-i} of the overlay. By running all-pairs shortest path algorithm on G_{-i}, the newcomer is able to obtain the pair-wise distance (delay) function $d_{G_{-i}}$. In addition to this information, the newcomer estimates d_{ij}, the weight of a potential direct overlay link from itself to node v_j, for all $v_j \in V_{-i}$. Using the values for d_{ij} and $d_{G_{-i}}$, the newcomer connects to G_{-i} using one of a number of wiring policies (discussed in Section 3.2).

Clearly, obtaining d_{ij} for all n nodes requires $O(n^2)$ measurements.[2] However, we note that these $O(n^2)$ measurements do not have to be announced or be continuously monitored. In particular, each node needs to make $O(n)$ measurements only once per "re-wiring epoch" to decide its neighbors but announces and keeps sending updates only for the k links that it chooses to establish and thus, the load on the link-state protocol is only $O(nk)$.

3.2 Neighbor Selection Policies in EGOIST

As its namesake suggests, the basic (default) neighbor selection policy in EGOIST is the Best-Response (BR) strategy described in Section 2.1, and detailed in [20]. Using BR, a node selects all its k neighbors so as to minimize a local cost function, which could be expressed in terms of some performance metric (*e.g.*, average delay to all destinations, maximum aggregate throughput to all destinations, etc). In addition to BR, we have also implemented the following neighbor selection policies in EGOIST so as to allow us to compare the performance of selfish neighbor selection to other policies.

k-Random: Using this strategy, a node selects one neighbor so as to form a cycle (thus ensuring connectivity of the overlay), and selects the remaining $k-1$ neighbors randomly.

k-Closest: Using this strategy, a node selects one neighbor so as to form a cycle (thus ensuring connectivity of the overlay), and selects the $k-1$ remaining neighbors to be the nodes with the minimum link cost (*e.g.*, minimum delay, maximum bandwidth, *etc.*)

k-Regular: Using this strategy, all nodes follow the same wiring pattern by using a common offset vector $o = \{o_1, o_2, \ldots, o_k\}$. Thus, node i connects to nodes $i + o_j \mod n$, $j = 1, \ldots, k$. In our system, we set $o_j = 1+(j-1)\cdot\frac{n-1}{k+1}$.[3] One way to visualize this is to consider that all nodes are placed in a ring according to their ids (like in a DHT). Thus, an offset vector makes each node use its k links to connect to other nodes so as to equally divide the periphery of the ring.

3.3 Dealing with Churn in EGOIST

As discussed above, EGOIST's BR neighbor selection strategy assumes a static setting in which nodes who *join* the overlay *never leave*. Clearly, in a typical setting, this is not the case as node churn is a hallmark of

[2]Notice that d_{ij} can be obtained through active or passive measurements depending on the metric of interest (see Section 4.1 for details).

[3]To simplify the presentation, we assume that $n-1$ is a multiple of $k+1$.

many overlay routing networks and P2P systems. The main issue with node churn is the fact that node leaves may result in network partitions, implying infinite link (and consequently path) costs, which are not possible to deal with using a BR strategy. One approach to dealing with churn is to re-formulate the BR objective function used by a node to take into account the churning behavior of other nodes. This, however, requires modeling of the churn characteristics of various nodes in an overlay, which may not be feasible nor realistic, especially for large networks [48].

In EGOIST we follow a different approach reminiscent of how k-Random and k-Closest policies ensure overlay connectivity. In particular, we deal with churn by ensuring that connectivity is assured through a wiring mechanism *other than* BR. To that end, and in addition to the neighbor selection strategies considered so far, we implemented a hybrid wiring strategy (HybridBR), in which each node uses k_1 of its k links to selfishly optimize its performance using BR, and "donates" the remaining $k_2 = k - k_1$ links to the system to be used for assuring basic connectivity under churn. We call this wiring "hybrid" because in effect two wiring strategies are in play – a selfish BR strategy that aims to maximize local performance and a selfless strategy that aims to maintain global connectivity.

There are several ways in which a system can use the k_2 donated links of each node to build a connectivity backbone. Young et al. [50] have proposed the use of k Minimum Spanning Trees (k-MST). Using k-MST—a centralized construction—to maintain connectivity is problematic, as it must always be updated (due to churn and to changes in edge weights over time), not to mention the overhead and complexities involved in establishing ($k_2/2$)-MSTs. To avoid these complexities, EGOIST uses a simpler solution that forms $k_2/2$ bidirectional cycles. Consider the simplest case $k_2 = 2$, which allows for the creation of a single bidirectional cycle. To accommodate a new node v_{n+1}, node v_n will disconnect from node v_1 and connect to v_{n+1}, whereas the latter will connect to v_1 to close the cycle. For higher $k_2/2$, the system decides $k_2/2$ *offsets* and then each node connects to the nodes taken by adding (modulo n) its id to each offset. If k_2 is small (*e.g.*, 2) then the nodes will need to monitor (*e.g.*, ping) the backbone links closely so as to quickly identify and restore disconnections. With higher k_2 the monitoring can be more relaxed due to the existence of alternative routes through other cycles. Computing BR using k_1 links *granted* the existence of the k_2 links can be achieved by simple re-formulation of the ILP model of [20] in which the decision variables Y_i corresponding to the nodes that receive high maintenance links will have preset values.

We have implemented this HybridBR (with $k_2 = 2$) in EGOIST. As hinted above, in our implementation, donated links are monitored aggressively so as to recover promptly from any disconnections in the connectivity backbone through the use of frequent heartbeat signaling. On the other hand, the monitoring and upkeep of the remaining BR links could be done lazily, namely by measuring link costs, and recomputing BR wirings at a pace that is convenient to the node–a pace that reduces probing and computational overheads without risking global connectivity.

To differentiate between these two types of link monitoring strategies (aggressive versus lazy), in EGOIST we allow the rewiring of a dropped link to be performed in one of two different modes: *immediate* and *delayed*. In immediate mode, rewiring is done as soon as it is determined that a link is dropped, whereas in delayed mode, rewiring is only performed (if necessary) at a pre-determined *rewiring epoch* T. Unless otherwise specified, we assume a delayed rewiring mode is in use.

4. EXPERIMENTAL EVALUATION

4.1 Cost Metrics

As we alluded earlier, a number of metrics could be used to measure the "cost" of traversing a link in an overlay setting. Clearly, the choice of an appropriate metric depends largely on the application at hand. In this section, we review the various cost metrics we have considered in our experiments. For each metric, we also discuss how this metric was estimated in EGOIST.

Link and Path Delays: Delays are natural cost metrics for many applications, especially those involving interactive communication (*e.g.*, gaming, or end-system multicast). To obtain the delay cost metric, a node needs to obtain estimates for its own delay to potential neighbors, and for the delay between pairs of overlay nodes already in the network. In EGOIST, we estimated the directed (one-way) link delays using two different methods: an active method based on `ping`, and a passive method using the `pyxida` virtual coordinate system [23, 22]. Using `ping`, the one-way delay is estimated to be half of the measured `ping` round-trip-times (RTT) averaged over enough samples. Clearly, a node is able to measure such a value for all of its direct (overlay) neighbors, and is also able to relay such information to any other nodes through the overlay link-state routing protocol. Using `pyxida`, the delay estimate is available directly through a simple query to the system.[4]

Node Load: For many overlay applications, it may be the case that the primary determinant of the cost of a path is the performance of the nodes along that path—*e.g.*, if traversal of nodes along the path incur

[4]Using `ping` produces more accurate estimates, but subjects the overlay to added load, whereas using `pyxida` produces less accurate estimates, but consumes much less bandwidth.

Algorithm 1 ρ=avbw($G(V, E),\ s \in V$)

```
1: Set $W = \{s\}$; and $\rho[s] = inf$;
2: for all $y \in V - \{s\}$ do $\rho[y] = d_{sy}$;
3: while $W \neq V$ do
4: begin find x = argmax$\{\rho[y] : y \notin W\}$;
5:     set $W = W \cup \{x\}$;
6:     for all $y \in V - W$ do
7:         $\rho[y] = \max\{\rho[y], \rho[x] + d_{xy}\}$
8: end
9: Return $\rho$;
```

significant overhead due to (say) context switching and frequent crossing of user/kernel spaces. Thus, in Egoist, we allow the use of a variation of the delay metric in which all outgoing links from a node are assigned the same cost, which is set to be equal to the measured load of the node. When applicable, the estimation of such a metric is straightforward as it requires only local measurements. In Egoist, we did this by querying the CPU load of the local PlanetLab node, and computing an exponentially-weighted moving average of that load calculated over a given interval (taken to be 1 minute in our experiments).

Available Bandwidth: Another important cost metric, especially for content-delivery applications, is the available bandwidth on overlay links. Different available bandwidth estimation tools have been proposed in the literature (see [42] for an exposition). In Egoist, we used `pathChirp` [36], a light, fast and accurate tool, which fits well with PlanetLab-specific constraints, namely: it does not impose a high load on PlanetLab nodes since it does not require the transmission of long sequences of packet trains, and it does not exceed the max-burst limits of Planetlab. `pathChirp` is an end-to-end active probing tool, which required the installation of sender and receiver `pathChirp` functionality in each Egoist node. The available bandwidth between a pair of nodes $v, u \in V_i$ is given by

$$AvailBW(v, u) = \min_{e \in P^*(v,u)} AvailBW(e),$$

where $P^*(v, u)$ denotes a path that connects v to u in G_{-i}, such that the edge in $P^*(v, u)$ with the minimal available bandwidth has higher available bandwidth than all corresponding edges among all the paths $P(v, u)$ that connect v to u (*i.e.*, we identify the path that yields the max-min available bandwidth). Thus, finding $P^*(v, u)$ and the bottleneck edge is a "Maximum Bottleneck Bandwidth" problem [11], which can be solved using a simple modification of Dijkstra's algorithm as shown in Algorithm 1.

Using available bandwidth as the cost metric requires us to also modify the local objective function for BR purposes. In particular, the best response for v_i may be based on a wiring s_i that maximizes the aggregate bandwidth out of a node given by

$$\sum_{v_j \in V_{-i}} \max_{w \in s} \min\left(AvailBW(e(v_i, w)), AvailBW(w, v_j)\right)$$

The above objective calls for the maximization of the *sum* of the bottleneck bandwidths to all destinations. A straightforward alternative formulation could also consider the maximization of the *minimum* of the bottleneck bandwidths to all destinations, so as to provide the best lower-bound on bandwidth to any destination.[5]

4.2 Baseline Experimental Results

In this section, we present performance results obtained through measurement of Egoist. These results allow us to draw comparisons between the various neighbor selection policies described in Section 3.2 for the various cost metrics described above. All the results in this section assume that node churn is not an issue – *i.e.*, once it joins the overlay, a node does not leave. Results showing the impact of node churn on Egoist performance are presented in Section 4.3.

Experimental Setting: We deploy Egoist on $n = 50$ PlanetLab nodes (11 in Europe, 7 in Asia, 1 in South America, and 1 in Oceania). Each of these nodes is configured to recompute its wiring every $T = 60$ seconds. Egoist nodes are not synchronized, thus on average a rewiring by some Egoist node occurs every $T/n = 1.2$ seconds. Whether a node ends up rewiring or not depends on the neighbor selection policy. For k-Random and k-Regular policies, and since our baseline experiments do not feature any node churn, it follows that these policies will not exhibit any rewiring. For k-Closest, rewiring would only be the result of dynamic changes in PlanetLab that result in changes to the cost metric, and hence what constitutes the closest set of neighbors. For BR, a node may rewire due to changes in PlanetLab conditions, but may also rewire simply as a result of another node's rewiring. While in theory [19, 20], BR strategies converge to some equilibrium in the Nash sense, we note that this is not likely to be the case for real systems such as Egoist, since dynamic changes of the underlying system (changes in link delays, bandwidth, and node load) are likely to result in perpetual rewiring by Egoist nodes. Setting the rewiring epoch T in Egoist has the effect of controlling the timescale of, and consequently the overhead incurred by, BR rewiring.

Each experiment presented in this section reflects the results obtained by running Egoist for 10 hours on PlanetLab on January 5th and January 15th, 2007. For each experiment, an individual cost metric is calculated for every one of the $n = 50$ nodes in the system.

[5]It is also implicit in this formulation that the available bandwidth of an edge is not affected by v_i's own traffic going through that edge (in the event that v_i will choose a wiring that uses this edge). This can be addressed with a more complicated formulation, which we omit due to space considerations.

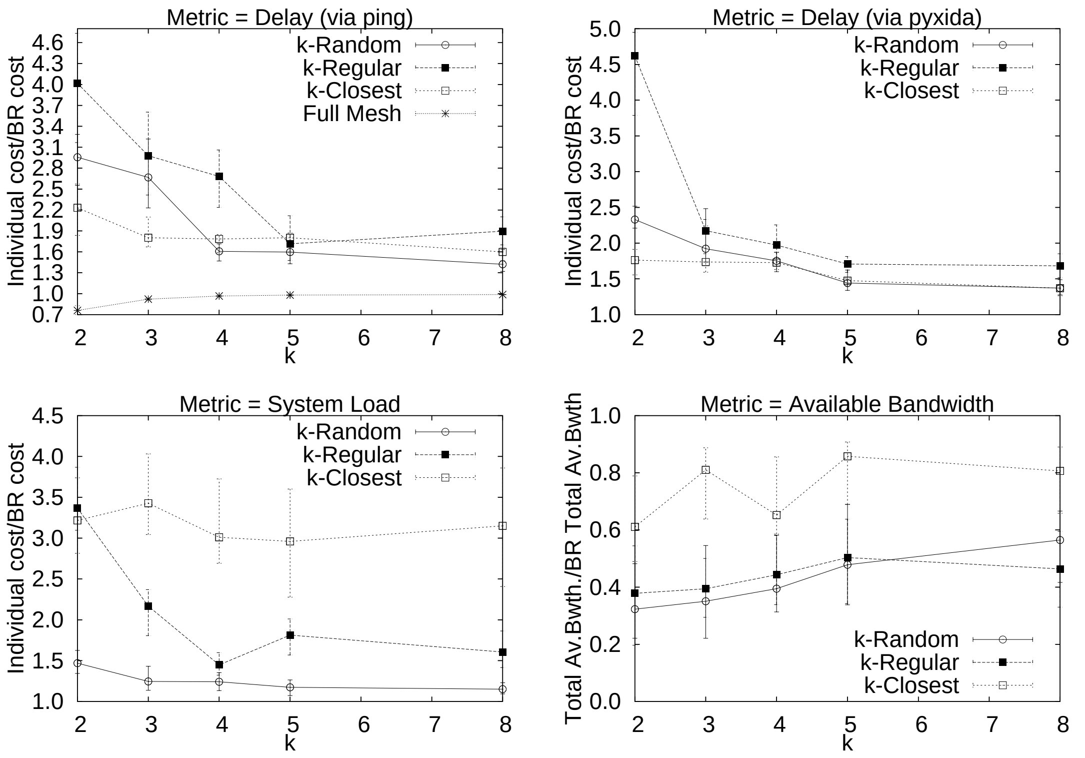


Figure 1: PlanetLab baseline experiments showing the individual costs for various neighbor selection policies (normalized with respect to BR costs) as a function of number of neighbors k for a 50-node EGOIST overlay: Cost metric is `ping` delays (top-left), `pyxida` delays (top-right), node CPU load (bottom-left), and available bandwidth (bottom-right).

The individual cost metric for a node reflects the cost of routing from that node to all other 49 nodes in the system, assuming a *uniform* routing preference over all destinations.[6] For each experiment we report the median of all $n = 50$ individual costs, as well as a range delimiting the 25^{th}-75^{th} percentiles for these costs.

Control Variables: In our first set of experiments, our aim is to identify for the three metrics of interest the payoff (if any) from adopting a selfish neighor selection strategy, *i.e.*, using a BR policy in Egoist. This payoff will depend on many variables. While some of these variables are *not* within our control (*e.g.*, the dynamic nature of the Internet as reflected by variability in observed PlanetLab conditions), others are within our control (*e.g.*, n, T, and the various settings for our active measurement techniques).

In order to neutralize the effect of extrinsic variables that are not within our control, experiments reporting on different neighbor selection policies were conducted *concurrently.* To do so, we deploy concurrent Egoist agents on each of the $n = 50$ PlanetLab nodes we use in our experiments, with each agent using a different neighbor selection strategy. In effect, each one of our experiments compares the performance of a *set* of concurrently deployed Egoist overlay networks, each resulting from the use of a particular neighbor selection policy.

One control variable that is particularly important is the *number of direct neighbors*, k, that an Egoist node is allowed to have. In many ways, k puts a premium on the significance of making a judicious choice of neighbors. For small values of k, choosing the right set of neighbors has the potential of making a bigger impact on performance, when compared to the impact for larger values of k. Thus, in all the results we present in this section, we show the performance of the various policies over a range of k values.

Overview of Performance Results: Before presenting specific performance results, we make two broad observations: first, in all of our experiments, using a BR policy in Egoist consistently yields the best performance. While such an outcome was anticipated by virtue of findings reported in [20] for a static setting, the results we present here are significant because they underscore the payoff in a *real* deployment, where the modeling assumptions made in prior works do not hold.

[6]We note that using a uniform routing preference will tend to deflate the advantage of BR neighbor selection – in other words, the results we present here are conservative, since unlike the other policies we considered, BR is capable of leveraging skew in preference to its advantage.

Second, in all of our experiments, with the exception of BR, no single neighbor selection policy was consistently better than all others across all metrics. In other words, while the performance of a given policy may approach that of BR for one metric while dominating all other policies, such policy dominance does not hold across all the metrics we considered.

Results for Delay Metric: Figure 1 shows the performance of the various neighbor selection policies in Egoist normalized with respect to that achievable using BR when the metric of interest is the overlay link/path delay over a range of values for k (with link delays measured using `ping` in the top-left plot, and using `pyxida` in the top-right plot). These results show that BR outperforms all the other wiring policies, especially when k is small, as anticipated in our discussion of the significance of k as a control variable. For example, for $k = 2$, the average delay experienced by an individual node could be anywhere between 200% and 400% *higher* than that achievable using BR. The performance advantage of BR in terms of routing delay stands, even for a moderate number of neighbors. For example, for $k = 5$, BR cuts the routing delay almost by half.

These results confirm the superiority of BR relative to other policies, but do not give us a feel for how close is the performance of Egoist using BR wiring to the "best possible" performance. To do so, we note that by allowing nodes to connect to all other nodes in the overlay (*i.e.*, by setting $k = n - 1$), we would be creating a complete overlay graph with $O(n^2)$ overlay links, obviating the need for a neighbor selection policy. Clearly, the performance of routing over such a rich overlay network gives us an *upper bound* on the achievable performance, and a lower bound on the delay metric. Thus, to provide a point of reference for the performance numbers we presented above, in the top-left plot in Figure 1, we also show the performance achieved by deploying Egoist and setting $k = n - 1$. Here we should note that this lower bound on delay is what a system such as RON [4] would yield, given that routing in RON is done over shortest paths established over a full mesh, and assuming that any of the $O(n^2)$ overlay links could be used for routing. These results show that using BR in Egoist yields a performance that is quite competitive with RON's lower bound. As expected, the difference is most pronounced for the smallest k we considered—namely, the lowest delay achievable using 49 overlay links per node is only 30% lower than that achievable using BR with 2 overlay links per node. BR is almost indistinguishable from the lower bound for slightly larger values of k (*e.g.*, $k = 4$).

With respect to the other heuristics, the results in the top plots in Figure 1 show that k-Closest outperforms k-Random when k is small, but that k-Random ends up outperforming k-Closest for slightly larger values of k. This can be explained by noting that k-Random ends up creating graphs with much smaller diameters than the grid-like graphs resulting from the use of k-Closest, especially as k gets larger. In all experiments, k-Regular performed the worst.

Results for Node Load: The bottom-left plot in Figure 1 shows the results we obtained using the node load metric, where the path cost is the sum of the loads of all nodes in the path. These results show clear delineations, with BR delivering the best performance over all values of k, k-Random delivering the second-best performance, and k-Closest delivering the worst performance as it fails to predict anything beyond the immediate neighbor, especially in light of the high variance in node load on PlanetLab.

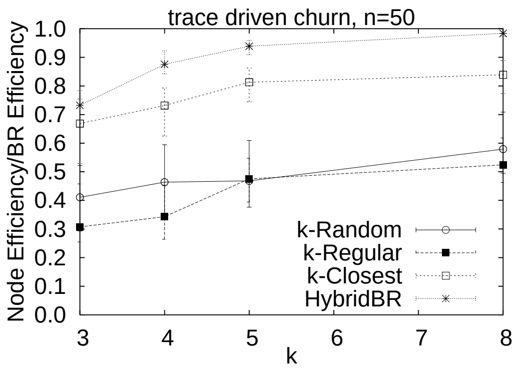
Results for Available Bandwidth: The bottom-right plot in Figure 1 shows the results we obtained using available bandwidth as the cost metric. Recall that, here, the objective is to get the highest possible agrregate bandwidth to all destinations (again, assuming a uniform preference for all destinations) – thus, larger is better. These results show trends that are quite similar to those obtained for the delay metric, with BR outperforming all other policies—delivering a two to four-fold improvement over the other policies, over a wide range of values of k.

4.3 Effect of Churn

In the original SNS formulation [20, 19], the graphs resulting from the SNS-game as well as from the empirical wiring strategies were guaranteed to be connected, so they could be compared in terms of average or maximum distance. Node churn, however, can lead to disconnected graphs, therefore we have to use a different metric. We chose the *Efficiency* metric [21], where the Efficiency ϵ_{ij} between node i and j ($j \neq i$) is inversely proportional to the shortest communication distance d_{ij} when i and j are connected. If there is no path in the graph between node i and j then $\epsilon_{ij} = 0$. The Efficiency ϵ_i of a node i defined as:

$$\epsilon_i = \frac{1}{n-1} \sum_{j \neq i} \epsilon_{ij}$$

The Efficiency of a graph is defined as the average of the node efficiencies. To evaluate the efficiency of Egoist overlays under churn, we allow each of the $n = 50$ nodes in the overlays to exhibit ON and OFF periods. During its ON periods, a node "joins" the overlay, performs rewiring according to the chosen policy, and fully participates in the link-state routing protocol. During its OFF periods, a node simply drops out from any activity related to the overlay. The ON/OFF periods we use in our experiments are derived from real data sets of the churn observed for PlanetLab nodes [16], with ad-



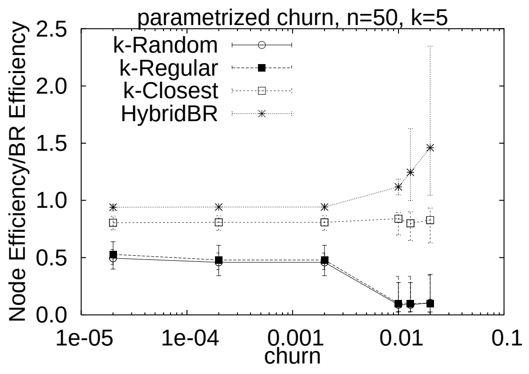


Figure 2: PlanetLab experiments with node churn showing the efficiency of neighbor selection policies (normalized with respect to BR) as a function of the number of neighbors k (left) and churn (right) for a 50-node EGOIST overlay.

justments to the timescale to control the rate of churn.

In addition to evaluating the efficiency of various neighbor selection policies we have considered so far, we also evaluate the efficiency of HybridBR (Section 3.3), which allows a node to "donate" $k_2 = 2$ of its links to ensure connectivity (*i.e.*, boost the efficiency of the overlay) while using BR for the remaining links.

The left plot in Figure 2 shows the achievable efficiency of the various neighbor selection policies when churn is present. As before, the efficiency of the various policies is normalized with respect to that achievable using BR, and is shown as a function of k. As with all the metrics we considered so far, BR outperforms all other policies (including HybridBR), but as EGOIST nodes are allowed to have more neighbors (*i.e.*, as k increases), the efficiency of the HybridBR approaches that of BR, with the efficiency of k-Closest decisively better than k-Random and k-Regular.

The above results imply that under the level of churn in these experiments, it is not justifiable for BR to donate two of its links simply to ensure connectivity, especially when k is small. Notice that BR overlays that get disconnected due to churn will naturally "heal" as soon as any of its active nodes decides to rewire. This is so because the (infinite) cost of reaching the disconnected nodes will act as an incentive for nodes to choose disconnected nodes as direct neighbors, thus reconnecting the overlay. As noted earlier, rewiring occurs every T/n units of time on average (1.2 seconds under our settings), which implies that the vulnerability of BR to disconnections due to churn is highest for smaller overlays and if rewiring is done infrequently. Said differently, the expected healing time for BR is $O(T/n)$.

Our last question then is whether at much higher churn rates, it is the case that the use of HybridBR would be justified. To answer this question, we changed the timescale of the ON/OFF churn processes to emulate more frequent joins and leaves. The right plot in Figure 2 shows the results by plotting the efficiency metric for the various policies as a function of the churn rate (on the x-axis), which we define (as in [16]) to be the sum of the fraction of the overlay network nodes that changed state (ON/OFF), normalized by time T:

$$Churn = \frac{1}{T}\sum_{events\ i}\frac{|U_{i-1}\ominus U_i|}{max\{|U_{i-1}|,|U_i|\}}$$

where U_i is the set of nodes when event i takes place and $\ominus$ is the symmetric set difference. Thus, a churn rate of 0.01 implies that, on average, 1% of the nodes join or leave the overlay per second. For an overlay of size $n = 50$, this translates to a join or leave event every two seconds.

As expected, when churn rate increases significantly to the point where the average time between churn events approaches $O(T/n)$), the efficiency of HybridBR eventually surpasses that of BR. The results also suggest that under such conditions, the efficiency of k-Random and k-Regular both fall dramatically, whereas that of k-Closest remains level with that of BR.

5. SCALABILITY VIA SAMPLING

In this section we address potential scaling limitations of EGOIST by describing methods that *sample* the large space of possible neighbors and apply BR algorithms to the sample. Such a technique might not be necessary for current overlay networks that are of small to moderate sizes, such as PlanetLab, as discussed later, but are likely to become essential in emerging overlays of massive scale. One such example we foresee is that of future "P2P reincarnations" of overlay routing that allow participating nodes to opportunistically choose overlay routes with minimal overhead. Unlike today's systems such as RON, which require central installation and maintenance by an interested party, these large systems would likely be self-organizing and self-regulating.

There are several aspects of an ORS that become potentially problematic at scale: the overhead of the underlying link-state protocol, the cost of performing local search to compute BR, and scaling questions associated with the sampling process itself. We view the scaling issues associated with link-state routing as modest, since in EGOIST we limit the number of monitored and announced links to much less than $O(n^2)$ (*i.e.*, when $k \ll n$), and thus the per-node communication complexity scales as a function of k and not n.

A more significant scaling issue is imposed by the computational complexity of computing best responses. It has been shown in [20] that computing an exact "Best Response" is an NP-hard problem but that approximate ones computed through local search perform nearly as well as exact ones. However, the local search [5], still impose substantial computational burden (polynomial number of iterations, each one requiring $n^{O(p)}$, p being a parameter of the algorithm). Such high order polynomial complexity becomes difficult to handle even for moderate n, especially when nodes must re-wire frequently to cope with the dynamics of the network. To handle such cases, we propose scaling down the input by computing BR based on a *limited* number of samples from the residual overlay graph. This enables us to run a computationally efficient algorithm (sampling) on the large input, and then run a computationally expensive BR algorithm on the scaled input. Later we will show that with an appropriate sampling technique in place, BR retains its performance edge over the other heuristics.

A natural approach would be to compute v_i's BR based on a sample of m nodes obtained through *unbiased random sampling* of the total n nodes of G_{-i}. This would limit the input to the parts of the distance function $d_{G_{-i}}$ that involve pairs that belong to the chosen sample. Also v_i would need to measure its distance to only those m samples. As we will show experimentally, such an approach has some value, but there is much more to gain by a simple "better than random" sampling.

Topology-Based Biased Random Sampling: The basic idea of our *topology-based biased random sampling* is to take $m' > m$ random samples and apply topological filters to keep those m that are likely to yield the best results. The heuristic approach we apply is to bias our samples towards nodes with the largest neighborhoods of radius r (*e.g.*, with the highest number of distinct nodes reachable in r hops). Defining $F(v_j)$ to be the size of the neighborhood of radius r around v_j, we give consideration to $|F(v_j)|$ as well as the distances of nodes within $F(v_j)$ from the perspective of the source v_i. This reflects the intuition that an ideal candidate for v_i has a large neighborhood of nodes, many of which are relatively close to v_i. Our ranking function b_{ij} establishes

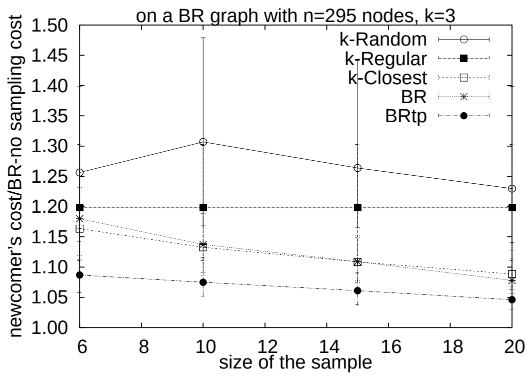


Figure 3: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a BR graph.

a priority order on candidates v_j as follows:

$$b_{ij} = \frac{|F(v_j)|}{\sum_{u \in F(v_j)} d(v_i, u)}$$

Using this ranking function, v_i chooses a sample of m nodes with the highest b_{ij} values and computes its BR based on these nodes only.

Finally, we need to verify that this sampling procedure itself is not prohibitive. Standard random-walk based methods can query a set of m' pseudorandomly-generated nodes in a k-regular graph with suitable expansion properties using $O(m' \log n / \log k)$ messages. Each node must be able to approximately maintain and express the number of nodes within its r-radius neighborhood, which requires $O(k^r)$ space. Nodes also must compute the b_{ij} values, which requires $O(m'k^r)$ distance lookups. All of this amounts to a reasonable overhead for the small fixed values of r and k that we focus on in this work.

Experimental Validation: We conduct trace-driven simulations on PlanetLab, as well as experiments on synthetic and AS topologies to evaluate the effectiveness of sampling. Due to space limitations, we only report on the PlanetLab simulations (results obtained in the other settings were similar). We use publicly available PlanetLab traces [49] containing delays obtained using `pings` between all pairs of PlanetLab sites. We test the four neighbor selection strategies of Section 3.2. In our simulation, an n-node network is constructed incrementally using the BR strategy (without sampling). A newcomer joins the network using one of the following strategies: k-Random, k-Regular, k-Closest, and BR, each with random sampling, and BR with topology-based sampling. In the simulation, $n = 295$, $k = 3$, and

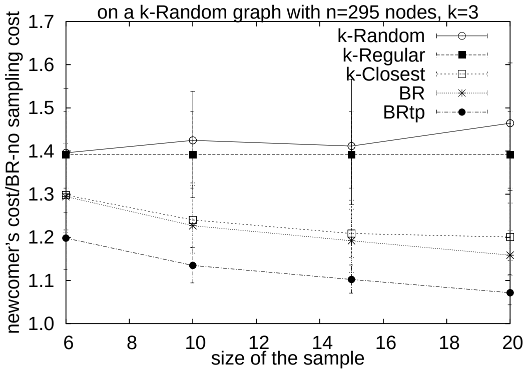


Figure 4: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a k-Random graph.

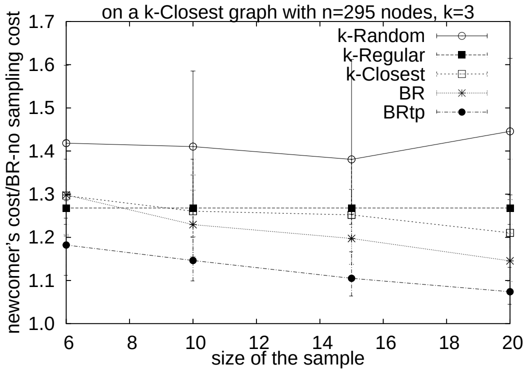


Figure 6: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a k-Closest graph.

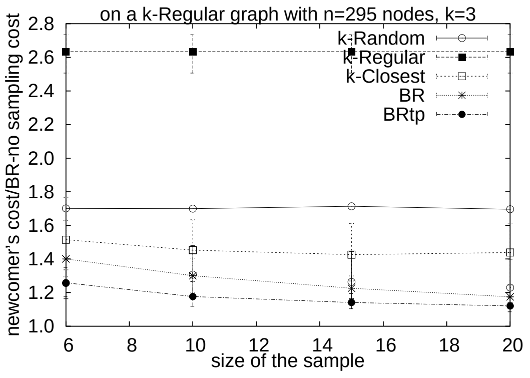


Figure 5: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a k-Regular graph.

the neighborhood size $r = 2$. In Figure 3, we plot the ratio of the newcomer's cost to the cost of using BR with no sampling for different sample sizes. The line labeled "BR" denotes the ratio when the newcomer uses BR with random sampling; "BRtp" denotes BR with topology-based sampling.

Our general observations across the experiments are that BR with sampling fares better than any of the three empirical rules, and that even for small m/n, the newcomer's cost ratio is not much larger than 1. We also find that topology-awareness in sampling improves the BR wiring significantly in all cases considered.

It is also worth mentioning that the performance of simple heuristics with random sampling in a BR graph is good, due to its highly optimized structure. In graphs formed by nodes that follow the previously mentioned random or myopic heuristics, we observed that the performance gain of topology-biased random sampling is substantially better compared to any other wiring policy which is based on random sampling (see Figs. 4, 5, and 6).

6. PEER-TO-PEER FILE SHARING

In this section we shift our attention to potential benefits from employing a BR neighbor selection strategy in an unstructured P2P file sharing application. We begin with a discussion of the applicability of the initial formulation of BR and move on to present a re-formulation that becomes advantageous in a P2P setting.

On the applicability of the original BR formulation: The first question we ask is whether the original BR formulation could be used for unstructured P2P file sharing applications – *i.e.*, as formulated in Section 2.1, what can BR do for P2P?

Unlike the case of overlay routing systems, the mapping of the BR formulation to the P2P file-sharing systems is not a direct one. The reason is that in P2P file sharing the outgoing traffic (*i.e.*, the search queries) target *objects*, wherever these may lie, instead of specific *nodes*, as assumed in the original formulation (and as is the case in overlay routing systems). Still the initial formulation of BR might be helpful. Consider a P2P file sharing system in which nodes maintain a figure of merit for each other node based on direct or third-party experience.[7] The merit value could, for example, indicate quality of content, correlation of interests, or

[7]Similar in spirit to reputation protocols.

the capacity or reliability of the node. Then, even if queries are for objects and not for nodes, it still is beneficial to have the queries reach meritorious nodes first before propagating further away in the network. One could then incorporate the merit value into the preference weights p_{ij} of the original BR formulation. This results in wirings in which nodes of higher merit are kept closer to the connecting node. Implementing this idea, however, requires addressing some non-trivial technical hurdles.

First, one must augment the current P2P protocols with additional functionality that will permit a node to gather the required information for computing a BR (namely the residual network). In the case of overlay routing, the link-state routing protocol running at the overlay layer provides this information, but in currently deployed P2P file-sharing systems, there is no equivalent capability. Second, even if the required information was to become available, the problem still remains that P2P applications have no way of performing shortest-path routing to a destination, which is a fundamental underlying assumption of the original formulation. This is because a querying node does not know *a priori* the identity of the destination node holding the file of interest. Employing full flooding would of course create an equivalent of shortest-path routing as queries would reach target nodes first through shortest-paths and then through non-shortest paths. Unfortunately, full flooding does not scale, and thus real unstructured P2P file-sharing systems rely on either scoped-flooding or random walks [28] for forwarding search queries. In scoped flooding, a successful search (meaning that the object is located) will indeed go over a shortest path. Objects that exist only outside the scope will simply not be reachable. With random walks, on the other hand, located objects are reached through paths that are not generally shortest-paths.

To understand the effects of non shortest-path routing we conduct the following experiment. We assume that the full topology of an unstructured P2P network is provided to a node, which then computes its BR as if shortest path routing is to be used. We then connect the node to the network according to the computed BR and let it start issuing queries that are forwarded using either scoped-flooding or random walks. We compare the cost (measured as the total distance to all destinations[8]) under this setting to the cost incurred by simple wiring strategies. We use as residual network the stable graphs obtained in Section 5 by having nodes compute BRs under a physical distance model obtained from a PlanetLab trace (we obtained similar results with physical distances taken from AS-level maps of the Internet and with synthetic maps generated using BRITE [29]).

[8]We assign a cost penalty equal to the diameter cost for unreachable nodes under scoped flooding.

Figure 7 (left) shows that under scoped flooding (with scope $r = 3$ hops), BR outperforms the other wirings by a significant margin. The performance of k-Closest and k-Regular wiring degrades as k increases, whereas for k-Random it improves as k increases. Figure 7 (right) shows the corresponding results under random walk (with 20 random walks for each source destination pair). The main observation from this figure is that contrary to scoped flooding, where BR retains much of its edge over the other policies, under random walk BR performs equally or even worse than these heuristics. This is attributed to the totally random nature of routing in this case, which is enough to obliterate any systematic attempt to capitalize on the structure of the residual graph.

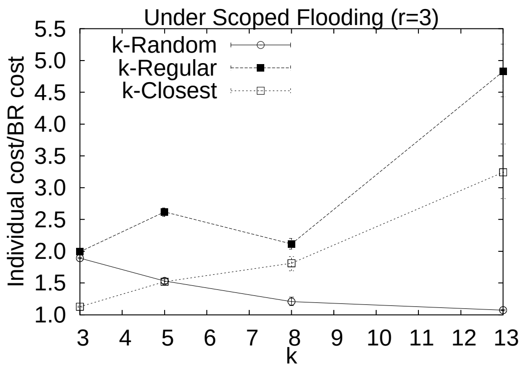
Considering (1) the discrepancy between the original formulation and current P2P file sharing systems, (2) the lack of protocol support for providing information on the residual graph, and (3) the performance penalty from applying BR in systems that use random walks, we conclude that a direct application of BR to current, deployed unstructured P2P file-sharing systems is not worth the effort. Instead, we reformulate the notion of BR to make it more natural and beneficial for use in a P2P file-sharing setting.

A reformulation of BR for scoped-flooding: Consider an unstructured P2P file sharing network employing scoped flooding of search queries with time-to-live r. Granted that in most such networks there's no *a priori* knowledge of other nodes' content, the search performance is optimized by maximizing the number of distinct nodes reachable by scoped flooding. This implies that first hop neighbors should be selected so as to cover as much as possible disjoint parts of the residual overlay topology. We reformulate our notion of BR so as to achieve this goal and compare with the corresponding search performance of k-Random which is the typical choice in many existing systems. We base our discussion on a two-tier unstructured P2P network (like KaZaA and the latest versions of Gnutella) with two types of nodes: Ordinary Nodes (ON) and Super Nodes (SN), which operates as follows:

(1) A newcomer node v_i connects to a bootstrap server and retrieves a set C with $m = |C|$ candidate SN's, from which it has to select k.

(2) The newcomer v_i contacts each one of the candidate SNs $v \in C$ and queries it for its list of first hop neighbors (and the type of each neighbor, ON or SN). Such capability is provided by most widely deployed unstructured P2P systems [44]. Then it queries all SN neighbors recursively up to distance $r-1$ from the initial candidate v.

(3) After receiving all such information, the newcomer



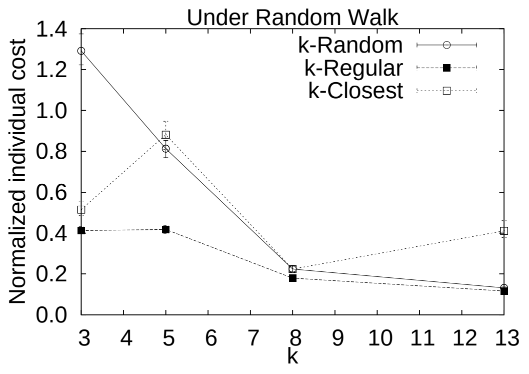


Figure 7: Individual cost for various wiring strategies (normalized with respect to BR): Under scoped flooding with time-to-live, r=3 (left) and under Random Walks (right).

Algorithm 2 s=greedy($\{F(v)\}_{\forall v \in C}$)

```
1: Set $s^{(0)} = \emptyset$ and $\Phi^{(0)} = \emptyset$;
2: for $i = 1 : k$ do
3:     $v = \arg\max_{v \in C} \left|\Phi^{(i-1)} \cup F(v)\right|$;
4:     $s^{(i)} = s^{(i-1)} \cup \{v\}$;
5:     $\Phi^{(i)} = \Phi^{(i-1)} \cup F(v)$;
6:     $C = C - \{v\}$;
7: Return $s = s^{(k)}$;
```

computes for each candidate SN v the set $F(v)$ of unique nodes (both ON and SN) that are reachable from it in $r-1$ hops.

(4) To compute its BR, v_i has to select a wiring s of cardinality k so as maximize the cardinality of $\Phi(s)$ over all possible wirings, where:

$$\Phi(s) = \bigcup_{v \in s} F(v)$$

A straight-forward exhaustive search can find such a BR wiring in $O(nkm^k)$ since there exist $\binom{m}{k} = O(m^k)$ possible wirings, and computing the cardinality of each one requires performing union operations on k sets $F(v)$, each one having size $O(n)$. Unfortunately, this pseudo-polynomial running time is essentially the best that can be achieved, as maximizing the cardinality of $F(s)$ is a variant of set cover that is easily seen to be NP-complete.

Thus, the approach that we follow in our experiments is to use a simple greedy algorithm of O(nmk) complexity (see Algorithm 2), which is capable of producing high-quality solutions, and provably optimal solutions for the special case of $k = 2$. The algorithm simply selects the ith set u_i so as to maximize $D(u_i)$, where $D(u_i) = F(u_i)/(F(u_i) \cap F(s_O^{(i-1)}))$.

To demonstrate the benefits of BR as reformulated for scoped-flooding, we use the Gnutella trace presented in [44]. This dataset provides a realistic snapshot of the Gnutella topology with over 305,000 ONs and SNs. We select an ON from this trace and let it be our "newcomer" node. Then we supply it with an unbiased random sample set C with m candidate nodes (as a bootstrap server for Gnutella would do). We compare the number of unique nodes reachable from this newcomer given its wiring as was reported in the dataset and according to our reformulated BR.

We uniformly at random select two sets of 30 ONs from the aforementioned data set, which are connected to two and three SNs respectively. Each of these ONs connects to the bootstrap server and retrieves candidate SNs ($m = 10$, including those to which it is currently connected).

In Figure 8, we plot the ratio of the unique nodes reachable with scoped-flooding for different values of time-to-live (r) using our new BR formulation relative to that reported in the Gnutella dataset. The reformulated BR computed through exhaustive search increases significantly the number of nodes reached, with similar improvements achieved when the greedy heuristic is used for the computation.

7. CONCLUSION

In this work we presented our attempt to utilize recent theoretic results on Selfish Neighbor Selection (SNS) and put them to work for the benefit of overlay routing and P2P file sharing applications. The mapping from the original SNS problem to actual overlay routing applications is a natural one, and thus our main challenge was to show through the development of our EGOIST prototype routing network for PlanetLab that Best-Response (BR) neighbor selection strategies can indeed be realized in practice. Such BR strategies fit naturally into overlay routing because on the one hand they provide a substantial performance boost as compared to simpler empirical strategies, and on the other hand scale much better than full-mesh approaches which require intensive monitoring of $O(n^2)$ links. We substantiated these benefits under different performance metrics, link

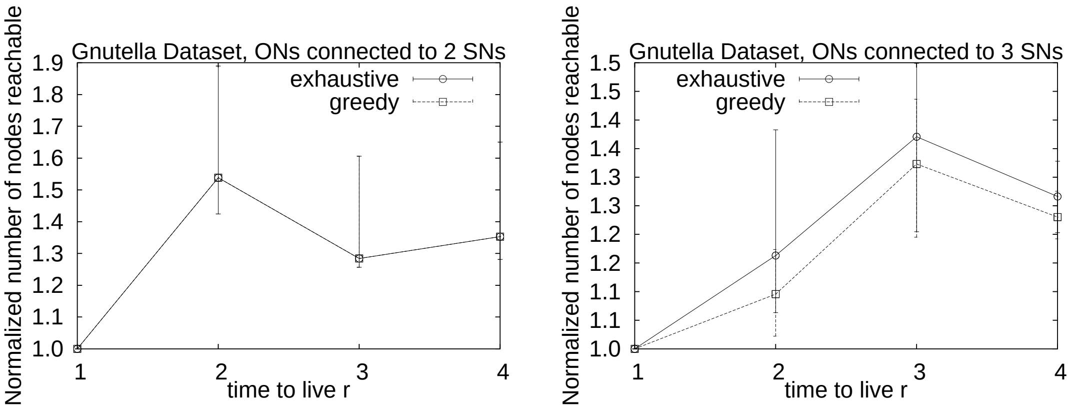


Figure 8: Gnutella simulations showing the improvement in node coverage using BR, reformulated for scoped flooding, and using exhaustive and greedy search algorithms. Improvement is relative to the coverage achieved using Gnutella's scoped flooding over its default wiring. Results are for ONs connected to two SNs (left) and three SNs (right).

monitoring methods, in static and churn-prone environments. Furthermore, we proactively equipped EGOIST nodes with the ability to compute BR's based on samples of the residual network, so as to be in position to handle possible future scale growth of classic overlay routing.

Shifting to unstructured P2P file sharing applications, we showed that applying the original BR formulation to P2P systems that perform scoped-flooding or random walk forwarding of requests is not beneficial. However, a simple reformulation of BR based on the realities of searching and bootstrapping in P2P can return significant performance boost in the quality of unstructured search.

There are several possible directions for future work. First, we would like to examine the effect of untruthful node behavior, where a selfish node lies while participating in the link-state routing protocol and announces fictitious weights for the links it maintains. Dealing with this behavior will require efficient auditing mechanisms to detect and possibly rewire around nodes making false claims. Second, we plan to offer EGOIST as a long-running service on PlanetLab, for application designers to use. Finally, at heart, EGOIST supports efficient many-to-many communications amongst nodes. We will explore leveraging this feature to simultaneously support multiple applications such as single-source multicast and distributed data stream querying over our common infrastructure.

8. REFERENCES

[1] Gnutella. http://www.gnutellanews.com.

[2] Kazaa peer-to-peer file sharing service. http://www.kazaa.com.

[3] S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash equilibria for a network creation game. In *Proc. of SODA '06*, Miami, FL, 2006.

[4] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. of ACM SOSP'01*, Banff, Canada, Oct 2001.

[5] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[6] A. Blanc, Y.-K. Liu, A. Vahdat, and S. Shenker. Designing incentives for peer-to-peer routing. In *Workshop on Economics of Peer-to-Peer Systems*, 2004.

[7] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making Gnutella-like P2P systems scalable. In *Proc. of ACM SIGCOMM '03*, pages 407–418, Karlsruhe, Germany, 2003.

[8] B.-G. Chun, R. Fonseca, I. Stoica, and J. Kubiatowicz. Characterizing selfishly constructed overlay routing networks. In *Proc. of INFOCOM'04*, 2004.

[9] B. Cohen. Incentives build robustness in bit torrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[10] J. Corbo and D. C. Parkes. The price of selfish behavior in bilateral network formation. In *Proc. of PODC'05*, Las Vegas, NV, 2005.

[11] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice Hall, 1994.

[12] Z. Duan, Z.-L. Zhang, and Y. T. Hou. Service overlay networks: SLAs, QoS, and bandwidth provisioning. *IEEE/ACM Transactions on Networking*, 11(6):870–883, 2003.

[13] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *Proc. of ACM PODC '03*, Boston, Massachusetts, 2003.

[14] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *Proc. of ACM EC '04*, pages 102–111, New York, NY, USA, 2004.

[15] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *in Proc. of ACM EC'04*. ACM Press, 2004.

[16] P. B. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. In *Proc. of ACM SIGCOMM '06*, Pisa, Italy, 2006.

[17] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. In *Proceedings of ACM SIGCOMM*, 2003.

[18] J. Han, D. Watson, and F. Jahanian. Topology aware overlay networks. In *Proc. of IEEE INFOCOM '05*, pages 2554–2565, Miami, FL, 2005.

[19] N. Laoutaris, R. Rajaraman, R. Sundaram, and S.-H. Teng. A bounded-degree network formation game, 2007. arXiv/CoRR cs.GT/0701071.

[20] N. Laoutaris, G. Smaragdakis, A. Bestavros, and J. Byers. Implications of selfish neighbor selection in overlay

networks. In *Proc. of IEEE INFOCOM '07*.

[21] V. Latora and M. Marchiori. Economic small-world behavior in weighted networks. *The European Physical Journal B*, 32:249–263, 2003.

[22] J. Ledlie, P. Pietzuch, and M. Parker. Pyxida. http://pyxida.sourceforge.net.

[23] J. Ledlie, P. Pietzuch, and M. Seltzer. Network Coordinates in the Wild. In *Proc. of NSDI '07*, Cambridge, MA, April 2007.

[24] Z. Li and P. Mohapatra. Impact of topology on overlay routing service. In *INFOCOM '04*, Hong Kong, 2004.

[25] Z. Li and P. Mohapatra. QRON: QoS-aware routing in overlay networks. *IEEE JSAC*, 22(1):29–40, Jan 2004.

[26] Y. Liu, H. Zhang, W. Gong, and D. F. Towsley. On the interaction between overlay routing and underlay routing. In *Proc. of IEEE INFOCOM '05*, pages 2543–2553, Miami, FL, 2005.

[27] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and a comparison of Peer-to-Peer overlay network schemes. *IEEE Communications Survey and Tutorial*, 7(2):72–93, 2005.

[28] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of ACM ICS '02*, pages 84–95, New York, NY, USA, 2002.

[29] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An Approach to Universal Topology Generation. In *Proc. of MASCOTS '01*, pages 346–354, Cincinnati, OH, Aug 2001.

[30] T. Moscibroda, S. Schmid, and R. Wattenhofer. On the topologies formed by selfish peers. In *Proc. of PODC'06*, Denver, Colorado, USA, 2006.

[31] A. Nakao, L. Peterson, and A. Bavier. A routing underlay for overlay networks. In *Proc of ACM SIGCOMM '03*, pages 11–18, 2003.

[32] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *Proc. ACM SIGCOMM'03*, pages 151–162, 2003.

[33] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *ACM SIGCOMM'01*, pages 161–172, San Diego, California, United States, 2001.

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

[35] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a dht. In *USENIX Annual Technical Conference*, 2004.

[36] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *Proc. of PAM'03*, La Jolla, CA, 2003.

[37] T. Roughgarden and E. Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.

[38] A. Rowstron and P. Druschel. Pastry: scalable, decentraized object location and routing for large-scale peer-to-peer systems. In *Proc. of IFIP/ACM Middleware'01*, Nov. 2001.

[39] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: Informed Internet routing and transport. *IEEE Micro*, 19(1):50–59, Jan./Feb. 1999.

[40] S. Seetharaman and M. Ammar. On the interaction between dynamic routing in the overlay and native layers. In *Proc. of IEEE INFOCOM '06*, Barcelona, Spain, 2006.

[41] S. Seetharaman, V. Hilt, M. Hofmann, and M. Ammar. Preemptive strategies to improve routing performance of native and overlay layers. In *Proc. of IEEE INFOCOM '07*, Anchorage, Alaska, 2007.

[42] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M. Fomenkov, and K. C. Claffy. Comparison of public end-to-end bandwidth estimation tools on high-speed links. In *Proc. of PAM'05*, Boston, MA, 2005.

[43] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.

[44] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *Proc. of IMC '06*, 2006.

[45] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. OverQoS: offering internet QoS using overlays. *SIGCOMM Comput. Commun. Rev.*, 33(1):11–16, 2003.

[46] K. Tamilmani, V. Pai, and A. E. Mohr. Swift: A system with incentives for trading. In *Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[47] L. Wang, K. Park, R. Pang, V. Pai, and L. Peterson. Reliability and security in the codeen content distribution network. In *USENIX*, 2004.

[48] Z. Yao, D. Leonard, X. Wang, and D. Loguinov. Modeling heterogeneous user churn and local resilience of unstructure d p2p networks. In *ICNP*, 2006.

[49] C. Yoshikawa. http://ping.ececs.uc.edu/ping/. Accessed on July 10, 2006.

[50] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. L. Peterson, and R. Wang. Overlay mesh construction using interleaved spanning trees. In *Proc of IEEE INFOCOM'04*, Hong Kong, 2004.

[51] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE JSAC*, 22(1):41–53, Jan. 2004.

[52] Y. Zhu, C. Dovrolis, and M. H. Ammar. Dynamic overlay routing based on available bandwidth estimation: A simulation study. *Computer Networks*, 50(6):742–762, 2006.