iPlane: An Information Plane for Distributed Services

Harsha V. Madhyastha* Tomas Isdal* Michael Piatek* Colin Dixon*
Thomas Anderson* Arvind Krishnamurthy* Arun Venkataramani†

Abstract— **In this paper, we present the design, implementation, and evaluation of the *iPlane*, a scalable service providing accurate predictions of Internet path performance for emerging overlay services. Unlike the more common black box latency prediction techniques in use today, the *iPlane* builds an explanatory model of the Internet. We predict end-to-end performance by composing measured performance of segments of known Internet paths. This method allows us to accurately and efficiently predict latency, bandwidth, capacity and loss rates between arbitrary Internet hosts. We demonstrate the feasibility and utility of the *iPlane* service by applying it to several representative overlay services in use today: content distribution, swarming peer-to-peer filesharing, and voice-over-IP. In each case, we observe that using *iPlane*'s predictions leads to a significant improvement in end user performance.**

1 Introduction

The Internet by design is opaque to its applications, providing best effort packet delivery with little to no information about the likely performance or reliability characteristics of different paths. While this is a reasonable design for simple client-server applications, many emerging large-scale distributed services depend on richer information about the state of the network. For example, content distribution networks like Akamai, Coral, and CoDeeN [1, 19, 59] have the ability to redirect each client to the replica providing the best performance for that client. Likewise, voice over IP systems such as Skype [53] use relay nodes to bridge hosts behind NAT/firewalls; the selection of these relay nodes can dramatically affect call quality. Peer-to-peer file distribution, overlay multicast, distributed hash tables, and many other overlay services [11, 33, 8, 2] can benefit from peer selection based on different metrics of network performance such as latency, available bandwidth, loss rate, and reliability. Finally, the Internet itself can benefit from more information about itself, e.g., ISPs can monitor the global state of the Internet for reachability and root cause analysis, routing instability, onset of DDoS attacks, etc.

If Internet performance was easily predictable, its opaqueness might be an acceptable state of affairs. However, Internet behavior is well-known to be fickle, with local hot spots, transient (and partial) disconnectivity, and triangle inequality violations all being quite common [51, 2]. Many large scale services adapt to this state of affairs by building their own proprietary and application-specific information plane. Not only is this redundant, but it prevents new applications from leveraging information already gathered by other applications. The result is often sub-optimal. For example, most implementations of the file distribution tool BitTorrent choose peers at random (or at best using round trip latency estimates); since downloads are bandwidth-dependent, this can yield suboptimal download times. By some estimates [48], BitTorrent accounts for roughly a third of backbone traffic, so inefficiency at this scale is a serious concern. Moreover, implementing an information plane is often quite subtle; for example, large-scale probing of end-hosts can raise intrusion alarms in edge networks as the traffic resembles a DDoS. This is the most common source of complaints on PlanetLab [49].

To address this, several research efforts, such as IDMaps [18], GNP [42], Vivaldi [13], Meridian [4], and PlanetSeer [62] have investigated providing a common measurement infrastructure for distributed applications. These systems provide only a limited subset of the metrics of interest, most commonly latency between a pair of nodes, whereas most applications desire richer information such as loss rate and bandwidth. Second, by treating the Internet as a black box, most of these services abstract away network characteristics and atypical behavior—exactly the information of value for troubleshooting as well as improving performance. For example, the most common latency prediction methods use metric embeddings which are fundamentally incapable of predicting which paths violate the triangle inequality [51, 63]. More importantly, being agnostic to network structure, they cannot pinpoint failures, identify causes of poor performance, predict the effect of network topology changes, or assist applications with new functionality such as multipath routing.

In this paper, we move beyond mere latency prediction and develop a service to automatically infer sophisticated network behavior. We develop an *Information Plane* (*iPlane*) that continuously performs measure-

*Dept. of Computer Science, Univ. of Washington, Seattle
†Dept. of Computer Science, Univ. of Massachusetts, Amherst

ments to generate and maintain an annotated map of large portions of the Internet, containing a rich set of link and router attributes. The *iPlane* uses structural information such as the router-level topology and autonomous system (AS) topology to predict paths between arbitrary nodes in the Internet. The path predictions are combined with measured characteristics of path segments to predict end-to-end path properties for a number of metrics such as latency, available bandwidth, and loss rate. The *iPlane* can also analyze isolated anomalies or obtain a global view of network behavior by correlating observations from different parts of the Internet.

The *iPlane* is designed as a service that distributed applications can query to obtain information about network conditions. Deploying *iPlane* as a shared service (as opposed to providing a library) has several benefits. First, a common *iPlane* can exploit temporal and spatial locality of queries across applications to minimize redundant measurement overhead. Second, the *iPlane* can selectively refresh its knowledge of valid regions of the IP address space based on the real query workloads. More generally, the *iPlane* can assimilate measurements made on behalf of all of its clients as well as incorporate information reported by clients to develop a more comprehensive model of Internet behavior over time. We note that all of these arguments have been recognized before [10, 60, 27], however a convincing validation has remained starkly absent.

Our primary contribution is in demonstrating feasibility of the *iPlane*, e.g., we can infer an annotated map of the Internet once every three hours with a measurement load of a few Kbps. In addition, we develop:

- A common explanatory model for abstracting the network structure and path properties like bandwidth, loss rate etc.
- A measurement infrastructure that is deployed on every active PlanetLab site and almost a thousand traceroute and Looking Glass server vantage points (with a lower intensity of probing).
- A toolkit of passive and active measurement and tomographic techniques for inferring path properties.
- Case studies of popular systems such as CDNs, swarming peer-to-peer file distribution, and VoIP. We demonstrate measurable benefits of applying the *iPlane* to each of these types of applications.

The *iPlane* is a modest step towards the vision of a knowledge plane [10] pioneered by Clark et al. The *iPlane* supplies information about the network and leaves the task of adapting or repairing to the client. Nevertheless, the collection, analysis, and distribution of Internet-scale measurement information is itself a challenging systems engineering problem, and the focus of this paper.

2 Design

We start by discussing the requirements of an Information Plane for distributed services before presenting a design that meets the requirements.

- *Accuracy*: The *iPlane* should accurately estimate a rich set of performance metrics such as latency, loss-rate, capacity, and available bandwidth.
- *Wide coverage*: The *iPlane* should be able to measure and predict the performance of arbitrary Internet paths, not just those involving the overlay. Many currently deployed overlay services, such as RON [2] and S^3 [35], limit their focus to the connectivity of the overlay nodes only.
- *Scalability*: The measurement process should not impose an undue communication load on the measurement infrastructure.
- *Unobtrusiveness*: Active probes of end-hosts should be coordinated and performed in an unobtrusive manner in order to minimize the possibility of raising intrusion detection alarms.
- *Cooperation*: The *iPlane* should be able to systematically integrate measurements from clients to account for routing asymmetry and infer end-hop properties.

2.1 Overview

The *iPlane* is designed to be deployed as an application-level overlay network with the overlay nodes collectively coordinating the task of generating and maintaining an "atlas" of the Internet. The atlas is both extensive and detailed—it comprises the topology of the Internet core and the core's connectivity to representative targets in the edge networks, complete with a rich set of static attributes (such as link delay and link capacity), and recent observations of dynamic properties (such as routes between network elements, path loss rates, and path congestion). The *iPlane* uses systematic *active measurements* to determine the attributes of the core routers and the links connecting them. In addition, the system performs *opportunistic measurements* by observing actual data transfers to end-hosts, exposing characteristics of the edge of the network that typically cannot be obtained from one-way probing.

The *iPlane* can use its collected repository of observed paths to predict end-to-end paths between any pair of end-hosts at the edge of the network. This prediction is made by carefully composing partial segments of known Internet paths so as to exploit the similarity of Internet routes. As routes from two nearby sources tend to converge when heading to the same destination, the *iPlane* makes path predictions by splicing a short path

Technique	Description	Goal	Section
generate probe targets	Obtain prefixes from Routeview's BGP snapshot and cluster groups of prefixes with similar routes.	coverage, lower measurement load	Section 2.2.1
traceroutes from vantage points	PlanetLab nodes probe all targets, while Traceroute/Looking Glass servers issue probes to a small subset of the targets.	map topology, capture path diversity	Section 2.2.1
cluster network interfaces	Identify network interfaces that are geographically colocated.	lower measurement load, build structured topology	Section 2.2.2
frontier algorithm	Schedule measurements of link attributes to PlanetLab nodes such that each link is probed by the vantage point closest to it.	accurate measurements, lower measurement load, balance measurement load	Section 2.3.1
measure link attributes	PlanetLab nodes measure the loss rate, capacity, and available bandwidth over a subset of paths in the Internet core.	richly annotated topology	Section 2.3.2
opportunistic measurements	Passive measurements that expose the structure and performance of the edge networks.	minimize obtrusiveness	Section 2.4
route composition	Compose segments of observed or reported paths to predict end-to-end paths between a pair of nodes.	path prediction, performance prediction	Section 2.5

Table 1: A summary of techniques used in the *iPlane*.

segment from the source to an arbitrary destination with an observed path segment going towards the destination. Once a path is predicted, the *iPlane* simply composes the measured properties of the path segments to predict the performance of the composite path. For instance, if the *iPlane* has to make a latency prediction, it simply adds the latencies associated with the path segments to make the overall prediction.

The *iPlane* can integrate hitherto unknown clients into its atlas using any one of the following operational scenarios depending upon the desired balance between accuracy, measurement load, and client participation. The simplest form of integration is to approximate the client's performance by a representative target in the same edge network (which is a client IP with the longest prefix match), resulting in a scheme that requires no additional probes. A more refined integration model requires a small number of vantage points to perform probes to determine their paths to the new client and then add these paths to the atlas for future path predictions. If the client desires greater prediction accuracy, it can voluntarily perform some probes and contribute the paths that it has discovered to the *iPlane*; multi-homed clients could benefit from such an operational model.

The rest of this section describes the various techniques used to develop a functional *iPlane* that has wide coverage, incurs modest measurement load without sacrificing coverage or detail, and uses topology structuring techniques to enable efficient measurement and accurate inference. The techniques are summarized in Table 1.

2.2 Mapping the Internet Topology

The *iPlane* requires geographically distributed vantage points to map the Internet topology and obtain a collection of observed paths. PlanetLab servers that are located at over 300 sites around the world currently serve as vantage points. We could also enlist the use of public Looking Glass/Traceroute servers, as long as they are used for low-intensity probing. Our primary tool for determining the Internet topology is `traceroute`, which allows us to identify the network interfaces on the forward path from the probing entity to the destination. We do however need to address the issue of what destinations are to be probed and how to convert the raw output of `traceroute` to a structured topology.

2.2.1 Probe Target Selection

BGP snapshots, such as those collected by RouteViews [41], are a good source of probe targets. The *iPlane* could achieve wide coverage for the topology mapping process by obtaining the list of all globally routable prefixes in BGP snapshots, and choosing within each prefix a target *.1* address that responds to either ICMP or UDP probes. However, such a strategy incurs high measurement load. Instead, the *iPlane* prunes this list by clustering prefixes into BGP atoms [5]. A BGP atom is a set of prefixes, each of which has the same AS path route to it from any given vantage point; in other words, each vantage point has the same AS path route to all of the prefixes within the atom. BGP atoms represent the knee of the curve with respect to measurement efficiency—probing within an atom might find useful data, but it is less likely to do so.

The *iPlane* uses the PlanetLab nodes to perform exhaustive and periodic probing of the representative targets. (In Section 2.7, we discuss optimizations that lower the measurement load once the initial topology has been discovered.) In addition, the *iPlane* schedules probes from public traceroute servers to a small random set of BGP atoms, typically making a few tens of measurements during the course of a day. The public traceroute servers serve as a valuable source of information

regarding local routing policies. Note that in the long run, a functioning *iPlane* may actually serve to decrease the load on the public traceroute servers as the *iPlane*, rather than the traceroute servers themselves, could be consulted for information on the Internet topology.

2.2.2 Clustering of Interfaces

Once the basic topology has been identified, the *iPlane* proceeds to measure a rich set of link attributes, such as loss rate and capacity. Unfortunately, the topology information gathered by traceroutes is at the granularity of network interfaces, and thus lacks the structuring required to enable efficient measurement and accurate prediction. First, the measured topology has redundant connectivity information, as two interfaces colocated on the same router would appear as distinct entities and will create the illusion of multiple links to be measured even though there is only a single physical link. Second, it also contains links that are not critical to performance prediction; links between routers belonging to the same AS in a PoP are typically low-latency and high capacity and need not be measured. Third, as we describe in Section 2.5, our path and performance prediction algorithms work by composing observed route segments from our measured topology of the Internet. Declaring two paths to have intersected only if we see the same network interface address on both is almost certainly too strict. Instead, we would like to say that two paths intersect even if they pass through different interfaces or routers as long as the interfaces have similar performance to end-hosts— operationally, are owned by the same AS and reside in the same PoP.

We therefore seek to partition all observed network interfaces into "clusters" and use this structured topology for more in-depth measurements and predictions. We define the clusters to include interfaces that are similar from a routing perspective, interfaces belonging to the same PoP, and interfaces within geographically nearby portions of the same AS; in other words, these are interfaces with similar routing and performance characteristics. Note that this clustering is performed on network interfaces in the Internet core, whereas the clustering of prefixes into BGP atoms was performed for end-host IPs. In fact, IP address-based clustering would be ineffective in the core as geographically distant interfaces are often assigned addresses in the same prefix.

First, the *iPlane* uses the Mercator [22] technique to identify interfaces that belong to the same router. UDP probes are sent to a high-numbered port on every router interface observed in traceroutes. Interfaces that returned responses with the same source address are declared as aliases.

Second, the *iPlane* determines the DNS names assigned to as many network interfaces as possible. It then uses Rocketfuel's `undns` utility [54] to determine the locations of these interfaces based on their DNS names. This step alone does not suffice for our purpose of clustering geographically co-located interfaces because: 1) several interfaces do not have a DNS name assigned to them, 2) undns does not have rules for inferring the locations of all DNS names, and 3) undns infers incorrect locations for some interfaces that have been misnamed.

So, thirdly, to cluster interfaces for which undns did not yield a location and to determine incorrectly inferred locations, we develop an automated algorithm that clusters interfaces based on responses received from them when probed from a large number of vantage points. We probe all interfaces from all of the *iPlane*'s PlanetLab vantage points using ICMP ECHO probes. We use the TTL value in the response to estimate the number of hops on the reverse path back from every router to each of our vantage points. Our hypothesis is that routers in the same AS that are geographically nearby will have almost identical routing table entries and hence, take similar reverse paths back to each vantage point.

To translate this hypothesis into a systematic algorithm for clustering, each interface is associated with a *reverse path length vector*. This is a vector with as many components as the number of vantage points, and the i^{th} component is the length of the reverse path from the interface back to the i^{th} vantage point. We define the cluster distance between two vectors to be the L1 distance—the sum of the absolute differences between corresponding components, divided by the number of components. In our measurements, we have observed that the cluster distance between reverse path length vectors of co-located routers in an AS is normally less than 2.

Based on the metric discussed above, we can now present a technique for validating undns results and also attributing locations to interfaces without DNS names. We start by initializing our clusters to contain those interfaces for which undns gave us a location. Interfaces that have been determined to be co-located in an AS are in the same cluster. We then determine interfaces for which undns has inferred an incorrect location. For each cluster, we compute the median reverse path length vector, whose i^{th} component is the median of the i^{th} components of the vectors corresponding to all interfaces in the cluster. The location of every interface that belongs to some cluster and has a cluster distance greater than 2 from the median vector for its cluster is invalidated. Now, we cluster all interfaces that do not belong to any cluster. For each interface, we determine the cluster in the same AS as the interface, with whose median vector the interface's vector has the least cluster distance. If this minimum cluster distance is less than 2, the interface it added to the chosen cluster, otherwise a new cluster is created. This clustering algorithm, when exe-

cuted on a typical traceroute output, clusters over 300K interfaces into about 25K clusters.

2.3 Measuring the Internet Core

After clustering, the *iPlane* can operate on a compact routing topology, where each *node* in the topology is a cluster of interfaces and each *link* connects two clusters. The *iPlane* then seeks to determine a variety of link attributes that can be used to predict path performance. To achieve this goal, a centralized agent is used to distribute the measurement tasks such that each vantage point of the *iPlane* is assigned to repeatedly measure only a subset of the links. The centralized agent uses the compact routing topology to determine the assignments of measurement tasks to vantage points, communicates the assignment, and monitors the execution of the tasks. Since the probes for these carefully constructed measurement activities might incur modest loads, only the *iPlane* infrastructure nodes (namely, the PlanetLab nodes) are used for these tasks.

2.3.1 Orchestrating the Measurement Tasks

There are three objectives to be satisfied in assigning measurement tasks to vantage points. First, we want to minimize the measurement load by requiring that each link attribute be measured by exactly one vantage point. Second, the measurement should be load-balanced across all vantage points, *i.e.*, each vantage point should perform a similar number of measurements. Third, in order to measure the properties of each link as accurately as possible, every link in the topology should be measured from the vantage point that is closest to it and is also capable of routing probes to the link; it is of course not meaningful to assign a link measurement to some vantage point that is incapable of generating valid Internet routes through it.

We have developed a novel "frontier" algorithm to perform the assignment of tasks to vantage points. The algorithm works by growing a frontier rooted at each vantage point in an uniform manner and having each vantage point measure only those links that are at its frontier. The centralized agent performs a Breadth-First-Search (BFS) over the measured topology in parallel from each of the vantage points. The agent goes through all the vantage points iteratively. Whenever a vantage point is taken up for consideration, the algorithm performs a single step of the BFS by following one of the traceroute paths originating at the vantage point. If it encounters a link whose measurement task has been assigned already to another vantage point, it continues the BFS exploration for the vantage point. When the BFS exploration encounters a new link that has not been seen before, it assigns the task of measuring the new link's attributes to the vantage point under consideration. This process continues until all the link measurements have been assigned to some vantage point in the system.

The centralized agent uses the above algorithm to determine the assignment of tasks, and then ships them to the respective vantage points. Each target link is identified by the traceroute path that the vantage point could use to reach the link and by its position within the traceroute path. If a vantage point is no longer capable of routing to the link due to route changes, the vantage point reports this to the centralized agent, who in turn reassigns the task to a different vantage point. If the routes have not changed, the vantage point probes the assigned link in order to measure various properties such as loss rate, bottleneck capacity and available bandwidth.

Most link attributes, however, cannot be directly determined by the vantage point. For instance, while measuring loss-rates, the vantage point can only measure the overall loss rate associated with the entire path from the vantage point to the target link; the individual link loss-rate would have to be inferred as a post-processing operation that is essentially a rerun of the frontier algorithm. Once all vantage points report their measurements back to the centralized agent, the agent can perform the BFS style exploration of the topology to infer link properties in the correct order. For instance, assume that a vantage point v had probed the path $v, \ldots, x, y$ and had obtained a (one-way) loss rate measurement of $l_{v,y}$ for the entire path. The centralized agent can then infer the loss rate along the link (x, y) after inferring the loss rates for each of the links in $v, \ldots, x$, composing these individual loss rates to predict the loss rate $l_{v,x}$ along the segment $v \ldots x$, and then calculating the loss rate for (x, y) using the equation $(1 - l_{v,y}) = (1 - l_{v,x}) \cdot (1 - l_{x,y})$. Since the link property inference is performed as a BFS traversal, we are guaranteed that all of the loss rates for the links along $v, \ldots, x$ have been inferred before we consider the link (x, y).

2.3.2 Measurement of Link Attributes

We next outline the details of the loss rate, bottleneck capacity and available bandwidth measurements performed from each vantage point. We note that many previous research efforts have proposed specific ways to measure each of these properties, but no widespread and continuous measurement has been attempted to date.

Loss Rate Measurements: We perform loss rate measurements along path segments from vantage points to routers in the core by sending out probes and determining the fraction of probes for which we get responses. Ideally, we would like to distinguish a probe loss from a lost response since the reverse path, in general, could be different from the forward path and is unknown. We experimented with the Tulip tool [40] that allows one to distinguish a forward path loss from a

reverse path loss by examining the IP-ID fields in responses to three back-to-back UDP probes.[1] Unfortunately, we determined that only 77% of Internet routers respond to UDP probes and only 26% respond to back-to-back UDP probes. Even if the UDP probes are replaced by ICMP probes, one cannot distinguish a lost probe from a lost response as the IP-ID counter value is rarely communicated in ICMP probe replies. We therefore settled on the simple method of sending TTL-limited singleton ICMP probes with a 1000-byte payload. When the probe's TTL value expires at the target router, it responds with a ICMP error message, typically with a small payload. When a response is not received, one cannot determine whether the probe or the response was lost, but there is some evidence from previous studies that the small packets tend to be preserved even when routers are congested [40]. We therefore currently attribute all of the packet loss to the forward path; the development of more accurate techniques is part of ongoing work.

Capacity Measurements: We perform capacity measurements using algorithms initially proposed by Bellovin [3] and Jacobson [28] that vary the packet size and determines the delay induced by increased packet sizes. For each packet size, a number of probes (typically, thirty to forty) of that size is sent to an intermediate router and the minimum observed round-trip time is noted. The minimum observed round-trip time corresponds to the scenario where the probe didn't observe any queueing delays. By performing this experiment for different packet sizes, one can determine the increased transmission cost per byte over a sequence of links. When this experiment is performed for a sequence of network links in succession, the capacity of each link could be determined. We implement this measurement technique as opposed to other techniques that observe the induced dispersion of back-to-back packets [31, 34], since such techniques report only the capacity of the bottleneck link along a segment but not the capacity of non-bottleneck links. Note that our capacity measurements may underestimate a cluster link if it consists of multiple parallel physical links, unless the router is configured to stripe packets across the parallel links.

Available Bandwidth Measurements: Once we have link capacities, we can probe for available bandwidth along path segments using packet dispersion techniques such as Spruce [55], IGI [25], Pathload [29]. A simple measurement is performed by sending a few, equally spaced, short probes at the believed capacity, and then measuring how much delay they induce. The slope of the delay increase will indicate how much background traffic arrived during the same time period as the probe. For instance, if the probes are generated with a gap of Δ_{in} through a path segment of capacity C and if the measured gap between between the probe replies is Δ_{out}, one can estimate the available bandwidth as $C \cdot (1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{in}})$. An important detail is that the packets have to be scheduled at the desired spacing, or else the measurement is not valid. Fortunately, even on heavily loaded PlanetLab nodes, it is possible to realize the desired scheduling most of the time.

[1] The IP-ID value typically corresponds to a router-specific counter that is incremented after each probe response.

2.4 Opportunistic Edge Measurements

To provide a comprehensive data set on which to infer current properties of paths to end-hosts, it is necessary for the *iPlane* to maintain an up-to-date map of the network that extends to the very edge. However, the measurement techniques outlined above are unlikely to work as active measurements do not elicit responses from most end-hosts. Also, measurements to end-hosts are frequently misinterpreted by intrusion detection systems as attacks. Hence, we pursue an opportunistic approach to data collection—measuring paths to end-hosts while interacting with them over normal connections. We participate in the popular file-distribution application BitTorrent [11] and gather measurements from our exchanges with the peers in this swarming system.

BitTorrent is used daily by thousands of end users to distribute large files. BitTorrent is one example of a large class of *swarming* data distribution tools, which have recently received much attention in the research community [33, 47, 8]. By participating in several BitTorrent swarms, we have the opportunity to interact with a large pool of end-hosts. We measure properties of the paths to peers while exchanging data with them as part of the swarming system.

We gather two kinds of measurements as part of our opportunistic measurement infrastructure.

- Packet traces of TCP flows to end-hosts. These traces provide information about packet inter-arrival times, loss rates, TCP retransmissions and round trip times. We use the inter-arrival times between data packets to measure bottleneck bandwidth capacities, as described further in Section 3.
- Traceroutes to end-hosts. When a peer requests a connection with our measurement node, we conduct a traceroute to that host. We record this data and add it to our atlas.

2.5 Performance Prediction

Next, we describe how to predict path properties between an arbitrary pair of nodes based on the above measurements. The prediction proceeds in two steps. First, we predict the forward and reverse paths connecting the two nodes. Second, we aggregate measured link-level properties to predict end-to-end path properties.

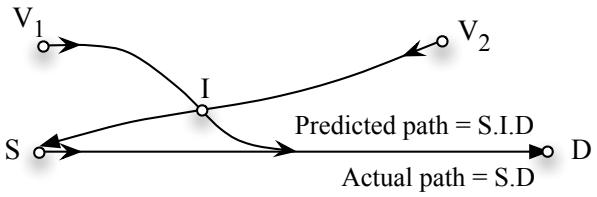

Figure 1: The path from S to D is obtained by composing a path going into S with a path going into D via an intersection point I close to S.

Path Prediction We use a technique developed by Madhyastha et al [38] based on composing observed path segments to predict unknown paths. Consider a source S and destination D. If S is a vantage point, then we simply return the measured path from S to D. Else, we determine an appropriate intersection point I in the measured subgraph of the Internet such that—(a) the AS hop count of the path $S.I.D$ is minimum, and (b) the latency from S to I is minimum, in that order (refer Figure 1). The underlying principle is similarity of routes, i.e., with a sufficiently large number of vantage points, the path to a destination (D) from any node (S) will be similar to the path from a vantage point or router (I) located nearby. Condition (a) encodes the default path selection criterion used by BGP in the absence of conflicting local preference policies. Furthermore, the extent of error introduced by policy routing is confined to a small path segment $S.I$ by virtue of condition (b). Note that the above technique is guaranteed to return a path. Every path of the form $S.V.D$, for each vantage point V, belongs to the measured subgraph. The caveat is that some of these predicted paths may be highly inflated due to the asymmetric nature of Internet routing. Madhyastha et al [38] note that adding a small number of measured paths originating from S significantly improves the prediction accuracy. In the proposed *iPlane* architecture, any client interested in greater prediction accuracy can contribute traceroute measurements. These measurements will then be used only for the purpose of answering queries issued by the contributing client, thereby limiting the risk of polluting the *iPlane* database with measurements from untrustworthy clients.

Path Properties Given predicted paths as above, we can estimate end-to-end properties by aggregating link-level properties. For example, we can estimate the one-way delay from S to D as the sum of measured link latencies along the predicted forward path. Round-trip delays are estimated as the sum of the delays along the forward and reverse paths. The loss rate on a path is estimated as the probability of a loss on any its constituent links while bandwidth is the minimum value across the links. We predict TCP throughput and total transfer time using widely accepted loss-rate-based models by Padhye et al [45] and Cardwell et al [7]. Recently, He et al [24] argued that the only way to accurately predict TCP throughput is to send TCP flows and use history-based predictors. Although we have not implemented these, our use of passive BitTorrent logs is amenable to incorporating such predictors. In practice, we do not need fine-grained link-level information to predict end-to-end properties. It suffices to know these values for the path segments used in path composition.

2.6 Query Interface

In this work we focus on techniques that enable inference of a rich set of structural, routing, and path performance properties of the Internet. However, a practical *iPlane* service also requires a careful design of the interface exposed to end users and ISPs keeping typical usage scenarios in mind. In this section, we consider design strategies for a query interface that will meet these goals.

The *iPlane* will export three kinds of interfaces:

1. *On-the-fly Queries*: The *iPlane* will enable applications to issue sophisticated queries over the annotated Internet graph on the fly. For example, (i) *get_detour(a, b, metric, relay_nodes[])* returns the best relay node to detour traffic from A to B that optimizes *metric* such as latency, loss rate, bottleneck capacity, TCP bandwidth etc, (ii) *select_peer(a, metric, peer_nodes[])* selects the peer closest to a node a with respect to *metric*, (iii) *select_peer(.)* or *get_detour(.)* may also specify a set of metrics seeking to optimize one and bounding the others. Such queries can be issued either by node a above, or by an overlay service provider (e.g., a BitTorrent tracker) to connect A to a good peer.
2. *Download the Internet*: A user will be able to obtain the annotated graph data structure of the entire Internet or specific popular regions. For example, a commercial CDN operating in the US may wish to analyze the Internet graph in North America to determine where to locate its caches.
3. *Network Newspaper*: The *iPlane* will also support a publish-subscribe interface that allows users to register for information updates about specific portions of the Internet graph. This interface is most useful to allow users to subscribe to their "view" of the Internet, i.e., all paths originating from a user to all BGP atoms, or insert triggers to be notified of specific events, e.g., when a critical link fails.

The on-the-fly query model has some limitations. First, each on-the-fly query incurs at least a round-trip time of querying latency and may not be appropriate for latency-sensitive applications. Second, handling Internet-scale query traffic especially during periods of peak load essentially requires re-engineering a system

as scalable and robust as DNS, but one that processes far more sophisticated queries—a formidable prospect. More realistically, we expect users to "download the Internet" at coarse-grained time-intervals and subscribe to updates to frequently accessed portions of the Internet graph.

We currently only support on-the-fly queries; the delicate task of defining a precise API is ongoing work.

2.7 Scalability

We now discuss the measurement load required to generate and maintain a frequently refreshed map of the Internet. First, we note that discovering the routing topology is crucial as all of our inference algorithms operate by predicting the path first. The BFS algorithm described in Section 2.3 needs as input traceroute probes from all vantage points to all address prefixes. However, we can reduce the running overhead of these traceroutes even further so that each link is probed mostly once. Thus, traceroutes from all vantage points to all prefixes are performed only once to initialize the system, while the running overhead of topology maintenance is small. The algorithm proceeds in two phases. In the first phase, the centralized agent optimistically assumes that no changes to the topology occurred since the previous phase and assigns to each vantage point v a set of path segments that it should probe using traceroutes; this assignment is obtained by simply running the frontier algorithm on the original traceroute graph. [2] All vantage points then report their measurements to the centralized agent. The agent then inspects these measurements to check if the routing topology is consistent, i.e., the paths from all vantage points to all prefixes has not changed since the previous phase. If topology changes have occurred, the agents computes a *difference set* of path segments that are assigned to each vantage point to complete the routing topology. The stationarity of Internet paths [61] suggests that the number of traceroutes in the second phase will be small.

With this optimization in place, we now quantify the measurement load associated with each technique in our measurement apparatus. The communication load is summarized in Table 2.7. Our main result is that the *iPlane* can produce an updated map of the Internet's routing topology with as little as 2Kbps of probe traffic per vantage point, and a map of link-level attributes with 5Kbps of probe traffic per vantage point, suggesting that the *iPlane* can refresh the Internet map once every 3 hours.

3 System Setup and Evaluation

In this section, we present details of our deployment of the *iPlane*. We provide an overview of the measurements we conducted as part of our deployment. We also outline the tests we conducted to validate various measurement techniques described in Section 2 and employed by the *iPlane*.

[2] Note that in the earlier discussions, the frontier algorithm was used on the compact cluster-level graph.

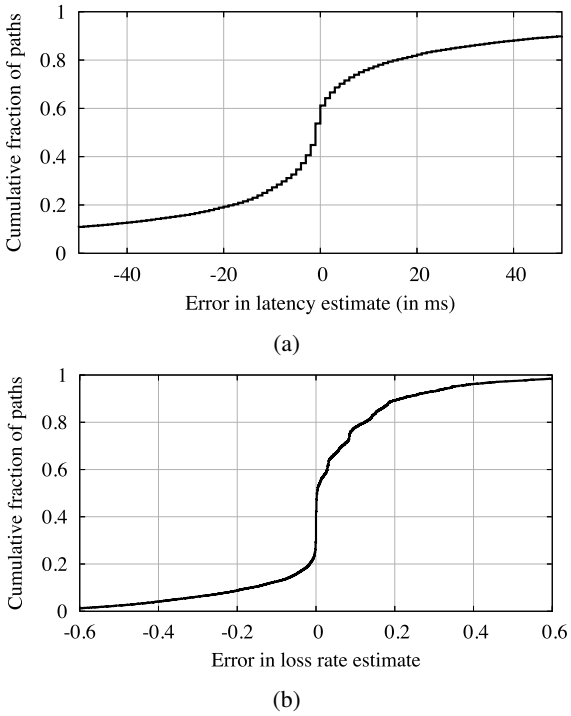

Figure 2: Distribution of errors in (a) latency, and (b) loss rate estimation.

3.1 Measuring the Core

We performed traceroutes from 147 PlanetLab nodes in distinct sites to map the topology of the core of the Internet. The targets for our traceroutes were .1 addresses in each of $47,983$ atoms determined from the RouteViews BGP snapshot. We probed all interfaces observed in our measured topology with UDP and ICMP probes, and clustered the interfaces based on their responses.

Once a map of the Internet's core was gathered, we employed our "frontier" BFS algorithm to determine paths to be probed from each of the 147 PlanetLab nodes. Each vantage point was assigned to measure only around 1500 paths. Loss rate, capacity, and available bandwidth were measured along each of the assigned paths. These measurements were then assembled to determine these properties for every cluster-level link that occurs in our measured topology.

To validate the predictions performed by the *iPlane* using the technique outlined in Section 2.5, we compared properties of paths between PlanetLab nodes with the corresponding values predicted by the *iPlane*. We considered paths between all pairs of PlanetLab nodes for validation, and measured the latency and loss rate along these paths. To predict the latency or loss rate

Measurement Task	Tool/Technique	Frequency	Probing complexity/rate
Topology Mapping	*traceroute*	Infrequently (over a day)	1000 vantage points × 50K atoms — 5Kbps
Clustering	*mercator* for alias resolution, ICMP-ECHO probes for reverse TTLs	Infrequently (over a day)	100 vantage points × 300K interfaces — 2Kbps
Capacity measurements	"frontier" BFS algorithm applied to cluster-level topology for path assignment, *pathchar* for bandwidth capacity	Infrequently (over a day)	200 vantage points × 1500 paths — 3Kbps
Loss rate and available bandwidth measurements	"frontier" BFS algorithm for path assignment, *tulip* for loss rate, *spruce* for available bandwidth	Continuous (every 3 hours)	200 vantage points × 1500 paths — 5Kbps
Topology Update	"frontier" BFS algorithm applied to router-level topology	Continuous (every 3 hours)	200 vantage points × 60K links — 2Kbps

Table 2: Complexity of measurements techniques used in the *iPlane*.

between a pair of nodes, we assumed that we only had 10 traceroutes from either node to aid in our prediction. The remaining 145 nodes were considered to be our vantage points. Each such experiment is independent to ensure no mixing of the measurement and validation set. Figure 2 plots the difference of latency and loss rate estimates made by the *iPlane* as compared to the true values. For 62% and 66% of the paths, the *iPlane*'s latency estimates have error less than 20ms, and loss rate estimates have error less than 10%, respectively.

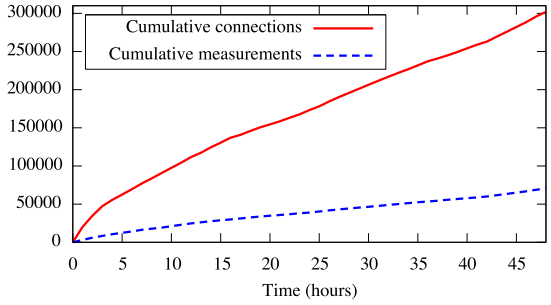

Figure 3: Cumulative opportunistic measurements obtained over a 48 hour period. Both the raw number of connections and those connections that produced actual measurements are shown.

3.2 Measurements to End Hosts

To measure the edges of the Internet, we deployed our modified BitTorrent client on 367 PlanetLab nodes. As described in Section 2.4, our infrastructure for measuring the edge also involves the millions of users who frequently participate in the BitTorrent filesharing application. Every hour, we crawl well-known public websites that provide links to several thousand `.torrent` files to put together a list of 120 popular swarms. The number of swarms for consideration was chosen so as to ensure the participation of several of our measurement vantage points in each swarm. The number of PlanetLab nodes designated to a swarm is proportional to the number of peers participating in it.

Each PlanetLab node runs a BitTorrent client that we have modified in several ways to aid in our measurements. First, the modified client does not write any data that it downloads onto disk. The client keeps only a small cache of recently received pieces in memory. Data in this cache is offered for upload so that we do not have to depend only on peers optimistically unchoking us to setup connections with them. Second, our client severs connections once we have exchanged $1MB$ of data, which suffices for purposes of our measurements. Finally, we introduce a *shadow tracker*—a database that coordinates measurements among all PlanetLab nodes participating in a single swarm. Instead of operating only on the set of peers returned by the original tracker for the swarm, our modified client also makes use of peers that it learns from the shadow tracker. Each client reports all peers it learns from the central tracker to the shadow tracker, and performs two queries on the shadow tracker for preferred measurement tasks. First, clients attempt to connect and exchange data with peers that have been observed but not yet measured. Second, clients then measure nodes that have been previously measured only from other vantage points. These modifications are crucial for measurement efficiency and diversity since typical BitTorrent trackers permit requesting only a restricted set (50–100) of participating peers once every 30 minutes or more. Such short lists are exhausted by our modified client quickly.

The rate at which we gather increasing measurements is summarized in Figure 3. During a 48 hour period, our measurement nodes have connected to 301, 595 distinct IPs. The number of unique IPs for which we gathered upload bandwidth capacity estimates were 70, 428. We connected to IPs in 3, 591 distinct ASs and 19, 639 distinct BGP prefixes. Our study spans end-hosts in 160 different countries.

The success of these measurements depends greatly on the popularity of the swarms in which we participate. As a result, our measurement rate fluctuates as popularity changes. These initial results show the promise of

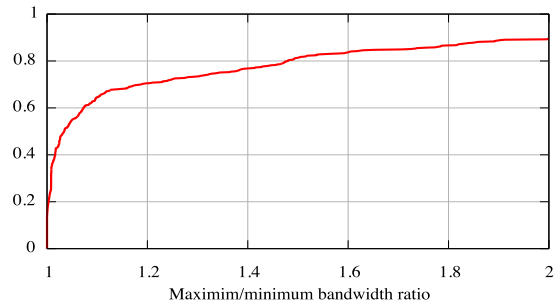

Figure 4: CDF of the ratio of maximum to minimum measured bandwidth capacity for /24 address blocks with multiple measurements.

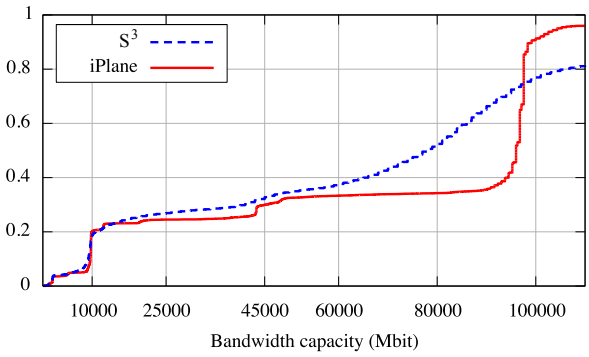

Figure 5: CDFs of estimated bandwidth capacity on paths between PlanetLab nodes as measured by *iPlane* and S^3.

opportunistic measurements obtained from peer-to-peer services, and are applicable even if services other than BitTorrent become more popular in the future.

3.3 Clustering of end-hosts

Although the data produced by our opportunistic strategy is extensive, it is by no means complete. We rely on clustering based on IP prefixes to generalize results for prefixes to which we have only a few measurements. In Figure 4, we explore the validity of this clustering assumption. For every /24 prefix in which we have measurements to multiple end-hosts from the same vantage point, we compute the ratio of the maximum to the minimum measured bandwidth capacity. For 70% of /24 prefixes, the capacities measured differ by less than 25%. During our 48 hour measurement period, we performed measurements to end-hosts in 61,294 /24 prefixes, which are representative of measurements to over 15 million end-hosts.

3.4 Validation of capacity measurements

Our edge bandwidth capacity measurement relies on inter-arrival times observed between data packets in the connections we maintain with BitTorrent peers. We implemented the multiQ [32] technique to infer end-to-end bottleneck bandwidth capacity from these inter-arrival

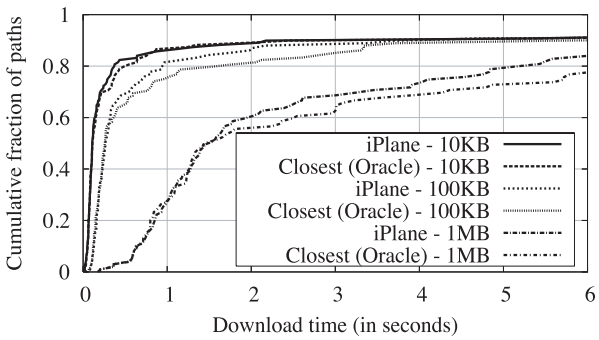

Figure 6: CDF of download times from replicas in the CDN chosen by the *iPlane* and from replicas closest in terms of latency. Each download time is the median of 5 measurements.

times. Although the performance of multiQ presented in previous studies is encouraging, with 85% of measurements based on data packets within 10% of the true bottleneck capacity, the unique properties of PlanetLab hosts motivated us to provide further validation. To verify that multiQ provides reasonable data in the presence of cross traffic, machines under heavy load and short TCP traces, we compared our measurements with those made by S^3 [15].

We setup a test torrent and had our measurement clients running on 357 PlanetLab nodes participate in this torrent. From this setup, we opportunistically measured the bottleneck bandwidth capacities between these PlanetLab nodes. The dataset we gathered from this experiment had 10,879 paths in common with measurements made by S^3 on the same day. Figure 5 compares the bandwidth capacities measured by the two methods. The measurements made by the *iPlane* closely match those of S^3 for capacities less than 10 Mbps. At higher bandwidth capacities, they are only roughly correlated. We attribute this difference to the use of user-level timestamps by Pathrate, the capacity measurement tool used by S^3. User-level timestamps for high capacity paths are likely to be inaccurate in the highly loaded PlanetLab environment. Our measurement setup makes use of kernel-level timestamps and is therefore less sensitive to high CPU load.

4 Application Case Studies

In this section, we show how popular applications can use the *iPlane*. We evaluate three distributed services for potential performance benefits from using the *iPlane*.

4.1 Content Distribution Network

Content distribution networks (CDNs) such as Akamai, Codeen and Coral [1, 59, 19] redirect clients to a nearby replica. The underlying assumption is that latency determines network performance.

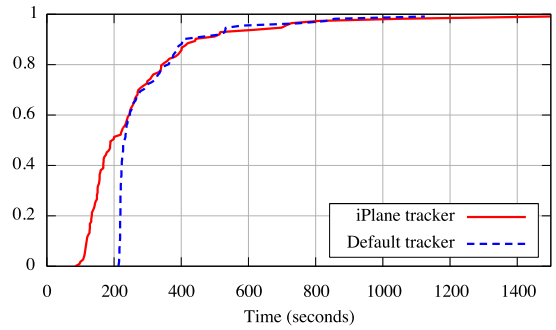

Figure 7: CDF of BitTorrent download completion times with and without informed peer selection at the tracker.

However, there is more to network performance than just latency. Models by Padhye et al [45] and Cardwell et al [7] indicate an inverse square root dependence on loss rate of TCP throughput and total transfer time. Even for small web documents, a loss of a SYN or a packet during slow start can considerably inflate transfer time. A CDN using the *iPlane* can track the latency, loss rate, and bottleneck capacity from each replica to the rest of the Internet. The CDN can then arrange for its name servers to redirect the client to optimize performance using the metric and model of its choice.

We evaluate potential performance gains that a CDN using the *iPlane* can provide to clients. We emulate a small CDN comprising 30 randomly chosen PlanetLab nodes. Each node serves 3 files of sizes 10KB, 100KB and 1MB. We use 141 other PlanetLab nodes to emulate clients. Each client downloads all 3 files from the replica that provides the best TCP throughput as predicted by the PFTK model [45] using the *iPlane*'s estimates of latency and loss rate. Each client also downloads the 3 files from the replica closest in terms of *actual* measured latency. Note that this comparison is against an optimum that cannot be achieved without extensive probing. A real CDN would only have estimated latencies available. Figure 6 compares the download times experienced by the clients in either case. Choosing the replica for optimized TCP throughput based on the *iPlane*'s predictions provides better performance than choosing the optimal replica in terms of latency. Though these results are only indicative, they suggest that CDNs could provide better performance by utilizing both latency and loss rate information exported by the *iPlane* instead of optimizing latency alone, as in OASIS [20] and other systems.

4.2 BitTorrent

We show how the *iPlane* can enable informed peer selection in popular file swarming systems like BitTorrent. In current implementations, a centralized BitTorrent *tracker* serves each client a random list of peers. Each client enforces a tit-for-tat bandwidth reciprocity mechanism that incents users to contribute more upload bandwidth to obtain faster downloads. However, the same mechanism also serves to optimize path selection at a local level—peers simply try uploading to many random peers and eventually settle on a set that maximizes their download rate. Because reasoning about peer quality occurs locally at each client, each client needs to keep a large pool of directly connected peers (60–100 for typical swarms) in order to achieve good performance even though at any time only a few of these (10–20) are actively engaged in data transfer with the client. This overhead is fundamental: with only local information, peers cannot reason about the value of neighbors without actively exchanging data with them. This lack of information is typical of popular peer-to-peer systems, and results in increased overhead and delayed convergence of path selection. Broadly, the *iPlane* enables a clean separation of the path selection policy from the incentive mechanism in peer-to-peer systems.

We built a modified tracker that uses the *iPlane* for informed peer selection. Instead of returning random peers, the tracker uses the *iPlane*'s loss rate and latency estimates to infer TCP throughput [45]. It then returns a set of peers to clients, half of which have high predicted TCP throughput and the rest are randomly selected. The random subset of peers are included to prevent the overlay from becoming disconnected (e.g., no North American node preferring a peer in Asia).

We used our modified tracker to coordinate the distribution of a 50 megabyte file over 117 PlanetLab nodes located in North America, Europe, and Asia. We measured the time taken by each of the peers to download the file after the seed was started. Figure 7 compares the download times observed against those of an unmodified tracker. Informed peer selection causes 50% of peers to have significantly better download times. Although preliminary, these performance numbers for BitTorrent are encouraging. We believe that better use of information from the *iPlane* could lead to even further improvements in performance. Our selection of 50% as the fraction of random peers was arbitrary, and we are currently investigating the tradeoff between robustness and performance in BitTorrent.

4.3 Voice Over IP

Voice over IP (VoIP) is a rapidly growing application that requires paths with low latency, loss and jitter for good performance. Several VoIP implementations such as Skype [53] require relay nodes to connect end-hosts behind NATs and firewalls. Choosing the right relay node is crucial to providing acceptable user-perceived performance. Reducing end-to-end latency is important since humans are sensitive to delays above a threshold. Low loss rates improves sound quality and reduces throughput consumed by compensating codecs. Metrics

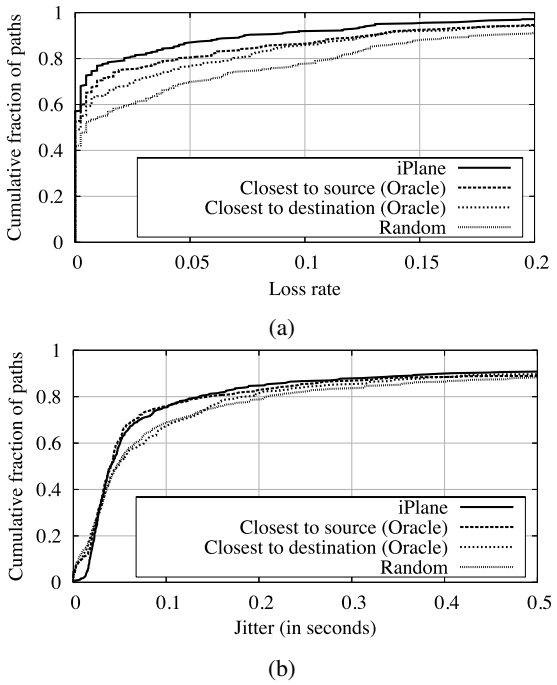

Figure 8: Comparison of (a) loss rate, and (b) jitter with and without use of the *iPlane* for end-to-end VoIP paths.

of user-perceived quality such as *mean opinion score* (MOS) [57] have been shown to be highly correlated with QoS models like the E-model [17] that depend on loss rate and end-to-end delay among other factors. Thus, VoIP applications can benefit from the *iPlane*'s predictions of latency and loss rate in choosing the best possible relay node.

To evaluate the *iPlane*'s ability to successfully pick good relay nodes, we emulated VoIP traffic patterns on PlanetLab. We considered 384 pairs of PlanetLab nodes, chosen at random, as being representative of end-hosts participating in a VoIP call. Between each pair, we emulated a call by sending a 10 KBps UDP packet stream via another PlanetLab node chosen as the relay node. We tried 4 different relay options for each pair chosen based on (i) the *iPlane*'s estimates of latency and loss rate, (ii) latency to the source, (iii) latency to the destination, and (iv) random choice. The *iPlane*-informed choice was obtained by first querying for the 10 relay options that minimize end-to-end loss and then, choosing the one that minimized end-to-end delay among these options. [3]

Each emulated call lasted for 60 seconds, and the end-to-end loss rate and latency were measured. Figure 8(a) shows that significantly lower loss rates were observed along relay paths chosen based on the *iPlane*'s predictions. Additionally, Figure 8(b) shows that the *iPlane* also helps to reduce jitter, which we computed as the standard deviation of end-to-end latency. These results demonstrate the potential for the use of the *iPlane* in VoIP applications. We are in the process of repeating this experiment running Skype coupled with the Oasis network stack [39] and employing the E-model to select and validate paths.

[3] The E-model uses a metric that is roughly $c_1.ln(1+c_2.L)+c_3.D$ for loss rate L and delay $D < 177.3$ms. We use the above simpler formula as we do not use compensating codecs.

5 Related Work

The *iPlane* bridges and builds upon ideas from network measurement, performance modeling, Internet tomography, and recent efforts towards building a knowledge plane for the Internet. We believe that an Internet-scale instantiation of the *iPlane* is greater than the sum of its parts, and relate individual contributions to prior work in the respective areas.

Information Plane Clark et al [10] pioneered the broad architectural vision of a knowledge plane to build large-scale, self-managing and self-diagnosing networks based on tools from AI and cognitive science. Several research efforts have since concretely addressed pieces of this problem.

Several existing systems focus primarily on the query processing engine of a network information plane. Sophia [60] is an information plane for large-scale networked systems such as PlanetLab that uses declarative programming to evaluate logical expressions in a distributed manner. PIER [27] presents a distributed query processing engine for monitoring networks and distributed databases. IrisNet [21] is an information plane for a "sensor Web" of end hosts such as PCs and Webcams and is based on a hierarchical data model using XML. SWORD [44], HP's OpenView [43] and IBM's Tivoli [58] are other such systems that focus on the query processing aspect of the system.

Furthermore, all of the above research as well as commercial systems manage information that represents nodes (e.g., PlanetLab nodes, routers in an ISP, or sensor devices) under control of the information plane. In contrast, our focus is on an information plane for the Internet spanning thousands of administrative domains and millions of routers and end-hosts with only a small number of vantage points under our control.

We discuss related networking research next.

Link Metrics IDMaps [18] is an early example of a network information service that estimates the latency between an arbitrary pair of nodes using a small set of vantage points as routing landmarks. Subsequently, Ng et al [42] discovered that Internet distances can be modeled as a k-dimensional Euclidean space. Such embeddings can be used to predict latencies between a large number of nodes by measuring latencies from a small number of vantage points to these nodes. Several refinements to this basic methodology such as Vi-

valdi [13], PIC [12], Tang et al [56] Lighthouse [50], Lim et al [36], Shavitt et al [52] followed. Meridian [4] enables closest replica selection using an algorithm based on multi-resolution proximity rings. Their analysis and intuition assumes an underlying growth-restricted metric space, i.e., distances obey the triangle inequality and, informally, the number of nodes in any ring is not too much higher than the number of nodes in a ring half its size. A key limitation of all of these techniques is that they treat the Internet as a black box and are only predictive, i.e., they do not explain why, if at all, their predictions are correct. As a result, they have serious systematic deficiencies, e.g., a significant fraction of Internet paths are known to have detours [51], however, metric embeddings like above, by definition, obey the triangle inequality and will predict no detours.

We use an explanatory path prediction model developed by Madhyastha et al[38], however they only consider latency prediction. Yu et al [26] developed a technique to estimate available bandwidth between arbitrary node pairs based of splicing observed path segments assuming that bottlenecks occur close to the ends. The OASIS [20] system by Freedman et al provides an overlay-based anycast service by mapping dynamically-sized address prefixes to geographic locations based on their proximity to a vantage point. All of these experiences and others [37] emphasize limitations of embedding-based techniques and the need for explanatory models to estimate latency as well as more sophisticated path metrics.

This paper builds upon earlier work by Madhyastha et al [38] in the following ways. First, we develop techniques to predict loss rate, capacity, and available bandwidth in addition to latency. Second, we develop an improved clustering algorithm to resolve aliases and detect PoPs. Finally, we design and implement the *iPlane* as a service compared to the measurement study in [38] and evaluate it using application case studies. We are in the process of integrating a complete query interface to the *iPlane* and the Oasis access system [39] at the user end.

Network Tomography Chen et al [9] proposed an algebraic approach to infer loss rates on paths between all pairs of nodes based on measured loss rates on a subset of the paths. Duffield et al [6, 16] proposed a multicast-based approach to infer link loss rates by observing loss correlations between receivers. Padmanabhan et al [46] proposed a Bayesian inference technique to infer lossy links by conducting passive measurements at a Web server in conjunction with active traceroutes. Our use of BitTorrent client logs can be viewed as an Internet-scale distributed extension of this methodology. Jaiswal et al [30] propose a "measurements-in-the-middle" technique to infer end-to-end path properties using passive measurements conducted at a router. Such inference techniques can be integrated into an *iPlane* to which both end-users and ISPs jointly contribute for mutual benefit. In contrast to these techniques, tomography is only one component of the *iPlane*—we also need to predict topology and performance between arbitrary node pairs in the Internet.

The Rocketfuel [54] and Doubletree [14] systems estimate ISP topologies by performing traceroutes from a small set of vantage points and efficiently pruning redundant traceroutes. Our frontier (Section 2.3) algorithm to prune traceroutes and loss rate probes is similar in spirit and drastically reduces the running overhead of topology discovery and tomography.

Failure Recovery The PlanetSeer [62] monitoring system by Zhang et al detects and characterizes network path anomalies. Their methodology is based on passive monitoring of traffic at nodes offering a wide-area service (such as content distribution) combined with active traceroute probes from different nodes to characterize the anomaly. For each source-destination pair, they maintain a reference (forward) path and compare it to the reachable prefix of the path after a failure to infer the cause. The SOSR [23] system by Gummadi et al suggests that a simple randomized strategy of choosing one of a small number (about 4) of detour nodes suffices to recover from most *recoverable* failures, i.e., failures that detour nodes can circumvent. Both PlanetSeer and SOSR suggest that roughly half of the failures are recoverable. In comparison, the *iPlane* focuses on predictable performance metrics as opposed to failure analysis or recovery.

6 Conclusion

The performance and robustness of overlay services critically depends on the choice of end-to-end paths used as overlay links. Today, overlay services face a tension between minimizing redundant probe overhead and selecting good overlay links. More importantly, they lack accurate method to infer path properties between an arbitrary pair of nodes not under their control. In this paper, we showed that it is possible to accurately infer sophisticated path properties between an arbitrary pair of nodes using a small number of vantage points and existing infrastructure. The key insight is to systematically exploit the Internet's structural properties. Based on this observation, we built the *iPlane* service and showed that it is feasible to infer a richly annotated link-level map of the Internet's routing topology once every hour. Our case studies suggest that the *iPlane* can serve as a common information plane for a wide range of distributed services such as content distribution, file swarming, and voice-over-IP.

References

[1] Akamai, Inc. home page. http://www.akamai.com.

[2] D. G. Anderson, H. Balakrishnan, M. F. Kaawhoek, and R. Morris. Resilient Overlay Networks. In *SOSP*, 2001.
[3] S. Bellovin. A best-case network performance model. Technical report, ATT Research, 1992.
[4] E. G. S. Bernard Wong, Aleksandrs Slivkins. Meridian: A lightweight network location service without virtual. In *SIGCOMM*, 2005.
[5] A. Broido and kc claffy. Analysis of routeViews BGP data: policy atoms. In *Network Resource Data Management Workshop*, 2001.
[6] R. Caceres, N. G. Duffield, J. Horowitz, and D. F. Towsley. Multicast-based inference of network internal loss characteristics. *IEEE Transactions on Information Theory*, 1999.
[7] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *INFOCOM*, 2000.
[8] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: high-bandwidth multicast in cooperative environments. In *SOSP*, 2003.
[9] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *SIGCOMM*, 2004.
[10] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski. A knowledge plane for the Internet. In *SIGCOMM*, 2003.
[11] B. Cohen. Incentives build robustness in BitTorrent. In *P2PEcon*, 2003.
[12] M. Costa, M. Castro, A. Rowstron, and P. Key. PIC: Practical Internet coordinates for distance estimation. In *ICDCS*, 2004.
[13] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *SIGCOMM*, 2004.
[14] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient algorithms for large-scale topology discovery. In *SIGMETRICS*, 2005.
[15] C. Dovrolis, P. Ramanathan, and D. Moore. Packet dispersion techniques and a capacity estimation methodology. *Transactions on Networking*, 2004.
[16] N. G. Duffield, F. L. Presti, V. Paxson, and D. F. Towsley. Inferring link loss using striped unicast probes. In *INFOCOM*, 2001.
[17] The E-model, a computational model for use in transmission planning. ITU-T Recommendation G.107 http://www.itu.int/.
[18] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A global Internet host distance estimation service. *Transactions on Networking*, 2001.
[19] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with Coral. In *NSDI*, 2004.
[20] M. J. Freedman, K. Lakshminarayanan, and D. Mazieres. OASIS: Anycast for any service. In *NSDI*, 2006.
[21] P. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. IrisNet: An architecture for a world-wide sensor web. *IEEE Pervasive Computing*, 2(4), 2003.
[22] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *INFOCOM*, 2000.
[23] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the reliability of Internet paths with one-hop source routing. In *OSDI*, 2004.
[24] Q. He, C. Dovrolis, and M. Ammar. On the predictability of large transfer TCP throughput. In *SIGCOMM*, 2005.
[25] N. Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling*, 21(6), 2003.
[26] N. Hu and P. Steenkiste. Exploiting Internet route sharing for large scale available bandwidth estimation. In *IMC*, 2005.
[27] R. Huebsch, J. M. Hellerstein, N. L. Boon, T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *VLDB*, 2003.
[28] V. Jacobson. Pathchar. ftp://ftp.ee.lbl.gov/pathchar.
[29] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. In *SIGCOMM*, 2002.
[30] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Formal analysis of passive measurement inference techniques. In *INFOCOM*, 2006.
[31] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi. CapProbe: a simple and accurate capacity estimation technique. In *SIGCOMM*, 2004.
[32] S. Katti, D. Katabi, C. Blake, E. Kohler, and J. Strauss. MultiQ: Automated detection of multiple bottleneck capacities along a path. In *IMC*, 2004.
[33] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *SOSP*, 2003.
[34] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *USITS*, 2001.
[35] S. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca. Measuring bandwidth between planetlab nodes. In *PAM*, 2005.
[36] H. Lim, J. C. Hou, and C.-H. Choi. Constructing an Internet coordinate system based on delay measurement. In *IMC*, 2003.
[37] E. K. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft. On the accuracy of embeddings for Internet coordinate systems. In *IMC*, 2005.
[38] H. V. Madhyastha, T. E. Anderson, A. Krishnamurthy, N. T. Spring, and A. Venkataramani. An explanatory model of path latency. Technical report, University of Washington, Computer Science and Engineering, 2006.
[39] H. V. Madhyastha, A. Venkataramani, A. Krishnamurthy, and T. E. Anderson. Oasis: An overlay-aware network stack. *OSR*, 2006.
[40] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet path diagnosis. In *SOSP*, 2003.
[41] D. Meyer. RouteViews. http://www.routeviews.org.
[42] T. S. E. Ng and H. Zhang. Predicting Internet network distance with coordinates-based approaches. In *INFOCOM*, 2002.
[43] OpenView home page. http://www.managementsoftware.hp.com/.
[44] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Scalable wide-area resource discovery. Technical Report Technical Report CSD04 -1334, University of California Berkeley, Berkeley, CA, USA, 2004.
[45] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. *CCR*, 28(4), 1998.
[46] V. N. Padmanabhan, L. Qiu, and H. J. Wang. Passive network tomography using bayesian inference. In *IMW*, 2002.
[47] V. Pai, K. Tamilmani, V. Sambamurthy, K. Kumar, and A. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *IPTPS*, 2005.
[48] A. Parker. CacheLogic. http://www.cachelogic.com/research/slide1.php.
[49] L. peterson. Personal communication.
[50] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. Lighthouses for scalable distributed location. In *IPTPS*, 2003.
[51] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: a case for informed Internet routing and transport. *IEEE Micro*, 19(1), 1999.
[52] Y. Shavitt and T. Tankel. On the curvature of the Internet and its usage for overlay construction and distance estimation. In *INFOCOM*, 2004.
[53] Skype home page. http://www.skype.com.
[54] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *Transactions on Networking*, 2004.
[55] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *IMC*, 2003.
[56] L. Tang and M. Crovella. Virtual landmarks for the Internet. In *IMC*, 2003.
[57] S. Tao, K. Xu, A. Estepa, T. Fei, L. Gao, R. Guerin, J. Kurose, D. Towsley, and Z.-L. Zhang. Improving VoIP quality through path switching. In *INFOCOM*, 2005.
[58] IBM Tivoli software home page. http://www.ibm.com/software/tivoli.
[59] L. Wang, K. Park, R. Pang, V. S. Pai, and L. L. Peterson. Reliability and security in the CoDeeN content distribution network. In *USENIX*, 2004.
[60] M. Wawrzoniak, L. Peterson, and T. Roscoe. Sophia: An information plane for networked systems. In *HotNets-II*, 2003.
[61] Y. Zhang, V. Paxson, and S. Shenker. The stationarity of Internet path properties: Routing, loss, and throughput. Technical report, AT&T Center for Internet Research at ICSI, http://www.aciri.org/, 2000.
[62] M. Zhao, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *OSDI*, 2004.
[63] H. Zheng, E. K. Lua, M. Pias, and T. Griffin. Internet routing policies and round-trip-times. In *PAM*, 2005.