RC22256 (W0111-064) November 21, 2001
Computer Science

IBM Research Report

On Exploiting Link State Techniques for Internet Mapping

Lisa D. Amini
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

On Exploiting Link State Techniques for Internet Mapping

November 9, 2001

ABSTRACT

Internet mapping, or network metrics measurement, is receiving significant attention due to applications in the strategic placement of data and server mirrors to improve the user perceived performance of the Internet. The challenge is in creating a scalable, Internet-wide service to effectively route users to a server based on network proximity, server capacity and load, path congestion, and bandwidth. Recent proposals call for distributing instrumentation to periodically probe end-to-end paths. However, active end-to-end probing presents the need to sacrifice accuracy for efficiency and scalability. In this paper, we examine the efficacy of exploiting Link State techniques to alleviate this contention. Link State routing protocols are well known at the network layer for efficiency, rapid convergence, and self-stabilization, but also for limitations in scaling to very large networks. We propose an adaptation of Link State routing techniques for efficient, wide-area server selection. We present solutions, in this application layer routing context, to scalability and policy-based routing issues endemic to hop-by-hop Internet measurement. We examine both active and passive path discovery to supplement Link State probing. Finally, we evaluate our techniques using generated Internet topologies as well as experimental data collected from Internet probing.

I. INTRODUCTION

Internet mapping is receiving significant attention due to applications in the strategic placement of data and server mirrors to improve the user perceived performance of the Internet. As noted by the authors of [1], the problem of locating a server or resource in a distributed environment has been the subject of investigation for nearly a decade. Deploying replicas of Web sites at distributed locations to reduce client access latency and enhance reliability has become commonplace. While much of the early work in this area targeted selecting from among a relatively small number of mirrors, recent developments are creating new challenges.

Content Delivery Networks (CDN's) have recently emerged to accelerate access to data via the Internet. As demand continues to outpace available end-to-end bandwidth from centralized Web servers, CDN's offer a valued service -- efficient, reliable content delivery from servers distributed throughout the Internet. While some Content Delivery Service Providers (CDSP's) scramble to gain customer share by building out globally distributed networks of servers, others are looking for ways to extend their reach and scale without expansive physical deployments. The concept of peering (or internetworking) CDN's to create a virtual application layer overlay to the Internet is getting significant attention in the Internet Engineering Task

Force (IETF). In this Content Internetworking (CI) [2] model, content hosted by a given CDN may also be replicated and served from any of the servers of CDN's with whom the hosting CDSP peers.

CI extends the requirements for server selection in a number of ways. First, the number of potential servers is significantly increased. Further, because selection is made across multiple administrative domains, mechanisms should afford both content owners and service providers the ability to impose policy constraints on the decision process. Finally, the selection criteria should include relatively stationary properties, such as network proximity and server capacity, and more transient properties, such as server load, path congestion, available bandwidth and location of the requested content.

Recent proposals, such as IDmaps [7], call for distributing instrumentation to periodically probe end-to-end paths. However, active end-to-end probing presents the need to sacrifice timeliness or accuracy for efficiency and scalability. That is, the overhead for end-to-end probing scales on $O(T^2)$ where T is the number of nodes (or Tracers) that are probing the network. Obviously, the number of Tracers and the frequency of probing must be limited to the minimal required to accurately report network distance on some time scale. The timeliness of such information is anticipated to be on the order of days and therefore, not reflect transient properties.

To alleviate the limitations imposed by end-to-end probing, we are investigating hop-by-hop path metric collection and dissemination. Link State technology, which employs hop-by-hop probing, is well known at the network layer for efficiency, rapid convergence, and self-stabilization, but also for limited ability to scale to very large networks [12]. In this paper, we investigate an adaptation of Link State techniques to enable efficient representation of both the transient and stationary factors in wide-area server selection. Our work is complementary to efforts such as the IETF's CI and others, such as Sonar [3] and HOPS [4], which would expose a single Internet-wide interface enabling clients to query for relative distances of server replicas. Our contribution is in proposing and assessing solutions, in this application layer routing context, to scalability and policy-based routing issues endemic to hop-by-hop Internet measurements. We have examined both active and passive path discovery to supplement Link State probing and have evaluated our techniques using generated Internet topologies as well as experimental data collected from Internet probing.

The remainder of the paper is structured as follows. We begin with background including our goals and how they align with certain network layer routing principles. We detail our proposal for adapting Link State techniques to efficient, wide-area server selection in Section III. In Section IV, we present the results of our experimental evaluation. We compare our proposal to related work in Section V and conclude in Section VI.

II. BACKGROUND

We are interested in enabling fast, effective server selection amongst globally distributed servers. Our key evaluation criteria are scalability, convergence in the presence of failures, and network overhead. Although we believe our proposal offers benefits applicable to any wide-area service or resource location problem, we discuss our proposal in the context of binding an IP address to a service or host name. We are most interested in providing a decentralized service so client-side agents, such as DNS [5] servers acting on behalf of clients, and eventually Sonar or HOPS servers, can quickly determine to which server a request for service should be routed.

For the remainder of this paper, we will refer to the network of interconnections between routers as a graph, $\mathcal{G}(\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is the set of nodes (or routers) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of links, $deg(v)$ is the degree of node v, and $d_{\mathcal{G}}$ is the diameter of $\mathcal{G}$. We use (u,v) to refer to the edge connecting nodes u and v, and Path(u, v) to refer to the set of edges traversed by a message flowing from node u to v.

A. Why Hop-by-Hop?

In determining an appropriate probe interval, two aspects of network overhead should be considered. First, the total number of messages traversing edges in a single interval and second, the number of messages traversing any given link in a single interval. To contrast the network overhead induced by end-to-end probing to that of hop-by-hop probing, we consider the cost, $c_{u,v}$=|Path(u,v)|, of a single probe as number edges that probe traverses. Given the graph in Figure 1, c_{ae} =3, while c_{ac} =1.

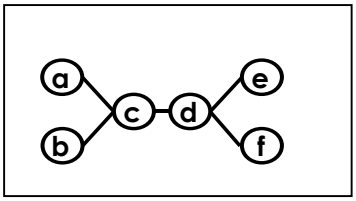


Figure 1: Six node example network.

The cost of a probe interval, i.e., the total number of messages traversing edges is:

$$C = \sum_{i,j \in T} c_{ij} = |T| * (|T| - 1) * \mu$$

where $T \subseteq \mathcal{V}$ is the set of Tracers and μ is the mean Tracer-to-Tracer path length. Alternatively, we can examine both total cost and cost per edge by defining $\mathcal{N}$ as a $|\mathcal{V}| \times |\mathcal{V}|$ matrix where n_{uv} is the number of probes traversing edge (u,v) in a single probe interval. The total number of messages traversing all edges in a single interval is

$$C = \sum_{u,v \in V} n_{uv} \ .$$

Given $t_1, t_2 \in T$, we define S as the set of nodes which lie on the path used to probe from t_1 to t_2, for any $t_1, t_2 \in T$. For a single probe interval, since Tracer probes every other Tracer once, n_{uv} must be at least 1 for any $u,v \in T \cup S$. Therefore, C is minimized when $n_{uv}=1$ for any $u,v \in T \cup S$, and $n_{uv} = 0$ otherwise.

With hop-by-hop probing, each node in $T'=T \cup S$ probes each adjacent node in T' once in a single interval. Therefore, with hop-by-hop probing $n_{uv} =1$ for any $u,v \in T'$ and $n_{uv} =0$ otherwise, and the minimum total cost results.

With end-to-end probing, S is the empty set and each node in $T'=T$ probes every other node in T'. We use the graph in Figure 1 to illustrate how the cost increases, or remains constant, as we decrease the size of T'. Assume we start with $T'=T \cup S = \{a,b,e,f\} \cup \{c,d\}$ and hop-by-hop probing for a minimum cost, C=10. As a hybrid strategy, we remove d from T' and have nodes c, e, f perform end-to-end probing amongst themselves. We note that n_{uv} is unchanged for all nodes except n_{cd} and n_{dc} which both increase by 1, thereby increasing the total cost. While the increase in this particular example is small, the increase is dependent on the network structure. In Section IV, we use the Tiers[10] and Inet[11] network topology generators to compare total and per edge costs for networks representative of the Internet.

B. Link State Principles

In this section, we briefly overview Link State routing principles. The discussion is based on the in-depth treatment in [12]. Link State technology was originally developed for the Arpanet and has led to several routing protocols, including OSPF and IS-IS. Link state protocols rely on each router periodically

probing the links to each of its neighbors so that a cost metric can be assigned to each neighbor. Each router creates a link state packet (LSP), which includes a list of neighbors and their associated metric. This LSP is generated whenever a new neighbor is discovered, a cost metric changes, or a link goes down. A router transmits any generated or received LSP to each of its neighbors using a distribution scheme that ensures each LSP travels over each link in the network exactly once. Since each router receives a full set of LSPs, each router has a complete topological map from which it can calculate a path to any other node. The algorithm most commonly used to compute paths is based on Dijkstra [13], and is referred to as Shortest Path First (SPF).

In addition to the benefits described in the previous section for hop-by-hop probing, Link State technology includes several properties that make it especially well suited to our application layer routing goals. First, because each router receives the same description of the topology, loops in route computations are avoided. Second, because the LSP for each v is small (O($deg(v)$) and is forwarded as-is, convergence occurs in O(d_G), and is independent of route computations. Third, SPF can be adapted to support multiple metrics to enable policy-based and application-dependent routing. Finally, entire partitions of the graph may be filtered to reduce computational complexity and storage – we will elaborate on the filtering aspects in Section III.

C. Network Layer *Packet* Routing vs. Application Layer *Request* Routing

While there are similarities to packet routing protocols, there are also significant differences between a routing protocol targeting efficient request routing and traditional routing protocols targeting efficient packet routing. We refer to entities that make application layer request routing decisions as Request Routers.

The following are differences between Packet and Request Routers, *i)* Request Routers choose among multiple routes to multiple destinations, as opposed to multiple routes to a single destination; *ii)* Request Routers base decisions on anticipated end-to-end route, as opposed to the current position in the path; *iii)* Instead of forwarding the request to the next router in the path, Request Routers may only supply reachability information. That is, an application may query a Request Router for an appropriate address to bind to a hostname, and then initiate a connection directly to that address – the request data need not pass through the Request Router; *iv)* Database overload, when a router has access to more reachability

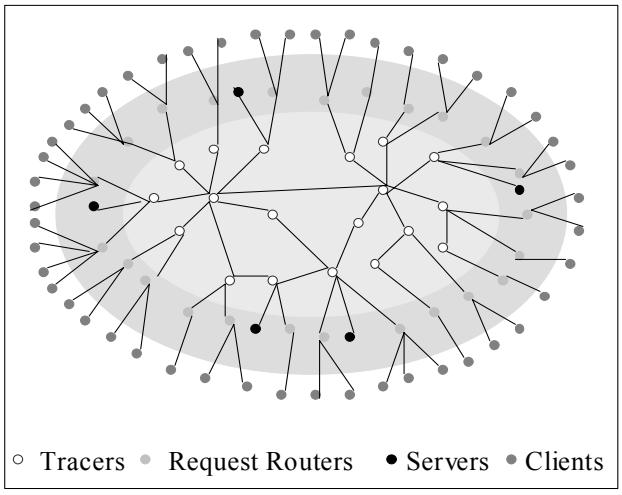


Figure 2: NEAR overlay components.

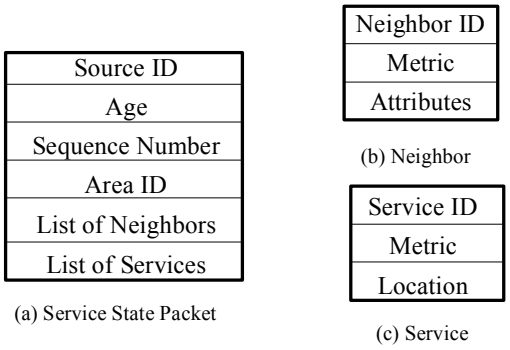


Figure 3: NEAR LSP contents.

information than it can maintain, is a serious matter for Packet Routers due to stringent response time requirements. It can be less serious for Request Routers.

All of the above listed factors play a crucial role in our ability to adapt LS techniques to the server selection problem.

III. ARCHITECTURAL OVERVIEW

In this section, we focus on the adaptations required to derive our application routing proposal from its Link State predecessors. We refer to our proposal as NEAR, for Network layer Enhanced Application Routing. As illustrated in Figure 2, the NEAR overlay consists of Tracers, Request Routers, and Clients. Tracers use Link State technology to periodically probe adjacent nodes (i.e., neighbors), maintain cost metrics, and advertise this information, in the form of LSP's, to adjacent Tracers and Request Routers. Servers may also establish sessions to Tracers to request the Tracer advertise the service provided, but do not probe or receive LSP's from Tracers. Request Routers use LSP's to construct topological maps used to respond to Client queries. Clients, such as client-side DNS servers, Sonar and HOPS servers, are responsible for providing an application layer interface to the NEAR infrastructure.

The NEAR LSP contains the elements shown in Figure 3. When a Tracer generates an LSP, it sets the Source ID to its IP address and Area ID to the IP network address and mask for the area it is advertising. Each LSP includes a list of the Tracer's neighbors, and a list of the services is the Tracer is advertising. The Sequence Number and Age elements are used by the LSP distribution scheme described later.

At startup, a Tracer will attempt to establish a reliable (TCP) connection to each of its neighbor Tracers over which LSP's will be communicated. Tracer-Request Router and Tracer-Server sessions are initiated by the Request Router or Server. Tracers will periodically probe links to neighbor Tracers and Servers, to maintain a cost metric for that neighbor. A Tracer generates an LSP whenever it discovers a new neighbor or service, the cost metric for a neighbor or service changes, or a service or link to a neighbor goes down.

Distribution Scheme: Each LSP must be transmitted, in a hop-by-hop fashion, to all other Tracers and Request Routers. We have adopted the distribution scheme used by most of today's link state routing protocols which ensures each LSP travels over each link in the network exactly once [12] with only minor adaptations.

The salient features of this adaptation are as follows. LSP's are sent to, but not received from Request Routers. Request Routers do not forward LSP's and may filter LSP's as discussed in later Path Monitoring sections. LSP's are not sent to Servers, although the metric for the Tracer-Server link is reflected in LSP's generated by the Tracer. A Request Router uses LSP's to construct a map of the topology, but this map is used only for effective server selection, not for request or packet forwarding. That is, once a server is selected for a given client request, packets exchanged between the client and server are forwarded by network layer routers, which mandates that our application layer overlay reflect the topology of the underlying packet network.

Server Selection and Sorting Algorithm: Most traditional link state routers use a form of Dijkstra's Shortest Path First (SPF) algorithm, which can be optimized to run in $O(\mathcal{E})$, where $\mathcal{E}$ is the number of Tracer-Tracer links in the network.

Figure 4 illustrates Dijkstra's algorithm as adapted for our service routing context. As shown, the list of shortest paths, **P**, is initialized to contain a zero metric path to the current node. The list of neighbors for this node, **T**, is accessed from its LSP. The closest

```
N=Path(myID,0,0,1) // id, metric, route, transits
P.insert(N);       // list of known shortest paths
do {
  T=N.getNeighborListFromSSP()
  while (n=T.extractNextClosestNeighbor() &&
        n.metric<INFINITY) {
    metric = N.metric + n.metric
    if ((q=P.find(n.NeighborID) &&
      metric >= q.metric) || !N.willTransit(n))
      continue;
    NewPath=(n.NeighborID, metric, route(N,n),
              n.transits)
    if (q) P.replaceByNeighborID(NewPath)
      else P.insert(NewPath)
    N.evaluated=true;
    N = P.NextUnevaluatedNode()
  }
} while (!T.isEmpty())
```

Figure 4: Server path calculation adapted from Dijkstra's Shortest Path First algorithm.

neighbor, **n**, is extracted from this list so the metric for the path to **n**, via **N**, may be computed. Only if **P** does not already contain a path, **q**, to **n** with a smaller metric and the path to **N** will provide transit for traffic to **n**, is the path, **NewPath**, recorded in **P** as the shortest path to **n**. Note that the elements in both **P** and **T** are kept ordered according to metric. The **route** for **NewPath** is the same as the shortest path to **N**, with **n** appended. The test, **willTransit**, checks whether the nodes in the path to **N** have the **transits** attribute. If not, then **n** must be in the same domain as the node that does not provide transit. In addition to calculating the shortest path to each Neighbor, our algorithm similarly evaluates the shortest path to each Service.

Topology Considerations: There are two significant limitations to the NEAR-SPF algorithm provided in Figure 4. First, at the time of this writing there were over 11,000 Autonomous Systems (AS's) in the Internet [14], which would render this O(*E*) algorithm too computationally intensive. Secondly, inter-AS, or Border Gateway Protocol (BGP), routing is policy-based, and does not necessarily reflect shortest paths [15].

We have investigated passive and active methods to more accurately represent the underlying packet topology. For each alternative, we assume Tracers are deployed to reflect the underlying AS connectivity by positioning a Tracer at the entry and exit points of each AS. However, the Tracer topology need not fully mirror the underlying packet routing topology, but instead, partial deployments as described in Section II.B are anticipated. We provide experimental evaluation of both techniques in Section IV.

Passive Path Monitoring: Passive monitoring can be achieved by having each Request Router establish a connection to its inter-AS (BGP) router. For each IP address prefix, a BGP router maintains a route entry, including the AS path, i.e., list of AS's in the path to the associated address prefix. Given the AS path for a given address prefix, a Request Router can construct the network path by utilizing the NEAR Tracers' LSP's. Specifically, given an IP address, the Request Router first performs a lookup of the AS path for this address. Starting with AS_i, where i=1 is the i^{th} entry in the AS path, the Request Router retrieves the list of LSP's maintained for AS_i. Then, reflective of what is referred to as hot-potato [17] routing, performs an SPF calculation to determine the path from the AS_i Tracer to the nearest Tracer in

AS_{i+1}. This procedure is repeated for each AS in the path, but each SPF calculation includes only the LSP's from Tracers in a single AS.

The advantages of passive monitoring for inter-AS path construction are fourfold. First, a resource impoverished Request Router can cache a list of AS's from paths of commonly received queries and discard all LSP's of Tracers in AS's that are not in this set[1]. Hence, the database overload issue fatal to Packet Routers results in only potential service degradation for Request Routers. Second, SPF algorithms use only LSP's within a given AS for a calculation, resulting in very fast, low-overhead computations. Third, Request Routers need not account for policy-based inter-AS routing as it is summarized in the AS path provided by the BGP router. Fourth, although BGP is slow to converge in the presence of failures [15], NEAR LSP's are disseminated in $O(d_G)$ and can provide advance warning that a given path should be avoided.

There are however, disadvantages to passive AS path monitoring. The path provided by the local BGP router represents routing behavior from the Request Router to the target IP address. A client may prefer to base server selection on the metric for the reverse path, from the target IP address to a local client. Our experiments (Section IV), indicate Internet paths are not symmetric. Second, the AS path provides only inter-AS Tracer selection guidance. That is, while hot-potato routing provides a useful heuristic for routing within an AS, it may not reflect the actual path taken [17].

Active Path Monitoring: This is actually a hybrid scheme that is based on the observation that overall, Internet paths are stable [19]. Active end-to-end path monitoring, instead of relying on the BGP AS path, uses mechanisms such as `traceroute` [16] to identify all the packet routers on a network path and thereby guide the construction of end-to-end metrics from LSP's. Active monitoring provides the same advantages as passive monitoring, with the added advantage that all nodes in the path are represented. The disadvantage of active monitoring is the overhead associated with probing and potential inaccuracies due to asymmetric routing. However, by probing only to retrieve the relatively stationary property of path, and collecting transient properties via hop-by-hop mechanisms, the overhead is anticipated to be less than a pure end-to-end probing scheme.

[1] We are currently developing tree based filtering schemes that would further reduce the number of LSP's to be disseminated.

IV. EXPERIMENTAL EVALUATION

The NEAR architecture exploits Link State routing principles to achieve a hop-by-hop Internet mapping service. Inferring end-to-end behavior from hop-by-hop measurements depends on a number of issues. In this section we analyze empirical data and simulations to evaluate the scalability, overhead and accuracy of our proposal.

A. Scalability

We are interested in the scalability and network overhead resulting from active probing, both from a hop-by-hop and from an end-to-end perspective. As discussed in Section II.B, the total cost, C, and per edge cost, $n_{uv,}$ in a single interval are affected by both the probing strategy and the network topology. We used the Tiers [10] and Inet [11] network topology generators to simulate the probing overhead using a number of placement and probing strategies. The Tiers generator generates hierarchical network structures by representing both inter-AS and intra-AS connectivity. The Inet generator generates a graph representing the inter-AS topology with power-law vertex degree frequency distribution. A detailed comparison of these topology generators can be found in [20].

Both Tiers and Inet generate topologies to reflect the administrative partitioning of today's Internet wherein each separately administered domain is classified as transit or stub [10]. A domain is classified as stub if the path connecting two nodes traverses that domain only if one of the nodes is in the domain. A domain is classified as transit if it does not have this restriction. We use this transit-stub classification to define our placement strategies.

We generated Tiers and Inet topologies for 3200, 5000 and 7000 node networks. For each topology, we tested strategies to reflect Tracer placement at selected stub domains, transit domains and random mixtures. For Inet topologies, MIN placement was achieved by placing Tracers at those nodes with the lowest edge degree, MAX by placing Tracers at nodes with the highest degree, and RAND by randomly selecting nodes with degree less than the average degree for the topology. For Tiers topologies, MIN

Simulation Parameters	
Topologies	Inet, Tiers
# of Nodes	3200, 5000,7000
Placement Strategies	MIN, MAX, RAND
Probe Strategies	E2E, HbH, Hybrid
# of Tracers	40, 80, 160

Table 1: Parameters for probe and placement strategy simulations.

placed Tracers on at most 1 node per stub domain, MAX on at most 1 node per transit domain and RAND randomly assigned at most 1 Tracer per transit or stub domain. Table I provides a summary of the simulation parameters.

The most interesting results were achieved from evaluating the cost per edge cost (n_{uv}) of end-to-end probing strategies, and the effect of probe strategy on total cost, (C). Figure 5 shows the skewed nature of per edge cost resulting from end-to-end probing of hierarchical networks. Plots for hop-by-hop probing are not represented since the $n_{uv} \leq 1$ for all edges. Similar skewing was exhibited, regardless of the topology generator, number of nodes or number of Tracers. As expected, skewing becomes more pronounced as Tracers move toward the fringes of the network.

To illustrate effect of probe strategy on C, we tested the incremental effect of going from a pure end-to-end probing strategy to a pure hop-by-hop strategy. Specifically, we started with $T'=T$ and calculated the C for the generated topology. We then added a single node $s \in S$ to T', allowed hop-by-hop probing amongst adjacent nodes, and recalculated C. For each of the topologies we continued this procedure until $T'=T \cup S$, and full hop-by-hop probing was enabled. Figure 6 depicts the effect of this progression from end-to-end, to hybrid, to hop-by-hop probing using the same topologies represented in Figure 5. The effects with MAX placement are significant, but not as dramatic as with MIN because the path lengths between transit nodes is relatively short.

The results clearly indicate the benefits of hop-by-hop probing strategy on both scalability and total network overhead.

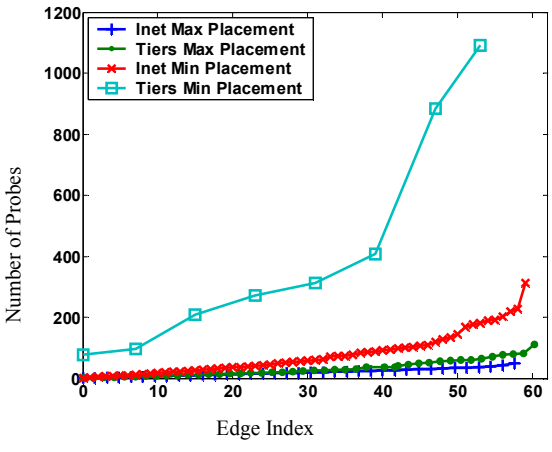


Figure 5: Effect of topology and placement strategy on const per edge (n_{uv}) with only 3200 nodes, 40 Tracers.

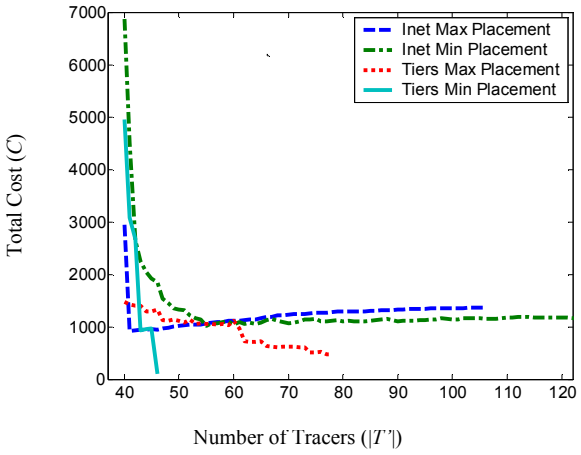


Figure 6: Effect of probe strategy on Total Cost (C) with 3200 nodes and 40 Tracers.

B. Accuracy

The timeliness of active measurement schemes is dependent on the probe interval. By reducing the total probe cost and the cost per edge, hop-by-hop techniques offer the ability to reduce the probe interval and thereby increase the timeliness of measurements. However, the accuracy of our service is also dependent on the ability to reliably reconstruct an end-to-end path metric from hop-by-hop measurements. In Section III.C we described both active and passive schemes to accomplish this reconstruction. In the following sections, we use empirical data to evaluate these schemes.

Traceroute Asymmetry: We are interested in knowing whether collecting the path information, either by `traceroute` or BGP's AS path, in a single direction will suffice in predicting the path in either direction. Asymmetry, the property of having Path(u,v)≠Path(u,v) for any $u,v \in \mathcal{V}$, is key to assessing both the passive and active path monitoring proposed in Section III. In [18], Paxson used `traceroute` probing to show that nearly half of all Internet paths included a major asymmetry. That is, about 20% of the end-to-end paths differed in two or more of the cities visited and about 30% differed in the AS's visited. Because Paxson's study was completed on data collected in 1995, we needed to re-assess the asymmetry of Internet paths.

To assess asymmetry in the Internet, we analyzed a dataset, $\mathcal{D}1$, originally collected for a stationarity study [18]. $\mathcal{D}1$ was collected using a pool of 189 public `traceroute` servers, of which one third were located in the United States, and the remaining two thirds were spread across 31 different countries. Each measurement consisted of a random pairing of two hosts with a `traceroute` command initiated in both directions.

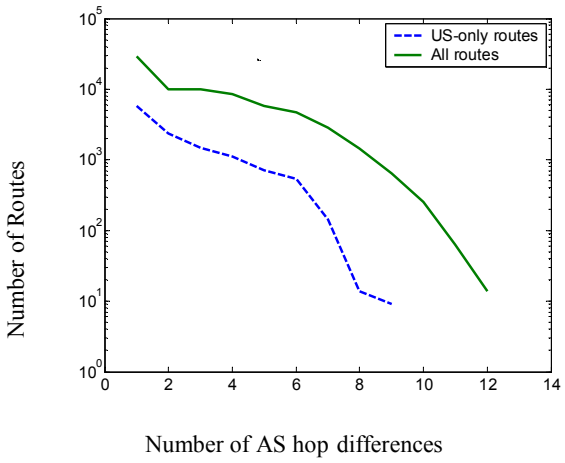
Measurements were made at Poisson intervals with a mean of 10 minutes between measurements initiated by the same host. Of the 220551 measurements made, 11% failed and 4% were incomplete due to the server failing before it delivered its results. As a result, we were able to analyze 72863 successfully paired routes, of which 11755 represented unique, end-to-end probes. Table II summarizes the characteristics of $\mathcal{D}1$.

Dataset $\mathcal{D}1$	
Total Nodes	189
Total Routes	220,551
Date Collected	12/1999-01/2000
Unique AS's Encountered	366
Avg AS Path Length	6.47
StdDev	2.33
Avg Router Path Length	16.45
StdDev	4.39

Table II: Summary of $\mathcal{D}1$ dataset.

Router level analysis of asymmetry requires the ability to identify routers with multiple interfaces so they can be treated as a single node. Although this association can be estimated by sending ICMP echo packets to interfaces, we estimate attempting to do so at this time would not be accurate since 18 months had passed since the data was originally collected. For this reason, our asymmetry analysis was limited to the AS path asymmetry, which provides a bound on the router layer asymmetry.

Figure 7: Routes per AS hop



We found that AS layer asymmetry had increased from 30% in 1995 to 60% in 2000. While Paxson's 1995 study had two thirds of the sites in the United States, as opposed to *D1*'s one third, the increase is still substantial. To ensure the difference was not due solely to the type of routes sampled, we calculated the asymmetry of the 12085 route pairs where both source and target in the United States; it was 53%. Additionally, for a full 81% of the asymmetric US-only routes, and 74% of all asymmetric routes, the AS path was different from the first hop.

However, as Figure 7 illustrates, the number of AS hop differences per route was generally small – 67% of the routes had 2 or fewer AS hop differences. To assess the feasibility of estimating reverse paths from based on forward `traceroute` probes, we are interested in the misprediction rate of our method. We were not able to calculate the misprediction rate for router level paths, for the same reason we were not able to assess router level asymmetry. However, we are able to calculate the AS path misprediction rate. We estimate the AS path misprediction rate by assigning a binary metric to each hop. The metric for any given edge is set to 0 if the edge is undesirable due to being either congested or otherwise unavailable. A value of 1 is assigned otherwise. Likewise, the path metric is assigned a value of 0 if any one of the edges comprising the path has a metric value of 0. The

Dataset *D2*	
Total Nodes	92
Total Routes	8372
Date Collected	11/2001
Avg AS Path Length	4.22
StdDev	1.77
Avg Router Path Length	11.57
StdDev	4.64

Table III: Summary of D2 dataset.

path metric is assigned a value of 1 only if all edges comprising the path have a metric of 1. The misprediction rate is then the sum of the probability of mispredicting the desirable (or undesirable) state of the path due one or more false positive, or false negative, nodes. For the $\mathcal{D}1$ dataset, the misprediction rate was a very encouraging 3.1%.

BGP AS path Asymmetry: Recall in Section III, we proposed both active `traceroute` probing and passive BGP monitoring to alleviate the need to represent inter-AS policy in our Link State-based mapping. To assess accuracy of AS path prediction based on BGP AS paths, we collected a second dataset, $\mathcal{D}2$. The $\mathcal{D}2$ dataset was collected using public Looking Glass [21] sites, allow `traceroute` commands to specified IP addresses, and querying of the local BGP router for the AS path associated with an IP address.

We attempted to find a mix of Looking Glass sites to match the one third to two thirds ratio of US to non-US sites of dataset $\mathcal{D}1$. The goal of maintaining this ratio was to have the datasets reflect similar properties. We were able to find 92 sites in which both the BGP query facility and the `traceroute` facility were active and met criteria for US to non-US ratio. However, a comparison of Table II and Table III illustrates measurable differences in the mean number of AS and router hops per route between $\mathcal{D}1$ and $\mathcal{D}2$. Based on an informal scan of the measurements, we believe this difference is best attributed to the fact that many of the successful probes involved instrumentation of network service providers within one to three hops of a backbone network.

We completed 8372 measurements of unique routes. We base our analysis on the 2209 unique routes for which we were able to collect a forward and reverse BGP path, as well as a forward and reverse end-to-end traceroute probe. The remaining measurements were discarded due to failure of one or more BGP or traceroute requests at either end. Failures were due to intermediate domains administratively blocked for ICMP traffic, inability for the BGP router to resolve the target IP address, or inability of the `traceroute` probes to reach the requested target.

Figure 8 shows the number of routes per AS hop difference for the reverse traceroute probe, the BGP forward path, and the BGP reverse path. While the per AS hop differences are similar, there were very interesting results regarding the ability of the BGP AS path to predict the actual path taken. Specifically, the forward BGP AS path correctly predicted only 69% the routes taken by the forward traceroute probes.

However, the reverse BGP AS path correctly predicted 49% of the routes taken by the reverse traceroute probe – whereas the forward `traceroute` probes achieved only 47% accuracy. Note that our tests utilized only the route labeled "best" by the BGP router queried. We are currently investigating the use of AS path data from multiple routes, as well as multiple BGP routers.

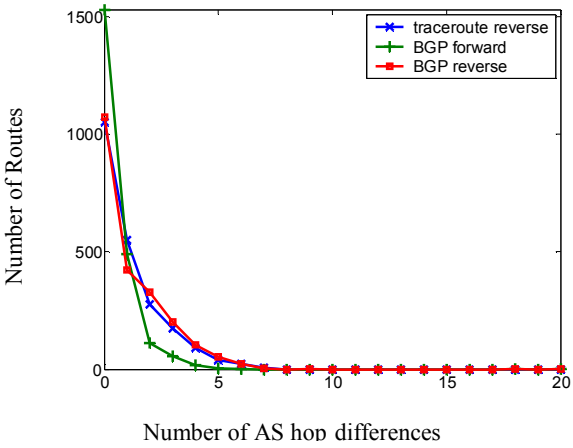


Figure 8: Routes per AS hop difference for BGP and traceroute monitoring

Summary: The `traceroute` and BGP data analyzed showed a significant degree of asymmetry in the Internet. However, because the number of hop differences was consistently low, we were able to achieve a 3.1% AS path misprediction rate by utilizing active probing. Further, the use of BGP for passive monitoring of forward and reverse paths appears promising.

V. RELATED WORK

Currently, application layer request routing is primarily accomplished through a number of ad hoc mechanisms. For example, round robin DNS [5] refers to the practice of returning a list of IP addresses for a single IP hostname so clients can be directed, in round robin fashion, to server replicas. Because this method results in skewed server loads and client response times, some implementers modify the authoritative DNS server to select a single IP address to associate with a hostname on a client-by-client basis. However, this effectively eliminates DNS caching and can increase client latency, as well as increasing the strain on an already stressed global DNS infrastructure [6].

Sonar [3] and HOPS [4] propose a single Internet-wide interface enabling clients to query the relative distances of server replicas. Our goal is to provide the infrastructure underlying services such as Sonar or Hops by collecting and disseminating network path metrics.

Several proposals have been made for active [7,8], passive [9] and hybrid [22] monitoring of network delays. Proposals employing active end-to-end techniques [7,8] focus on the relatively stationary properties of the network path. Our work differs by considering hop-by-hop techniques to enable representation of

transient, as well as the more stationary properties. While passive techniques, such as [9] outperform any active technique in terms of scalability, they are limited by the need to modify client software. Our work also differs from [23] which proposes a hybrid scheme which partitions the Internet approximately 250 into regions and probes inter-region metrics. Namely, we do not require aggregation or clustering to achieve scalability.

VI. CONCLUSIONS

The problem of resource and data location in distributed environments continues to be challenge. In this paper, we have examined the efficacy of exploiting Link State techniques in an effort to enable efficient, Internet-wide server selection. We have shown that a hop-by-hop probing strategy offers significant reductions in the total probe cost and cost per edge skewing over existing active, end-to-end probe techniques. We have provided algorithms which enable construction of end-to-end path metrics from the combination of hop-by-hop metrics, the AS path, and Internet routing heuristics. Our initial analysis of supplementing hop-by-hop probing with passive AS path monitoring demonstrates promise for the ability to enable new application layer routing services, including ability to filter extraneous topology data, to incorporate multiple metrics in the selection process, and the ability to effect policy-based application layer routing. Overall, this study has provided positive results regarding the use of Link State principles to represent the transient and stationary metrics useful for wide-area server selection.

REFERENCES

1. S. Bhattacharjee, Z. Fei, "A Novel Server Selection Technique for Improving the Response Time of a Replicated Service," *Proceedings of IEEE Infocom*, March/April 1998.
2. M. Day, B.Cain, G.Tomlinson. *A Model for CDN Peering*, Work in Progress, draft-day-cdnp-scenarios-05.txt, November 2000.
3. K. Moore, J. Cox, S.Green, "Sonar – a network proximity service," Internet-Draft, http://www.netlib.org/utk/projects/sonar, February 1996.
4. P. Francis, "Host Proximity Service (HOPS)," URL: http://www.ingrid.org/hops, July 1998.
5. P. Mockapetris, "RFC 1034: Domain Names – Concepts and Facilities," November 1987, ftp://ftp.isi.edu/in-notes/rfc1034.txt
6. A. Shaikh, R. Tewari, M. Agarwal. "On the Effectiveness of DNS-based Server Selection, *Proceedings of IEEE Infocomm*, 2001.
7. P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang, "IDMaps: A Global Internet Host Distance Estimation Service," *IEEE/ACM Transactions on Networking*, October 2001.
8. M. Crovella and R. Carter, "Dynamic Server Selection in the Internet," *Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95)*, August 1995.
9. S. Seshan, M. Stemm, R. Katz, "SPAND: Shared Passive Network Performance Discovery," *USENIX Symposium on Internet Technologies and Systems*, December 1997.
10. K. Calvert, M. Doar, and E. W. Zegura, "Modeling internet topology, " *IEEE Communications Magazine*, June 1997.
11. C. Jin, Q.Chen, S.Jamin, "Inet: Internet Topology Generator," University of Michigan Technical Report, CSE-TR-433-00, September, 2000.
12. R. Perlman. *Interconnections: Bridges, Routers, Switches and Internetworking Protocols*, Addison-Wesley, 2000.
13. E. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1:269-271, 1959.
14. Telstra Internet Network Performance Reports, "BGP Table Statistics", http://www.telstra.net/ops/bgp/index.html, October 2001.
15. H.Tangmunarunkit, R. Govindan, S. Shenker, D. Estrin. "The Impact of Routing Policy on Internet Paths," *Proceedings IEEE Infocom*, 2001.
16. V. Jacobson, Traceroute software, ftp://ftp.ee.lbl.gov/traceroute.tar.gz, 1989.
17. L. Subramanian, V. Padmanabhan, R. Katz. "Geographic Properties of Internet Routing: Analysis and Implications," University of California at Berkeley, Berkeley, California, Technical Report MSR-TR-2001-89, September 2001.
18. V. Paxson, "End-to-End Routing Behavior in the Internet," *IEEE/ACM Transactions on Networking*, Vol.5, No.5, pp.601-615, October 1997.
19. Y. Zhang, V. Paxson, S. Shenker, "The Stationarity of Internet Path Properties: Routing, Loss, and Throughput," ACIRI, Palo Alto, California, Technical Report, May 2000.
20. S. Jamin, C. Jin, Y. Jin, D. Raz, Y.Shavitt, L.Zhang. "On the Placement of Internet Instrumentation," *Proceedings of IEEE INFOCOM*, March 2000.
21. T. Kernen, Traceroute Organization, http://www.traceroute.org.
22. A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal and O. Spatscheck, "CDN Brokering," *Web Caching and Content Distribution Workshop*, June 2001.