Dasu: Pushing Experiments to the Internet's Edge

Mario A. Sánchez* John S. Otto* Zachary S. Bischof* David R. Choffnes†
Fabián E. Bustamante* Balachander Krishnamurthy‡ Walter Willinger‡
*Northwestern U. †U. of Washington ‡AT&T Labs-Research

Abstract

We present Dasu, a measurement experimentation platform for the Internet's edge. Dasu supports both controlled network experimentation and broadband characterization, building on public interest on the latter to gain the adoption necessary for the former. We discuss some of the challenges we faced building a platform for the Internet's edge, describe our current design and implementation, and illustrate the unique perspective it brings to Internet measurement. Dasu has been publicly available since July 2010 and has been installed by over 90,000 users with a heterogeneous set of connections spreading across 1,802 networks and 147 countries.

1 Introduction

Our poor visibility into the network hampers progress in a number of important research areas, from network troubleshooting to Internet topology and performance mapping. This well-known problem [8, 39] has served as motivation for several efforts to build new testbeds or expand existing ones by recruiting increasingly large and diverse sets of measurement vantage points [15, 33, 37]. However, capturing the diversity of the commercial Internet (including, for instance, end-hosts in homes and small businesses) at sufficient scale remains an elusive goal [17, 25].

We argue that, at its roots, the problem is one of incentives. Today's measurement and experimentation platforms offer two basic incentive models for adoption – cooperative and altruistic. In cooperative platforms such as PlanetLab [31] and RIPE Atlas [35] an experimenter interested in using the system must first become part of it. Other platforms such as SatelliteLab [15] and DIMES [37] have opted instead for an altruistic approach in which users join the platform for the betterment of science. All these efforts build on the assumption, sometimes implicit, that the goals of those hosting the platform and the experimenters that use it are aligned. As much of the Internet's recent growth occurs in residential broadband networks [1] this assumption no longer holds.

This paper presents Dasu — a platform for network measurement experimentation and the Internet's edge built with an alternate model that explicitly aligns the objectives of platform hosts and experimenters. Dasu[1] is designed to support both broadband characterization and Internet measurement experiments and leverage their synergies. Both functionalities benefit from wide network coverage to capture network and broadband service diversity. Both can leverage continuous availability to capture time-varying changes in broadband service levels and to enable long-running and time-dependent measurement experiments. Both must support dynamic extensibility to remain effective in the face of ISP policy changes and to enable purposefully-designed, controlled Internet experiments. Finally, both functionalities must be available at the edge of the network to capture the end users' view of the provided services and offer visibility into this missing part of the Internet [9].

This paper focuses on Dasu's support for Internet measurement experimentation, outlining our current design and how it addresses some of the key challenges raised by our goals.[2] Dasu has been publicly available since June 2010 and is currently in use by 90,222 users with a heterogeneous set of connections spreading over 1,802 networks and across 147 countries.

We make the following contributions in this work:

- We present the design and implementation of Dasu — an extensible platform for measurement experimentation from the Internet's edge.
- We describe the current deployment of Dasu and present results demonstrating how the participating nodes collectively offer broad network coverage,

[1]Dasu is a Japanese word that can mean "to reveal" or "to expose".
[2]Please see [6] for a general description of Dasu's support for broadband characterization.

high availability and fine-grained synchronization to enable Internet measurement experimentation.

- We use three case studies to illustrate the unique perspective that a platform like Dasu brings to Internet measurement. In the process, we demonstrate Dasu's capabilities to (i) simplify traditional measurements (e.g., examining routing asymmetry), (ii) reveal fundamental shortcomings in existing measurement efforts (e.g., mapping AS-level topology), and (iii) conduct novel experiments for original system evaluations (e.g., examining the effectiveness of a recently-proposed DNS extension).

The rest of this paper is structured as follows. We put our work in context and provide further motivation in Sec. 2. In Sec. 3 and 4 we outline the design and implementation of Dasu and characterize our current deployment. We present cases studies that illustrate the benefits of a measurement experimentation platform that runs at the Internet's edge in Sec. 5. Finally, we discuss future work and present our conclusions in Sec. 6.

2 Background and Motivation

The lack of network and geographic diversity in current Internet experimentation platforms is well a known problem [8, 39]. Most Internet measurement and system evaluation studies rely on dedicated infrastructures [3, 5, 7, 31] which provide relatively continuous availability at the cost of limited vantage point diversity (i.e. with nodes primarily located in well-provisioned academic or research networks that are not representative of the larger Internet).

Several research projects have pointed out the pitfalls when attempting to generalize results of network measurements taken with a limited network perspective (e.g. [8, 10, 27, 32, 45]). For example, consider the differences in paths between PlanetLab nodes and between nodes in residential networks. These two sets traverse different parts of the network [9], exhibit different latency and packet loss characteristics [12, 20] and result in different network protocol behaviors [16].

2.1 Goals and Approach

An experimental platform for the Internet should be deployed at scale to capture network and service diversity. It should be hosted at the network edge to provide visibility into this opaque part of the Internet. Such a platform should allow dynamic extensibility in order to enable purposefully-designed, controlled measurement experiments, without compromising end-host security. To support time-dependent and long-running experiments, it should offer (nearly) continuous availability. Last, it should facilitate the design and deployment of experiments at the network edge while controlling the impact on the resources of participating nodes and the underlying network resources.

Dasu is an experimental platform designed to match these goals. To capture the diversity of the commercial Internet, Dasu supports both Internet measurement experimentation and broadband characterization and leverages their synergies. In its current version, Dasu is built as an extension to the most popular large-scale peer-to-peer system – BitTorrent.[3] The typical usage patterns and comparatively long session times of BitTorrent users means that Dasu can attain nearly continuous availability to launch measurement experiments. More importantly, by leveraging BitTorrent's popularity, Dasu attains the necessary scale and coverage at the edge of the network. Dasu is tailored for Internet network experimentation and, unlike general-purpose Internet testbeds such as PlanetLab, does not support the deployment of planetary-scale network services.

2.2 Challenges

Both strengths and challenges of a platform like Dasu stem from its inclusion of participating nodes at the Internet's edge. For one, the increased network coverage from these hosts comes at the cost of higher volatility and leaves the platform at the "mercy" of end users' behavior. The types of experiments possible in such a platform depend thus on the clients' availability and session times since these partially determine the maximum length of the experiment that can be safely assigned to clients. Such a platform must provide a scalable way to share measurement resources among concurrent experiments with a dynamic set of vantage points. It must also guarantee the safety of the volunteer nodes where it is hosted (for instance, by restricting the execution environment), and ensure secure communication with infrastructure servers. Last, to control the impact that experiments may have on underlying network and system resources, the system must support coordinated measurements among large numbers of hosts worldwide, each of which is subject to user interaction and interference.

2.3 Related Work

Dasu shares goals with and builds upon ideas from several prior large-scale platforms targeting Internet experimentation. Most active measurement and experimentation research relies on dedicated infrastructures (PlanetLab [31], Ark [7], Looking Glass servers). Such infrastructures provide relatively continuous availability and nearly continuous monitoring at the cost of limited vantage point diversity. Dasu targets the increasingly "invisible" portions of the Internet, relying on a direct

[3] A stand-alone version of Dasu has been developed and we plan to release it in June 2013.

incentive model to ensure large-scale adoption at the Internet edge.

Several related projects use passive measurements or restricted active measurements from volunteer platforms to capture this same perspective (e.g., [15, 33, 35, 37, 38, 42]). In contrast, Dasu is a software-based solution with a much broader set of measurement vantage points that has been achieved by altruistic and hardware-based systems, and supports a programmable interface that enables complex, coordinated measurements across the participating hosts. As such, Dasu shares some design goals with Scriptroute [40] and SatelliteLab [15]). Unlike Scriptroute, Dasu is intended for large scale deployment on end users' machines, and relies on incentives for user adoption at scale. Dasu also enables programable measurements without requiring root access, avoiding potential security risks and barriers to adoption. SatelliteLab adopts an interesting two-tier architecture that links end hosts (satellites) to PlanetLab nodes and separates traffic forwarding (done by satellites) from code execution. In Dasu, experiment code generates traffic directly from hosts at the network edge.

Several systems have proposed leveraging clients in a P2P system to measure, diagnose and predict the performance of end-to-end paths (e.g., [11, 28]. Dasu moves beyond these efforts, exploring the challenges and opportunities in supporting programmable experimentation from volunteer end hosts.

3 Dasu Design

In this section, we provide an overview of Dasu's design, discuss several system's components and briefly describe the API supporting measurement experiments.

3.1 System Overview

Dasu is composed of a distributed collection of clients and a set of management services. Dasu clients provide the desired coverage and carry on the measurements needed for broadband characterization and Internet experimentation. The *Management Services*, comprising the Configuration, Experiment Administration, Coordination and Data services, distribute client configuration and experiments and manage data collection. Figure 1 presents the different components and their interactions.

Upon initialization, clients use the *Configuration Service* to announce themselves and obtain various configuration settings including the frequency and duration of measurements as well as the location to which experiment results should be reported. Dasu clients periodically contact the Experiment Administration Service, which assigns measurement tasks, and the Coordination Service to submit updates about completed probes and retrieve measurement limits for the different experiment tasks. Finally, clients use the Data Service to report the results of completed experiments as they become available.

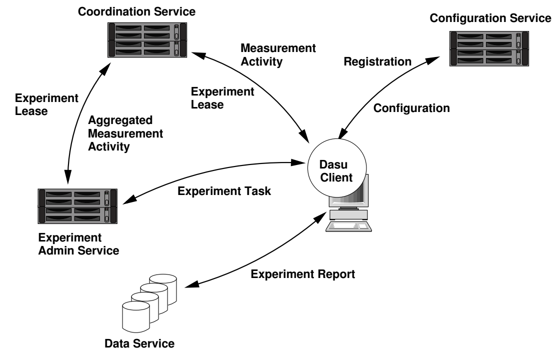


Figure 1: *Dasu system components.*

3.2 Experiment Specification

Dasu is a dynamically extensible platform designed to facilitate Internet measurement experimentation while controlling the impact on hosts' resources and the underlying network. A key challenge in this context is selecting a programming interface that is both flexible (i.e., supports a wide range of experiments) and safe (i.e., does not permit run-away programs). We rejected several approaches based on these constraints and our platforms goals. These include offering only a small and fixed set of measurement primitives as they would limit flexibility. We also avoided providing arbitrary binary execution as handling the ramifications of such an approach would be needlessly complex.

We opted for a rule-based declarative model for experiment specification in Dasu. In this model, a rule is a simple `when-then` construct that specifies the set of actions to execute when certain activation conditions hold. A rule's left-hand side is the conditional part (`when`) and states the conditions to be matched. The right-hand side is the consequence or action part of the rule (`then`) i.e., the list of actions to be executed. Condition and action statements are specified in terms of read/write operations on a shared working memory and invocation of accessor methods and measurement primitives. A collection of rules form a program and a set of related programs define an experiment.

The rule-based model provides a clean separation between experiment logic and state. In our experience, this has proven to be a flexible and lightweight approach for specifying and controlling experiments. Experiment logic is centralized, making it easy to maintain and extend. Also, strict constraints can be imposed on rule syntax, enabling safety verification through simple static program analysis.

Dasu provides an extensible set of measurement primitives (modules) and a programmable API to combine them into measurement experiments. Tables 1

Method	Params.	Description
addProbeTask	<probe> <params> [<times>] [<when>]	Submit measurement request of the specified type.
commitResult	<report>	Submit completed experiment results to data server.
getClientIPs	[]	Return network information about the client including the list of IP addresses assigned (both public and private).
getDnsServers	[]	Return the list of DNS servers configured at the client.
getEnvInfo	[]	Return information about the plugin and the host node, including OS information and types of measurement probes available to the experimenter.

Table 1: *Dasu API – Methods.*

<table>
<tr><th>Probe</th><th>Params.</th><th>Description</th></tr>
<tr><td>PING</td><td><dest-list (IP/name)></td><td>Use the local host ping implementation to send $ECHO\_REQUEST$ packets to a host.</td></tr>
<tr><td>TRACEROUTE</td><td><dest-list (IP/name)></td><td>Print the route packets take to a network host.</td></tr>
<tr><td>NDT</td><td>[<server>]</td><td>Run the M-Lab Network Diagnostic Tool [29].</td></tr>
<tr><td>DNS</td><td>[<server>] | [<timeout>] | [<tcp/udp>] | [<options>] | <DNS-msg>] | <dest-list></td><td>Submit DNS resolution request to a set of servers.</td></tr>
<tr><td>HTTP</td><td>[server] | [<port>] | [<HTTP-Req>] | <url-list></td><td>Submit HTTP request to a a given $< host, port >$ pair.</td></tr>
</table>

Table 2: *Dasu API – Measurement modules currently supported.*

and 2 provide a summary of this API and the current set of measurement primitives supported. The API includes some basic accessor methods (e.g. `getClientIps`, `getDnsServers` and `getEnvInfo`). The method `addProbeTask` serves to request the execution of measurements at a given point in time. The `commitResult` method allows results from the experiment to be submitted to the Data Service after completion.

Dasu provides low-level measurement tools that can be combined to build a wide range of measurement experiments. Currently available measurement primitives include traceroute, ping, Network Diagnostic Tool (NDT) [29], HTTP GET and DNS resolution. While this set is easily extensible (by the platform administrators) we have found it sufficient to allow complex experiments to be specified clearly and concisely. For instance, the experiment for the Routing Asymmetry case study (Sec. 5.1) was specified using only 3 different rules with an average of 24 lines of code per rule.

Measurements primitives are invoked asynchronously by the *Coordinator*, which multiplexes resources across experiments. Progress and results are communicated through a shared *Working Memory*; through this working memory, an experiment can also chain rules that schedule measurements and handle results.

In addition to these active measurements, Dasu leverages the naturally-generated BitTorrent traffic as passive measurements (particularly in the context of broadband characterization [6]) by continuously monitoring the end-host Internet connection. Devising an interface to expose these passively collected measurements to experimenters is part of future work.

A Simple Example. To illustrate the application of rules, we walk through the execution of a simple experiment for debugging high latency DNS queries. Figure 2 lists the rules that implement this experiment. When rule #1 is triggered, it requests a DNS resolution for a domain name using the client's configured DNS server. When the DNS lookup completes, rule #2 extracts the IP address from the DNS result and schedules a ping measurement. After the ping completes, rule #3 checks the ping latency to the IP address and schedules a traceroute measurement if this is larger than 50 ms.

3.3 Delegating Code Execution to Clients

Dasu manages concurrent experiments, including resource allocation, via the Experiment Administration Service. As clients become available, they announce their specific characteristics (such as client IP prefix, connection type, geographic location and operating system) and request new experiment tasks. The *Experiment Administration (EA) Service* assigns tasks to a given client based on experiment requirements and characteristics of available clients (e.g. random sample of DSL users in Boston).

In the simplest of experiments, every Dasu client assigned to an experiment will receive and execute the same experiment task (specified as a stand-alone rules file). Dasu also enables more sophisticated experiments where experimenters specify which clients to use and how to execute tasks based on client characteristics.

```
rule "(1) Resolve IP address through local DNS"
 when
  $fact : FactFireAction(action=="resolveIp");
 then
  addProbeTask(ProbeType.DNS, "example.com");
end

rule "(2) Handle DNS lookup result"
 when
  $dnsResult : FactDnsResult(toLookup=="example.com")
 then
  String ip = $dnsResult.getSimpleResponse();
  addProbeTask(ProbeType.PING, ip);
end

rule "(3) Handle ping measurement result"
 when
  $pingResult : FactPingResult()
 then
  if ( $pingResult.getRtt() > 50 )
   addProbeTask(ProbeType.TRACEROUTE, $pingResult.ip );
end
```

Figure 2: *Example measurement experiment for debugging high latency DNS queries.*

Dasu adopts a two-tiered architecture for the EA Service, with a primary server, responsible for resource allocation, and a number of secondary servers in charge of particular experiments. The *Primary EA* server acts as a broker, allocating clients to experiments, by assigning them to the responsible secondary server, based on clients' characteristics and resource availability. The *Secondary EA server* is responsible for task parameterization and allocation of tasks to clients according to the experiment's logic. While the customized task assigned to a client is generated by the experiment's secondary server, all communication with Dasu clients is mediated by the primary server who is responsible for authenticating and digitally signing the assigned experiments.

Submitting External Experiments. Dasu supports third-party experiments through the two-tier architecture described above. Authorized research groups host their own Secondary EA server, with security and accountability provided through the Primary EA server.

In addition to providing a safe environment for executing experiments, all experiments submitted to Dasu are first curated and approved by the system administrators before deployment. This curation process serves as another safety check and ensures that admitted experiments are aligned with the platform's stated goals.

3.4 Security and Safety

Safely conducting measurements is a critical requirement for any measurement platform and particularly for one deployed at the Internet edge. We focus on two security concerns: protecting the host and the network when executing experiments. We expand on the former here and discuss the latter in the following section.

To protect the host, Dasu uses a sandboxed environment for safe execution of external code, ensures secure communication with infrastructure servers, and carefully limits resource consumption.

Experiment Sandbox. To ensure the execution safety of external experiments, Dasu confines each experiment to a separate virtual machine, instantiated with limited resources and with a security manager that implements restrictive security polices akin to those applied to unsigned Java applets. In addition, all Dasu experiments are specified as a set of rules that are parsed for unsafe imports at load time, restricting the libraries that can be imported. Dasu inspects the experiment's syntax tree to ensure that only specifically allowed functionality is included and rejects a submitted experiment otherwise.

Secure communication. To ensure secure communication between participating hosts and infrastructure servers, all configuration and experiment rule files served by the EA Service are digitally signed for authenticity and all ongoing communications with the servers (e.g. for reporting results) are established over secure channels.

Limits on resource consumption. Dasu must carefully control the load its experiments impose on the local host, as well as minimize the impact that users' interactions (i.e., with the host and the application) can have on experiments' results. To this end, Dasu limits consumption of hosts' resources[4] and restricts the launching of experiments to periods of low resource utilization; the monitored resources include CPU time, network bandwidth, memory and disk space.

To control CPU utilization, Dasu monitors the fraction of CPU time consumed by each system component (including the base system and each different probe module). Dasu regulates average CPU utilization by imposing time-delays on the activity of individual probe modules whenever their "fair share" of CPU time has been exceeded over the previous monitoring period. Dasu also employs watchdog timers to control for long-running experiments.

To control bandwidth consumption, Dasu passively monitors the system bandwidth usage and launches active measurements only when utilization is below certain threshold (we evaluate the impact of this policy on experiment execution time in Sec. 4.3). Dasu uses the 95th percentile of client's throughput rates measured by NDT to estimate the maximum bandwidth capacity of the host and continuosly monitors host network activity (using the commonly available `netstat` tool). Based on pre-computed estimates of approximate bandwidth consumption for each probe, Dasu limits probe execution by only launching those that will not exceed the predetermined average bandwidth utilization limit. Additionally Dasu relies on a set of predefined limits on the number

[4] Currently 15% of any monitored resource.

of measurement probes of each type that can be launched per monitored interval. While clients are allowed to dispense with their entire budget at once, the combined bandwidth consumed by all probe modules must remain below the specified limit.

To restrict memory consumption, Dasu monitors the allocated memory used by its different data structures and limits, for instance, the number of queued probe-requests and results. Measurement results are offloaded to disk until they can successfully be reported to the Data Service. Disk space utilization is also controlled by limiting the size of the different probe-result logs; older results are dropped first when the pre-determined quota limits have been reached.

3.5 Coordination

In addition to controlling the load on and guaranteeing the safety of volunteer hosts, Dasu must control the impact that measurement experiments collectively may have on the underlying network and system resources. For instance, although the individual launch rate of `ping` measurements is limited, a large number of clients probing the same destination can overload it.

To this end, Dasu introduces two new constructs - *experiment leases* and *elastic budgets*, to efficiently allow the scalable and effective coordination of measurements among potentially thousands of hosts. In the following paragraphs, we describe both constructs and Dasu's approach to coordination.

Experiment Leases. To support the necessary fine-grained control of resource usage, we introduce the concept of experiment leases. In general, a *lease* is a contract that gives its holder specified rights over a set of resources for a limited period of time [19]. An *experiment lease* grants to its holder the right to launch a number of measurement probes, using the common infrastructure, from/toward a particular network location. Origin and/or targets for the probes can be specified as IP-prefixes or domain names (other forms, such as geographic location, could be easily incorporated).

Experiment leases are managed by the EA Service. The Primary EA server ensures that the aggregated use of resources by the different experiments is within the specified bounds. Secondary EA servers are responsible for managing experiment leases to control the load imposed by their particular experiments. To coordinate the use of resources by the Dasu clients taking part in an experiment, we rely on a distributed coordination service [23]. The Coordination Service runs on well-provisioned servers (PlanetLab nodes) using replication for availability and performance. Clients receive the list of coordination servers as part of the experiment description.

Before beginning an experiment, clients must contact a coordinator server to announce they are joining the experiment and obtain an associated lease. As probes are launched, the clients submit periodic updates to the coordination servers about the destinations being probed. The EA Service uses this information to compute estimated aggregate load per destination and to update the associated entries in the experiment lease. Before running a measurement, the Coordinator checks whether it violates the constraint on the number of probes allowed for the associated source and destination, and if so delays it. After a lease expires, the host must request a new lease or extend the previous one before issuing a new measurement. The choice of the lease term presents a trade-off between minimizing overhead on the EA Service versus minimizing client overhead and maximizing its use.

Elastic Budget. An experiment lease grants to its holder the right to launch a number of measurement probes (i.e., *a budget*) from/toward a particular network location. Due to churn and user-generated actions, the number of measurement probes a Dasu client can launch before lease expiration (i.e., the fraction of the allocated budget actually used) can vary widely. To account for this, Dasu introduces the idea of *elastic budgets* that expand and contract based on system dynamics.

Elastic budgets are computed by the EA Service and used to update bounds on experiment leases distributed to Dasu clients. The EA Service calculates the elastic budget periodically based on the current number of clients participating in the experiment, the number of measurement probes allowed, assigned and completed by each client. The EA Service uses this elastic budget to compute measurement probe budgets for the next lease period for each participating client. This approach is well suited for experiments where the server knows a priori what destinations each client should probe. In the case of experiments where the destinations to be probed are not assigned by the server, but obtained by the clients themselves (through a DNS resolution for example), the same approach can be used if we conservatively assume that a client will launch the maximum number of probes per unit of time whenever it is online.

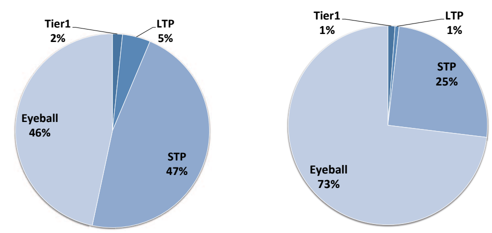
3.6 Synchronization

Dasu also provides support for Internet experiments that require synchronized client operation (e.g. [34, 41]).

For coarse-level synchronization, Dasu clients include a cron-like probe-scheduler that allows the scheduling of measurements for future execution. All Dasu clients periodically synchronize their clocks using NTP. Assuming clients' clocks are closely synchronized, an experiment can request the "simultaneous" launch of measurements

Region	Penetration	Dasu Total	Dasu Total Countries
North America	78.6 %	21.45 %	60 %
Oceania/Australia	67.5 %	3.82 %	6 %
Europe	61.3 %	59.25 %	73 %
L. America/Carib.	39.5 %	1.68 %	65 %
Middle East	35.6 %	1.52 %	73 %
Asia	26.2 %	2.59 %	57 %
Africa	13.5 %	9.66 %	34 %

Table 3: *Internet penetration*[6] *and Dasu coverage (as percentage of its total population of 90,222) by January 2013.*



(a) Peer Distribution **(b)** AS Distribution

Figure 3: *Distribution of Dasu peers per AS (left). Distribution of ASes covered by Dasu peers (right).*

by a set of clients. We have found this to be sufficient to achieve task synchronization on the order of 1-3 seconds.

For finer-grained synchronization (on the order of milliseconds), Dasu adopts a remote triggered execution model. All synchronized clients must establish persistent TCP connections with one of the coordination servers. These connections are later used to trigger clients actions at a precise moment, taking into account network delays between clients and coordination servers.

4 Deployment

We have implemented Dasu as an extension to a popular BitTorrent client [43] as it offers a large and widespread client population and a powerful interface for extensions. We have made Dasu publicly available since June 2010.

To participating users, Dasu provides information about the service they receive from their ISP [6, 36]. Access to such information has proven sufficient incentive for widespread subscription with over 90K users who have adopted our extension with minimum advertisement.[5]

This section demonstrates how Dasu clients collectively provide broad network coverage, sufficiently high availability and fine-grained synchronization for Internet experimentation.

4.1 Dasu Coverage

We show the coverage of Dasu's current deployment in terms of geography and network topology. Table 3 lists broadband penetration in each primary geographic region and compares these numbers with those from our current Dasu's deployment.

Given the high Internet penetration numbers in Europe and North America, the distribution of Dasu clients per region is not surprising. Note, however, the penetration of Dasu clients per region, measured as the percentage of countries covered. As the table shows, Dasu penetration is over 57% for most regions and is particularly high for Latin America/Caribbean (65%) and the Middle East (73%), two of the fastest growing Internet regions. Even in Africa Dasu penetration reaches 34%.

We also analyze Dasu's network coverage in terms of ASes where hosts are located. With our existing user-base at the end of January 2013, we have Dasu clients in 1,802 different ASes. We classify these ASes following a recently proposed approach [13], as follows:

- Tier-1: 11 known Tier-1s
- LTP: Large (non tier-1) transit providers and large (global) communications service providers
- STP: Small transit providers and small (regional) communication service providers
- Eyeball: Enterprise customers or access/hosting providers

Figure 3a uses this classification to illustrate where Dasu peers are deployed. As the figure shows, 93% of Dasu peers are located in small transit providers and eyeball ASes; with only minimal presence in large transit and Tier-1 providers. Figure 3b presents the distribution of all the ASes covered by Dasu peers. This figure shows that 73% of the ASes covered by Dasu are eyeball ASes, highlighting the effectiveness of Dasu as a platform for capturing the view from the network edge.

4.2 Dasu Dynamics

In this section, we show that the churn from Dasu clients is sufficiently low to support meaningful experimentation. This churn is a result of both the volatility of Dasu's current hosting application (i.e. BitTorrent) and that of the end systems themselves. In the following analysis, we focus on the hosting application dynamics. In particular, we investigate what portion of clients are online at any moment, and whether their session times support common measurement durations.

First, we analyze Dasu clients' availability, using the percentage of clients online at any given hour over a 31-day period. Figure 4 plots this for the month of January 2013. The fraction of available clients during the period varies, on average, between 39% and 44% of the total number of unique users seen during a day, with a total of 1,473 active unique users for the month. With respect to

[5]Upon download, users are informed of both roles of Dasu. Users can, at any point, opt to disable experiments from running and/or reporting performance information, without losing access to Dasu's broadband benchmarking information.

[6]`http://www.internetworldstats.com`

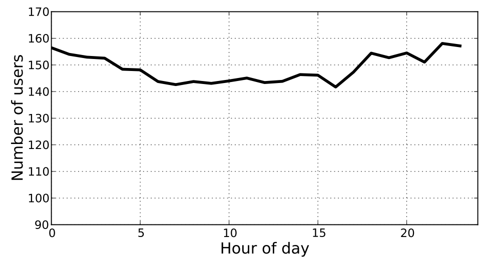


Figure 4: *Number of online Dasu clients over a 24-hour period. The fraction ranges from 39-44% of the total number of unique users, on average.*

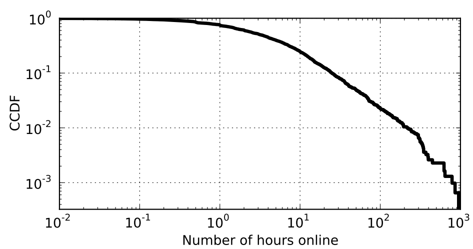


Figure 5: *Session time distribution of Dasu clients (time between their joining and leaving the system).*

the overall stability of the platform, for the same month of January 2013, we saw a total of 1,303 installs, 61 user uninstalls and 21 users who disabled reporting while continuing to run Dasu.

Next, we analyze how the duration of experiments is limited by client session times. Session time is defined as the elapsed time between it joining the network and subsequently leaving it. The distribution of clients' session times partially determines the maximum length of the measurement tasks that can be "safely" assigned to Dasu clients. Figure 5 shows the complementary cumulative distribution function of session times for the studied period. The distribution is clearly heavy-tailed, with a median session time for Dasu clients of 178 minutes or ≈ 3 hours.

Given an average session time, the fraction of tasks that are able to complete depends on the duration of the

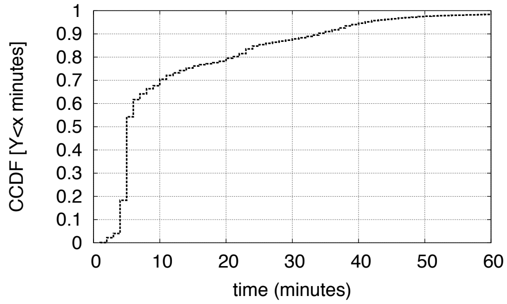


Figure 6: *Task time distribution for completed tasks by Dasu clients. The median task successfully completes in <5 minutes.*

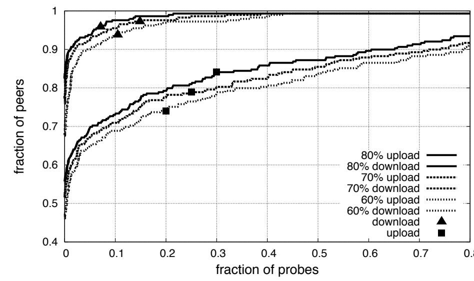


Figure 7: *Distribution of fraction of probes per peer that are delayed due to bandwidth constraints at the client.*

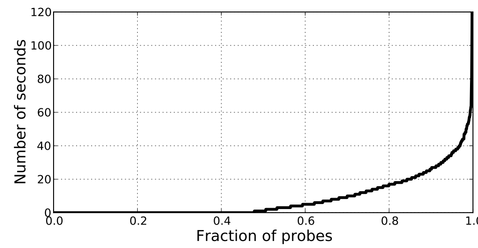


Figure 8: *Distribution of experiment probe submission for clients. Over 55% are launched $< 1sec.$ after being scheduled.*

task – a function of the number of actual measurements and the load at the client. Figure 6 shows the distribution of task completion times for all experiments completed by Dasu peers over a 3-week period. All experiments during this period were done in the context of the case study on IXP mapping (Sec. 5), were an experiment task consists of a set of traceroutes issued by clients to discover potential peerings. The figure shows that the median task is able to successfully complete in less than 5 minutes. The plot also shows that nearly all tasks are able to complete successfully in the face of churn, with 70% of tasks finishing in less than 12 minutes.

4.3 Controlling Experimentation Load

To minimize Dasu's impact on host application performance and to ensure that user interactions do not interfere with scheduled measurements, Dasu enforces pre-defined limits on the number of probes executed per unit time and schedules measurements during low utilization periods. We evaluate the impact of one of these restrictions (on bandwidth utilization) on experiment execution by determining the portion of scheduled measurements delayed.

Figure 7 shows a CDF of the fraction of probes delayed by clients due to different bandwidth utilization constraints (60%, 70% and 80%), taken from a random subset of clients over a two-week period. The distribution shows, for instance, that capping at a download utilization of 80%, every scheduled probe can be launched immediately for 85% of the peers, and that for 98% of

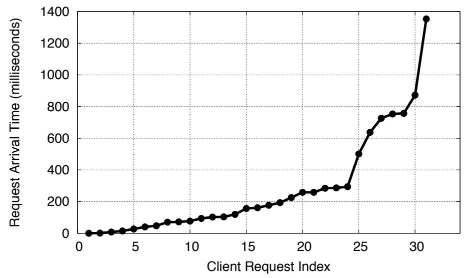


Figure 9: *Request arrival times at the target server. Approximately 80% of requests arrive within 300 ms.*

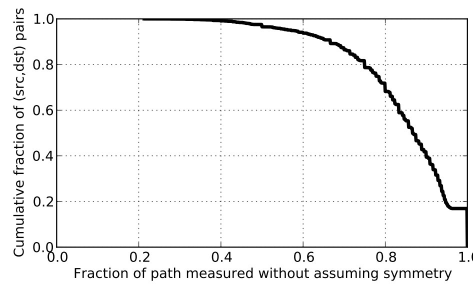


Figure 10: *CCDF of fraction of Dasu-PL path hops that can be directly measured using IP Options probes. 17% of paths reply to probes at each hop, meaning that we can determine the complete reverse path.*

the peers less than 20% of the probes would require any delay. In contrast, a smaller fraction of probes (60%) experience no delay when an 80% utilization limit is imposed on the upload direction. This is expected, since broadband users are often allocated lower upload bandwidth than download.

Fig. 8 shows the queueing time of probes assigned to Dasu clients for a given experiment over a 1-week period. The figure shows that over 55% of the probes are launched in less than a second after being scheduled.

4.4 Client Synchronization

To evaluate the granularity of Dasu's fine-grain synchronization capabilities, we run an experiment where Dasu clients were instructed to simultaneously launch an HTTP request to an instrumented web server. For a span of five minutes, approximately 30 clients were recruited to cooperate in the experiment. Following Ramamurthy et al. [34], as clients joined the experiment they were instructed to measure their latency to the target server as well as to the Coordination Server and to report back their findings.

At the end of the five minutes, clients were scheduled to launch their measurements (having adjusted each request based on their measured latencies) while we logged the arrival times of each incoming HTTP request at the target server. We repeated this experiment 10 times. Figure 9 shows the mean arrival time of each request with a crowd size of 31 clients. About 80% of the requests arrive within 300ms of each other, and 91% of the requests arrive within 1s of each other. This result is on par with the synchronization of 100s of milliseconds reported by Ramamurthy et al. [34]

Variations in the arrival times of the top 20% of requests are due to queuing delays in broadband networks [16] and errors in estimating the latency between clients and the coordinator server.

5 Case Studies

In this section, we present three case studies that illustrate the unique perspective our edge-based platform brings to Internet measurement and serve as examples of experiments made possible using Dasu.

5.1 Extending Earlier Experiments: Routing Asymmetry

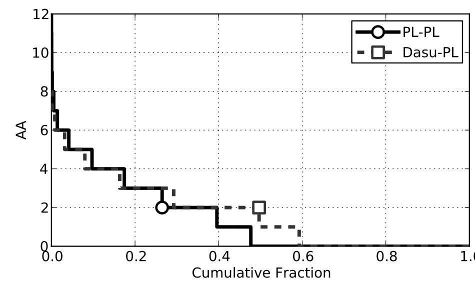
Routing asymmetry can impact the results of measurement tools such as traceroute. For instance, estimates of delay between hosts are subject to errors if the forward and reverse path differ. We extend the work by He et al. [21], comparing routing asymmetry for research and commercial networks, by examining the paths between stub (Dasu) and research (PlanetLab) networks.

Ideally, one would like to control the hosts at both ends of a path to determine the forward and reverse paths between them, and have both endpoints probe the path concurrently to minimize the impact of factors such as network load or time-of-day on routing decisions. The Reverse Traceroute system [24] provides a useful approach to determine the reverse path even when one controls only one of the end points. The approach, however, is not always effective as it cannot probe reverse paths in networks where routers do not reply to IP Options probes.

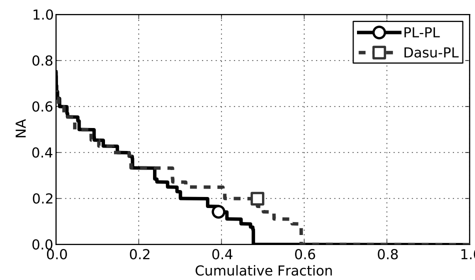
A number of features of Dasu make it possible to conduct an accurate analysis of path asymmetry between nodes located in stub networks vs. research networks including the ability to schedule experiments and to synchronize the launching of measurements across nodes.

We find that for ≈28% of the paths tested between Dasu clients and PlanetLab nodes (out of 8,046) reverse traceroute would be forced to make an incorrect symmetry assumption because a segment of the reverse path transits at least one AS that does not appear on the forward path and that does not respond to IP Options probes. Figure 10 shows that only 17% of the paths between Dasu and PlanetLab nodes respond to IP Options probes at every hop, this in contrast to the over 40% of paths between PlanetLab nodes reported in [24].

To study routing asymmetry between pairs of Dasu-PlanetLab (Dasu-PL) and PlanetLab-PlanetLab (PL-PL)

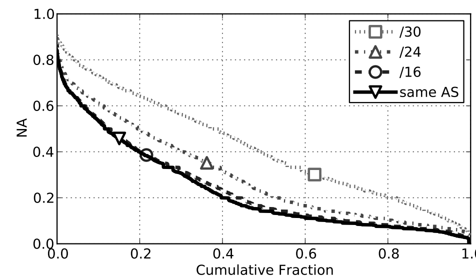


(a) Absolute Asymmetry

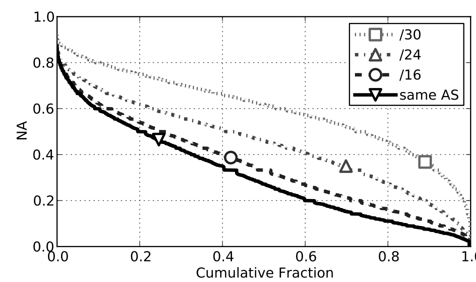


(b) Normalized Asymmetry

Figure 11: *CDFs of AS-level asymmetry in Dasu-PL and PL-PL paths; ≈60% of Dasu-PL paths show some degree of asymmetry, vs. 48% of PL-PL paths.*



(a) PlanetLab-PlanetLab



(b) Dasu-PlanetLab

Figure 12: *CDFs of link-level normalized asymmetry using different heuristics for IP to link mapping. Link-level NA is much lower for PL-PL paths than Dasu-PL paths.*

nodes, we launched probes across 8,046 paths between Dasu clients and PlanetLab nodes, and across 10,067 paths between two PlanetLab nodes. To ensure accurate measures of routing asymmetry we had hosts at both endpoints probe the path concurrently.

We measure routing asymmetry by following the methodology described in [21]. This method maps hops in the forward path to those of the reverse path (either at the link-level or AS-level) and assigns a value of 0 if the hops are identical and a value of 1 if they are different. Through dynamic programming, it then selects the mappings for each path that results in the minimal distance. The minimal composite dissimilarity between a forward and reverse path is referred to as the *Absolute Asymmetry (AA)*, while the length-based *Normalized Asymmetry (NA)* is defined as AA normalized by the length of the round-trip path.

To compare the asymmetry in the AS-level paths between the two sets of paths (i.e., Dasu-PL and PL-PL), figures 11a and 11b show the cumulative distributions of the AS-level AA and NA metrics, respectively. It can be observed that the Dasu-PL paths not only have a higher percentage of asymmetric routes, but also display a higher magnitude of asymmetry than the PL-PL paths. To compare the two datasets at the link-level, we again follow the approach described in He et al. [21] and use their heuristics to determine if two IP addresses correspond to the interfaces of the same link. These heuristics consider two IP addresses to belong to the same point-to-point link if they belong to the same /30, /24, /16, or AS. For each of the four heuristics, Figures 12a and 12b show the cumulative distributions of the resulting NA metric for the PL-PL and Dasu-PL paths, respectively. As noted in [21], the first (/30) and last (AS) heuristics provide the upper and lower bounds, on the observed Internet routing asymmetry at the link level. While figures 11a and 11b show that the two sets of paths exhibit differences in routing asymmetry at the AS-level, figures 12a and 12b show these differences are significantly more pronounced at the link-level but depend greatly on the heuristics used.

5.2 Questioning Existing Experiments: Inferring AS-level Connectivity

The model of the Internet as a hierarchically-structured or "tiered" network of networks is changing [14, 18, 26]. The emergence of new types of networks (e.g., content providers, web hosting companies, CDNs) and their resulting demands on the Internet have induced changes in the patterns of interdomain connections; however, the precise degree and nature of these changes remains poorly understood.

Internet exchange Points (IXPs) are an important part of the rapidly developing Internet ecosystem because they facilitate the changes, enabling direct connections between member ASes. A recent study of a large European IXP has shown that some of the largest IXPs (e.g., DEC-IX and AMS-IX) handle traffic volumes that

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	prefixes
x	*x*	-	-	-	-	-	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	-	*x*	*x*	*A*
✓	✓	✓	✓	✓	-	-	-	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	*B*

Table 4: *Prefix-based peering at Amsterdam Internet Exchange (AMS-IX) between two ASes. Columns show the hour, local time. Legend: '✓' probes crossed IXP; '*x*' probes did not cross IXP; '-' no probes.*

are comparable to those carried by some of the largest global ISPs and support peering fabrics consisting of more than 60% of all possible peerings among their 400-500 member ASes [2]. However, despite their importance, there exists little to no publicly available information about who is peering with whom nor about the nature of these peerings.

These changes in the network's structure demand changes to how we have traditionally conducted experiments. For instance to question the standard assumption of homogeneity that has been made when inferring AS-level connectivity at IXPs [4, 22, 44] – where a single traceroute between two ASes members of an IXP is sufficient to declare that these ASes are, as a whole, connected in the AS graph by a peer-peer link – we require an endemic population of vantage points that allows for finer-grained measurements.

Dasu provides an ideal platform to examine the validity of such assumptions. Its widespread and diverse user base provides vantage points in multiple prefixes within the same AS which allows us to identify prefix-specific features that could not be identified from a single location in the network. Additionally, Dasu's near-continuous availability of vantage points allows us to study temporal effects that are critical for the observed kind of peering. Lastly, conducting this kind of targeted experiments involving specific prefixes in specific ASes at particular IXPs relies critically on the programmability of Dasu.

To evaluate the validity of this homogeneity assumption, we set up an experiment to launch multiple traceroute probes, between the same pair of member ASes of a given IXP, from vantage points located inside different prefixes of the source AS and at different hours of the day. We found that about 15% of the peering links that Dasu discovered violated the assumed homogeneity condition. Depending on the prefixes, the probes either crossed the given IXP or were sent instead via one of the source AS's upstream providers.[7] Table 4 shows a concrete example of such fine-grained peering observed between two ASes at AMS-IX. By probing for peerings between AS1 and AS2 repeatedly from different prefixes in the ASes and separating the probes by the peers' local time, we obtained a view of these well-covered peerings throughout the day. For each data point in the table we corroborated the result across multiple traceroute probes and obtained thus an example of a consistent prefix-based peering – while probes launched from source prefix A towards AS2 are never seen crossing the IXP, probes launched from source prefix B towards AS2 seem to always go through the IXP.

In short, the discovery of such fine-grained or prefix-specific peering arrangements is proof that the traditional view that a single type of AS peering applies uniformly across all prefixes of an AS is no longer tenable. This finding has clear implications for measurement and inference of AS-level connectivity and poses new challenges and requirements for the platforms and techniques used for this type of studies.

5.3 Performing Novel Experiments: Evaluating a Recently-proposed DNS Extension

The *edns-client-subnet* EDNS0 extension (`ECS`) was developed to address the problems raised by the interaction between the DNS-based redirection techniques commonly employed by CDNs and the increasing use of remote DNS services. CDNs typically map clients to replicas on the location of the client's local resolver; since the resolvers of remote DNS services may be far from the users, this can result in reduced CDN performance. `ECS` aims to improve the quality of CDN redirections by enabling the DNS resolver to provide partial client location (i.e. client's IP prefix) directly to the CDN's authoritative DNS server. `ECS` is currently being used by a few public DNS services (e.g., Google DNS) and CDNs (e.g. EdgeCast) and can improve CDN redirections without modifications to end hosts.

To understand the performance benefits of the proposed `ECS` extension and capture potential variations across geographic regions would require access to a large set of vantage points. These vantage points should be located in access networks around the world and allow issuing the necessary interrelated measurement probes. These are some of the unique features that Dasu offers.

Dasu's extensibility allows for the creation and addition of a new probe module to generate and parse ECS-enabled DNS messages. Additionally, Dasu's user base allows us to obtain representative measurement samples from diverse regions and compare trends across geographic areas by looking at the relationships between raw CDN performance, relative proportions of clients affected by the extension, and the degree of performance improvement provided by the extension.

This experiment extends the work by Otto et al. [30], which examined the impact of varying the amount of

[7] The various reasons for why certain ASes engage in such non-traditional peering arrangements is beyond the scope of this paper.

information shared by ECS (i.e. prefix length) and compared its performance to a client-based solution. We first obtain CDN redirections to edge servers both with the ECS extension enabled and disabled. Specifically, we query Google DNS (8.8.8.8) for an EdgeCast hostname. To obtain a redirection with ECS disabled, our DNS probe module sends a query with the ECS option that specifies 0 bytes of the client's IP prefix—this effectively disables the extension's functionality. For the ECS-enabled query, we provide the client's /24 IP prefix. After obtaining CDN edge server redirections with and without ECS's help, we conduct HTTP requests to both sets of CDN edge servers to measure the application-level performance in terms of latency to obtain the first byte of content. For the results from each client, we compare the median performance with and without ECS being enabled.

We analyze results from a subset of 1,185 Dasu clients that conducted this experiment over a 4 month period from September 12th, 2011 to January 16th, 2012.[8] Figure 13 shows the relationship between HTTP latency with ECS disabled and the performance benefits (latency savings) with ECS enabled. We classify users by geographic region; the percentages listed in the legend indicate the fraction of all sampled clients from that region. In all regions, sampled clients are located in a diverse set of networks; even in Oceania—the region with fewest clients—we cover 9 ISPs in Australia and 4 in New Zealand. The figure plots the subset of samples in which EDNS impacted HTTP performance.

While we find clients in all these regions that obtained HTTP performance improvements with ECS enabled, the samples tend to cluster by region. Although clients in North America and Western Europe both typically see HTTP latencies between 20 and 200 ms, the North American clients generally obtain higher percentage savings. This would indicate that the CDN's infrastructure in North America is relatively dense in comparison to that of the public DNS service's deployment. Clients in Oceania typically have relatively high HTTP latencies between 200 and 1000 ms with ECS disabled—but commonly realize savings of 70–90% with ECS enabled. This is likely a result of the specific deployments of the CDN and DNS services; although there are actually CDN edge servers near to clients in this region, it appears that the nearest Google DNS servers are farther away, resulting in reduced HTTP performance when ECS is disabled. Finally, we compare the number of clients with benefits from ECS between Eastern Europe and Oceania; while clients in Oceania actually comprise a slightly smaller fraction of the overall sample, the number of clients that actually observed better performance is much higher than for clients in Eastern Europe.

[8]Each participating client runs the experiment once over that time.

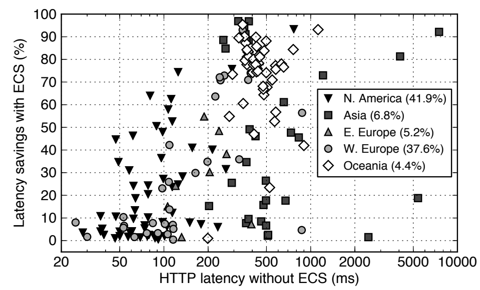


Figure 13: *HTTP latency vs. the performance benefits provided by ECS, by geographic region. Percentages in the legend indicate the geographic composition of the dataset.*

6 Conclusion

We presented Dasu, a measurement experimentation platform for the Internet's edge that supports and builds on broadband characterization as an incentive for adoption. We described Dasu's design and implementation and used our current deployment to demonstrate how participating nodes collectively offer broad network coverage, high availability and fine-grained synchronization to enable Internet measurement experimentation.

Dasu represents but a single point in a large design space. We described our rational for our current design choices, but expect to revisit some of these decisions as we learn from our own and other experimenters' use of the platform.

We presented three case studies that demonstrate Dasu's capabilities and illustrate the unique perspective it brings to Internet measurement. As part of ongoing work, we are exploring the use of node availability prediction for experimentation, approaches to ensure the integrity of experimental results, and allowing fine-grained control of experiments by end users.

The Dasu client is open source and available for download from http://azureus.sourceforge.net/plugin_details.php?plugin=dasu.

Acknowledgements

We would like to thank Lorenzo Alvisi, our shepherd Rebecca Isaacs, and the anonymous reviewers for their invaluable feedback. We are always grateful to Paul Gardner for his assistance with Vuze and the users of our software. This work was supported in part by the National Science Foundation through Awards CNS 1218287, CNS 0917233 and II 0855253 and by a Google Faculty Research Award.

References

[1] Broadband & IPTV progress report. http://www.broadband-forum.org/news/

download/pressreleeases/2012/BBF_IP&TV_Presentation12.pdf, March 2012.

[2] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a large european ixp. In *Proc. of ACM SIGCOMM*, 2012.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. ACM SOSP*, 2001.

[4] B. Augustin, B. Krishnamurthy, and W. Willinger. IXPs: mapped? In *Proc. of IMC*, 2009.

[5] BGP4. IPv4 Looking Glass Sites. http://www.bgp4.net/wiki/doku.php?id=tools:ipv4_looking_glasses.

[6] Z. S. Bischof, J. S. Otto, M. A. Sánchez, J. P. Rula, D. R. Choffnes, and F. E. Bustamante. Crowdsourcing ISP characterization to the network edge. In *Proc. of W-MUST*, 2011.

[7] Caida. Ark. http://www.caida.org/projects/ark/.

[8] M. Casado and T. Garfinkel. Opportunistic measurement: Spurious network events as a light in the darkness. In *Proc. of HotNets*, 2005.

[9] K. Chen, D. R. Choffnes, R. Potharaju, Y. Chen, F. E. Bustamante, D. Pei, and Y. Zhao. Where the sidewalk ends: extending the Internet AS graph using traceroutes from P2P users. In *Proc. ACM CoNEXT*, 2009.

[10] D. R. Choffnes and F. E. Bustamante. Pitfalls for testbed evaluations of Internet systems. *SIGCOMM Comput. Commun. Rev.*, April 2010.

[11] D. R. Choffnes, F. E. Bustamante, and Z. Ge. Crowdsourcing service-level network event monitoring. In *Proc. of ACM SIGCOMM*, 2010.

[12] D. R. Choffnes, M. A. Sánchez, and F. E. Bustamante. Network positioning from the edge: an empirical study of the effectiveness of network positioning in P2P systems. In *Proc. IEEE INFOCOM*, 2010.

[13] A. Dhamdhere and C. Dovrolis. Ten years in the evolution of the Internet ecosystem. In *Proc. of IMC*, 2008.

[14] A. Dhamdhere and C. Dovrolis. The Internet is flat: modeling the transition from a transit hierarchy to a peering mesh. In *Proc. ACM CoNEXT*, 2010.

[15] M. Dischinger, A. Haeberlen, I. Beschastnikh, K. P. Gummadi, and S. Saroiu. SatelliteLab: Adding heterogeneity to planetary-scale network testbeds. In *Proc. of ACM SIGCOMM*, 2008.

[16] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proc. of IMC*, 2007.

[17] FCC. Broadband network management practices – en banc public hearing, February 2008. http://www.fcc.gov/broadband_network_management/hearing-ma022508.html.

[18] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. The flattening Internet topology: natural evolution, unsightly barnacles or contrived collapse? In *Proc. of PAM*, 2008.

[19] C. G. Gray and D. R. Cheriton. Leases: An efficient fault-tolerant mechanism for distributed file cache consistency. In *Proc. ACM SOSP*, 1989.

[20] 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 *Proc. USENIX OSDI*, 2004.

[21] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker. On routing asymmetry in the Internet. In *Proceedings of IEEE Globecom*, 2005.

[22] Y. He, G. Siganos, M. Faloutsos, and S. Krishnamurthy. Lord of the links: a framework for discovering missing links in the Internet topology. *IEEE/ACM Transactions on Networking*, 2009.

[23] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed. ZooKeeper: wait-free coordination for Internet-scale systems. In *Proc. USENIX ATC*, 2010.

[24] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. Van Wesep, T. Anderson, and A. Krishnamurthy. Reverse traceroute. In *Proc. of USENIX NSDI*, 2010.

[25] Keynote. Internet health report. http://internetpulse.net/.

[26] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Proc. of ACM SIGCOMM*, 2010.

[27] J. Ledlie, P. Gardner, and M. Seltzer. Network coordinates in the wild. In *Proc. of USENIX NSDI*, 2007.

[28] H. M. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: an information plane for distributed systems. In *Proc. USENIX OSDI*, 2006.

[29] MLabs. Network diagnostic tool. http://www.measurementlab.net/run-ndt/.

[30] J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante. Content Delivery and the Natural Evolution of DNS: remote dns trends, performance issues and alternative solutions. In *Proc. of IMC*, 2012.

[31] PlanetLab. PlanetLab. http://www.planet-lab.org/.

[32] H. Pucha, Y. C. Hu, and Z. M. Mao. On the impact of research network based testbeds on wide-area experiments. In *Proc. of IMC*, 2006.

[33] M. Rabinovich, S. Triukose, Z. Wen, and L. Wang. Dipzoom: The Internet measurements marketplace. In *Proc. IEEE INFOCOM*, 2006.

[34] P. Ramamurthy, V. Sekar, A. Akella, B. Krishnamurthy, and A. Shaikh. Remote profiling of resource constraints of web servers using mini-flash crowds. In *Proc. USENIX ATC*, 2008.

[35] RIPE. RIPE atlas. http://atlas.ripe.net/.

[36] SamKnows. Accurate broadband information for consumers, governments and ISPs. http://www.samknows.com/.

[37] Y. Shavitt and E. Shir. DIMES: Let the Internet measure itself. *SIGCOMM Comput. Commun. Rev.*, 35(5), October 2005.

[38] C. R. Simpson, Jr and G. F. Riley. NETIhome: A distributed approach to collecting end-to-end network performance measurements. In *Proc. of PAM*, 2004.

[39] N. Spring, L. Peterson, A. Bavier, and V. Pai. Using PlanetLab for network research: Myths, realities and best practices. *ACM SIGOPS Op. Sys. Rev.*, 2006.

[40] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: a public Internet measurement facility. In *Proc. USENIX USITS*, 2003.

[41] A.-J. Su, D. Choffnes, A. Kuzmanovic, and F. Bustamante. Drafting behind Akamai: Travelocity-based detouring. In *Proc. of ACM SIGCOMM*, Sep. 2006.

[42] S. Sundaresan, W. de Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Broadband internet performance: a view from the gateway. In *Proc. of ACM SIGCOMM*, 2011.

[43] Vuze. Vuze. http://www.vuze.com/.

[44] K. Xu, Z. Duan, Z.-L. Zhang, and J. Chandrashekar. On properties of Internet exchange points and their impact on AS topology and relationship. In *NETWORKING*. 2004.

[45] R. Zhang, C. Tang, Y. C. Hu, S. Fahmy, and X. Lin. Impact of the inaccuracy of distance prediction algorithms on Internet applications: an analytical and comparative study. In *Proc. IEEE INFOCOM*, 2006.