EGOIST: Overlay Routing using Selfish Neighbor Selection

Georgios Smaragdakis
BOSTON UNIVERSITY
gsmaragd@cs.bu.edu

Nikolaos Laoutaris
TELEFÓNICA RESEARCH, BARCELONA
nikos@tid.es

Azer Bestavros
BOSTON UNIVERSITY
best@cs.bu.edu

John W. Byers
BOSTON UNIVERSITY
byers@cs.bu.edu

Mema Roussopoulos
INSTITUTE OF COMPUTER SCIENCE, FORTH
mema@ics.forth.gr

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 an existing overlay, 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 analytically tractable, especially via game-theoretic analysis. In this paper, we unify these two thrusts by using insights gleaned from novel, realistic theoretic models in the design of EGOIST – a prototype overlay routing system that we implemented, deployed, and evaluated on PlanetLab. Using measurements on PlanetLab and trace-based simulations, we demonstrate that EGOIST's neighbor selection primitives significantly outperform existing heuristics on a variety of performance metrics, including delay, available bandwidth, and node utilization. Moreover, we demonstrate that EGOIST is competitive with an optimal, but unscalable full-mesh approach, remains highly effective under significant churn, is robust to cheating, and incurs minimal overhead. Finally, we discuss some of the potential benefits EGOIST may offer to applications.

1. INTRODUCTION

Motivation: Overlay networks are used for a variety of applications ranging from routing [4] to content distribution [40] to peer-to-peer file sharing. 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 domains, in which 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, as exemplified in studies relating to selfish (source) routing [28] and free riding [16] in P2P file-sharing networks.

Selfish behavior has many implications for connectivity management. In particular, it creates additional incentives for nodes to rewire, not only for operational purposes (bootstrapping and substituting nodes that went off-line), but also for seizing opportunities to incrementally maximize the local connection quality to 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 such selfish behavior, which include the obvious benefits to selfish nodes, but more surprisingly, to the network as a whole. 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) [21, 20]. This SNS formulation focused on characterizing the emergent overlay topology when overlay nodes behave selfishly and employ "Best-Response" (BR) neighbor selection strategies. Using BR a node chooses the best k neighbors that optimize its connection quality to the overlay, granted knowledge of how other nodes have connected among themselves.

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 shortest-path overlay routing performs much better over SNS topologies than over random and myopic ones.[1] Left unanswered in this prior work, though, is whether it is practical to build SNS-inspired overlays, how to incorporate additional metrics other than delay, *e.g.*, bandwidth, whether such overlays would be robust against network dynamics and whether they 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 prototype *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.

Our contributions can be summarized as follows. We first demonstrate through real measurements on PlanetLab that overlay routing atop EGOIST is significantly more efficient than systems utilizing common heuristic neighbor selection strategies under multiple performance metrics, including delay, system load 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 link announcements proportional to nk compared to n^2 for a full mesh topology. We also

introduce sampling techniques for reducing the link announcement overhead even further. Third, to accommodate high-churn environments, we introduce a hybrid extension of the "Best-Response" (BR) 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. Fourth, we consider the impact of cheaters – nodes that announce false information in order to benefit themselves, or harm the network. While such behavior can be identified and eliminated through the use of appropriate mechanisms, we show that EGOIST remains robust even without the use of such mechanisms. Finally, we discuss how EGOIST can provide a redirection stepping-stone for the benefit of applications that perform multi-path transmission in order to increase their effective maximum end-to-end rate, or their robustness to delay jitter and loss.

2. BACKGROUND

2.1 Basic Definitions

Let $V = \{v_1, v_2, \ldots, v_n\}$ denote a set of overlay routing 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} (in general, $d_{ji} \neq d_{ij}$). 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 the underlying IP path connecting 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 to 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 [21] as a strategic game where nodes are the

---

[1]Note that the only selfishly-motivated action of a node is the choice of its direct overlay neighbors and *not* the choice of the overlay path to various destinations. As we discuss later in section 2.2, this is not about "selfish routing", but rather it is about standard (shortest-path) routing over a selfishly constructed overlay topology.

players, wirings are the strategies, and C_i's are the cost functions. It was shown that under hop-count distance, obtaining the BR of v_i requires solving an asymmetric k-median problem on the residual wiring S_{-i} and is, therefore, NP-hard. In [20] 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 of that 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 inspired by the SNS game [21, 20]. While these works presented basic theoretic and experimental results, they did not consider any of the practical systems issues that are covered in this paper, such as dealing with churn in realistic network conditions or achieving high global performance without the computational and control message overheads required by theoretical formulations. Network Creation Games that predate SNS [15, 3, 12, 27, 10] have considered settings in which nodes may buy as many links (neighbors) as they like and thus differ fundamentally from our work, in which constraints on the number of neighbors play a central role.[2] Also, fundamentally different is the work on Selfish Routing [28, 32], in which the network topology is part of the input to the game, and selfish source routing is the outcome. In a way, this is the inverse of our work, in which network-based (shortest-path) routing is an input of the game, and topology is the outcome.

A number of routing overlay systems have been recently proposed [33, 4, 25, 24, 43, 26, 19, 44, 34, 35, 38, 14]. 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 [19] routing protocol to route at the overlay layer. Some overlay routing systems optimize route hop count [24, 34, 35], others optimize for application delay [33, 4, 28, 25, 43, 26, 19], and others optimize for available bandwidth [44]. These works assume that either all overlay nodes are under central control and thus obediently follow simple empirical neighbor selection strategies as discussed earlier, or bypass the issue altogether by assuming that some fixed overlay design is already in place. With reference to the employed metric, in our work, we provide mechanisms to support optimization of *all* aforementioned metrics and leave it up to the application designers to choose the most suitable one.

Selfish behavior has been studied in the context of providing incentives for nodes to route traffic for others [7].[3] Such works are complementary to ours since we assume that an external mechanism exists for incentivizing forwarding for other nodes. Chawathe et al. [9] proposed mechanisms for dealing with selfish nodes that lie about their capacities to avoid receiving queries. While we visit some of these issues in this paper, we note that this prior work did not focus on neighbor selection nor did it impose any constraints on node degrees.

In structured DHTs, proximity neighbor selection has been proposed to make the overlay topology match the underlying IP topology as much as possible [29, 18] in order to achieve faster lookups: Nodes can choose the physically closest nodes from a set of candidate nodes. While this approach gives nodes some flexibility in choosing neighbors selfishly, the set of nodes from which the choice must be made is constrained by node ID. In our work, we focus on *unstructured* networks and do not constrain a node's choices except to cap its degree, a practice in line with all currently deployed unstructured networks, and justified by system implementation/performance concerns (*e.g.*, fragmentation of up-link bandwidth).

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 its (weighted) sum of distances to all destinations under shortest-path routing. 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 and which are incorporated in EGOIST's implementation – 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,

---

[2] We also note that in our target (overlay networks) setting, it is more realistic to impose a limit on the number of neighbors as opposed to a price on the link to a neighbor. This latter assumption is better justified for connectivity at the physical as opposed to overlay layer.

[3] The use of incentives has also been studied in other contexts that are fundamentally different from ours, *e.g.*, P2P file sharing [16, 11, 39].

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 of 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.[4] 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 monitor and send updates only for the k links that it chooses to establish, with $O(n)$ measurements to all nodes in the overlay done much less frequently – namely once per *wiring epoch*, which is defined as the period T between two successive evaluations by a node of its set of candidate links and possible adoption of a new wiring (*i.e.*, re-wiring) based on such evaluation. *Since re-wiring is much less frequent than monitoring of the established k links, the load imposed by the link-state protocol is only $O(nk)$ and not $O(n^2)$.*

3.2 Neighbor Selection Policies in EGOIST

As its namesake suggests, the default neighbor selection policy in EGOIST is the Best-Response (BR) strategy described in Section 2.1, and detailed in [21]. 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). Since obtaining an exact BR is NP-hard under both delay [21] and throughput, in Section 4.1, we employ fast approximate versions based on local search to reduce computational costs and enhance scalability (more in Section 5). In addition to BR, we have also implemented the following neighbor selection policies, to enable comparative evaluation.

***k*-Random**: Each node selects k neighbors randomly. If the resulting graph is not connected, we enforce a cycle.

***k*-Closest**: Each node selects its k neighbors to be the nodes with the minimum link cost (*e.g.*, minimum delay from it, maximum bandwidth, *etc.*). Again, if the graph is not connected, we enforce a cycle.

***k*-Regular**: In this case, all nodes follow the same wiring pattern dictated by a common offset vector $o = \{o_1, o_2, \ldots, o_k\}$, used as follows: node i connects to nodes $i + o_j \mod n$, $j = 1, \ldots, k$. In our system, we set $o_j = 1 + (j - 1) \cdot \frac{n-1}{k+1}$.[5] One way to visualize this is to consider that all nodes are placed on a ring according to their ids (as with 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

EGOIST's BR neighbor selection strategy assumes that existing nodes never leave the overlay. Therefore, even in an extreme case in which some nodes are reachable through only a unique path, a node can count on this path always being in place (re-wirings by other nodes will not tear it down as this would also disconnect them [20]). Overlay routing networks (*e.g.*, RON [4]) are not inherently prone to churn to the extent that file-sharing P2P-networks [17, 30] are. Nonetheless, nodes may occasionally go down, or network problems may cause transient disconnections until successive re-wirings establish new paths. One could 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, particularly for large networks [41].

In EGOIST we follow a different approach reminiscent of how k-Random and k-Closest policies ensure overlay connectivity. We introduce 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 by providing redundant routes.

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. [43] 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

---

[4] Notice that d_{ij} can be obtained through active or passive measurements depending on the metric of interest (see Section 4.1 for details).
[5] To simplify the presentation, we assume that $n - 1$ is a multiple of $k + 1$.

the k_2 links can be achieved by simple re-formulation of the ILP model of [21], in which the decision variables Y_i are set to 1 in correspondence to the nodes that receive high maintenance links.

We have implemented HybridBR in EGOIST. As hinted above, 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 re-wiring of a dropped link to be performed in one of two different modes: *immediate* and *delayed*. In immediate mode, re-wiring is done as soon as it is determined that the link is dropped, whereas in delayed mode re-wiring is only performed (if necessary) at the preset *wiring epoch T*. Unless otherwise specified, we assume a delayed re-wiring mode is in use.

3.4   Dealing with Cheaters in EGOIST

In this paper, we make the case for selfish neighbor selection in the sense that we do not consider selfishness in selecting one's neighbors as an anti-social behavior that needs to be mitigated. In this section, we briefly examine harmful ways in which a node may be selfish (or "cheat" its way through), as well as possible countermeasures.

The most blatant form of cheating is *free-riding*, *i.e.*, using the system to route one's own traffic but denying routing to any incoming traffic from other nodes. Dealing with such behavior has been the subject of a number of studies, including the works in [7, 8] which propose the adoption of reputation and repudiation or punishment mechanisms that act as incentives for nodes to route, and/or expel misbehaving nodes from the system. These studies are orthogonal to and thus complement our work.

A more elaborate way for a node to cheat is to announce false information via the link-state protocol to discourage others from picking it as an upstream neighbor. For example, a node can cheat by falsely announcing larger-than-actual delays for its potential outgoing links. One could add mechanisms to detect this type of cheating. If the construction of the overlay is based on passive measurements obtained from a virtual coordinate system (as discussed in Section 4.1), then nodes could periodically select a random subset of remote nodes and "audit them" by asking the coordinate system for the delays of the outgoing links of the audited nodes and comparing them to the actual

values that the audited nodes declare on the link-state routing protocol. Similar audits can be designed using active probing by sending traffic and measuring its delay and comparing it to the expected delay based on the delays that nodes on the end-to-end path declare. In Section 4.5, we evaluate the impact of broadcasting false information to cheat the system: we show that even without the use of the aforementioned audit mechanisms, EGOIST is robust to this form of cheating.

4.   EXPERIMENTAL EVALUATION

4.1   Cost Metrics

As we alluded earlier, a number of metrics could be used to measure the "cost" of traversing an overlay link. 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 incorporated in EGOIST and discuss how each such metric was evaluated in our implementation.

Link and Path Delays: Delays are natural cost metrics for many applications, especially those involving interactive communication. 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 estimate 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, 2]. Using `ping`, one-way delay is estimated to be one 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`, delay estimates are available through a simple query to the `pyxida` system.[6]

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 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

---

[6]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=AvailBW($G(V, E), \ s \in V$)

1: Set $W = \{s\}$; and $\rho[s] = \infty$;
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 ρ;

---

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 [36] for an exposition). In EGOIST, we used `pathChirp` [31], a light-weight, 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 requires 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

$$AvailBW(v, u) = \max_{p \in P(v,u)} AvailBW(p),$$

where the available bandwidth for a path p is given by:

$$AvailBW(p) = \min_{e \in p} AvailBW(e),$$

and $P(v, u)$ denotes the set of paths that connects v to u. Thus, finding $P^*(v, u)$ that maximizes the available bandwidth between v and u, and the bottleneck edge, is a "Maximum Bottleneck Bandwidth" problem which can be solved using a simple modification of Dijkstra's [13].

Using available bandwidth as the cost metric also requires us to modify the local objective function for computing BR wirings. 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 average of the bottleneck bandwidths to all destinations. In Appendix A.1 we show that finding a local wiring s_i that maximizes this objective function is an NP-hard problem. Thus in our implementation we used a fast local-search heuristic that we verified to be within 5% of optimal in the tested scenarios (we also added infinity costs to guarantee connectedness). Notice that it is straightforward to use the above definitions to produce alternative formulations, *e.g.*, consider the maxi-

mization of the *minimum* of the bottleneck bandwidths to all destinations.[7]

4.2 Baseline Experimental Results

In this section, we present performance results obtained through measurement of EGOIST. These results allow us to make 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 4.4.

Experimental Setting: We deployed EGOIST on $n = 50$ PlanetLab nodes (30 in North America, 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 wiring epoch $T = 60$ seconds. EGOIST nodes are not synchronized, thus on average a re-wiring by some EGOIST node occurs every $T/n = 1.2$ seconds. Whether a node ends up re-wiring 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 re-wiring. For k-Closest, re-wiring would only be the result of dynamic changes in PlanetLab that result in changes to the cost metric in use (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 re-wiring. While in theory [20, 21], 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 re-wiring by EGOIST nodes. Setting the wiring epoch T in EGOIST has the effect of controlling the timescale of, and consequently the overhead incurred by, BR re-wiring.

Each experiment presented in this section reflects the results obtained by running EGOIST for 10 hours on PlanetLab on January 5th, January 15th and September 15th 2007.

Performance Metric: To be able to compare the impact of neighbor selection on the quality of the resulting overlay, throughout this paper we use the *routing cost* (for an individual node or averaged over all nodes) as the main performance metric. For each experiment, an individual cost metric is calculated for every one of the

---

[7]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.
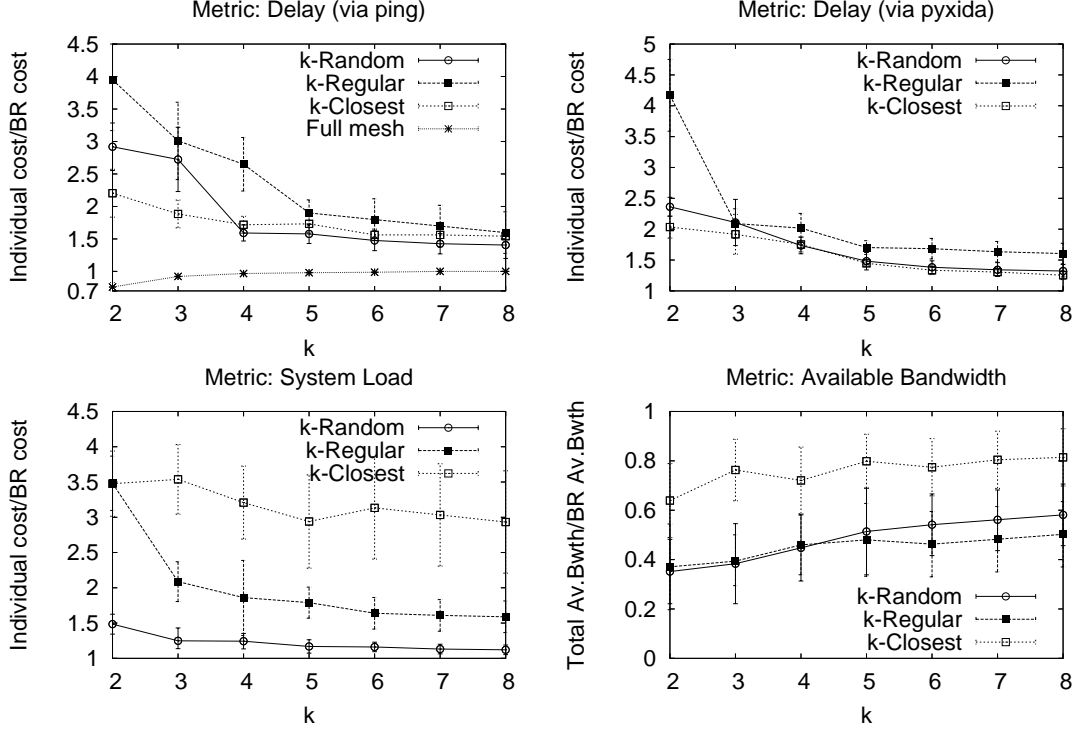
Figure 1: PlanetLab baseline experiments showing the individual costs for various neighbor selection policies (normalized with respect to BR costs) as a function of number of neighbors k for a 50-node EGOIST overlay: Cost metric is `ping` delays (top-left), `pyxida` delays (top-right), node CPU load (bottom-left), and available bandwidth (bottom-right).

$n = 50$ nodes in the system. 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.[8] For each experiment we report the mean of all $n = 50$ individual costs, as well as the 95th-percentile confidence interval.

To facilitate comparisons between various neighbor selection strategies, we often report the *normalized routing cost*, which is the ratio of the cost achievable using a given strategy to that achievable using BR.

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 neighbor 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 con-

ducted *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 [21] 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 work do not hold.

---

[8]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: Fig. 1 shows the performance of the various neighbor selection policies in EGO-IST 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 Fig. 1 we also show the performance achieved by deploying EGO-IST 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 EGO-IST 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 Fig. 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 Fig. 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 Fig. 1 shows the results we obtained using available bandwidth as the cost metric. Recall that, here, the objective is to get the highest possible *aggregate* 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-fold to four-fold improvement over the other policies, over a wide range of values of k.

4.3 Measurement and Re-wiring Overheads

The traffic injected in the network for maintaining EGO-IST is very small. In this section we quantify EGOIST's overheads and validate our expectations through experimental evaluations.

Active Measurement Load: As mentioned in Section 4.2, with no node churn, k-Random and k-Regular do no feature any re-wiring, and thus do not incur active measurement load. For k-Closest and BR, the active measurement load is identical.

When the cost metric is delay via `ping`, ICMP messages of size 320 bits each (`ECHO` requests/replies) are exchanged once per wiring epoch T. Notice that for established links, there is no need for active measurements since the cost metric for a link would be available by virtue of its use. Thus, the overhead is $\approx (n - k - 1) \cdot 320/T$ bps per node. Using `pyxida`, a single (`http`) request/reply to the `pyxida` server yields the (virtual coordinate space) distances between the node initiating the request and all other nodes in the overlay. This is clearly more efficient than using `ping`, as it injects $\approx (320 + 32n)/T$ bps per node.[9] When the metric is system load, there is no overhead imposed on the network as the system load is measured locally at

---

[9]Measurements showed that a rate of one message per (one minute) wiring epoch per node was sufficient to sustain a coordinate system in PlanetLab.
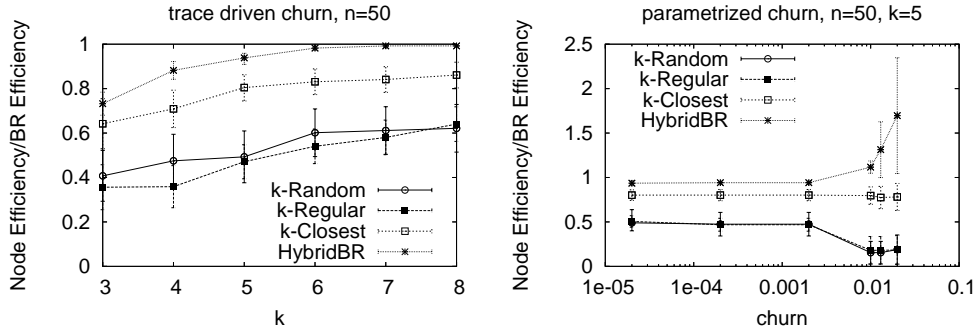
Figure 2: PlanetLab experiments with node churn showing the efficiency of neighbor selection policies (normalized with respect to BR) as a function of the number of neighbors k (left) and churn (right) for a 50-node EGOIST overlay.

each node (using the `loadavg` tool). Finally, when the metric is available bandwidth, our experimental results showed that the bandwidth needed for accurate probing of available bandwidth between two nodes in the overlay is less than 2% of the available bandwidth between these two nodes.

Link-State Protocol Load: The overhead (in terms of additional injected traffic) imposed by the link-state protocol is also low. Each node broadcasts a packet with its ID, its neighbors' IDs and the cost of the established links to its k neighbors every $T_{\text{announce}} < T$. The header and padding of the link-state protocol messages require a total of 192 bits, and the payload per neighbor requires 32 bits. Thus, the overhead in terms of injected traffic on the overlay is $\approx (192 + 32 \cdot k)/T_{\text{announce}}$ bps per node. In our experiments we set T_{announce}=20 secs.

Re-wirings Overhead: Fig. 3 (left) shows the total number of re-wirings per (one minute) epoch for the entire overlay over time. The results suggest that the re-wiring rate decreases fast as EGOIST reaches a "steady state" and that the re-wiring rate is minimal for small values of k. Here we note that as k increases the re-wiring rate increases, but the improvement (in terms of routing cost) is marginal, as a small number of outgoing links is sufficient to significantly decrease the cost. This is evident in Fig. 3 (center). Finally, we also note that the re-wiring rate can significantly be decreased (with marginal impact on routing cost) by requiring that re-wiring be performed only if connecting to the "new" set of neighbors would improve the local cost to the node by more than a given threshold ϵ. We refer to this modified version of BR as BR(ϵ). Fig. 3 (right) confirms this by showing the number of re-wirings and resulting performance when $\epsilon = 10\%$.

4.4 Effect of Churn

In the original SNS formulation [21, 20], 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 max-

imum distance. Node churn, however, can lead to disconnected graphs, therefore we have to use a different metric. For that purpose, we choose the *Efficiency* metric, 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:

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

To evaluate the efficiency of nodes in 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 re-wiring 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 [17], with adjustments to the timescale to control the intensity 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 (see 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 Fig. 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 (*i.e.*, the typical churn in
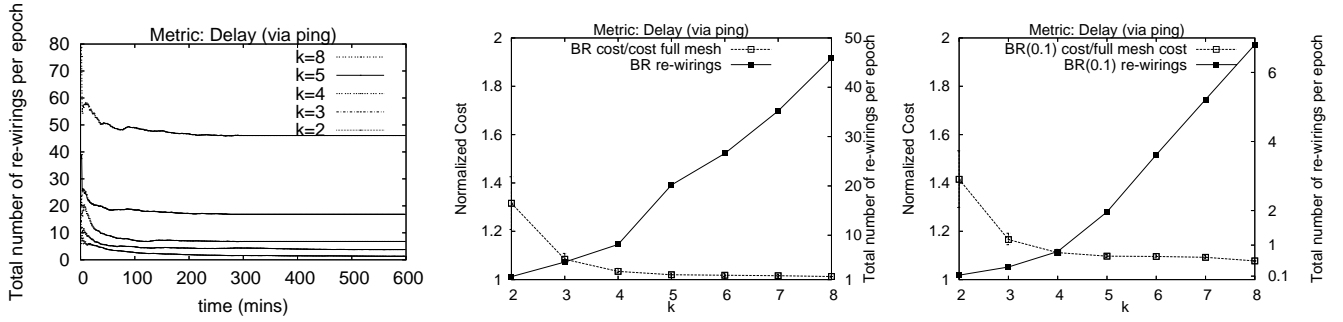
Figure 3: PlanetLab experiments showing the total number of re-wirings per epoch in the system (left), and the relationship between individual cost and total number of re-wirings per epoch in the system with exact best response and an approximate best response with $\epsilon = 10\%$ (center and right respectively), as a function of the number of neighbors k in our EGOIST overlay.

PlanetLab), 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, re-wiring 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 re-wiring 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 Fig. 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 [17]) 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 new set of nodes in the overlay following an event i that alters the membership in the set of nodes that participate in the overlay, 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 both k-Random and k-Regular fall dramatically, whereas that of k-Closest remains level with that of BR.

4.5 Vulnerability to Abuse

As we discussed in section 3.4, nodes (which we term as "free riders") may attempt to cheat by misrepresenting their cost to neighbors in order to benefit from EGOIST without contributing their own resources to the overlay. Due to the combinatorial nature of the optimization problem underlying BR re-wiring, it is very hard for individual free riders to derive the proper costs that will lead to wirings that will be of benefit to them individually, while harming others. Theoretical results [6] advocate that such behavior may even lead to worse equilibria for free riders in routing games. Thus, in this section, we present results from a series of experiments aimed to assess EGOIST's vulnerability to free riders that misrepresent the cost of their outgoing links (simply by inflating them), in the hopes of discouraging others from selecting them as neighbors.

As described in Section 3.4, one could add mechanisms to detect when free riders make such false representations. These mechanisms would take the form of passive or active measurement audits from other nodes. Determining how often nodes should perform such random audits and what these nodes do when cheating nodes are identified can be complex. Thus, it would be preferable if one can show that the impact from such abuse is minimal. Clearly, an assessment of the impact of the full spectrum of possible false announcements is beyond the scope of this paper. Thus, we only consider the impact from inflated delay announcements by a single node and by a variable fraction of the nodes.

In Fig. 4 (left), we show the impact from a single free rider announcing link costs that are twice as high as the real ones. The figure shows the individual cost for both the free rider and for all other nodes for different values of k. The cost for both types of nodes (free rider and non-free rider) is very close to the cost without the presence of the free rider. We also evaluate the robustness of EGOIST in the presence of many free riders (up to one-third of the population). These results, shown in Fig. 4 (right), yield consistent observations even when the number of outgoing links is very small
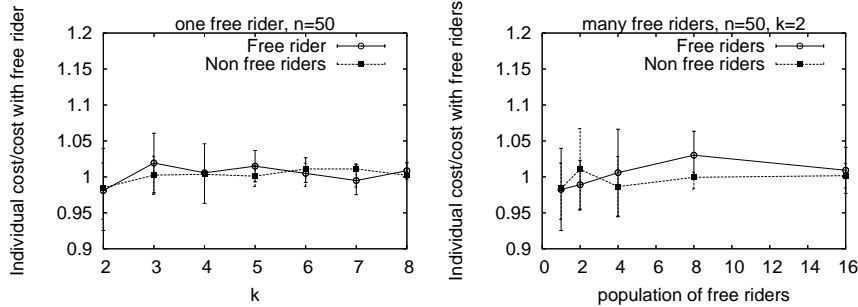
Figure 4: PlanetLab experiments with free riders showing the robustness of neighbor selection policies (normalized with respect to BR) as a function of the number of neighbors k in the presence of one free rider (left) and many free riders for $k = 2$ (right) for a 50-node EGOIST overlay.

$(k = 2)$, which is the setting in which the impact of bad re-wirings is amplified. These results provide evidence that EGOIST is fairly robust to abuse by free riders, even without the deployment of auditing mechanisms.[10]

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 compute BR wirings based on only these samples. Such a technique might not be necessary for current overlay networks that are of small to moderate sizes, such as PlanetLab, 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 overlay routing network 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, not n.

A more significant scaling issue is imposed by the computational complexity of computing best responses. As mentioned before, computing an exact BR is an NP-hard problem. Approximate solutions based on local search perform well in practice. However, even local search [5], imposes substantial computational burden (polynomial number of iterations, each one requiring $n^{O(p)}$, $p \in [1, k]$ being a parameter of the algorithm). Such high order polynomial complexity becomes difficult to handle for large 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 biased sampling.

Topology-Based Biased Random Sampling: The basic idea 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 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

---
[10]Similar observations were also obtained when the abuse amounted to advertising lower values than the actual delays.
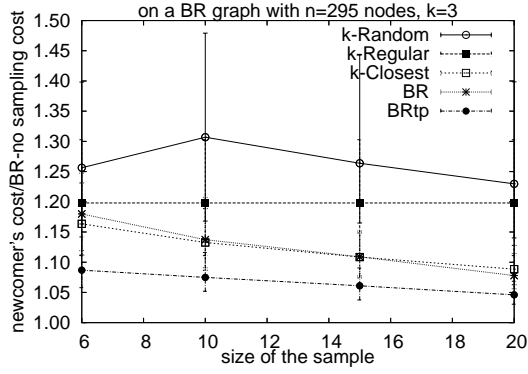
Figure 5: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a BR graph.
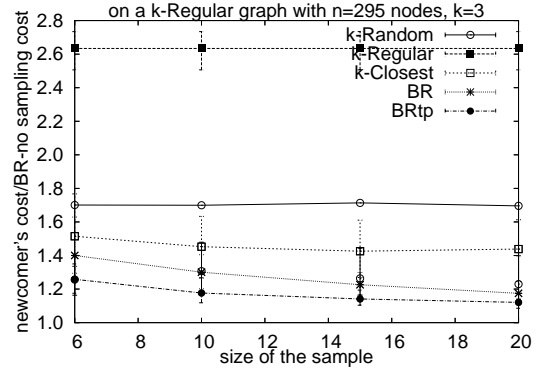

Figure 7: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a k-Regular graph.
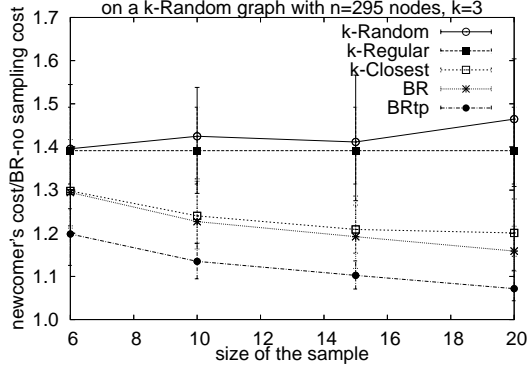

Figure 6: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a k-Random graph.
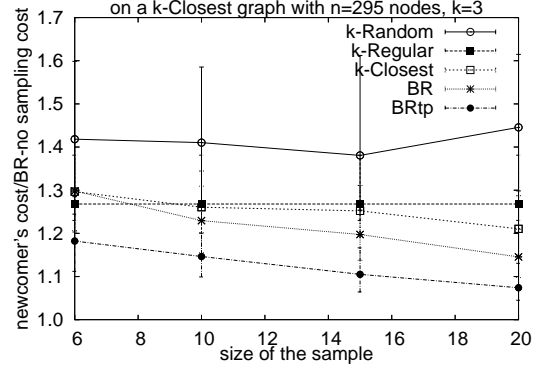

Figure 8: PlanetLab Simulation. The cost incurred by simple wiring strategies (k-Random, k-Regular, k-Closest with random sampling), BR with random sampling, and BR with topology-based biased random sampling (normalized against the cost of BR with no sampling) in a k-Closest graph.

based methods can query a set of m' pseudorandomly-generated nodes in a k-regular graph with suitable expansion properties using $\mathrm{O}(m' \log n / \log k)$ messages. Each node must be able to approximately maintain and express the number of nodes within its r-radius neighborhood, which requires $\mathrm{O}(k^r)$ space. Nodes also must compute the b_{ij} values, which requires $\mathrm{O}(m'k^r)$ 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: To assess scalability, instead of our 50 node PlanetLab prototype, we use a publicly available trace [42] containing delays obtained using `pings` between all pairs of existing PlanetLab sites, as well as synthetic topologies from BRITE and real

AS topologies. We use these data sets to conduct simulations. Due to space limitations, we only report on the PlanetLab simulations (results obtained in the other settings were similar). 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 the neighborhood size $r = 2$. In Fig. 5, 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.
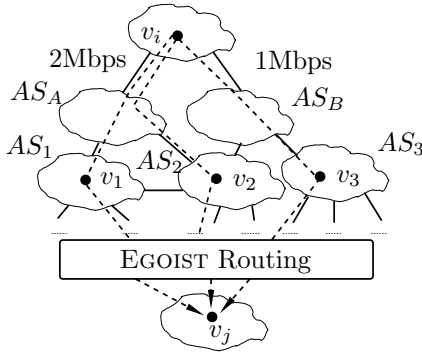
Figure 9: Source node v_i sending to target v_j through its $k = 3$ immediate EGOIST neighbors v_1, v_2 and v_3. v_i takes full advantage of its 2-homed AS_i: $v_i \rightarrow v_1$ and $v_i \rightarrow v_2$ use the maximum allowed bandwidth at the peering point with AS_A (2 Mbps), whereas $v_i \rightarrow v_3$ uses the maximum allowed bandwidth at the peering point with AS_B (1 Mbps). Assuming no bottlenecks exist further down, this gives an aggregate transmission rate of 3 Mbps, whereas any single-path scheme (even with parallel connections) would have limited to 1 or 2 Mbps.

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. 6, 7, and 8).

6. DISCUSSION AND APPLICATIONS

EGOIST is a general purpose overlay routing network that can be used by applications to supplement traditional IP routing. The main difference between an EGOIST overlay and other routing overlays is that by virtue of its BR-wiring strategy, an application contacting its local EGOIST node can be assured that this node will provide better paths than a node that connects to the overlay non-selfishly, using previously-mentioned random or myopic heuristics. Stated otherwise, the selfish selection of neighbors in EGOIST is just a manifestation of the desire of local applications to get the best possible service for themselves. In the rest of this section we discuss two example applications and attempt to quantify potential benefits based on our PlanetLab prototype. We leave full implementation and detailed evaluation of such applications to future work.

6.1 Multipath File Transfer

File transfer applications can take advantage of redirection opportunities offered by (bandwidth-based) EGOIST to increase their effective end-to-end transmission rates by performing multipath transfers through first-hop neighbors. The idea is quite simple: Source node v_i uses EGOIST to establish up to k parallel sessions to a target node v_j, each one redirected through a different first-hop EGOIST neighbor $v_l \in s_i$. Each session requires establishing two virtual channels over EGOIST: $v_i \rightarrow v_l$ (single-hop overlay path) and $v_l \rightarrow v_j$ (multi-hop overlay path). The purpose of redirection through neighbors is to take advantage of potentially multihomed source and target ASes (henceforth AS_i and AS_j) and thus alleviate bottlenecks caused by session-level[11] traffic shaping and rate-limiting at AS peering points [1]. As long as the number of EGOIST neighbors k is sufficiently larger than $|AS_i|$, the number of ASes to which AS_i has a peering relationship, there is good chance that at least one overlay neighbor is behind each peering point. Redirecting through this neighbor permits v_i to utilize up to the maximum allowed rate at that peering point (see Fig. 9 for an illustration). If peering points permit a given maximum rate for each session, the aforementioned multi-path redirection can increases the maximum total rate out of v_i by up to a multiplicative factor $|AS_i|$ (observe that establishing the same number of parallel connections going over the same path would not yield the same benefit, since they will all be part of a single session, and hence be subject to the same rate limits at peering points). Of course, the real end-to-end benefit can be much smaller due to bottlenecks on the overlay paths from v_l to v_j, especially in the last hops before closing-in on the target v_j (large $|AS_j|$'s working again in favor of the application). To get a feeling for the potential benefits on a real topology, we perform the following experiment.

In our 50-node EGOIST overlay, we select a source-target pair and we estimate the available bandwidth that can be realized if the source establishes k parallel connections going through its immediate neighbors. Then we compare this value with the available bandwidth that is realized when the source routes the traffic using the unique path to the destination offered by IP. We repeat the experiment for all source-target pairs and we plot the average along with the 95th-percentile confidence intervals in Fig. 10. Furthermore, we estimate the theoretically maximum available bandwidth that can be realized when all peers allow multipath redirections for all source-target pairs (*i.e.*, when the total bandwidth becomes equal to a max-flow from v_i to v_j).

6.2 Real-time traffic over IP

Applications that transmit real-time (*i.e.*, delay- and

---

[11] A session identified as a (source,target) IP pair.
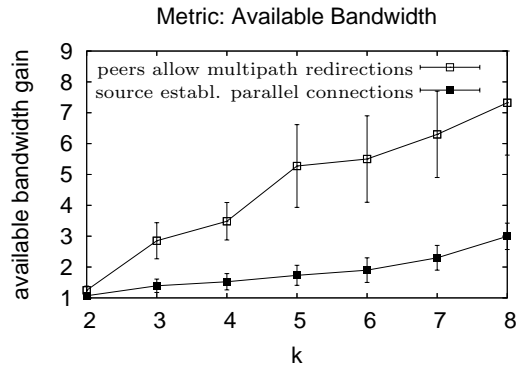
Metric: Available Bandwidth

Figure 10: PlanetLab experiment showing the available bandwidth gain between source node u_i and target node u_j when the source establishes k parallel connections and when all the peering points between the aforementioned nodes, allow multi-path redirection in our 50-node EGOIST overlay.
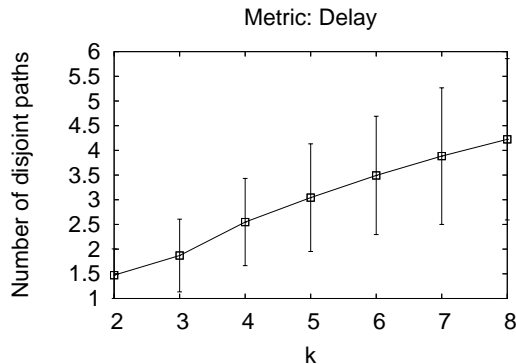
Metric: Delay

Figure 11: PlanetLab experiment showing the available bandwidth the number of disjoint paths between the source node u_i and target node u_j when the source establishes k parallel connections in our 50-node EGOIST overlay.

loss-sensitive) traffic can use the redirection infrastructure of (delay-based) EGOIST to send additional copies of the original stream through multiple disjoint paths, thus improving the chance that at least one copy of every packet will reach its destination before the designated playout time [22]. Some P2P voice-over-IP (VoIP) applications, like *Skype*, are already in position to implement such schemes as they have achieved a huge user-base that provides ample opportunities for redirection. EGOIST on the other hand can assist applications that have not yet achieved high penetration (*e.g.*high quality video-conferencing) and thus would otherwise have to rely on delay jitter-prone single-path delivery. Substantiating this claim requires measuring precise timing information and making sure that OS introduced delays do not interfere with the purpose of the experiment. We leave such elaborate experiments to future work. Our initial results show that the number of disjoint paths increases linearly with the number of parallel connections, as illustrated in Fig. 11.

7. EGOIST ARTIFACTS

Our EGOIST prototype is currently deployed on PlanetLab. A live demonstration of the overlay routing topolopy maintained by EGOIST can be accessed from the EGOIST project web site at `http://csr.bu.edu/sns/`. Traces from all experiments used in this paper (as well as others which we could not present due to space limitations) are also available from the project web site. Upon publication of this work, EGOIST source code will be released to the research community.

8. CONCLUSION

In this paper we have shown how recent theoretical results on Selfish Neighbor Selection (SNS) could be leveraged for overlay routing applications. Through the development and deployment of our EGOIST prototype routing network on PlanetLab, we have established that Best-Response (BR) neighbor selection strategies can indeed be realized in practice, that they provide a substantial performance boost when compared to simpler empirical strategies, and that they scale much better than full-mesh approaches which require intensive monitoring of $O(n^2)$ links. We have substantiated these benefits under different performance metrics, active and passive link monitoring strategies, in static and churn-prone environments, and in the presence of truthful and untruthful nodes. Furthermore, we proactively equipped EGOIST nodes with the ability to compute their best responses based on samples of the residual network, so as to be in position to handle possible future scale growth in overlay routing networks.

9. REFERENCES

[1] P2P-2-ISP Peace Pipe Could Ease Bandwidth.
   `http://www.wired.com/software/webservices/news/2007/08/p2p`.
[2] Pyxida. `http://pyxida.sourceforge.net`.
[3] S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and
   L. Roditty. On Nash equilibria for a network creation game.
   In *SODA*, 2006.
[4] D. Andersen, H. Balakrishnan, F. Kaashoek, and
   R. Morris. Resilient overlay networks. In *SOSP*, 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] M. Babaioff, R. Kleinberg, and C. H. Papadimitriou.
   Congestion games with malicious players. In *EC*, 2007.
[7] A. Blanc, Y.-K. Liu, A. Vahdat, and S. Shenker. Designing
   incentives for peer-to-peer routing. In *P2PEcon*, 2004.
[8] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and
   D. Wallach. Secure routing for structured peer-to-peer
   overlay networks. In *Proc. of OSDI*, 2002.
[9] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and
   S. Shenker. Making Gnutella-like P2P systems scalable. In
   SIGCOMM, 2003.
[10] B.-G. Chun, R. Fonseca, I. Stoica, and J. Kubiatowicz.
   Characterizing selfishly constructed overlay routing
   networks. In *INFOCOM*, 2004.
[11] B. Cohen. Incentives build robustness in bit torrent. In
   Workshop on Economics of Peer-to-Peer Systems, 2003.
[12] J. Corbo and D. C. Parkes. The price of selfish behavior in
   bilateral network formation. In *PODC*, 2005.

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

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

[15] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *PODC'03*.

[16] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *EC*, 2004.

[17] P. B. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. In *ACM SIGCOMM*, 2006.

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

[19] J. Han, D. Watson, and F. Jahanian. Topology aware overlay networks. In *INFOCOM*, 2005.

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

[21] N. Laoutaris, G. Smaragdakis, A. Bestavros, and J. Byers. Implications of selfish neighbor selection in overlay networks. In *INFOCOM*, May 2007.

[22] N. Laoutaris and I. Stavrakakis. Intrastream synchronization for continuous media streams: A survey of playout schedulers. *IEEE Network Magazine*, 16(3), May 2002.

[23] J. Ledlie, P. Pietzuch, and M. Seltzer. Network Coordinates in the Wild. In *NSDI*, April 2007.

[24] Z. Li and P. Mohapatra. Impact of topology on overlay routing service. In *INFOCOM*, 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 *INFOCOM*, 2005.

[27] T. Moscibroda, S. Schmid, and R. Wattenhofer. On the topologies formed by selfish peers. In *PODC*, 2006.

[28] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *SIGCOMM*, 2003.

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

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

[31] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *PAM*, 2003.

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

[33] 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.

[34] S. Seetharaman and M. Ammar. On the interaction between dynamic routing in the overlay and native layers. In *INFOCOM*, 2006.

[35] S. Seetharaman, V. Hilt, M. Hofmann, and M. Ammar. Preemptive strategies to improve routing performance of native and overlay layers. In *INFOCOM*, 2007.

[36] 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 *PAM*, 2005.

[37] G. Smaragdakis, N. Laoutaris, A. Bestavros, J. Byers, and M. Roussopoulos. Improving the Performance of Overlay Routing and P2P File Sharing using Selfish Neighbor Selection. Technical Report BUCS-TR-2007-006.

[38] 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.

[39] K. Tamilmani, V. Pai, and A. E. Mohr. Swift: A system with incentives for trading. In *P2PEcon*, 2004.

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

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

[42] C. Yoshikawa. http://ping.ececs.uc.edu/ping/.

[43] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. L. Peterson, and R. Wang. Overlay mesh construction using interleaved spanning trees. In *INFOCOM*, 2004.

[44] 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.

APPENDIX

A.1 NP-hardness of maximizing the sum of bottleneck bandwidths

Consider a node s that wants to connect to a network composed of m nodes $v_i \in C$, $1 \le i \le m$ and n nodes $u_j \in S$, $1 \le j \le n$, so as to maximize the sum of bottleneck bandwidths to all destinations as described in Sect. 4.1. Each node v_i has directed links of bandwidth b_2 to a subset S_i of the nodes of S. Node s has k links of bandwidth b_1 which it wants to use for connecting to k distinct nodes of $C \cup S$ (see Fig. 12 for an illustration). Establishing a link to a node of S increases the utility of s by at most b_1 independently of how it uses its remaining $k - 1$ links. Establishing a link to a node of C increases its utility by b_1 plus b_2 times the number of nodes of S that s reaches by following the new link. When s can reach a node u_j through direct links to more than one nodes v_i we pick exactly one of the paths $s \to v_i \to u_j$ to be the bottleneck path for destination u_j (all have bandwidth $\min(b_1, b_2)$ so it doesn't matter which one we choose). Granted this construction, it is clear that s will establish direct links only to nodes of C. More over, it will have to choose those nodes of C that maximize the number of distinct reachable nodes of S. Therefore, a solution that maximizes the sum of bottleneck bandwidths to nodes of $C \cup S$ implies an optimal solution to the MAX-UNIQUES problem which is shown in Appendix A.2 to be NP-hard. Therefore, maximizing the sum of bottleneck bandwidths is also an NP-hard problem.
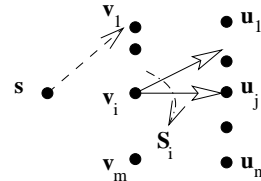

Figure 12: Reduction from MAX-UNIQUES to max sum of bottleneck bandwidths.

A.2 NP-hardness of MAX-UNIQUES

Given m subsets S_i of a set S with $|S| = n$, MAX-UNIQUES is an optimization problem that requires choos-

ing k of these subsets so as to maximize the cardinality of their union U^k. The corresponding decision problem dubbed UNIQUES asks whether there exists a collection of k subsets whose union has cardinality σ. UNIQUES is obviously NP-complete since for $\sigma = n$, a solution to UNIQUES implies a solution to the SET-COVER problem that asks whether there exist k subsets S_i that cover S. Therefore, MAX-UNIQUES is NP-hard.