Placing Relay Nodes for Intra-Domain Path Diversity

Meeyoung Cha[†], Sue Moon[†], Chong-Dae Park[‡],
and Aman Shaikh[*]

CS/TR-2005-214

May 17, 2005 (revised July 6, 2005)

K A I S T
Department of Computer Science

[†] Meeyoung Cha and Sue Moon are supported by Korea Science and Engineering Foundation (KOSEF) through Advanced Information Technology Research Center (AITrc). [‡] Chong-Dae Park is supported by Brain Korea 21 (BK21) project through the school of information technology in KAIST. [*] Aman Shaikh is with AT&T Labs – Research, at 180 Park Avenue, Florham Park, NJ 07932, USA.

Placing Relay Nodes for Intra-Domain Path Diversity

Meeyoung Cha[†], Sue Moon[†], Chong-Dae Park[‡], and Aman Shaikh[*]

Abstract

To increase reliability and robustness of mission-critical services in the face of routing changes, it is often desirable and beneficial to take advantage of path diversity provided by the network topology. One way of achieving this inside a single Autonomous System (AS) is to use two paths between every Origin-Destination (OD) pair. One path is the default path defined by the intra-domain routing protocol; the other path is defined as an overlay path that passes through a strategically placed relay node. The key question then is how to place such relay nodes inside an AS, which is the focus of this paper.

We propose two heuristic algorithms to find the positions of relay nodes such that every OD pair has an overlay path, going through a relay node, that is disjoint from the default path. When it is not possible to find completely disjoint overlay paths, we allow overlay paths to have overlapped links with default paths. Since overlapped links diminish the robustness of overlay paths against a single point of failure, we introduce the notion of penalty for partially disjoint paths.

We apply our algorithms on three different types of topology data – real, inferred, and synthetic – and show that our algorithms find relay nodes of close-to-minimum penalty. Using daily topology snapshots and network event log, we also show that our choices for relay nodes are relatively insensitive to network dynamics; which is very important for a placement algorithm to be viable and practical.

1. Introduction and Motivation

Link and router failures are frequent in the Internet [20], [24]. Routing protocols are used to detect such failures and route around them. However, the convergence time for routing protocols to route around failures is often in the order of seconds or minutes [11], [15]. The downside of such long convergence time is that certain end-to-end connections may experience seconds or minutes of outage [7]. To increase reliability and robustness of mission-critical services in the face of temporary end-to-end path outages, it is often desirable and beneficial to take advantage of *path diversity* provided by the network topology.

One way of exploiting path diversity is to use a node inside the network to relay packets over an alternate path that is different from the default routing path; we term this alternate path as an *overlay path*. Previous work on overlay routing has focused on selecting good relay nodes based on measured metrics or QoS (Quality of Service) constraints, assuming relay nodes are already deployed (*e.g.* RON [5], Detour [27], or OverQoS [33]). However, none of these works tackle the problem of *placing the relay nodes well*, which is the focus of this paper.

To benefit from an overlay network of relay nodes, we envision applications using both the default and overlay paths all the time thereby doubling the consumed network bandwidth. We believe that this redundant bandwidth usage is justified when users of these applications are willing to pay or the total bandwidth consumed is not significant, as in the case of VoIP applications.

Routing in the Internet forms a two-level hierarchy: inter-domain and intra-domain. BGP (Border Gateway Protocol) [31] is the *de facto* standard inter-domain routing protocol. BGP route selection process is governed by policies set forth by network administrators of individual domains or ASes (Autonomous Systems). On the

[†] Meeyoung Cha and Sue Moon are supported by Korea Science and Engineering Foundation (KOSEF) through Advanced Information Technology Research Center (AITrc). [‡] Chong-Dae Park is supported by Brain Korea 21 (BK21) project through the school of information technology in KAIST. [*] Aman Shaikh is with AT&T Labs – Research, at 180 Park Avenue, Florham Park, NJ 07932, USA.

other hand, several routing protocols are used for intra-domain routing; OSPF [23], IS-IS [8], and EIGRP [4] being the popular ones. These protocols assign weights to links and employ shortest path routing in terms of the link weights. The complete end-to-end path is a concatenation of several shortest paths within ASes and inter-AS links chosen by individual ASes' BGP policies, and is not determined by a single AS or policy. Thus path diversity for end-to-end connections that span multiple ASes should be addressed in both intra- and inter-domains. For this work, we focus on the relay placement problem within a single domain or an AS by exploiting the path diversity available within such networks, and leave the problem of relay node placement in the inter-domain context as future work. To the best of our knowledge, this is the first paper to address the relay placement problem in the intra-domain context.

Within an AS, the overlay path consists of two shortest paths: one from the source to the relay node and the other from the relay node to the destination. We assume that every node is a relay candidate, where relay nodes are simply routers with a relaying capability. Our aim is to find positions of relay nodes such that every OD (Origin-Destination) pair inside a domain has an overlay path that is completely disjoint from the default path. Unfortunately in reality, it is often not possible to find completely disjoint paths for all OD pairs. As a result, we allow overlap between the default and overlay paths while keeping the overlap as low as possible. In this work, we report that a large portion of OD pairs fail to have completely disjoint paths due to topological structures or link weights. For some realistic topologies, failures are over 75%. However, it is still beneficial to have "partially" disjoint paths with minimum overlap. To quantify the extent of the overlap and resulting quality degradation of overlay paths, we introduce the notion of penalty, and develop heuristic algorithms to find relay nodes that incur close to minimum penalty.

We evaluate our algorithms on three different types of topology data – real, inferred, and synthetic. We show that with a small number of relay nodes (typically fewer than 10% of the total number of nodes), network resilience increases significantly against single link failures. We also use daily topology snapshots and network event log from a tier-1 ISP to evaluate the efficacy of the algorithms against network dynamics. Specifically, we show that the relays selected by our algorithms not only provide complete protection against 75.3% of failure events and over 99% protection against 92.8% of failure events, but they also remain effective over several months under dynamic network conditions.

The rest of the paper is organized as follows. We describe related work in Section 2. In Section 3, we formulate the relay node placement problem and present a definition of penalty with some practical considerations. We also propose our heuristic placement algorithms in this section. The evaluation of placement algorithms follows in Section 4. In Section 5, using daily snapshots and event log, we show how our relay nodes perform in the face of network dynamics. We discuss issues for further work in Section 6, and conclude in Section 7.

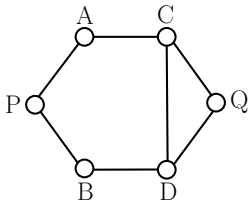
2. Related Works

Exploiting path diversity for fault tolerance and load balancing was first introduced by Maxemchuck as *dispersity routing* [21]. Since then, quite a few papers focusing on the selection of good overlay paths based on measured metrics or QoS constraints have appeared in the literature [5], [27], [33]. Recent works have proposed ways of using overlay networks in the security context. Lee *et al.* have proposed a distributed way of constructing an overlay network against link attacks [16]. Li *et al.* have proposed using overlay paths for resilient delivery of security updates [17]. All these proposals, however, assume that the relay (or overlay) nodes have already been deployed.

In terms of real-life deployment, many overlay networks have been constructed, often in an ad-hoc fashion. MBone, the overlay network for multicast communication, comprises of multicast-capable border routers at ASes and a set of intermediary nodes [9]. PlanetLab is a network of over 500 Linux PCs all around the world that serves a large number of research projects involved in testing, deploying, and debugging new services [3], [25]. PlanetLab nodes are hosted by volunteers; and no topological constraint has been imposed on how they are placed. Resilient Overlay Network (RON) is an application-layer overlay on top of the existing Internet routing substrate [5]. RON has about 50 machines that are located world-wide, but the majority (80%) are in the United States.

Server placement problems hold some similarities with our work in that they also focus on finding an optimal solution for resource locations [13], [26]. Often these problems are formulated as a k-median or

k-center problem [10]. In k-median problem, the objective is to find k medians among all possible positions to minimize the sum of distances from each vertex to its nearest median. In k-center problem, the objective is to minimize the maximum of distances to its nearest center. The objective of our problem is to minimize the sum of overlaps between the default and overlay paths. The formulation of our problem is unique in that: (1) our work focuses on providing disjoint paths and assign a relay node to each OD pair (while k-median and k-center assign a median or a center to each node); (2) the objective term to minimize is the overlap between two paths (while in other problems, it is distance or delay); and (3) our problem lies in a non-metric space. A cost function in a metric space must to be positive and symmetric, and to satisfy the triangular inequality. However, our problem is in non-metric space. A distance function in metric space needs to be positive, symmetric, and satisfy the triangular inequality. Figure 1 illustrates an example of non-metric space. Let OD pairs in the figure be (A, B) and (C, D), and the relays be P and Q. Then, we see that triangular inequality does not hold: the cost between (A, B) and P is greater than the sum of other costs.



$$\mathcal{K}_{AB}(P) = 2$$
$$\mathcal{K}_{AB}(Q) = 0$$
$$\mathcal{K}_{CD}(P) = 0$$
$$\mathcal{K}_{CD}(Q) = 0$$

$$\mathcal{K}_{AB}(P) > \mathcal{K}_{AB}(Q) + \mathcal{K}_{CD}(P) + \mathcal{K}_{CD}(Q)$$

Figure 1. Relay node placement problem is in non-metric space (triangular inequality does not hold).

3. Relay Node Placement Problem

We model a network as a graph $G(V, E)$, where V is a set of nodes and E is a set of directed links between pairs of nodes. A path is a finite non-null sequence of nodes and links between a pair of nodes. We term the start node of a path as an origin, the end node as a destination, and the node pair as an OD pair. Every link in the network is assigned a weight, and the cost of a path is measured as the sum of the weights of all links along the path. As we limit our study to intra-domain routing, we assume that Shortest Path First (SPF) routing based on link weights is used. If two paths do not have any common link between them, we call them *disjoint*.

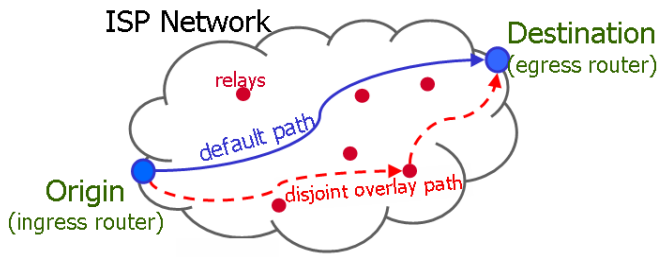


Figure 2. Traffic for an OD pair is routed along two paths: default path (determined by the intra-domain routing protocol running within the AS) and overlay path (that passes through a strategically placed relay node).

Figure 2 depicts the idea of using disjoint overlay paths. If packets from the origin are duplicated and sent along disjoint overlay paths, any disruption on either path causes no impact on the other path. However, if multiple links fail simultaneously, both paths may be affected. In real networks, the chance of network components located physically apart to fail at the exactly same moment is extremely slim. As the goal of this work is to improve network resilience in the face of transient routing instability (that is, during the period of routing convergence), we only consider single link or router failure events throughout this paper.

We now define the relay node placement problem as follows. Given a network G and the number of relay nodes k (constrained by available network resources), we want to find k positions of relays in the network such

that every OD pair finds an overlay path that is maximally disjoint from the default path. We use the concept of penalty to quantify the overlap in paths. Our approach is for static analysis of network path diversity based on the topology of a network, and we simply assume that equal amount of traffic flows between origin and destination of every OD pair. However, real traffic matrix of a network is highly dynamic, and we discuss how we can relax the assumption of homogeneous traffic matrix in Section 6.

Before we introduce our method to find relay nodes for disjoint overlay paths, we introduce how path diversity is characterized in typical ISP networks. Then, we illustrate the key concept of our idea along with two relay placement algorithms.

3.1. Path Diversity in Intra-Domain Routing and Its Impact on Relay Selection

Studies show that path diversity is available in IP layer topologies of typical ISP networks [12], [34]. Figure 3 shows an example of a large AS, consisting of a collection of physical locations called Point-of-Presences, or PoPs. Within a PoP, an access router (denoted as AR) is connected to two or more backbone routers (denoted as BR) with equal link weights for fault tolerance and load balancing [12]. Typically, parallel links between a pair of two PoPs are assigned the same weight. As a result of such link weight assignment, multiple shortest paths exist between the access routers in two PoPs, and they are called *Equal Cost Multi-Paths (ECMP)*.

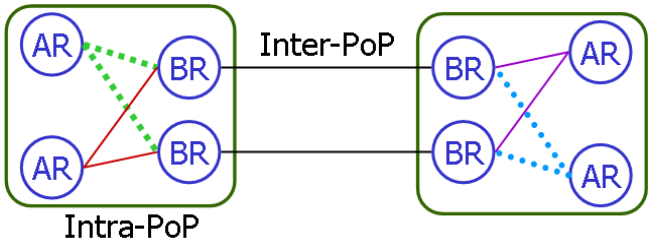


Figure 3. Path diversity is available in typical ISP networks. It is often not possible to find completely disjoint overlay paths for all node pairs.

When there exist ECMP between an OD pair, traffic is split equally among the multiple shortest paths, but each individual flow (a group of packets with the same 5-tuple: source IP address, source port, destination IP addresses, destination port, and protocol) is routed along only one path[1].

ECMP play a positive role against link failures. Sridharan *et al.* reports that ECMP are helpful in avoiding transient loops against link failures [30]. However, end-to-end connections are still susceptible to outages from link failures, and disjoint overlay paths should provide increased level of protection against detrimental impact of routing changes.

Since a node has finite degree, ECMP may exhaust all links out of a source for an OD pair and leave no link for a disjoint overlay path. In this work, we report that a significant portion of OD pairs fail to have completely disjoint paths due to topological structures or link weights; for some realistic topologies, as many as 75% of OD pairs failed to have completely disjoint paths. In this case, we are forced to have overlapped links between the default and overlay paths. Overlapped links will diminish robustness since a network is less resilient to link or router failures. Toward that end, we introduce a notion of penalty to quantify the quality degradation of overlay paths when they overlap with the default paths.

3.2. Measure of Penalty

First, we consider a way to quantify the impact of a particular link failure on a path. We use notation $o \rightarrow d$ to denote a collection of shortest paths from node o to d. When there is only one path between o and d, we treat $o \rightarrow d$ as a single path. When there are multiple shortest paths between o and d, we assume traffic is evenly split among those paths and treat $o \rightarrow d$ as a collection of paths. We define an indicator variable,

[1]Often, hash functions are used to equally split traffic amongst ECMP and forward packets of a flow along the same path.

$\mathcal{I}_{o,d,l}$, as the probability that a packet routed from o to d encounters the failed link, l. That is, $\mathcal{I}_{o,d,l}$ is the conditional probability that path $o \to d$ fails given that link l fails.

$$\mathcal{I}_{o,d,l} = P[o \to d \text{ fails} \mid l \text{ fails}] \tag{1}$$

The indicator variable quantifies the impact of a particular link failure on a given path. When $\mathcal{I}_{o,d,l} = 1$, a packet from o to d always goes through link l. Therefore, $o \to d$ will certainly fail, if link l fails. Otherwise, if l is not used on any path of $o \to d$, $\mathcal{I}_{o,d,l} = 0$. In this case, failure of l is irrelevant to $o \to d$. When $\mathcal{I}_{o,d,l}$ is between 0 and 1 (say p), it means that some paths in $o \to d$ include l and others do not. If a packet is routed through a path that includes the failed link, it will be lost. Therefore, $o \to d$ will fail with probability p if l fails. This happens when ECMP exist. We say $o \to d$ is *affected* by a link failure of l, if $\mathcal{I}_{o,d,l} > 0$. Figure 4 shows an example of how traffic is evenly split among multiple shortest paths and how $\mathcal{I}_{o,d,l}$ value is computed for every link.

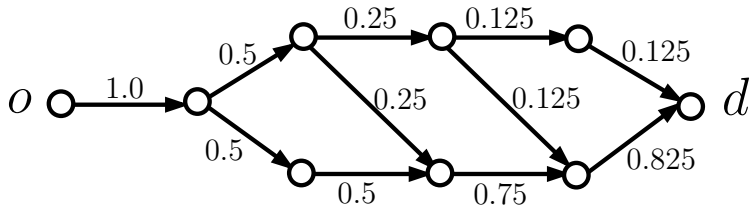


Figure 4. From o to d, traffic is evenly split among the shortest paths. For each link l, $I_{o,d,l}$ value is given.

We calculate $\mathcal{I}_{o,d,l}$ by extending Dijkstra's shortest path algorithm as follows. Given a network and a source node, we store all the shortest paths to each destination instead of storing a single path. Let $G^*[o,d]$ be a subgraph of G induced by all the shortest paths from source o to destination d. For convenience, we consider $G^*[o,d]$ as a directed acyclic graph (dag). Once dag $G^*[o,d]$ is found for an OD pair, we traverse $G^*[o,d]$ by inserting a unit amount of virtual flow at the root (*i.e.*, source node). We let the total amount of the incoming flow be equal to that of the outgoing flow (which is split evenly to all the outgoing links of the dag). Finally, the amount of flow assigned on each link equals $\mathcal{I}_{o,d,l}$.

Given an OD pair (o,d), we use $\mathcal{K}_{od}$ to denote the probability that a single link failure affects path $o \to d$, and calculate it as follows.

$$\begin{aligned} \mathcal{K}_{od} &= P[\ o \to d \text{ fails} \mid \text{single link failure}\ ] \\ &= \frac{1}{|E|} \sum_{l \in E} \mathcal{I}_{o,d,l} \end{aligned} \tag{2}$$

We use notation $o \to r \to d$ to denote the overlay path from node o to d via relay r, which is formed by concatenating the default shortest paths from the source to the relay (*i.e.*, $o \to r$) and the relay to the destination (*i.e.*, $r \to d$). Figure 5 shows an example of a default path (drawn in solid line) and an overlay path (drawn in dotted line).

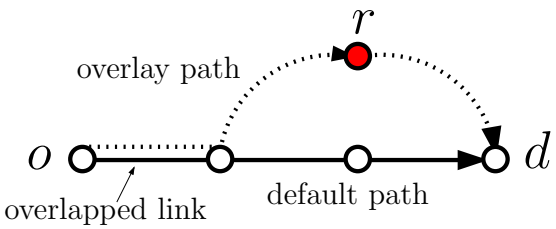


Figure 5. Solid and dotted lines denote the default and overlay paths, respectively. For resilience, we introduce the notion of penalty based on the overlapped links.

Now let's consider when an overlay path is used along with the default one. If link l is included in both of the paths (as the overlapped link in the figure), failure of link l affects both the default and overlay paths.

If l is used in only one of the paths, failure of l does not affect the other path. That is, either $o \rightarrow d$ or $o \rightarrow r \rightarrow d$ is irrelevant to the link failure on l; path between o and d is resilient to the failure of l. Therefore, we consider a fraction of traffic carried on overlapped links (between the default and overlay paths) as a measure of penalty for using partially disjoint paths.

Given an OD pair (o, d) and a relay r, we use $\mathcal{K}_{od}(r)$ to denote the probability that a single link failure affects $o \rightarrow d$ and $o \rightarrow r \rightarrow d$ simultaneously, and calculate it as follows.

$$\begin{aligned} \mathcal{K}_{od}(r) &= P[\text{ both } o \rightarrow d \text{ and } o \rightarrow r \rightarrow d \text{ fail} \\ &\qquad\qquad \mid \text{single link failure }] \\ &= \frac{1}{|E|} \sum_{l \in E} \mathcal{I}_{o,d,l}(\mathcal{I}_{o,r,l} + \mathcal{I}_{r,d,l}) \end{aligned} \tag{3}$$

We can see from the definition that the penalty of a relay node is zero when the relay provides a completely disjoint overlay path for an OD pair. It is interesting to note that the penalty value directly reflects the quality of an overlay path. If this value is fairly small, overlay paths have very few overlapped links with the default paths. Accordingly, network is more resilient to arbitrary single link failures. For network resilience against single link failures, we propose using overlay paths that are as disjoint as possible from the default paths. Or equivalently, we aim at finding a set of relay nodes that minimizes the penalty in (3) for all OD pairs.

Having defined the penalty for a given OD pair and a single relay node, let us now extend the definition to a set of relay nodes, R. Since our objective is to determine the positions of relay nodes such that every OD pair has a maximally disjoint overlay path, we should select a relay $r \in R$ that yields the least penalty value in (3). Accordingly, the penalty value of (o, d) under a relay set R is:

$$\mathcal{K}_{od}(R) = \min\{\mathcal{K}_{od}(r) | r \in R\}. \tag{4}$$

Finally, we define the *total penalty*, $\mathcal{P}(R)$ of using a relay set R for *all* OD pairs as follows:

$$\mathcal{P}(R) = \sum_{\forall o,d} \mathcal{K}_{od}(R), \quad R \subset V, |R| \leq k. \tag{5}$$

Given this, our objective is to find a subset R of V such that the $\mathcal{P}(R)$ is minimized, where $|R|$ is not greater than a given value, k.

3.3. Placement Strategies

In this subsection, we present our placement strategies to find a relay set R of a fixed size k such that the total penalty in (5) is minimized. Given a set size, k, an optimal solution is a subset R of V with the least total penalty. We denote the optimal solution as Optimal in the rest of the paper. The optimal solution can be formulated using 0-1 integer programming (IP). The IP formulation of our problem is given in the Appendix. While Optimal gives the best result, it is unlikely that an efficient method for solving it exists due to computational complexity. In our simulation, we compute Optimal for only limited cases when k is significantly small compared to n. In the following, we present two efficient heuristic algorithms: greedy selection and local search. These two heuristics are simple and intuitive while delivering good performance.

3.3.1. Greedy Selection Algorithm

In our greedy selection, we begin with an empty set R. Then we add a relay node r one by one, that incurs the maximum decrease in the total penalty given by (5). We iterate this process k times. We refer to this approach as Greedy.

3.3.2. Local Search Algorithm

We start with an arbitrary set of k relays and keep improving our solution with a single swap. A single swap involves removing a relay $r \in R$ and adding a new relay $r' \notin R$, if the total penalty is reduced. We repeat single swaps until there are no improvements. As its name suggests, the solution produced by this algorithm

is a local optimum that may or may not be far away from the global optimum. We refer to this approach as Local.

Refer to detailed algorithms of Greedy and Local in the Appendix. For comparison, we consider two other potential strategies: one that chooses a random set of relay nodes, referred to as Random, and the other that selects nodes in a decreasing order of node degree, referred to as Degree.

4. Evaluation of Placement Algorithms

In this section we perform detailed analysis of our Greedy and Local algorithms, along with the two other Random and Degree algorithms. In Section 4.1, we introduce topological datasets used in our evaluation. In Section 4.2, we compare the four placement algorithms in terms of total penalty they incur. In Section 4.3, we delve into the topological structures of networks and investigate their impact on certain relay nodes.

4.1. Types of Network Topologies

We use datasets drawn from three different types of topologies – real, inferred, and synthetic. We have access to only a small number of real topologies, which is a significant limiting factor in exploiting topological diversity. Thus, we supplement our evaluation with inferred and synthetic topologies. Table 1 summarizes the network topologies used in evaluation. Each topology is listed with its type, name, number of nodes and links, and minimum and maximum node degrees. The last column, marked as *Failed*, shows the percentage of OD pairs that fail to have completely disjoint paths. If a pair (o, d) fails to find a relay $r \in V$ such that the default and overlay paths have no overlap, then the node pair is said to *fail* to have completely disjoint overlay paths. This condition is checked as following.

$$\mathcal{K}_{od}(V) = \begin{cases} > 0 & \text{if } (o,d) \text{ fails to have completely disjoint paths} \\ 0 & \text{otherwise} \end{cases}$$

After analyzing each failure case, we note that a large portion of OD pairs fail to have completely disjoint paths due to topological structures or link weights. For some realistic topologies, failures up to 75%, and above 90% for some synthetic topologies. This finding motivates us to find "partially" disjoint paths with minimum overlap. In the following we will introduce each topology in detail.

Table 1. Summary of datasets

Topology Type	Topology Name	Nodes #	Links #	Degree (min,max)	Failed (%)
Real	Abilene	11	14	2, 3	36.36
	Backbone	$\approx$ 100	$\approx$ 200	$\approx$ 2, 10	53.89
Inferred	Exodus	79	147	1, 12	43.58
	Ebone	87	161	1, 11	67.83
	Tiscali	161	328	1, 29	77.67
Synthetic	BA50-2	50	81	2, 17	38.96
	BA100-2	100	197	2, 25	30.02
	HOT	171	440	1, 10	93.71
	mesh	64	112	2, 4	98.43
	torus	64	128	4, 4	60.96
	ring	64	64	2, 2	4.69

Note: the number of bi-directional links is given in *Links (#)* column. Since we consider unidirectional links in our algorithm, $|E|$ should be doubled. For example, Abilene has 28 unidirectional links.

Two real topologies we use are those of Abilene and an operational tier-1 ISP backbone. *Abilene* is a high-performance Internet2 backbone network for universities and research laboratories in the United States [1]. We use the Point-of-Presence (PoP) level map of Abilene. We assume a unit weight for each link (which essentially simulates a hop-count based routing). Figure 6 depicts the topology of Abilene, where three relays

chosen are denoted in triangles[2]. The *operational tier-1 ISP backbone*, simply referred to as the Backbone in the rest of the paper, has an order of magnitude more nodes and links than Abilene. Due to proprietary nature of the data, Table 1 only provides approximate values for the parameters of the topology. We also have real link weights from the Backbone. The Backbone has topological structure similar to the one illustrated in Figure 3, where border routers have two-dimensional square mesh connectivity.

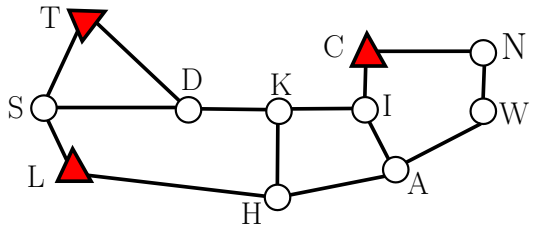
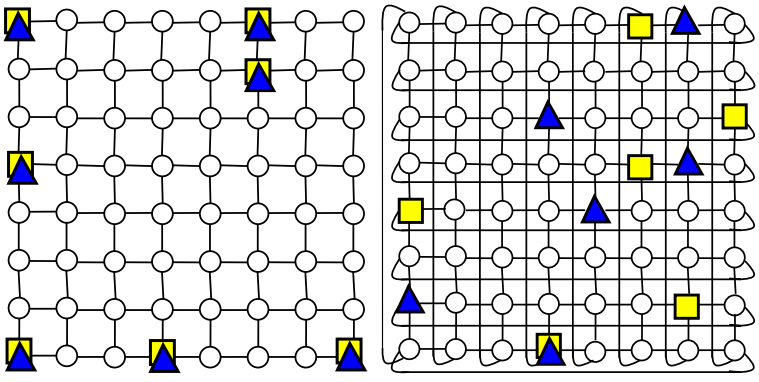


Figure 6. Abilene network

We use three of the inferred ISP topologies generated by *Rocketfuel*, a router-level ISP topology inference engine [29]. Here, link weights are assigned proportional to geographical distances (or delay). It is important to observe that Rocketfuel topologies are subject to inference errors. In particular, Teixeira *et al.* showed that Rocketfuel has significantly more path diversity than the real topology in case of the Sprint network [34]. We note that overestimated path diversity may give better result with our algorithms by increasing the chances for finding disjoint overlay paths. It should also be noted that the result in [34] only applies to Sprint's Rocketfuel topology, and may or may not hold for other topologies.

Synthetic topologies are generated using *BRITE*, an Internet topology generator [2]. We use the flat *Albert-Barabasi* model [6] which generates router-level topologies. Each of the BRITE-generated router-level topology has n nodes and minimum d number of edges, and is denoted BAn-d. Our settings in BRITE reflect incremental growth and preferential connectivity [22]. We use latency of each link as its weight, where latency is calculated proportional to distances between nodes. Node placement is based on a heavy tailed distribution. While the Albert-Barabasi model approximates scale-free networks, recent work by Li *et al.* suggests a model – where low degree nodes are in the center and high degree nodes are at the edge of the network – which is considered a better reflection of real networks [18]. We use their *Heuristically Optimal Topology (HOT)* model, which consists of 49 core routers and 122 gateway routers. We use unit link weights for all links.

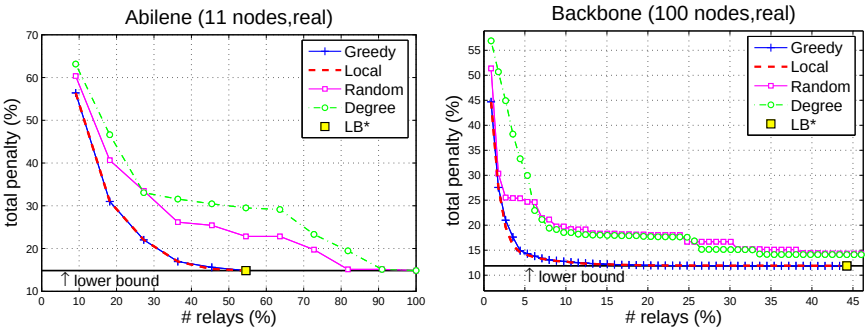


(a) mesh (b) torus

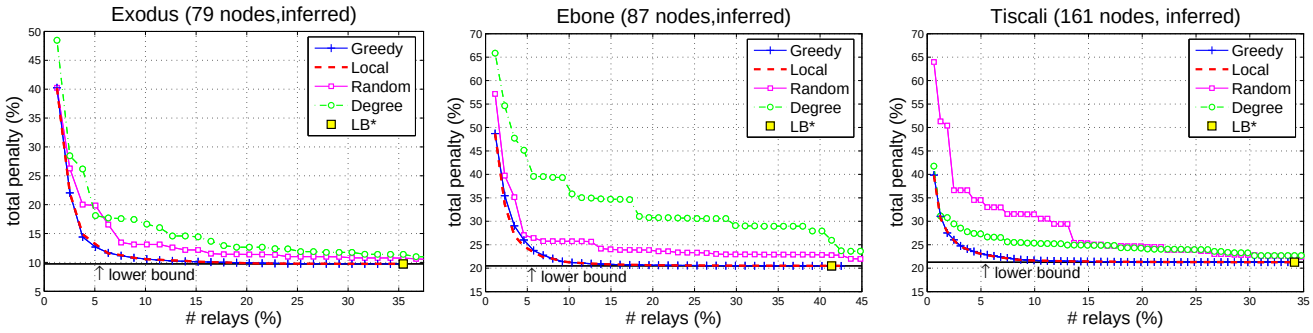
Figure 7. Synthetic networks

We consider three more networks with regular structures, a mesh, a torus, and a ring, each of which has 64 nodes. A *mesh* has its nodes placed in a 8 by 8 square grid, where nodes at the edge of the square grid have lower node degrees than the ones in the center. A *torus* has a similar topology to that of a mesh, but the edges wrap around the torus when they cross the square grid's boundary. Nodes in a *ring* network form a circular

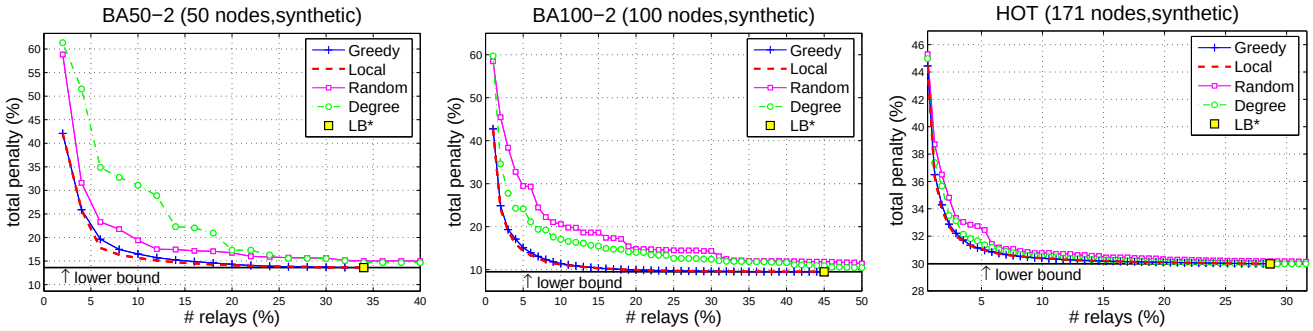
[2]When $k = 3$, the same set of relays is chosen for all three algorithms: Local, Greedy, and Optimal as in Figure 6.



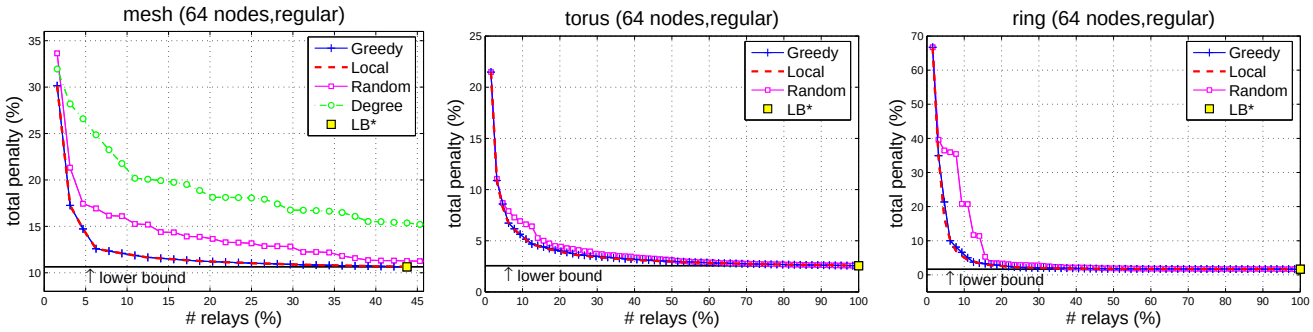
(a) Real topologies



(b) Rocketfuel inferred topologies



(c) BRITE generated topologies and HOT model



(d) Synthetic topologies

Figure 8. Performance comparison of placement heuristics on real, inferred, and synthetic topologies. Since nodes in torus and ring networks have the same number of node degrees, we do not include Degree plot for those networks.

shape. We use unit link weights for all links. Figure 7 depicts mesh and torus networks, where 6 nodes are selected as relays, denoted in triangles (Local) and rectangles (Optimal). While these regular graphs are not particularly realistic in their network configurations, they provide a more neutral context for evaluating our placement algorithms. By considering a variety of networks, we hope to avoid drawing conclusions that may be attributable purely to topological idiosyncracies of a particular network.

4.2. Comparison of Total Penalty

We now assess the performance of our heuristic algorithms. Figure 8 compares the four placement algorithms in terms of total penalty in (5) they incur. The number of the relay nodes has a range of 1 to k ($k \leq n$). Total penalty is normalized such that 100% represents when only default paths are used (*i.e.*, when the default and overlay paths are identical).

As all our algorithms are based on heuristics, it is hard to fathom how far they are from the best case. As it is hard to capture the minimum total penalty for all possible k values, we only compute the lower bound on total penalty when all n nodes are used as relays. We define a lower bound, LB, of $\mathcal{P}(R)$ in (5) as below:

$$\text{LB} = \mathcal{P}(V). \tag{6}$$

Conceptually, LB captures the notion of each OD pair designating a relay that incurs the least amount of penalty among all n nodes. Under the assumption of using a single relay per node pair, LB represents the least total penalty for any topology. A horizontal straight line at the bottom of each graph in Figure 8 indicates LB. While LB represents the least total penalty, finding the minimum number of relay nodes that achieve LB is another problem. We use a Set Cover approximation method [28] and obtain an estimate for the minimum number of relay nodes that produce LB. This estimate is denoted as LB* and marked with a square in the figure.

In all cases, Greedy and Local consistently perform better than Random and Degree. This is plausible since Random represents the case when no planning is used in relay node placement, while Degree places relay nodes at the top k nodes in terms of node degree. Degree's relays are likely to be heavily involved in the default paths of many OD pairs, increasing the number of overlapped links when used as relays.

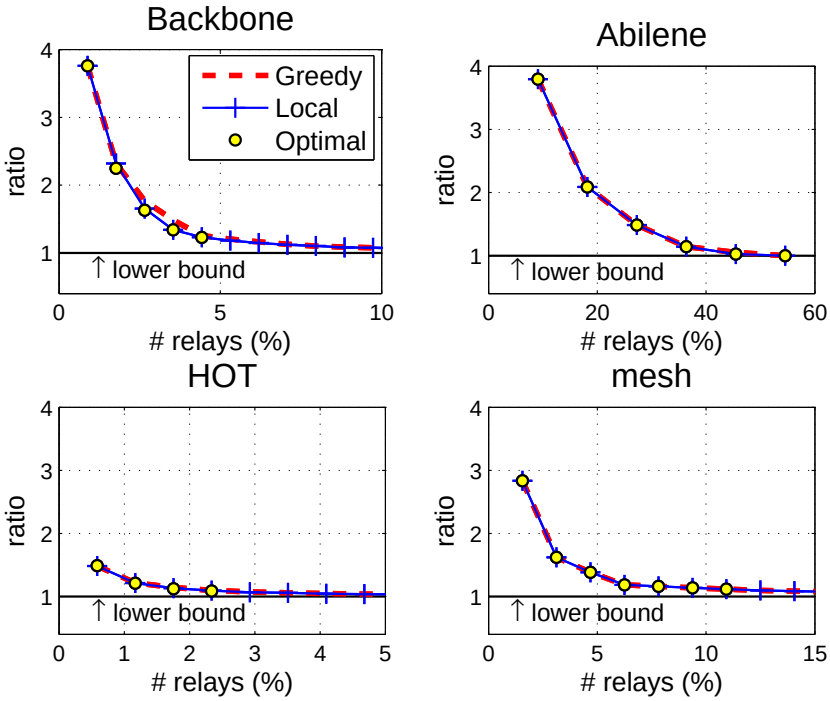


Figure 9. Ratio of total penalty against the lower bound

Intuitively, as we place more relays in the network, OD pairs are likely to find overlay paths with less total penalty. However, the unit gain in total penalty will saturate when an enough number of relays are placed in the network. This intuition is evident in Figure 8, where all curves flatten out after a while. The knee point of saturation is different for each heuristic; Local and Greedy tend to reach the knee points with smaller numbers of relays.

In all graphs in Figure 8, the gap between the total penalty of our heuristics and that of LB is substantial when less than 5% of nodes are chosen as relays, and it is hard to know how close the total penalty is to Optimal for the given number of relay nodes. Although we cannot compute Optimal for all possible values of k due to computational complexity, we can compute it when k is significantly small compared to n. For a subset of topologies, namely, Abilene, Backbone, HOT, and mesh, we calculate results from Optimal and compare with Local, Greedy, and Optimal. Figure 9 plots the total penalties from Optimal, Greedy, and Optimal as a ratio against that of LB. Ratio of 1 indicates that the total penalty is the same as LB (which is calculated assuming the number of relays is n). Even though Local and Greedy are not optimal, total penalties from these simple and fast heuristics are almost identical to those from Optimal. We conclude that our placement heuristics are simple and intuitive, while delivering near-optimal performance.

4.3. Relay Nodes

So far, we have evaluated the performance of our placement heuristics on a set of topologies. We now discuss how the topology structure affects the placement heuristics and analyze the properties of the relay nodes.

4.3.1. Impact of Topology Structure

We observe that in Figure 8, HOT model has a LB value of 30%, which is larger than most other networks, where LB is around 10%. We discuss what attributes to this high variance and how structures of the topology in general (and Internet-like topologies in particular) affect the selection of the relay nodes.

For each topology, we examine OD pairs for whom only very little or no improvement is achieved with overlay paths. Consider the Abilene topology in Figure 6 with three relay nodes. Assuming hop-count based routing is used, traffic between an OD pair (S, N) is evenly split between two paths: the upper path that goes via $DKIC$ and the lower path that goes via $LHAW$. In this case, all possible overlay paths result in total penalty of at least $2/28$. Similar case applies for (T, W). Another apparent example is between the node pair (C, I), where disjoint overlay path cannot be found using the given relays.

From our analysis, pathological cases where OD pairs fail to find good quality overlay paths are prevalent in typical ISP networks. This is due to the fact that a typical ISP network topology is not completely random, but has structural regularities. For example, the number of links connected to a node does not vary over a wide range, but is limited by the maximum number of slots and ports on a router. Also routers located at one PoP are connected in such a way that traffic out of the PoP is aggregated and sent out through a small number of routers, thus forming a certain hierarchy between routers as illustrated in Figure 3. Between an arbitrary pair of nodes (AR,BR) within a PoP, it is unlikely that there is a good quality overlay path. HOT model has the highest LB since all 122 gateway routers are singly linked to one of the 49 core routers. Figure 10 shows the

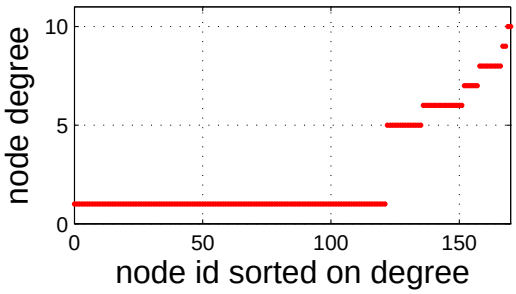


Figure 10. Node degree distribution of HOT network

node degree distribution of HOT network, where the overall distribution of degree is heavy-tailed. A torus, on the other hand, has many paths between the OD pairs, and therefore, overlay paths are often no worse than the default paths.

4.3.2. Relay Node Properties

To see which nodes are chosen as relays for each heuristic, we measure the following three metrics and observe whether relays are selected at the core or at the edge of the network.

- Node degree - the number of incident links of a node
- Hop count - average hop count to other nodes
- Path weight - average path weight (the sum of link weights along the path) to other nodes

Figure 11 plots the distributions of the above three metrics of the relay nodes selected by Greedy, Local, Random, and Degree (denoted as G, L, R, and D, respectively). For each metric and placement heuristic, we set the number of relay nodes to 5 and 10% of the nodes. The maximum, median, and minimum values of the relay node distribution are displayed using error bars. For each metric, the maximum, median, and minimum for the entire topology are drawn in dotted lines.

In terms of node degree, relay nodes by Local and Greedy are selected near the median distribution of the overall nodes for Ebone and Tiscali networks. As expected, Degree'srelay nodes are those with high node degrees. The result of Random varies for each trial, but the median degree of the relay nodes by Random stays close to that of the overall nodes. In case of HOT model, relay nodes by Local and Greedy show unusual distribution in their node degree as well as in hop count and path weight. This is due to the heavy-tailed node degree distribution of the HOT model.

When 5% of nodes are chosen as relays, Degree yields significantly smaller hop counts and path weights than Local and Greedy for Ebone network. This implies routing in Ebone network is done in a way that OD pairs prefer paths (by IGP costs) that go via nodes with high degree compared to nodes with low degree. Therefore, high degree nodes are accessible with smaller hop counts by arbitrary nodes. When 10% of nodes are chosen as relays, the gap between the distribution of each heuristic becomes less noticeable. In case of Tiscali network, similar observations hold true, however in less noticeable form. This is because Tiscali network has more high degree nodes than Ebone network.

In summary, Local and Greedy tend to avoid nodes that are the highest in node degree at the cost of taking a "detour." Even though high degree nodes seem like a good choice in terms of hop count and path weight metrics, we note that those nodes are likely to be heavily involved in default paths of many OD pairs, making them inappropriate for use as relays.

In large networks, detour through disjoint overlay paths could cause some OD pairs to traverse longer paths, thereby increasing the delay between those OD pairs. In practice, operational backbone networks provision their networks such that the average load on each link and the average end-to-end propagation delay are below a certain limit agreed upon in Service Level Agreements (SLA). Our algorithms can be easily modified to meet both these requirements of networks if we are given with the traffic matrices (for both the default and overlay traffic), latency of each link, and the detailed service requirements. (More discussions follow in Section 6.)

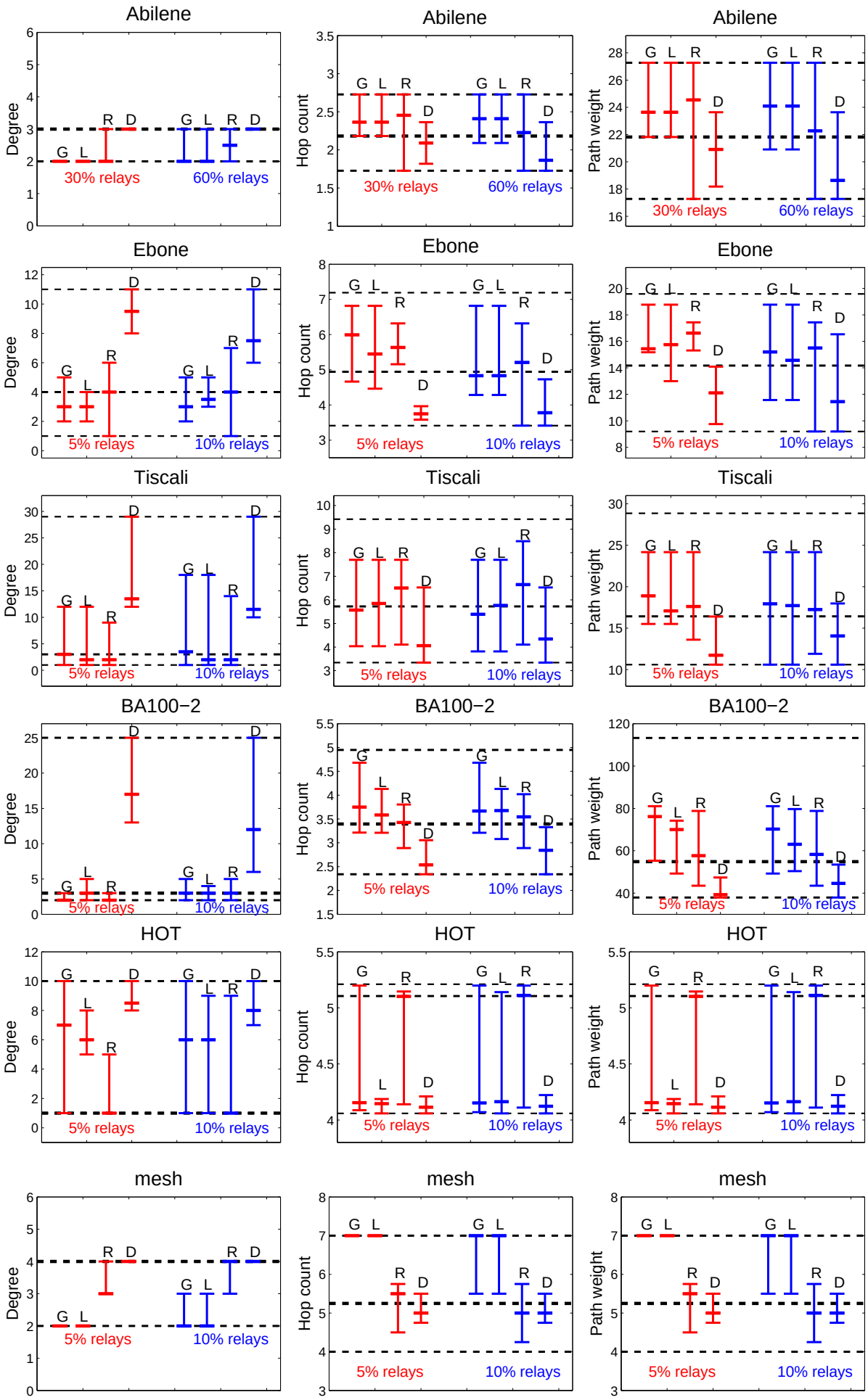


Figure 11. Distribution of the relay nodes by Greedy (G), Local (L), Random (R), and Degree (D) heuristics

5. Simulation under Dynamic Network Conditions

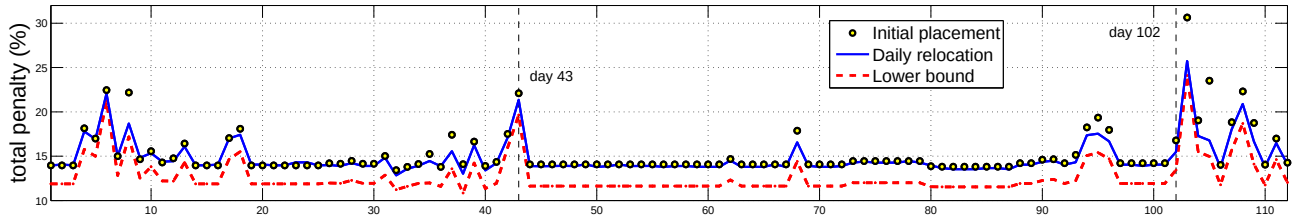
Our relay node placement algorithms are based on the assumption that network topologies remain fixed and we have complete knowledge of the underlying topologies. In practice, network topologies are dynamic in that there are frequent link and router failures, whether they are caused by manual operations or unplanned events. In this section, we examine how our heuristic algorithms perform under dynamic network conditions. In particular, we conduct two sets of simulations: (1) we use a set of network topology snapshots of three months, taken at the same hour each day, and examine whether the relay nodes selected at the beginning of the period are still effective over time; (2) we use a network event log of six months and calculate the fraction of traffic that is affected by network events with and without relay nodes. We use the topology snapshots and event log of the Backbone. In the following simulation, again, we assume equal amount of traffic flow between all node pairs. We state that this traffic matrix is hypothetical and does not reflect the real traffic volume of the Backbone.

5.1. Using Daily Topology Snapshots

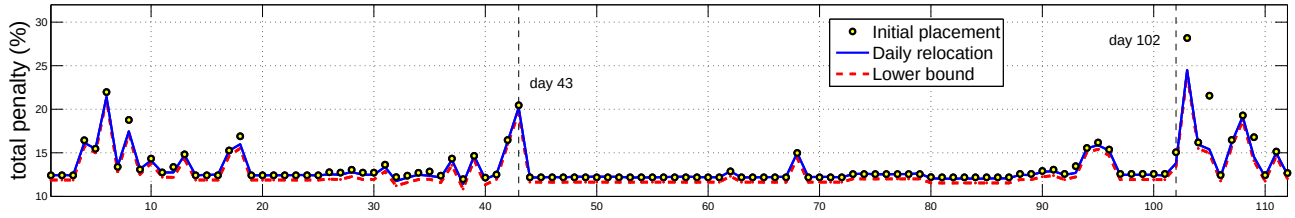
We use daily topology snapshots of 113 days taken during October 1, 2004 to January 22, 2005. Each snapshot has its own set of IGP link weights. The overall numbers of nodes and links vary with a standard deviation of 1.41 and 3.3, respectively. We examine how much path diversity the relay nodes fixed on Day 1 (Oct. 1, 2004) can provide over the time period, compared to the case of relay nodes changing every day based on daily topology snapshots. For simulation, we use Greedy and choose 5 and 10 percents of the nodes as relays.

In Figure 12, we display time-series evolution of total penalty in (5) from three sets of simulations over 113 days. In the first simulation, the set of relay nodes is determined on Day 1 and remains fixed throughout the entire time period (referred to as *initial placement*). In the second simulation, the set of relay nodes is refreshed (optimized) daily based on the corresponding topology snapshot (referred to as *daily relocation*). In the third simulation, we calculate LB in (6) per daily snapshot. Again, the total penalty here is normalized such that 100% represents the case when only default paths are used in each snapshot.

Figure 12 shows that the relay nodes from the initial placement scheme perform nearly as well as the daily relocation scheme. When the number of relay nodes increases from 5% to 10%, then both schemes almost match the lower bound, LB. We note that our placement algorithms are not very sensitive to network dynamics. As long as the network topology does not change in a grand scale (*e.g.*, partition of network), our relay nodes selected based on a topology snapshot accommodate dynamic network conditions (*e.g.*, several router/link failures or link weight changes) well.



(a) Initial placement vs daily relocation using 5% of nodes as relays



(b) Initial placement vs daily relocation using 10% of nodes as relays

Figure 12. Comparison of total penalty by different relay placement strategies

The normalized total penalty in the figure mostly remains under 15%, but fluctuates over 15% a few times, noticeably on Days 6, 43, 103, and 108. This is mainly due to the fact that snapshots of the Backbone were taken during the maintenance window, and thus the network might have not converged yet at the time the snapshot was taken. In fact, we have verified that, in topology snapshots where total penalty surged, several link weights were set high, making those links unavailable in the actual routing and decreasing the chance for OD pairs to find a good quality overlay path; thus the total penalty may increase.

However, the fluctuation might be from other sources as well. In case of Day 43, we notice a sudden increase in the hypothetical traffic volume carried on one of major links (*e.g.*, links that are heavily used in default paths of many OD pairs). Since a large portion of OD pairs share the major link, it is also likely that overlay paths themselves go through the link. This overlap in the default and overlay paths may have increased the total penalty. For the fluctuations from Days 102 to 113, we have confirmed from the ISP that major upgrades have been performed on the Backbone (*e.g.*, new high bandwidth links, router operating system updates, etc).

Again, using daily snapshots of the Backbone, we note that our choices for relay nodes are relatively insensitive to network dynamics; which is very important for a placement algorithm to be viable and practical.

5.2. Using Network Event Log

We use a network event log that spans a six-month period from June 1 to November 30, 2004. The log contains five types of events: router up, router down, link up, link down, and link weight change. When a router comes up or goes down, all links incident on the router also come up or down. Sudden link or router down events usually cause temporary traffic loss for a number of OD pairs, resulting in service disruption. On the other hand, for router/link up and link weight change events, shortest paths are recomputed and OD pairs may experience a route change (or a traffic shift) in their default paths. However, such a change has less detrimental impact compared to router/link failures [7]. Therefore, we only focus on router/link down events in our simulation. It should be noted that our algorithm is applicable and effective against routing instability caused by router/link up and link weight change events as well.

We assume that each node re-calculates its routes immediately and instantaneously after each event. We realize this assumption by updating the topology and recomputing the shortest paths after each event. Relay nodes, used in the analysis, are chosen based on the topology snapshot at the beginning of the event log (*i.e.*, June 1st, 2004), and are kept unchanged even though the topology changes as events unfold. We use Greedy and choose three and five relay nodes for the simulation. For each event (single/multiple link and/or router failures), we calculate the fraction of hypothetical traffic affected due to the failure with and without relay nodes. As defined earlier, a single link failure on l affects OD pair (o, d) by $I_{o,d,l}$, which is the fraction of traffic assigned to that particular link. In this way, we determine the fraction of traffic affected due to the failure for every OD pair.

Figure 13 plots the simulation result, where the x-axis is the percentage of the affected traffic and the y-axis is the CDF of network events. The plot has three graphs. The first one (drawn in a solid line) shows traffic affected when only default paths are used. The second graph (drawn in a dash-dotted line) uses both default and overlay paths with three relay nodes, and the last (drawn in dashed line), with five relay nodes.

When only default paths are used, 35.9% of failure events have no impact on traffic. Though lower than 50%, its impact turns out to be less than we have expected. Detailed analysis of these events show that link weights were manually set high before the corresponding link failure event. Setting a link weight to a larger value forces traffic to bypass the link, allowing a “graceful” link shutdown. The remaining events impact only a small fraction of traffic in the network; for 65.5% of failure events, less than 1% of traffic is affected.

When three relay nodes are used, they provide complete resilience against 52.9% of failure events, which is a 17% increase, compared to no relay node case. Better still, up to 77% of failure events affect 1% or less of traffic. When five relay nodes are used, network resilience to real failures increases further. In this case, using overlay paths provide complete protection against 75.3% of failure events and over 99% protection against 92.8% of failure events. It is also worth noting that a small number of relay nodes chosen at the beginning of the period remains effective in providing resilience against failures over the entire course of six months.

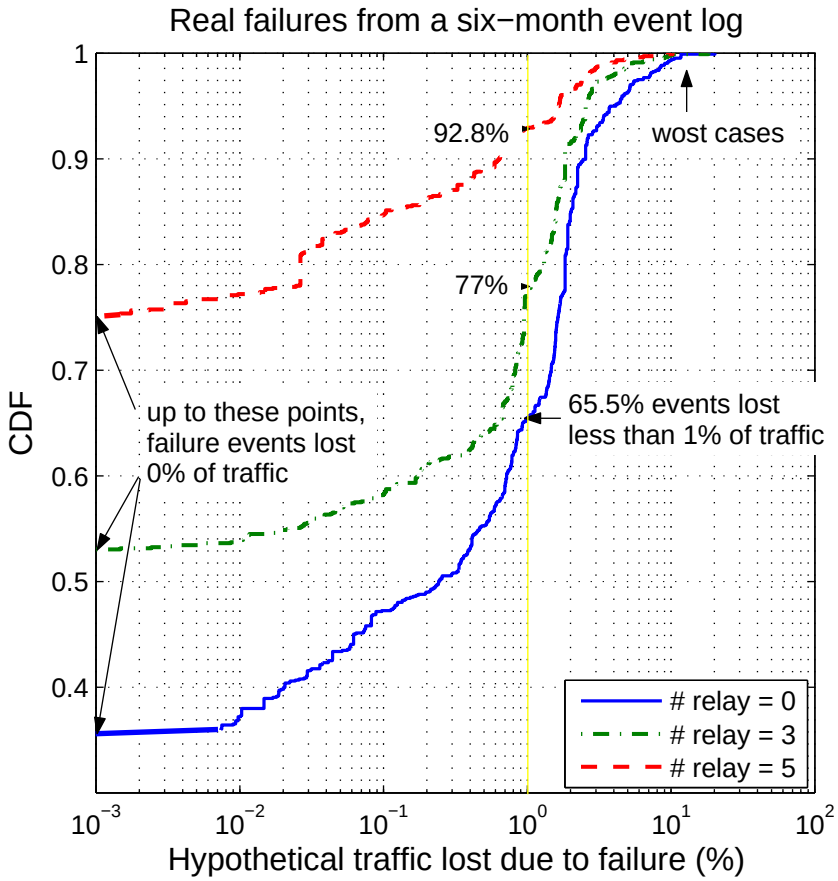


Figure 13. Impact of failure events on hypothetical traffic with and without relay nodes for a tier-1 ISP network

6. Discussion and Future Work

In this section we look at a number of ways in which our work can be extended. Addressing them is beyond the scope of this paper, so we leave them as part of our future work.

• *Relay Architecture for Service Overlay Network:* We have envisioned relay nodes forming an infrastructure, *i.e.*, a service overlay network, for a value-added service. As stated in the introduction, we have assumed that relays are simply routers with relaying capability. If routers allow the IP option of loose-source routing and end hosts use them, the service overlay can be deployed without any modification to the existing routers. Unfortunately, most service providers disable loose-source routing due to the security threat it poses and the processing load on the router CPU. An alternative is to realize the relay nodes by attaching servers to routers as proposed in [14].

We expect certain routers may not be suitable as relays (because of their locations, limited numbers of interfaces, or constrained CPU). We also note that average end-to-end delay of overlay paths should be below a limit agreed upon in SLA. We realize these extra requirements by defining a set of routers usable as relays for a given OD pair as *good*, and the others as *bad*. For example, a router with certain hardware specifications or higher may be set as good for all OD pairs, whereas a router that provides an overlay path with delay twice or more than that of the default path may be set as bad for the given OD pair. Then, we incorporate this information by redefining the penalty of a relay r for OD pair (o, d) as $\mathcal{K}'_{od}(r)$.

$$\mathcal{K}'_{od}(r) = \begin{cases} \mathcal{K}_{od} & \text{if } r \text{ is bad for } o \text{ and } d \\ \mathcal{K}_{od}(r) & \text{if } r \text{ is good for } o \text{ and } d \end{cases}$$

By replacing (3) with $\mathcal{K}'_{od}(r)$, we enforce that bad relays are not selected by the OD pairs in (4) and (5).

• *Reflecting Real Traffic Matrix:* In this work, we have assumed that equal amount of traffic flows between all OD pairs. However, in real ISP networks, our assumption on homogeneous traffic matrix does not hold.

We can easily modify our penalty measure to reflect the real traffic matrix as follows. Let $\mathcal{M}(i,j)$ denote the amount of relative traffic volume such that $\sum_{\forall i,j \in V} \mathcal{M}(i,j) = 1$. Then, by simply multiplying the penalty measures in (2) and (3) with $\mathcal{M}(o,d)$ yields the amount of traffic that is affected by a single link failure. Given an OD pair (o,d) and a relay r, let $\mathcal{K}^*_{od}$ and $\mathcal{K}^*_{od}(r)$ denote the amount of default and overlay traffic affected by a single link failure, respectively. Then, we calculate them as follows.

$$\mathcal{K}^*_{od} = \mathcal{M}(o,d) \cdot \mathcal{K}_{od} \quad (7)$$

$$\mathcal{K}^*_{od}(r) = \mathcal{M}(o,d) \cdot \mathcal{K}_{od}(r) \quad (8)$$

By replacing (3) with $\mathcal{K}^*_{od}$, (5), now (5) denotes the amount of traffic affected by a single link failure for all OD pairs when we are given a relay set. Greedy and Local heuristics are still useful in relay placement considering the real traffic matrix.

• *Relay Placement in Inter-Domain Setting:* A natural extension of our work is relay placement algorithm that provide disjoint overlay paths for traffic that span multiple ASes. Path diversity in inter-domain routing is more complicated, due to scale (*e.g.*, consider all combinations of source and destination) and the policy-governed route selection of BGP. Since most ASes do not publish their routing policies (such as local preferences in BGP), we may need to infer inter-domain routing paths from the publicly available BGP feeds as in [19]. Potential challenges include: (1) AS-level path inference (since BGP is policy-based); (2) asymmetries of AS paths [24] (*i.e.*, forward and backward paths may require different relays); and (3) realistic traffic matrix that span multiple ASes. Noting that BGP's best path selection is based on a destination prefix instead of a destination AS, relay nodes should be selected per prefix, rather than per AS. Finding a small set of relay nodes that minimizes the number of overlapped links under different and partly unknown routing regimes is far more challenging. We leave the overlay design issues in inter-domain setting for future work.

• *Physical Layer Path Diversity:* Our work considers layer-3 path diversity, which is distinct from the physical layer path diversity. At the physical layer, disjoint IP layer paths may run over the same optical fiber. For more robustness against link failures, cross-layer check for disjoint paths should be added [32]. Large ISP networks have access to their physical-layer topology map, and thus intra-domain path diversity may be strengthened greatly by considering these maps.

7. Conclusions

In this work, we identify the problem of relay node placement in an intra-domain setting for path diversity. An end-to-end connection may use more than one path to guard against temporary outages from frequent network changes, provided that those paths are completely disjoint. In reality, unfortunately, it is often not possible to find completely disjoint paths for all node pairs. We formalize the notion of penalty to quantify the quality degradation when partial overlap between the default and overlay paths is allowed, and present two efficient heuristic algorithms that choose relay nodes with the penalty close to minimum. Using three different types of network topologies, network snapshots, and network event log, we show that a very small number of relay nodes (typically fewer than 10% of the total number of nodes), are sufficient to provide much heightened level of protection against everyday network changes.

There are a number of applications that can exploit path diversity for improved QoS within an AS (*e.g.*, on-line game traffic, VoIP, resilient security updates, backup line for banking system's private network). In such scenarios, placing relays hold critical issue, which should benefit the most from our work. We also believe that there is more room for further research in this area.

References

[1] *Abilene Network*, Advanced Networking for Leading-edge Research and Education. `http://abilene.internet2.edu/`.

[2] *BRITE*, Boston university Representative Internet Topology gEnerator. `http://www.cs.bu.edu/brite/`.

[3] *PlanetLab*. `http://www.planet-lab.org`.

[4] R. Albrightson, J. J. Garcia-Luna-Aceves, and J. Boyle. EIGRP-a fast routing protocol based on distance vectors. In *Proceedings of Networld/Interop'94*, May 1994.

[5] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, October 2001.

[6] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[7] C. Boutremans, G. Iannaccone, and C. Diot. Impact of link failures on VoIP performance. In *Proceedings of Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, May 2002.

[8] R. W. Callon. RFC 1195: Use of OSI IS-IS for routing in TCP/IP and dual environments, December 1990.

[9] S. Casner and S. Deering. First IETF Internet audiocast. *ACM Computer Communication Review*, 22(3):92–97, 1992.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, 2nd edition*. MIT Press/McGraw Hill, 2001.

[11] T. G. Griffin and B. J. Premore. An experimental analysis of BGP convergence time. In *Proceedings of the Ninth International Conference on Network Protocols (ICNP'01)*, November 2001.

[12] G. Iannaccone, C. N. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier-1 backbone. *IEEE Network Magazine*, 18(2):13–19, 2004.

[13] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. On the placement of Internet instrumentation. In *Proceedings of IEEE INFOCOM*, March 2000.

[14] C. Kommareddy, T. G'uven, B. Bhattacharjee, R. La, and M. Shayman. Intradomain overlays: architecture and applications. Technical Report, UMIACS-TR 2003-70, 2003.

[15] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. *IEEE/ACM Transactions on Networking*, 9(3):293–306, 2001.

[16] P. P. Lee, V. Misra, and D. Rubenstein. Distributed algorithms for secure multipath routing. In *Proceedings of IEEE INFOCOM*, March 2005.

[17] J. Li, P. Reiher, and G. Popek. Resilient self-organizing overlay networks for security update delivery. *IEEE/JSAC, Special Issue on Service Overlay Networks*, 22(1):189–202, 2004.

[18] L. Li, D. Alderson, W. Willinger, and J. Doyle. A first-principles approach to understanding the Internet's router-level topology. In *Proceedings of ACM SIGCOMM*, August 2004.

[19] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang. On AS-level path inference. In *Proceedings of ACM SIGMETRICS*, June 2005.

[20] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, and C. Diot. Characterization of Failures in an IP Backbone. In *Proceedings of IEEE INFOCOM*, March 2004.

[21] N. F. Maxemchuk. Dispersity routing. In *Proceedings of International Conference on Communications (ICC)*, June 1975.

[22] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. *ACM Computer Communication Review*, 30(2):18–28, 2000.

[23] J. Moy. RFC 2328: OSPF version 2, 1998.

[24] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM TON*, 5(5):601–615, 1997.

[25] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A blueprint for introducing disruptive technology into the Internet. In *Proceedings of HotNets–I*, October 2002.

[26] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *Proceedings of IEEE INFOCOM*, April 2001.

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

[28] P. Slavík. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2):237–254, 1997.

[29] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM*, August 2002.

[30] A. Sridharan, S. Moon, and C. Diot. On the correlation between route dynamics and routing loops. In *Proceedings of ACM SIGCOMM IMC*, October 2003.

[31] J. W. Stewart. *BGP4: Inter-Domain Routing in the Internet*. Addison Wesley, 1998.

[32] J. Strand, A. Chiu, and R. Tkach. Issues for routing in the optical layer. *IEEE Communications Magazine*, 39(2):81–87, 2001.

[33] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: Offering Internet QoS using overlays. In *Proceedings of HotNets–I*, October 2002.

[34] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker. In search of path diversity in ISP network. In *Proceedings of ACM SIGCOMM IMC*, October 2003.

Appendix

1.1. Integer Programming Formulation

The objective function has a form that maximize (or minimize) the sum of all variables. More specifically, the objective functions are stated as the following. The 0-1 variable $y_i, i \in V$, indicates whether the location i is selected as a relay, and the 0-1 variable $x_{ijz}, i, j, z \in V$, indicates whether OD pair (j, z) is assigned to the relay at i:

$$\text{minimize} \quad \sum_{i,j,z \in n} \mathcal{K}_{jz}(i)\ x_{ijz} \tag{9}$$

$$\text{subject to} \quad \sum_{i \in V} x_{ijz} \geq 1 \quad \text{for each} \quad j, z \in V, \tag{10}$$

$$x_{ijz} \leq y_i \quad \text{for each} \quad i, j, z \in V, \tag{11}$$

$$\sum_{i \in V} y_i \leq k, \tag{12}$$

$$x_{ijz} \in \{0, 1\}, \quad \text{for each} \quad i, j, z \in V, \tag{13}$$

$$y_i \in \{0, 1\}, \quad \text{for each} \quad i \in V. \tag{14}$$

The set of constraints (10) ensures that each OD pair (j, z) is assigned to some relay $i \in V$, the set of constraints (11) ensures that, whenever OD pair (j, z) is assigned to a relay i, then a relay must have been selected at i, and (12) ensures that at most k relays are chosen. Due to computational complexity, solving the problem is usually done relaxing the constraints (13) and (14) and allowing the x_{ijz} and y_i to take rational values between 0 and 1.

1.2. Greedy Selection

Following steps describe how Greedy heuristic works.

$$P(R) = \sum_{\forall o,d \in V} \min\{\mathcal{K}_{od}(r) | r \in R\}$$

Algorithm 1 GreedySelection

Input: A graph $G(V, E)$ of a network
Output: A set of relay nodes R

```
R ← ∅
while |R| < k do
  for all r_i ∈ V \ R do
    Calculate P(R ∪ {r_i})
  end for
  Let r_k = arg min_{r_j ∈ V\R} P(R ∪ {r_j})
  R ← R ∪ {r_k}
end while
```

1.3. Local Search

Following steps describe how Local heuristic works.

Algorithm 2 LocalSearch

Input: A graph $G(V, E)$ of a network
Output: A set of relay nodes R

```
Let R ⊂ V be an arbitrary subset of size k
while there are no changes in R do
  for all v ∈ R and v' ∈ V \ R do
    if P(R) > P(R − v + v') then
      R ← R − v + v'
    end if
  end for
end while
```