Overlay Node Placement: Analysis, Algorithms and Impact on Applications

Sabyasachi Roy[1], Himabindu Pucha[1], Zheng Zhang[1], Y. Charlie Hu[1], and Lili Qiu[2]
[1]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907
[2]Department of Computer Science, University of Texas at Austin, Austin, TX 78712

***Abstract*— Overlay routing has emerged as a promising approach to improving performance and reliability of Internet paths. To fully realize the potential of overlay routing under the constraints of deployment costs in terms of hardware, network connectivity and human effort, it is critical to carefully place infrastructure overlay nodes to balance the trade-off between performance and resource constraints. In this paper, we investigate approaches to perform intelligent placement of overlay nodes to facilitate (i) resilient routing and (ii) TCP performance improvement. We formulate objective functions to accurately capture application behavior: reliability and TCP performance, and develop several placement algorithms, which offer a wide range of trade-offs in complexity and required knowledge of the client-server location and traffic load. Using simulations on synthetic and real Internet topologies, and PlanetLab experiments, we demonstrate the effectiveness of the placement algorithms and objective functions developed, respectively. We conclude that an approach, hybrid of random and greedy approaches, provides the best tradeoff between computational efficiency and accuracy. We also uncover the fundamental challenge in simultaneously optimizing for reliability and TCP performance, and propose a simple unified algorithm to achieve the same.**

I. INTRODUCTION

Overlay routing has recently emerged as a promising approach to improving efficiency and reliability of Internet paths. In overlay routing, an end host has the flexibility in routing its traffic to its destination through one or multiple intermediate overlay nodes. By properly selecting the intermediate overlay nodes, the end host can optimize a variety of objectives. For example, overlay routing has been used to improve reliability of Internet paths in Detour and RON [1], [2]. It has also been used for improving TCP throughput [3], [4] by breaking the end-to-end feedback loop between a source and a destination into multiple pipelined sub-loops using overlay nodes. Moreover, it has been used to perform multipath routing to optimize end-to-end performance [5]. While previous work has focused on overlay routing given a set of overlay nodes, this paper addresses the problem of *choosing the set of overlay nodes to maximize the gain due to overlay routing*.

There are two broad types of overlay networks: peer-to-peer networks and infrastructure overlay networks. A peer-to-peer network (p2p) is a highly dynamic environment governed by the churn of the peer nodes. Compared with highly dynamic p2p, an infrastructure overlay has much better connectivity, higher persistence and availability. Moreover, it is typically managed by a single administrative entity. Therefore, infrastructure overlays are the most effective in fully realizing the potential benefits of overlay routing.

While such a dedicated infrastructure overlay network deployment provides the target applications or users the benefit of optimizing overlay routing performance, it also comes at a significant cost in terms of time, hardware, network connectivity, and human effort in maintaining the system. Thus, it is critical to carefully place infrastructure overlay nodes to balance the trade-off between overlay application performance and resource constraints.

This paper investigates approaches to perform intelligent placement of infrastructure overlay nodes for a generic set of overlay routing applications. The placement problem is stated as: *Given M possible locations for overlay nodes and a budget of k possible nodes, how to place k overlay nodes to optimize some application-specific performance metric.*

As a concrete example, we focus on the problem of overlay node placement in the context of the following two overlay routing applications: (i) improving routing reliability using SOSR [6] that employs a single overlay node in the overlay path, and (ii) improving TCP throughput using SLOT [4] that leverages multiple overlay hop paths. In this paper, we develop and evaluate a set of heuristics to achieve improved application performance. Besides performance, one of the primary focus of our heuristics is to allow for *incremental deployment*, as incremental deployability is indispensable under practical constraints of budget and dynamic user traffic patterns.

We study the pros and cons of various placement heuristics via simulations under a variety of topologies and network conditions. We then conduct real experiments on the PlanetLab testbed to observe how well the different heuristics, which try to optimize objective functions designed based on stable topological and network properties such as underlying routing paths and path RTTs, perform in terms of more abstract application layer performance metrics such as failure-recovery (reliability) and TCP throughput gains that can not be easily incorporated into objective functions.

Since an overlay is typically used for more than one application, it is critical to simultaneously support diverse applications using a common overlay infrastructure. Motivated by this observation, we further explore how to place overlay nodes to simultaneously improve the performance of multiple applications, in particular, reliability and TCP performance.

We make the following contributions in this paper.

- We propose objective functions to accurately capture application behavior, namely, reliability and TCP per-

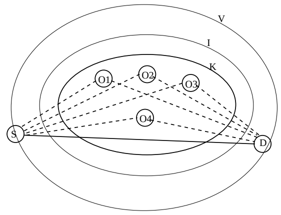

Fig. 1. Example scenario explaining the working of SOSR.

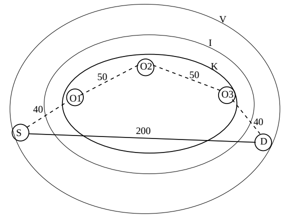

Fig. 2. Example scenario explaining the working of SLOT.

formance. We formulate placement problems aiming to optimize the proposed objective functions, and analyze their complexity.

- We develop a series of heuristics that operate with varied levels of topology and traffic information, and provide different tradeoffs between complexity and performance. We extensively evaluate the heuristics using simulations based on real and synthetic network topologies. Our results show that on average intelligent placement techniques outperform random schemes by up to a margin of 100% for reliability (SOSR) and 200% for TCP performance (SLOT).
- We demonstrate the effectiveness of the objective functions and heuristics in capturing and optimizing abstract application performance metrics, namely, reliability and TCP performance via PlanetLab experiments. The experiments show that intelligent placement techniques recover from 100% more number of failures with 80% probability for SOSR, and achieve 50% more throughput gain for 40% of the cases for SLOT, as compared to random schemes.
- We identify the fundamental challenge in simultaneously optimizing for reliability and TCP performance, and propose a simple unified algorithm to simultaneously optimize both objectives.

II. Background

In this section, we provide a brief background on the two sample overlay routing applications considered for the placement problem in this paper. The applications are chosen so that the placement problem can be studied under two different conditions: (1) a scenario where a single overlay node is chosen in overlay routing (SOSR), and (2) a scenario where multiple overlay hops are leveraged in overlay routing (SLOT).

A. SOSR

Failure recovery using overlay routing has been studied in Detour [1] and RON [2]. Different from the previous work, SOSR [6] proposes one-hop source routing approach to mitigate Internet path failures. More specifically, upon path failures the source node randomly chooses four nodes as relays to re-route traffic. SOSR is shown to recover from 20-56% failures. Figure 1 shows how a source node S uses overlay paths S-O1-D, S-O2-D, S-O3-D, and S-O4-D to reroute traffic.

B. SLOT

Next we consider SLOT (Shortened Loop Overlay Transport) [4], an *infrastructure-based* solution that aims to increase the throughput of the standard TCP by exploiting the well-known relationship between TCP throughput and round-trip-time. SLOT leverages overlay routing to break up an end-to-end TCP connection into multiple shortened TCP (sub-)connections (with smaller RTTs) forming a multi-overlay-hop path over which data bytes are *pipelined* to enable end-to-end transport. Since the throughput of an end-to-end SLOT path is constrained by that of the worst overlay hop, SLOT chooses a multi-hop path that minimizes the maximum RTT of the sub-connections. Figure 2 shows an example scenario: the direct path between S and D with RTT 200 is split into a four-hop path: $S - O1 - O2 - O3 - D$ with an effective path RTT of 50.

III. The Overlay Placement Problem (OPP)

In this section, we first define the generic overlay placement problem. We then develop objective functions for two special instances of OPP, namely, SOSR and SLOT. Note that the design of the objective functions is completely governed by the application it is designed to optimize. Finally, we formulate the respective overlay placement problems, namely, SOSR-Overlay Placement Problem (SOSR-OPP) and the SLOT-overlay placement problem (SLOT-OPP). The problems are similar in nature but differ in the way in which the objective function is defined.

Definition 1: (Generic Overlay Placement Problem) We state the overlay placement problem as a graph-theoretic problem. Given a graph $G = (V, E)$, where V is a set of N nodes and E is a set of edges between these nodes that denote the underlying routing edges, a set $I \subseteq V$ of M nodes that are potential nodes for intermediary placement, and a set $C \subseteq V \times V$ of client-server pairs (c, s). The OPP problem is to find a set $K \subseteq I$ of k nodes such that it optimizes some overlay routing objective function.

In this paper, we consider *uncapacitated* overlay node placement problem that assumes unlimited capacity at each overlay site. This is a reasonable assumption since it is much easier and less costly to add a node at an existing site than adding a node at a new site. Throughout the paper, we use overlay sites and overlay nodes interchangeably.

A. SOSR

a) Design of an objective function to capture reliability: In order to capture reliability characteristics of overlay routes, we develop the objective function obj_{sosr}. It assumes information about underlying routing hops between nodes and the traffic volume between the client-server pairs. Given this information the objective function obj_{sosr} is defined as

$$obj_{sosr} = \frac{\sum_{(c,s)\in C} T[c,s] * LO[(c,s), H(c,s)]}{\sum_{(c,s)\in C} T[c,s]} \quad (1)$$

where $T[c, s]$ is the traffic flowing between a client c and a server s, $H(c, s) \subseteq K$, and $LO[(c, s), H(c, s)]$ is the average pairwise overlap between a set of $|H|+1$ paths, namely, one *direct path* from c to s, and $|H|$ *single-hop* detoured paths between the same two nodes. Overlap between two paths is defined as the number of links (edges in the graph) that are common between the two paths. The direct path between c and s is the shortest hop path between the two nodes and the detoured path via a hop m is a path consisting of a direct path from c to m followed by another direct path from m to s.

While an overlay path containing multiple intermediate hops may provide more flexibility and less overlap with the direct path, measurements in [6] have shown that one-hop detour is sufficient to significantly reduce Internet routing failures. Hence in this paper, we will use one intermediate hop for overlay routing for reliability.

b) Significance of the LO factor: While it is important to find alternative paths with minimum overlap with the direct path, it is imperative that the alternative paths themselves have as low pairwise overlap among themselves as possible. This is captured by the LO factor in obj_{sosr}. By calculating the *pairwise overlap* between all the paths, i.e., $|H|$ detoured paths and a direct path (($|H|+1$)-choose-2 pairs), it is ensured that all the alternative paths have minimum failure correlation and provide reliability under multiple failures. [7] shows that two paths experiencing failures show a correlation that depends on the pairwise path overlap.

The cardinality of the set $H(c,s)$ is critical. It should be not too small, otherwise, there would not be enough options to choose from, and it should not be too large because it is impractical and inefficient to detour data through such a large number of alternative paths. A suitable choice for $|H|$ is four, as shown in [6] based on Internet experiments. As in SOSR, set $H(c,s)$ is randomly chosen from the set K for each client-server pair. Thus, it is important to choose the set K in a manner such that obj_{sosr} is low irrespective of what set is chosen as $H(c,s)$.

c) SOSR Overlay Placement Problem (SOSR-OPP): Given an objective function obj_{sosr} to capture reliability of overlay routing, we formally define the optimal placement problem for SOSR.

Definition 2: (k-Overlay Placement Problem for SOSR (k-SOSR-OPP))
INSTANCE: Same as the Generic OPP.
QUESTION: Is there a set $K \subseteq I$, such that $obj_{sosr} \leq B$, where obj_{sosr} is given by (1) and B is a constant.

B. SLOT

a) Design of an objective function to capture TCP performance: Similar to SOSR, we develop obj_{slot} to capture TCP performance of overlay routes. It also assumes that the information about the RTTs between client and server pairs and the corresponding traffic volume are known. Given this information the objective function obj_{slot} is defined as

$$obj_{slot} = \frac{\sum_{(c,s)\in C} T[c,s] * \frac{RTT[c,s]}{ORTT[c,s,k]}}{\sum_{(c,s)\in C} T[c,s]} \tag{2}$$

where $RTT[c,s]$ is the RTT of the direct path between c and s, $ORTT[c,s,l]$ is the best overlay RTT possible between nodes c and s after l nodes have been chosen from K. The matrix $ORTT[c,s,l]$ is calculated in an incremental manner. Assuming that at any given time we have chosen i intermediaries, and we choose the $(i+1)$-th intermediate node a, then $ORTT[c,s,i+1] = min(ORTT[c,s,i], max(ORTT[c,a,i], ORTT[a,s,i]))$. Note that unlike SOSR, for SLOT each client-server pair (c,s) is allowed to route via multiple intermediate hops.

b) SLOT Overlay Placement Problem (SLOT-OPP): Given an objective function obj_{slot} to capture TCP performance of overlay routing, we formally define the optimal placement problem for SLOT.

Definition 3: (k-Overlay Placement Problem for SLOT (k-SLOT-OPP))
INSTANCE: Same as the Generic OPP.
QUESTION: Is there a set $K \subseteq I$, such that $obj_{slot} \geq B$, where obj_{slot} is given by (2) and B is a constant.

IV. Complexity Analysis

In this section, we first prove that OPP is NP-hard for SOSR when one intermediate hop per pair is used in the overlay path. We then prove OPP is NP-hard for SLOT where multiple intermediate hops are allowed in the overlay path.

A. SOSR-OPP is NP-hard

We prove that SOSR-OPP is NP-hard by first reducing it to a decision problem k-SOSR-OPP. We then translate it to derive the k-SOSR-OPP-modified problem. Finally, we show that k-SOSR-OPP-modified problem is the same as the k-median problem [8]. This sequence of steps prove that k-SOSR-OPP is NP-complete and hence SOSR-OPP is NP-hard.

Definition 4: (The k-median problem (k-MP))
INSTANCE: Given a weighted graph $G' = (V', E')$, with edge-weights $w(e)$. Given $I' \subseteq V'$, and $C' \subseteq V'$.
QUESTION: Is there a set of nodes K' where $K' \subseteq I'$ and $|K'| = k$, such that the objective function $\sum_{c\in C'} w(e_c) \leq B$, where $w(e_c) \geq 0$ and is the weight of the lowest-weighted edge from c to any node $k' \in K'$?

Definition 5: (Modified k-Overlay Placement Problem for SOSR (k-SOSR-OPP-modified)) We modify k-SOSR-OPP problem to a different version. From a graph $\bar{G} = (\bar{V}, \bar{E})$, in k-OPP-SOSR, we construct another graph $G = (V, E)$.
INSTANCE: A graph $G = (V, E)$ is constructed, where $V = I \cup U$, where I and U are disjoint sets and $I = \bar{I}$. Each $u \in U$ corresponds to a pair $(c,s) \in \bar{C}$. The edges $e \in E$ are such that $e = (i,u)$, i.e., the endpoints of such edges contain one node from I and one from U. The edge-weights of edge e is given by $ew(e) = ew(i,u) = ew(i,(c,s)) = LO[(c,s), H]$, where H is a singleton set such that $H = u$.
QUESTION: Is there a set $K \subseteq I$, such that $\sum_{u\in U} w(e_u) \leq B$, where $w(e_u) = min(ew(j,u))$, $\forall j \in K$.

Theorem 1: The k-SOSR-OPP problem is NP-complete.
The detailed proof is omitted due to space limit and can be found in [9].

B. SLOT-OPP is NP-hard

SLOT-OPP is different from SOSR-OPP in two ways. First, SLOT-OPP tries to maximize its objective function while SOSR-OPP tries to minimize its objective function. Second, multiple intermediate hops are allowed in SLOT-OPP while SOSR-OPP uses only a single overlay hop. Therefore we cannot directly use the proof for Theorem 1. Instead, we first obtain a decision version of the SLOT-OPP problem, called k-SLOT-OPP. We then prove that k-SLOT-OPP is NP-complete by reducing an instance of k-MP to an instance of k-SLOT-OPP. This proves that SLOT-OPP is NP-hard. The details of these steps can be found in [9].

V. Incremental Placement Algorithms

In this section, we present a number of heuristic algorithms for the overlay placement problem. The algorithms are the same for both SOSR-OPP and SLOT-OPP, and differ in the objective functions used. In general, all such heuristics help us choose K intermediary nodes out of N nodes.

All the heuristics described below are designed in a way so that they are incremental in nature. We only consider incremental algorithms since they make infrastructure overlay administration practical, i.e., it is impractical to change the entire deployment every time a new node is to be added. Below we classify the heuristics into three main categories.

A. Uninformed Heuristics

Heuristics in this category are completely unaware of the network conditions and topology information such as the path latencies, the traffic between clients and servers, and the underlying routing layer topology. Thus, the placement decision is completely random. We would call such an heuristic as $Random$. The Random heuristic serves as a lower bound of the performance for the rest of the heuristics.

B. Partially-informed Heuristics

Such heuristics have only partial information about the network. Heuristics in this category are unaware of the relatively dynamic network conditions such as the path latencies and the traffic between clients and servers. They make their decisions based on network topology properties which are relatively stable.

a) Node Degree Based (NDB): The NDB algorithm greedily chooses nodes with large numbers of edges attached to it. Intuitively, for SLOT, a high-degree node can split a TCP connection into small RTT segments as it can be reached from a client or a server in a few number of hops. However, for SOSR, it is counter-intuitive to choose high-degree nodes as such nodes are more likely to be traversed by different paths between different pair of nodes, leading to more likelihood of overlap between such paths. On the contrary, such nodes may also yield low path overlap as such nodes can be reached using few hops, thus, reducing the scope of overlap. Thus, the behavior of the NDB heuristic is not obvious and hence, worth exploring. Note that this algorithm uses the degree based on the routing edges at the underlying routing layer.

C. Fully informed Heuristics

The heuristics in this category assume complete information about the traffic and latencies for SLOT, and traffic and physical route information for SOSR, between the client-server pairs, and thus can make more informed decisions at the cost of higher computational complexity. Since the heuristics are incremental, such information can be easily collected by logging the traffic from the current deployment of the infrastructure overlay. We note that such information may change over time and consequently the quality of placement generated by such heuristics may change accordingly.

a) Traffic Aware Greedy (TAG): The TAG algorithm works as follows. At each step of the algorithm, one new intermediary is selected. The choice of such an intermediary is based on adding which node gives the best value of the objective function. The steps of the algorithm are as follows. If there are a total of M nodes from which the intermediaries can be chosen and m nodes have already been selected as intermediaries (denoted as set S), to select the $(m+1)$-th node, we iterate over the remaining $M-m$ candidate nodes. At each iteration, we add one node to the set S and re-calculate the objective function for the new set S. The node that gives the best value of the objective function is chosen as the $(m+1)$-th node. Note that this is the *optimal incremental* algorithm when a *single* node is added at each step. Given that incremental deployment is of prime importance, this algorithm serves as the upper bound of performance among all placement algorithms as it is the optimal among all incremental algorithms. The complexity for this algorithm is $O(k(M+W)^2M)$ for SLOT, where W is the number of nodes that are either a client or a server, and the complexity for SOSR is $O(k|C|Md)$, where $|C|$ is the cardinality of the set C of client-server pairs. A detailed derivation of the time complexities can be found in [9].

D. Hybrid Heuristics

Hybrid heuristics are a hybrid of traffic-aware heuristics and Random heuristic. Such heuristics tradeoff performance with computational complexity.

a) Traffic Aware Greedy-p (TAG-p): The TAG-p algorithm chooses the set of intermediaries randomly but incrementally as follows. It generally follows TAG and differs in the number of candidate nodes that are screened at each step. In particular, when choosing the $(m+1)$-th node, it iterates over a set of $min(p, M-m)$ nodes randomly chosen from the set of $M-m$ remaining candidate nodes. Out of the p nodes, it chooses the node that gives the best objective function as the $(m+1)$-th node. Thus, the complexity of TAG-p is smaller than TAG by a factor of $\frac{M}{p}$. Note that TAG-p where p is equal to M becomes equivalent to TAG.

VI. Simulations

In addition to the factors captured in the objective functions, the ultimate performance of various applications is affected by additional factors, for example, the TCP window size and Internet congestion in the case of SLOT. Hence, it is imperative to study the performance of various heuristics in a real testbed. Nevertheless, to understand the strengths and weaknesses of the various placement strategies and their comparative behavior under varying topologies and network conditions, it is worthwhile to conduct experiments under a controlled setup. For this purpose, we first perform a comprehensive simulation study in this section. We then present the Internet experiments in the next section.

A. Methodology

To evaluate the performance of various algorithms and to understand the importance of placement of nodes, we use simulations on a variety of network topologies and workloads. We use two fundamentally different types of graph topologies, namely, random graphs and hierarchical graphs. For random graphs, we use the GTITM [10] topology generator. We use two different models of random graphs: pure-random model and waxman model. Results for GTITM-waxman are similar to the pure-random model, hence, we do not show the corresponding results in this paper for the sake of brevity. The results can be found in [9]. For hierarchical graphs we use the Inet [11] topology generator. GTITM also generates hierarchical graphs using a transit-stub model, but the Inet generated graphs reflect the Internet structure more closely [11]. Hence, for the sake of brevity we only show results for Inet-generated hierarchical graphs. With GTITM, we use a network topology of 1000 ($N = 1000$) nodes. For Inet, we generate 3037-node topologies ($N = 3037$). We run simulations on several different topology instances generated by each topology model and present the average behavior over all the instances. For the simulations, we consider all the nodes in the network as potential intermediaries ($M = N$). For each simulation run, we incrementally choose 30 nodes ($k = 30$) as intermediaries.

For client-server traffic, we use real web proxy traces provided by `www.ircache.net` for 10 proxies, geographically distributed across the US. Each proxy has a per-day trace for the week of 20th October, 2005. Each of these traces are parsed to extract client to server request patterns. Since the set of IP addresses of the client-server pairs in the web log is much larger than the number of nodes in the network, we cluster the clients and servers using IP-address-based aggregation and map them to the nodes in the network. To reduce the running time of the simulations, we prune the number of clients and servers by choosing the top 200 clients and server nodes (after clustering) that correspond to the most traffic-generating client-server pairs. The rest of the nodes are assumed not to generate any traffic.

We also conducted simulations on real Internet traces to compare the different heuristics. For SOSR, we used the same trace as used in [12] which consists of BGP routing data from seven geographically dispersed BGP peers. Each BGP table entry consists of a sequence of ASes called the AS paths. We created a topology based on these AS paths. Thus in this topology the edges are the links between ASes in the AS paths, as opposed to edges between routers. The intuition here is that diversifying the ASes in the paths traversed increases the resilience when one AS is down.

For SLOT, we used a topology trace used in [13]. The topology consists of 89 Traceroute Gateways that serve as the set of potential intermediaries, and a set of 3130 IP addresses that serve as clients and servers. These nodes make up a “virtual network”. The traceroute gateways measure RTTs among themselves. For the path between a client and a server, the RTT is approximated as the sum of the RTTs of the client to the closest traceroute gateway and of the server to the closest traceroute gateway and the RTT between the two gateways, similarly as in [13]. We used separate Internet traces for SOSR and SLOT as they had no information about RTTs and routing layer topologies, respectively.

B. Results - SOSR

In all simulations and experiments with SOSR, we used $LO[(c,s), H(c,s)]$ such that $H(c,s)$ is a set of 4 nodes chosen randomly from among all potential intermediaries for each pair (c,s). For each heuristic, we plot the objective function as presented by (1).

a) Inet: Figure 3 shows that careful placement of the nodes clearly benefits the reliability performance of the system. TAG gives almost 70% less overlap than the Random strategy. As expected TAG-p achieves progressively better performance as p is increased and gives a good tradeoff between cost and performance. Specifically, TAG-5 and TAG-20 outperform Random by 50%, 60%, respectively. Also, the NDB heuristic performs as poorly as Random.

b) GTITM: Figures 4 asserts the results obtained for Inet topologies and the relative qualitative performance of the heuristics are retained. For GTITM-pure-random, TAG outperforms Random by 200%, and TAG-5, TAG-20 outperform Random by around 100% and 150%, respectively.

There are two major differences between the results of GTITM and Inet topologies. The first point of difference lies in the actual values of the overlap. GTITM gives lower values of overlap than Inet because of Inet’s hierarchical tree structure. Intuitively, Inet topology can be thought of as a set of trees, as shown in Figure 5, containing nodes with high degrees serving as the roots of those trees, depicted by nodes A, B, and C. A low degree node has to traverse the root of the respective tree it belongs to in order to reach another node. For example, the direct path from $E1$ to $E2$ passes through A and B (the roots of trees $E1$ and $E2$ reside in). Thus, irrespective of which node is chosen as an intermediary to detour through, there is a significant overlap between the direct and the detoured path. For example, for the endpoint pair $E1 \rightarrow E2$, no matter which node is chosen as an intermediary, there will be always overlaps at links $l1$, $l2$, $l3$ and $l4$. This leads to a higher value of average overlap for the Inet topologies. The second difference is that the performance gap between Random and TAG is smaller for Inet. It is also due to the hierarchical structure mentioned before. Since the overlap is high irrespective of the heuristic used, the scope of percentage

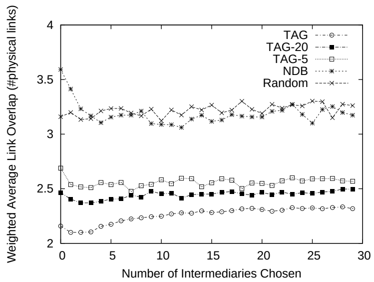

Fig. 3. Average path overlap for SOSR under Inet generated networks.

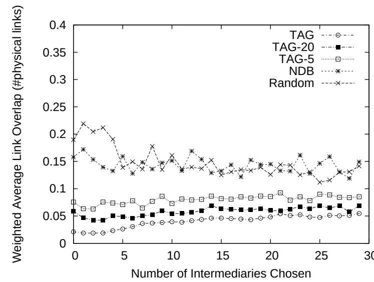

Fig. 4. Average path overlap for SOSR under GTITM-pure-random generated networks.

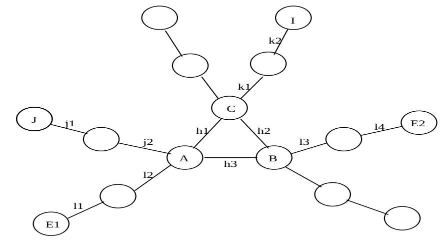

Fig. 5. A hierarchical graph with nodes with varied degrees is a forest of tree structures.

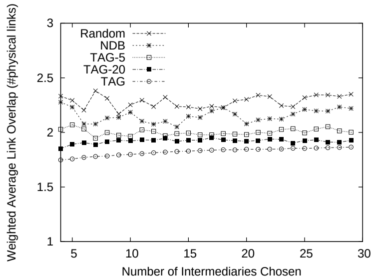

Fig. 6. Average path overlap for SOSR under real Internet traces.

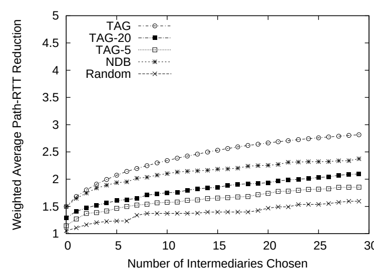

Fig. 7. PathRTT reduction for SLOT under INET generated network.

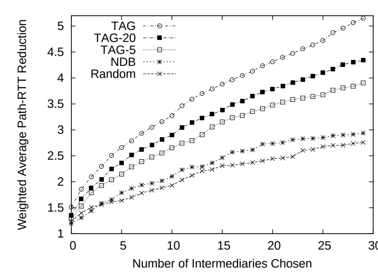

Fig. 8. PathRTT reduction for SLOT under GTITM-pure-random generated network.

improvement is less, and hence the performance gap between TAG and Random is smaller.

One of the interesting observations was that none of the curves are monotonically decreasing, i.e., increasing the number of intermediaries picked does not change the overall performance of any of the heuristic significantly. The fact that we minimize the pairwise overlap while *randomly* choosing 4 overlay nodes per pair explains that. Moreover, for TAG the curve tends to go up as more nodes are selected because TAG chooses the best 4 nodes as the first 4 nodes. This behavior is due to the fact that as more nodes are added, the random-4 technique causes nodes other than the best 4 nodes to be chosen, and hence the average path overlap increases. Thus, if the best-4 nodes were chosen instead of the random-4, the gap between TAG and Random strategies would have been larger, and the results show the lower bound on the performance gap between the two heuristics. Moreover, this observation does not imply that choosing a small number of intermediaries is sufficient because a larger number of intermediaries provides more redundancy, and hence more resilience to multiple concurrent path failures.

c) Real Trace: Figure 6 shows the relative performance of SOSR under a scenario constructed based on the Internet trace. A difference of about 50% can be observed between TAG and Random schemes. TAG-5 and TAG-20 also outperform Random by around 30% to 40%.

C. Results - SLOT

In all simulations and experiments with SLOT, for each heuristic we plot the objective function as presented by (2).

a) Inet: Figure 7 shows that proper placement of the intermediaries is critical to the good performance of SLOT, under Inet topologies. The TAG heuristic is easily the best and outperforms the Random placement significantly with about 100% improvement. Another point worth noting is that TAG-p performs reasonably well as p is increased. TAG-5 and TAG-20 outperform Random by around 50%. Finally, as the number of intermediaries increases, the gain due to using SLOT also increases and so does the performance gap between the TAG and Random heuristics. This indicates that placement becomes more critical as the number of intermediaries chosen increases as long as the number of intermediaries chosen, k, is much less the number of potential intermediaries M.

Interestingly, NDB outperforms TAG-20. Again, such a behavior can be explained on the basis of the hierarchical structure of the Inet topology. Choosing high-degree nodes as intermediaries allows connections to be broken into segments with small RTTs because a high-degree node can be reached from a client or a server in a few number of hops. For example, consider Figure 5. For simplification, we assume that all links have the same RTT R. Thus, in order to split a connection between $E1$ and $E2$, choosing A or B or C is more efficient than choosing I or J. This is so because in the former case the total path RTT is reduced from 5R to 3R, whereas in the latter case the improvement is zero. Random selection selects a large number of end nodes such as I and J, but NDB automatically chooses high-degree nodes such as A, B and C.

b) GTITM: Figures 8 shows similar relative performance results as Figure 7 and confirm the findings with Inet topology. Specifically, TAG, TAG-5, TAG-20 outperform Random by around 100%, 50% and 60%, respectively. As in SOSR, one of the differences is the value of the *Weighted Average PathRTT Reduction* metric between the results obtained for GTITM and Inet topologies. The reason for this is that Inet captures the node degree skew better than the random graphs generated by GTITM. The skew makes routing for Inet more constrained (the shortest paths between any two nodes pass through the few high-degree nodes), and hence the gains due to using

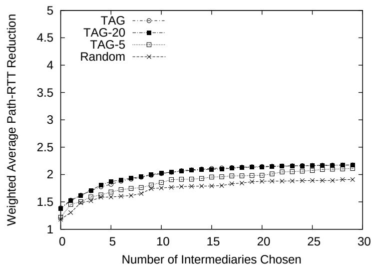

Fig. 9. PathRTT reduction for SLOT under real Internet traces.

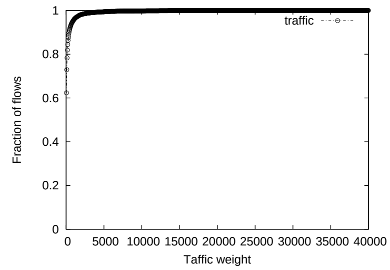

Fig. 10. CDF of traffi c weights across all the fbws in the network showing a heavily skewed distribution.

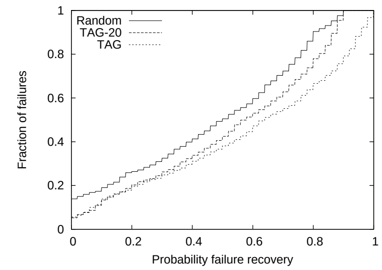

Fig. 11. CDF of failure recovery probability for SOSR under PlanetLab deployment.

SLOT is less. Nevertheless, this difference manifests itself in all the heuristics and the relative performance gap between TAG and Random remains close to 100%, similar to what was observed for the Inet topologies. Another difference is the relative performance of NDB heuristics for the graphs generated by Inet and GTITM. Unlike with Inet graphs, NDB performs as poorly as Random for GTITM graphs because the node-degrees in the GTITM is uniformly random.

c) Real Trace: Figure 9 shows the relative performance of SLOT under a scenario constructed based on the Internet trace described in (§VI-A). A difference of about 30% can be observed between the TAG and Random schemes. Note that this difference is much smaller compared to the synthetic scenarios generated using GTITM and Inet because the number of potential intermediaries is 89 in this case as compared to 3037 and 1000 for Inet and GTITM, respectively, which in turn reduces the scope for improvement. Again, TAG-20 performs as well as TAG because of the small number of intermediaries. We do not show the results for NDB as we could not obtain the degree information for each node.

D. Implications

We have observed that relative performance of the heuristics are different depending on the type of topology, namely, random or hierarchical. Thus, depending upon the topology, application and computational resource constraints, appropriate heuristics should be chosen. For example, under hierarchical topologies, NDB can be used for SLOT. For both SOSR and SLOT, under all topologies, TAG-p can be used for a good tradeoff between the complexity and performance.

Under cases where the topology is very large, intermediaries can be chosen in two steps. In the first step, an initial set of nodes can be chosen using NDB (e.g. SLOT under hierarchical topology) or TAG-p (with a small p value), and then the final set can be selected from the nodes chosen in the first step using TAG or TAG-p (where p is large). For the NDB heuristic, the AS-level topology (obtained from BGP routing data) can be used to estimate node-degrees, and nodes residing in ASes with high-degrees can be selected.

E. Robustness to traffic changes

The above results hold true assuming perfect knowledge about the traffic generated by each node. In practice, perfect knowledge is not possible and only estimates can be obtained.

TABLE I

ROBUSTNESS OF THE TAG HEURISTICS TO TRAFFIC FLUCTUATION FOR SOSR AND SLOT. THE NUMBERS SHOW THE AVERAGE AND STANDARD DEVIATION (FORMAT: AVG, STD) OF THE PERCENTAGE CHANGE IN OBJECTIVE FUNCTIONS WITH IMPERFECT AND PERFECT TRAFFIC KNOWLEDGE.

Topology	SOSR	SLOT
INET	0.4, 0.2	0.3, 0.03
GTITM	4, 0.5	1.5, 0.4

In this section, we study the robustness of the proposed heuristic placement algorithms with respect to such imperfect knowledge of traffic demands.

Our approach is to salt the input traffic information with random noise, similar to the methodology used in [12]. In particular, we perturb the volume of the traffic from an initial value of d to a random value between 0 and $2d$. We feed the salted input traffic to the placement algorithms and compute the list of intermediaries. Using the set of intermediaries thus generated, we calculate the objective functions using the accurate traffic values. We compare the objective functions thus obtained, with the case where intermediaries were chosen with perfect knowledge.

Table I shows the average and the standard deviation of the percentage differences in the objective function between the cases with imperfect and perfect input. The numbers are of the form $average, standard\ deviation$. It can be observed that the differences are in the range of 0 to 4% for SOSR and 0 to 2% for SLOT. The respective standard deviations are also very close to zero. Thus the robustness results confirm that the heuristics are resilient to traffic fluctuations. The results only present the behavior of the TAG heuristic. The TAG-p heuristics behave similarly, and we do not show those results for brevity. Note that we show results for only traffic-aware heuristics as NDB and Random heuristics will choose the same set of intermediaries irrespective of what the traffic values are.

The observation that the heuristics that take traffic into account actually are robust to large traffic variations seems contradictory. This can be explained as follows. The traffic values have a skewed Zipf distribution, i.e., a few flows have a high traffic weight, while a large number of flows have small traffic weights, as shown in Figure 10. Traffic-aware heuristics tend to choose intermediaries such that high-traffic flows are benefited the most. Due to the high skew in the traffic, on average, high-traffic flows remain high-traffic and low-traffic remain low-traffic even if traffic values are perturbed by a factor of 100%. Previously, it was shown in [12] that input data about traffic is relatively stable, and that future can be predicted using past behavior.

VII. Internet Experiments

The simulation study provides insight on the pros and cons of the different placement strategies that try to optimize the two objective functions. To understand how the gains seen in simulation results translate into application performance in practice, we conducted experiments on a real testbed: PlanetLab. The PlanetLab experiments reflect how well the objective functions based merely on stable topological properties such as path overlap and path RTT, capture more abstract and fickle application level performance metrics such as failure-recovery (reliability) and TCP throughput gain, respectively.

For each of the two applications, SOSR and SLOT, we assume a client and server population and traffic from clients to servers. We also assume a set of possible overlay locations (M). Out of these M nodes, we run our heuristics to pick k overlay nodes for the given setup. We then deploy the offline generated placement and rerun the overlay applications to measure their gain in performance. To the best of our knowledge, we are the first to show the impact of informed placement algorithms on application performance using real-time experiments using an Internet-wide infrastructure, i.e., PlanetLab.

A. SOSR

1) Experiment Setup: Our client set consists of the 10 web proxies from `www.ircache.net` at 10 different locations, each corresponding to a node in PlanetLab. The proxy traces serve as the traffic pattern between the clients and the servers around the world. To keep the real experiments tractable, we chose the top 20 servers for each client that served the most number of requests as our sample traffic pattern. We picked 162 nodes from the PlanetLab testbed served as the M possible overlay locations. To run our placement heuristics, we collected topology information for the above setup by performing traceroute. This topology was then input to the heuristics to generate $k = 10$ overlay nodes as output.

To validate the gain from the placement, we instrument reliability measurements from each client to their respective servers via the chosen placement and compare the reliability performance across placements. The reliability measurements are done as follows: Each client pinged the 20 servers periodically until it detected a loss, after which it went to a *fail* mode. In this mode, it pinged the server at a higher rate and also pinged server via all the intermediaries. In case it did not get back a reply from the intermediary it concluded that the path through that intermediary also failed. Out of all the failures the transient failures and server failures were filtered out. This way of measurement is similar to the procedure used in [6]. Such a trace is collected for a period of 6 days.

The failure recovery trace was used to calculate the performance of each heuristic. For each heuristic, we calculate the probability that failures are recovered by using the intermediaries generated by that heuristic as follows. First, the set of intermediate nodes chosen by the heuristic, denoted as A, was obtained. Then the subset of intermediate nodes that could actually recover from the failure if used for overlay routing, denoted as B, was obtained based on the reliability measurement. The probability was then calculated as $\frac{|B|}{|A|}$. Note that $|A| = k = 10$.

Over all client-server pairs, we registered 20269 loss incidences out of which 1648 were inferred as long term failures. 287 long term failures were concluded to be server failures, leaving 1361 *recoverable failures*. For the probability calculation we considered only recoverable failures in determining the relative performance of each placement scheme.

2) Results: Figure 11 shows the CDF of the probability with which a certain percentage of failures were recovered using the different sets of intermediaries chosen by the different heuristics. It can be observed that TAG recovers from the failures with a higher probability than Random and TAG-20. Also, TAG-20 gives a good tradeoff between performance and complexity as observed in our simulation studies. In particular, Random is not able to recover for about 15% of the cases, whereas TAG and TAG-20 fail to recover only for 5% of the cases. TAG can recover from 40% of the failures with a probability 80% or higher. TAG-20 can recover 30% of failures with a probability of 80% or higher, whereas Random can recover from only 20% of failures with a probability of 80% or higher. In summary, the above results show that informed placement using the objective function developed indeed provides better reliability than a random placement scheme in Internet experiments.

B. SLOT

1) Experiment Setup: The traffic and the topology setup for SLOT is similar to SOSR except that the top 50 servers were chosen for each client instead of 20. This choice was largely due to the long-term nature of SOSR experiments which limits its scale. Also, for the traffic from each client to a server, only those URLs that were responsive and of size 1MB-1GB were chosen because small files do not gain from using SLOT. To run our placement heuristics, we then collected latency information for the above setup by measuring RTTs between the different nodes (clients, servers and intermediaries. TCP ACK/RST was used to measure RTTs.

To compare SLOT performance across different placements, we ran the SLOT system with the placement generated overlay set. Each client downloaded the URLs using wget from the servers in its traffic trace using SLOT and directly, and computed the gain as $\frac{Xput_{SLOT}}{Xput_{Direct}}$.

2) Results: Figure 12 shows the CDF of the per client-server pair throughput gain for each of the heuristics. It can be observed that using intermediaries selected by TAG offers higher gains than using intermediaries selected by Random. In particular, TAG and TAG-20 achieve a gain of 1.8 or more

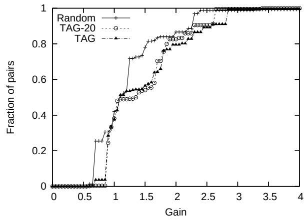

Fig. 12. CDF of pathRTT reduction for SLOT under PlanetLab deployment.

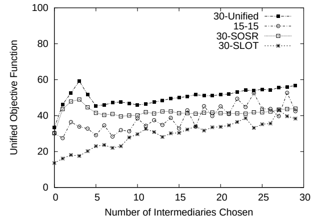

Fig. 13. Unified objective function under GTITM-pure-random generated networks.

for 40% of the pairs as opposed to a gain of only 1.2 or more for Random for the same number of pairs. TAG-20 achieves performance as good as TAG, which asserts the observations made in the simulation studies that TAG-20 achieves a good tradeoff between performance and complexity. In summary, the results show that informed placement indeed provides better performance in terms of TCP throughput than a random scheme in Internet experiments.

VIII. GENERALIZATION TO MULTIPLE APPLICATIONS

In this section, we formulate a unified overlay placement problem that simultaneously optimizes multiple objective functions for overlay routing, and we show that the unified OPP can be easily handled by heuristic placement algorithms proposed in this paper.

a) Unified Overlay Placement Problem: We consider the scenario where overlay routing is used to improve multiple performance metrics, for example, by the same ISP operator. In this case, it is administratively efficient to deploy a single infrastructure overlay, for example, for both resilience and TCP performance improvement, and possibly other performance metrics, than having a separate overlay for each performance metric. In the following, we consider the specific case of unified overlay placement problem for SOSR and SLOT. In our study, we choose an objective function defined as

$$obj_{sosr\_slot} = \frac{obj_{slot}}{obj_{sosr}} \tag{3}$$

where obj_{sosr} and obj_{slot} are defined in (1) and (2), respectively. We now formally define the unified SOSR/SLOT optimal placement problem.

Definition 6: (k-Overlay Placement Problem for SOSR and SLOT)
INSTANCE: Same as the Generic OPP.
QUESTION: Is there a set $K \subseteq I$, such that $obj_{sosr\_slot} \leq B$, where obj_{sosr_slot} is given by (3) and B is a constant.

b) Extended Heuristics: The greedy TAG heuristic presented in (§V-C) can be easily extended to solve the unified optimal placement problem as follows. As before, the heuristic incrementally adds one overlay node at a time. At each step, it picks the new node as the node that in conjunction with the already selected nodes results in the largest increase in the new objective function.

c) Performance Results: We compared the unified objective function obtained under different ways of choosing nodes. The first was to simply use TAG with the objective function obj_{sosr_slot} to incrementally select 30 nodes, denoted as 30-$unified$. The second method employed was to incrementally select 30 nodes using TAG with obj_{sosr}, denoted as 30-$SOSR$. The third method employed was to incrementally select 30 nodes using TAG with obj_{slot}, denoted as 30-$SLOT$. Finally, the fourth method chooses 15 nodes using TAG while optimizing obj_{sosr} and 15 nodes using TAG while optimizing obj_{slot}. This gave us a set of 30 nodes. The 30 nodes are ordered by alternating between the nodes selected by TAG using obj_{sosr} and obj_{slot}. This set is denoted as 15-15.

For each set of 30 nodes chosen, we incrementally calculate two scalar values, namely obj_{sosr} and obj_{slot} to obtain the corresponding obj_{sosr_slot} using (3). We conducted the above simulations only on GTITM-pure-random due to space limitation. Figure 13 shows that the unified approach outperforms the three alternative ways of selecting the set of 30 intermediaries and hence is effective in optimizing the unified objective function.

d) Designing the unified objective function: The idea of having a unified objective function is to enable us to choose a set of intermediaries, while optimizing a *single* objective function (obj_{sosr_slot}), that would lead to both (i) best reliability, and (ii) best TCP performance. However, we observed a conflicting nature of the two overlay routing problems addressed, namely, SOSR and SLOT: optimizing obj_{slot} tends to select intermediaries close to the direct path, while optimizing obj_{sosr} tends to pick nodes that are far away from the direct path in order to minimize overlap. Thus, designing a unified objective function that would improve both obj_{sosr} and obj_{slot} simultaneously, is challenging. We observed that in practice, 15-15 appears to be a good *generic unified algorithm* for selecting intermediaries that benefit applications requiring SLOT as well as SOSR. Details are omitted due to lack of space and can be found in [9].

IX. RELATED WORK

The node placement problem has been studied under various contexts [14], [15], [16], [12], [13], [17], [18], [19], [20], [21] such as Web server/cache placement. For example, in [21] the authors propose a dynamic programming based solution to proxy placement assuming the Internet as a tree topology. In [12], the authors study a Web server replica placement problem to minimize the cost for clients to access data. It develops a greedy placement algorithm, which is shown to yield close-to-optimal performance. It also shows that incorporating client location and workload information is critical to good performance. In [13], [18], the authors consider a variant of the mirror placement problem where mirrors can be placed only at a restricted set of hosts. They study this problem from the point of view of minimizing the maximum, average, and 95th percentile of the client-server latencies. They propose and study different placement They show that there is a rapid diminishing return to placing more mirrors in terms of both client latency and server load balance. In [15], the authors study the cache location problem. In particular, they study the model of a network that aims to minimize the average access delay for a single web-server. They experimentally study the effects of their algorithms using real web data transfer and find remarkable consistency over time in the relative amount of web traffic from the server along a path. In [16], the authors study the problem of replica placement to remove the requirement of knowing traffic and client locations. In [22], the placement problem was studied in the context of placing proxies for multicast function with proxy servers with the objective of minimization of delay time, minimization of bandwidth consumption, or both. In [14], the authors study a similar problem as ours with the goal of optimizing routing latency and reliability. However, their problem formulation is different from ours. They take an engineering approach, and look at which ASes, at how many Ases and where within the ASes nodes should be placed by analyzing topology dumps. They do not have definite metrics that they optimize. In comparison, we take an algorithmic approach, formally define the metrics for reliability and latency, prove the NP-completeness of the problems, and develop practical placement algorithms for a variety of networks varying in size and topology.

Overlay node placement is fundamentally different from Web servers/cache placement. In the latter, the goal is to push the servers/caches as close to the clients as possible to minimize the clients' access cost. In comparison, overlay nodes are intermediate nodes connecting the servers and the clients. Proximity towards clients alone is insufficient. Instead, the goal is to optimize the efficiency of the path from the server to the client through one or more given overlay nodes.

X. Conclusion

As overlay networks play an increasingly important role in the Internet, it becomes critical that they should be well designed to satisfy the application requirements. In this paper, we focus on one important aspect of overlay network design – strategically placing overlay nodes to satisfy the target applications. We investigate the placement problem in the context of two overlay applications: improving reliability of routing and improving TCP performance. We develop objective functions to accurately capture application behavior in terms of reliability and TCP performance. We then develop incremental placement algorithms for both applications, and demonstrate their effectiveness using simulation and PlanetLab experiments. We observed that intelligent placement algorithms significantly outperform a random heuristic. A hybrid approach combining a greedy and a random approach gives a good tradeoff between performance and computational complexity. To this end, we propose TAG-p heuristics which perform similarly to TAG. We also observe that the heuristics are resilient to traffic fluctuations.

Recognizing the need to support diverse applications using a common overlay infrastructure, we present a general framework of placing overlay nodes using a unified objective. We apply it to the two overlay applications and our results show that the two applications in consideration are contradictory in nature. We find that an approach based on selecting half the nodes with each objective function independently provides a simple, natural and efficient unified heuristic. In our ongoing work, we are studying generalizing our placement algorithms to handle more applications, capacitated overlay node placement, and topology changes. We are also interested in identifying a few general metrics that reflect the optimization objectives of overlay applications so that overlay node placement is effective for both today and future overlay applications.

Acknowledgment

This work was supported in part by NSF CAREER award grant ACI-0238379 and by NSF grant CNS-0627020.

References

[1] 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*, vol. 19, pp. 50–59, January 1999.

[2] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proc. of ACM SOSP*, 2001.

[3] Y. Liu, Y. Gu, H. Zhang, W. Gong, and D. Towsley, "Application level relay for high-bandwidth data transport," in *Proc. of GridNets*, 2004.

[4] H. Pucha and Y. C. Hu, "Slot: Shortened Loop Overlay Transport," TR-ECE-05-12, Purdue University, Tech. Rep., July 2005.

[5] T. Fei, S. Tao, L. Gao, and R. Guerin, "How to select a good alternate path in large peer-to-peer systems?" in *Proc. of IEEE INFOCOM*, 2006.

[6] K. P. Gummadi, H. Madhyastha, S. D. Gribble, H. M. Levy, and D. J. Wetherall, "Improving the Reliability of Internet Paths with One-hop Source Routing," in *Proc. of OSDI*, 2004.

[7] V. Padmanabhan, L. Qiu, and H. Wang, "Server-based inference of internet link lossiness," in *Proc. of IEEE INFOCOM*, April 2003.

[8] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[9] S. Roy, H. Pucha, Z. Zhang, Y. C. Hu, and L. Qiu, "Overlay node placement: Analysis, algorithms and impact on applications," TR-ECE-07-??, Purdue University, Tech. Rep., January 2007.

[10] K. Calvert and E. Zegura, "Internetwork topology models (gt-itm)," http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm.

[11] H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger, "Towards capturing representative as-level internet topologies," in *Proc. of ACM SIGMETRICS*, June 2002.

[12] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas." in *Proc. of IEEE INFOCOM*, 2001.

[13] S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," in *Proc. of IEEE INFOCOM*, 2001.

[14] J. Han, D. Watson, and F. Jahanian, "Topology aware overlay networks," in *Proc. of IEEE INFOCOM*, Apr. 2005.

[15] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 568–582, 2000.

[16] P. Radoslavov, R. Govindan, and D. Estrin, "Topology-informed internet replica placement," in *Proc. of WCW*, 2001.

[17] J. Xu, B. Li, and D. Lee, "Placement problems for transparent data replication proxy services," *IEEE/ACM Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1383–1398, 2002.

[18] E. Cronin, S. Jamin, C. Danny, and R. Yuval, "Constrained mirror placement on the internet," *IEEE/ACM Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1369–1382, 2002.

[19] M. Karlsson and M. Mahalingam, "Do we need replica placement algorithms in content delivery networks," in *Proc. of WCW*, 2002.

[20] S. Shi and J. Turner, "Placing servers in overlay networks," in *Proc. of SPECTS*, 2002.

[21] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the internet." in *Proc. of IEEE INFOCOM*, 1999.

[22] M.-Y. Wu, Y. Zhu, and W. Shu, "Placement of proxy-based multicast overlays," *The International Journal of Computer and Telecommunications Networking*, vol. 48, no. 4, pp. 627–655, 2005.