A measurement experimentation platform at the Internet's edge

Mario A. Sánchez† John S. Otto† Zachary S. Bischof† David R. Choffnes‡
Fabián E. Bustamante† Balachander Krishnamurthy* Walter Willinger°[1]
†Northwestern University ‡Northeastern University *AT&T Labs-Research °Niksun, Inc.

***Abstract*—Poor visibility into the network hampers progress in a number of important research areas, from network troubleshooting to Internet topology and performance mapping. This persistent, well-known problem has served as motivation for numerous proposals to build or extend existing Internet measurement platforms by recruiting larger, more diverse vantage points. Capturing the edge of the network, however, remains an elusive goal.**

We argue that at its root the problem is one of incentives. Today's measurement platforms build on the assumption that the goals of experimenters and those hosting the platform are the same. As much of the Internet growth occurs in residential broadband networks, this assumption no longer holds.

We present a measurement experimentation platform that reaches the network edge by explicitly aligning the objectives of the experimenters with those of the users hosting the platform. Dasu – our current prototype – is designed to support both network measurement experimentation and broadband characterization. Dasu has been publicly available since July 2010 and has been installed by over 100,000 users with a heterogeneous set of connections spreading across 2,431 Autonomous Systems (ASes) and 166 countries. We discuss some of the challenges we faced building and using a platform for the Internet's edge, describe its design and implementation, and illustrate the unique perspective its current deployment brings to Internet measurement.

***Index Terms*—Internet Measurements, Internet Experimentation.**

I. 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 [1], [2] 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 [3]–[6]. Today's measurement and experimentation platforms offer two basic incentive models for adoption – cooperative and altruistic. In cooperative platforms such as PlanetLab [7] and RIPE Atlas [8] an experimenter interested in using the system must first become part of it. Other platforms such as SatelliteLab [3] and DIMES [5] have opted instead for an altruistic approach in which users join the platform for the betterment of science. Despite these efforts, capturing the diversity of the commercial Internet – including end-hosts in homes and small businesses – at sufficient scale remains an elusive goal [9], [10].

We present a measurement experimentation platform that reaches the network edge by explicitly aligning the objectives of the experimenters with those of the users hosting the platform. Dasu[2] – our current prototype – is designed to support network measurement experimentation and broadband characterization. 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 [11]. Dasu has been publicly available since June 2010 and has been installed by 100,118 users with a heterogeneous set of connections spreading over 2,431 166 Autonomous Systems (ASes) and across 166 countries.

The strengths and challenges of this new class of experimental platforms come from their inclusion of end users' devices at the network edge. Increased network coverage comes, for starters, at the price of radically more complex and less dependable platform environments at homes and coffee shops. The scale, dynamics and diversity of these new settings raise questions about the class of experiments feasible and the way in which experiments are declared, deployed and their execution controlled – *How does one define experiments for thousands of highly volatile nodes? How can we efficiently utilize the newly available resources for experimentation while controlling the performance impact of these experiments on users hosting the platform and the Internet as a whole?*

In this paper, we address key algorithms and system design issues in building and using a larger-scale, experimental platform for the Internet's edge. Our main contributions are:

- We characterize the challenging setting of edge-network experimentation platforms and show that, despite their increasing complexity, they offer a sufficiently stable and resource-rich environment to host a platform's vantage points (§III).
- We present the design and implementation of Dasu – an extensible platform for measurement experimentation for

[1] Work done while at AT&T Labs–Research.

[2] Dasu is a Japanese word that means "to reveal" or "to expose".

the Internet's edge (§IV).

- We characterize Dasu's current deployment and present results demonstrating how the participating nodes collectively offer broad network coverage, sufficiently high availability and fine-grained synchronization to enable Internet measurement experimentation (§V).
- We present a case study that illustrates the unique perspective that a platform like Dasu brings to Internet measurement (§VII) and discuss our early experiences sharing it with third-party experimenters (§VI).

In the following section, we provide further motivation and point out the goals and challenges of our work. We present our closing thoughts and some open areas of future work in Section IX.

II. Goals and Challenges

The lack of network and geographic diversity in current Internet experimentation platforms has been long recognized [1], [2]. Most Internet measurement and system evaluation studies rely on dedicated infrastructure [7], [12], [13] which include nodes primarily located in well-provisioned academic or research networks and are not representative of the larger Internet — with end-hosts in homes, small businesses and Internet cafes, connected over DSL, dial-up and cable.

Several research projects have pointed out the implications this "lack of representativeness" has on efforts to generalize the results of network measurements and have cast doubts on the conclusions drawn from evaluations of networked systems (e.g. [2], [14]–[17]). A comparative analysis of the paths between PlanetLab nodes and between nodes in residential networks illustrates some of these issues. These two sets of paths have been shown to traverse different parts of the network [11], exhibit different latency and packet loss characteristics [18], [19] and result in different network protocol behaviors [20].

We argue that an experimental platform that captures the network and geographic diversity of the Internet, should be deployed at end user's devices, at scale. It should be hosted at the network edge, to provide visibility into this missing part of the Internet. To support time-dependent and long-running experiments, such a platform should also 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. Dasu is an experimental platform designed to match these goals.

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. Its first instantiation was built as an extension to the most popular large-scale peer-to-peer system – BitTorrent. Both strengths and challenges of a platform like Dasu stem from its inclusion of measurement 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' behaviors. 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 experiments that can be safely assigned to clients. As we show in Sec. V, typical usage patterns and comparatively long session times of BitTorrent users means Dasu can attain nearly continuous availability to launch measurement experiments. Any such 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 efficiently utilize its vantage points while controlling the impact that experiments may have on underlying network and system resources, such a platform must support coordinated measurements among large numbers of hosts worldwide, each of which is subject to user interaction and interference.

These challenges are not bound to Dasu or its current instantiation but, we argue, are common to any sufficiently large, network experimentation platform for the Internet's edge. Since Dasu's first deployment, we have developed and released a stand-alone version of this client and designed an altogether new platform based on namehelp, an end-host solution to the tensions between content distribution and public DNS usage [21].

III. The complex edge-network environment

A key challenge of any experimental platform for the Internet's edge is the unmanaged complexity of its vantage points' environments. Compared with the managed setting of typical experimental platforms (e.g., dedicated servers in academic/research networks, as in Planetlab), the environments hosting these new platforms' vantage points can include multiple networked devices that frequently join and leave the network and support a handful of users and applications, all competing for limited computational and network resources.

Among these challenging environments, home networks are probably among the most complex and their complexity seems posed to only worsen in the immediate future.[3] Considering this raises as a question on the feasibility of experimentation platforms for the Internet's edge, in this section we present results from an analysis of home networks based on a dataset of passive and active measurements collected from Dasu clients over a 6-month period between February 24, 2012 and August 23, 2012. We use this dataset to characterize home networks and device usage, and show that despite their growing complexity, they offer a sufficiently stable and resource rich environment to host an edge-network measurement platform.

A. Exploring the complexity of home networks

To analyze the complexity of today's home networks, we first measure the number of networked devices typically found at home. We estimate [4] this by counting the number of devices found for a subset of ≈4.6K of our clients' home networks

[3] http://www.tvtechnology.com/cable,-satellite,-iptv/0149/smart-connected-devices-shipments-to-exceed–billion-by–says-idc/224955

[4] This approach could miss devices without –and, potentially, over-count a few devices with– multiple UPnP services

using UPnP –Universal Plug and Play– discovery messages.[5] UPnP is a set of networking protocols that allows networked devices to transparently discover each other and establish functional network services (data sharing, communications, and entertainment). During the measurement period, each client periodically broadcasted UPnP discovery messages and recorded, for each responding device, the reported information including the device's UUID and UDN, device type, model name and number.

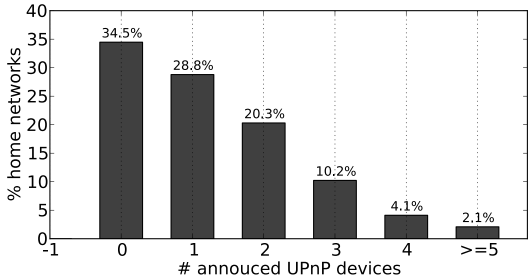


Fig. 1: UPnP-enabled devices in home networks.

Figure 1 shows the histogram of home networks by the number of UPnP announced devices found. While 34.5% of sampled home networks have no UPnP devices announcing their presence, over 65% of them have at least 1 device, and more than 16% have 3 or more devices. While a first approximation, these results are sufficient to illustrate the complexity of today's home networks, that is, for nearly two-thirds of locations, we must consider the impact of other devices on the network.

B. Detecting Cross-traffic

As the number of devices connected to the home network increases, so does the likelihood that the access link will be used by multiple devices simultaneously, potentially interfering with measurement experiments. To detect the presence of cross traffic from other devices one can leverage UPnP-enabled gateways, if available. By periodically querying these gateways for traffic counters across their WAN interface, it is possible to identify periods when the number of packets or bytes sent or received is high enough to impact experiments and simply discard (if passive) or postpone (if active) those measurements [22] [23].

Considering that the goal of the UPnP protocol is to simplify the management of home networks, it would not be surprising to find that the presence of UPnP enabled gateways increases with the complexity of home networks increases. Our dataset shows this is indeed the case. Figure 2 presents a histogram of the prevalence of homes with UPnP-enabled gateways clustered according to the number of detected devices; the bar graph show the expected trend with the percentage of UPnP-enabled gateways increasing from 45% to 81% as the number of extra devices (i.e., beyond the host) goes from one to four.

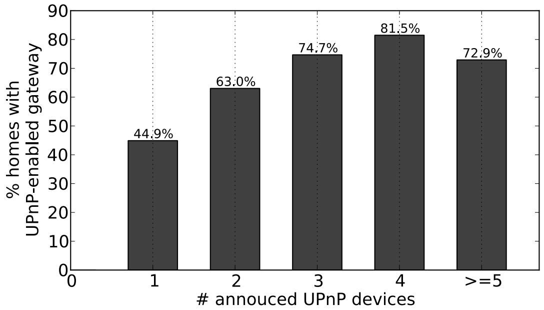


Fig. 2: Prevalence of UPnP-enabled gateways in sampled homes, clustered based on number of UPnP-enabled devices announced.

Even in environments with a single, unmanaged host, other applications can generate sufficient cross-traffic to invalidate results of certain concurrent experiments. To detect traffic originating from other applications in the same host, one can rely on `netstat`, a network statistics tool available in all major platforms, to capture the number of bytes sent and received from the host and compare it against the amount of traffic monitored by our client.

C. Access-link Utilization

The previous paragraphs argues that it is possible to detect the presence of cross-traffic in the most complex of edge-network settings, but leave open the question of there being sufficiently long time periods with sufficiently low or no cross-traffic to launch experiments. To answer this, we use data collected between December 2012 and January 2013 from 1,377 different end users and select settings with UPnP-enabled gateways that properly report traffic counters (≈40% of them). Using this dataset, we analyze the fraction of time users are alone in the network and quantify the utilization of home networks' access link.

Network utilization is known to vary significantly during a day. We define the *common hour* for each user as the median utilization hour and use this as representative of each user. To compute the common hour, we measure, for every hour an individual user is online, the fraction of time in which the traffic in their access link (captured using UPnP counters) is higher than that generated by the end host (captured using `netstat`) and sort these utilization values in increasing order. From this, we select the *common hour* for each user as the hour over which 50% of sampled-hours fall.

Using this common period, we first look at the fraction of time the user's traffic shares the access link with traffic generated by other devices in the network. Figure 3 shows that for the *common hour* over 60% of users see no other traffic in the network (i.e. they are sole users of the access link) and for an additional 23% of users the fraction of time that the link is also utilized by other devices is less than 50% (<30 minutes).

The figure also shows that for 13% of the users their common hour have a utilization of 1, which means that in their representative hour the access-link is always shared with other devices in the network. We next look at the access-link utilization for this set of users. For each hour the users

[5] This dataset, as any other, may have some inherent bias but we posit that Dasu/BitTorrent users can be seen as early adopters and thus, in a sense, offer a worst-case scenario in terms of the level of complexity in home networks.

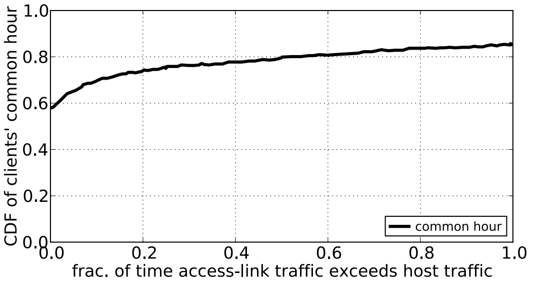
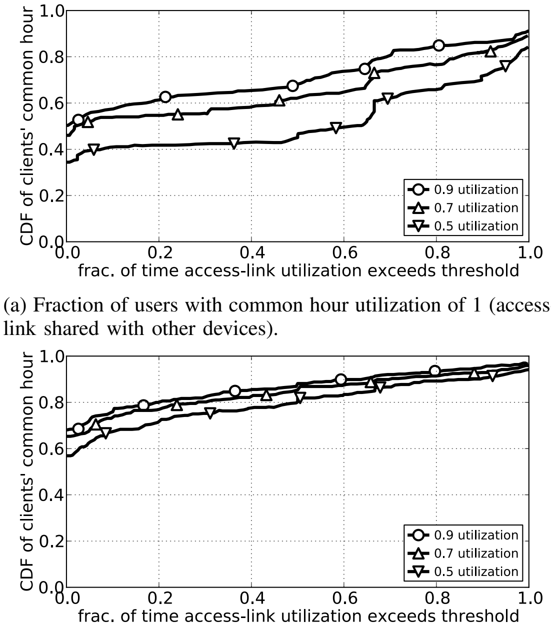


Fig. 3: Fraction of time the host machine shares the access-link with other devices in the network.



(a) Fraction of users with common hour utilization of 1 (access-link shared with other devices).

(b) All users.

Fig. 4: Fraction of time access-link utilization surpasses various utilization thresholds, (4a) for users with common hour utilization of 1, i.e. access-link shared with other devices; (4b) for all users.

are online we compute the fraction of time in which the utilization of their access-link (captured using UPnP counters) is higher than different thresholds of their maximum link capacity (obtained through the use of NDT [24]). Figure 4a shows the fraction of time the utilization of the access-link surpasses different utilization thresholds. As the figure shows, for 35% of these clients, the traffic present in their access-link never exceeds 50% of their link capacity at the common hour, and this fraction increases to 48% and 52% if we move our thresholds to 70% and 90% respectively. These results suggest that, even for users with high probability of encountering cross traffic in their network, the utilization of their link still allows for some measurements.

Finally, we extend this analysis to include every user in our dataset. The results are depicted in Figure 4b and show that for 60% of the clients the traffic in their access-link never exceeds 50% of their link capacity at the common hour, and that fraction increases to almost 70% when looking at link capacity of 90%. These results indicate that there is significant available capacity for running network measurement without adversely affecting other applications using the network.

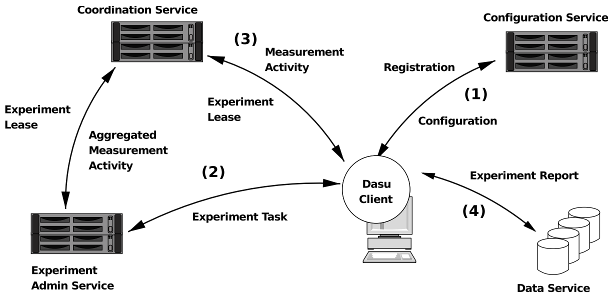


Fig. 5: Dasu system components.

We have shown that a measurement platform hosted at end-users' devices can effectively detect the presence of cross-traffic from other applications and devices in the network, and that there are sufficient opportunities for launching measurement experiments that do not affect nor are affected by concurrent end-user activity.

IV. Dasu Design

The following paragraphs describe Dasu's design, focusing on its support for measurement experimentation and partially guided by the findings reported in our previous section. After a brief system overview, we describe key system's components and the experimentation model they support.

A. 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 5 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.

B. The Dasu Client

Dasu runs at the edge of the network either as a standalone client or in the context of a network-intensive hosting application. Each client (Fig.6) includes a set of *Probe Modules* to passively collect application metrics and run active measurements. For experimentation, Dasu provides low-level measurement tools in the form of active probe modules that can be combined to build a wide range of measurement experiments. Currently available measurement primitives include traceroute, ping, Network Diagnostic Tool (NDT), HTTP GET and DNS resolution and JavaScript testing support (through PhantomJS [25]).

In addition to these active measurements, Dasu leverages the host's naturally-generated traffic as passive measurements (particularly in the context of broadband characterization [26]) by continuously monitoring the end-host Internet connection. This *Passive Monitoring* module is responsible for collecting relevant statistics from the host (as described in III-B) as well as the hosting application if one is being used. Devising an interface to expose these passively collected measurements to experimenters is part of future work.

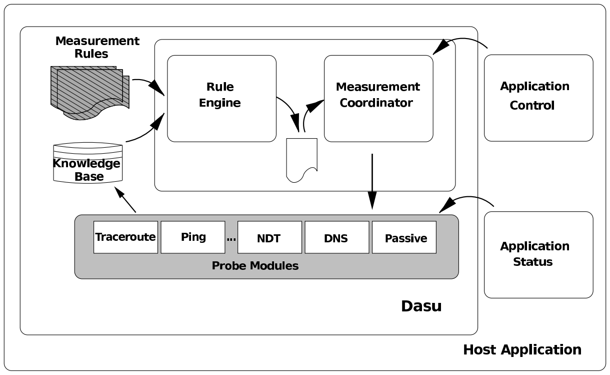


Fig. 6: Dasu client architecture.

C. Experiment Specification

Dasu is 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).

Related measurement platforms offer a variety of models covering the whole spectrum in terms of flexibility and expressiveness for experiment specification. These models range from a handful of parameterized commands, as in DIMES and Looking Glass servers, to a generous subset of a portable programming language, as in Seattle and Fathom [27]. Given our goal of supporting measurement experimentation rather than more general prototype testing and deployment, we opted instead for a rule-based declarative model for experiment specification.

In a rule-based 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 (modules). A collection of rules form a program and a set of related programs define an experiment.

Dasu provides experimenters with a set of measurement modules and a programmable API to combine them. Rules can be chained together to schedule measurements and handle results using the available measurement modules, comprising a measurement experiment. Although the number of measurement modules available is finite, this set is easily extensible by platform administrators. Primitives are transparently and easily updated without end user interaction and without the need to update the base client itself. While this architecture limits experimenters from implementing their own measurement primitives, it removes the security and safety issues associated with providing low-level access to raw sockets or access to privileged network ports.

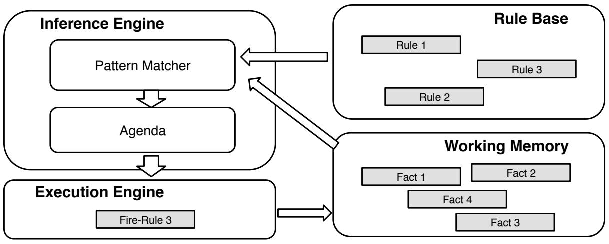


Fig. 7: Rule-based program execution environment.

The rule-based model provides a clean separation between experiment logic and state. By imposing strict constraints on rule syntax, rules can be safely verified through simple static analysis. In our experience, despite constraints, rules have proven to be a flexible and lightweight approach for specifying and controlling experiments. For instance, an experiment to compare routing asymmetry between commercial and research networks that examines the paths between the stub (Dasu) and research (PlanetLab) networks can be specified using only 3 different rules with an average of 24 lines of code per rule.

Tables I and II 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. Measurements 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. (Fig. 7).

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 8 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.

D. Detecting cross-traffic

To detect cross-traffic from other applications in the same host, Dasu leverages passively collected traffic statistics

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 I: Dasu API – Methods.

Probe	Params.	Description
PING	<dest-list (IP/name)>	Use the local host ping implementation to send $ECHO\_REQUEST$ packets to a host.
TRACEROUTE	<dest-list (IP/name)>	Print the route packets take to a network host.
NDT	[<server>]	Run the M-Lab Network Diagnostic Tool [24].
DNS	[<server>] | [<timeout>] | [<tcp/udp>] | [<options>] | <DNS-msg>] | <dest-list>	Submit DNS resolution request to a set of servers.
HTTP	[server] | [<port>] | [<HTTP-Req>] | <url-list>	Submit HTTP request to a a given $< host, port >$ pair.
PhantomJS	[url] | [<jsScriptt>]	Run functional javaScript tests against a given $< url >$.

TABLE II: Dasu API – Measurement modules currently supported.

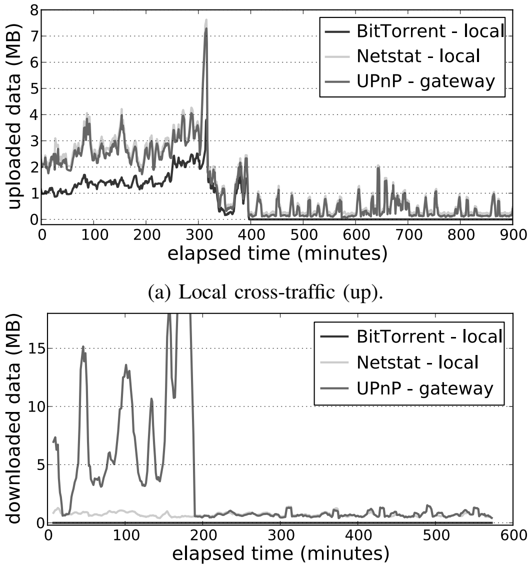
```
rule "(1) Resolve IP address through local DNS"
 when
  $fact : FactFireAction(action=="resolveIp");
 then
  addProbeTask(ProbeType.DNS, "example.com");
end

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
```

Fig. 8: Measurement experiment for debugging high latency DNS queries.

through the netstat tool, and the relevant signals collected from the hosting application (BitTorrent) when appropriate. For networks with UPnP-enabled gateways, Dasu periodically queries for traffic counters across the access link. While gateway UPnP traffic counters are not always accurate, such instances can be easily identified and accounted for. Dasu employs the technique described by DiCioccio et al. [28] where UPnP-enabled home gateways are periodically queried to measure traffic in the home network.

We now present some concrete examples of how traffic counters from UPnP-enabled gateways allow Dasu to disambiguate between different scenarios inside the network. Using data collected from our Dasu users we show how the presence of internal traffic can be identified and separated from traffic that uses the access link, both from the local host and other devices within the network.

Local cross-traffic and no cross-traffic. Figure 9a plots the upload activity of one Dasu client for a span of 15 hours in June 2012. Each of the three signals in the graph represents the number of downloaded bytes as reported by BitTorrent, netstat, and the gateway counters, respectively, in intervals of 30 seconds increment. The figure shows that the client is solely responsible for all the traffic present in the access link, but BitTorrent is not the only active application. As the figure shows, the curves that correspond to the local netstat counters and the UPnP-counters at the gateway overlap through the entire collection period (i.e., the client is the only device using the access link), but the curve that corresponds to BitTorrent traffic is much lower than that of netstat for the first five hours (300 minutes) of the session.



(a) Local cross-traffic (up).

(b) Cross-traffic (down).

Fig. 9: Traffic scenarios within the home network: (9a) local cross-traffic from other applications and no cross-traffic and (9b) download cross-traffic.

Cross-traffic from other devices. Figure 9b shows a different scenario, where there is significant cross-traffic from other devices in the home network. The figure plots the download activity seen from a client over a span of five hours. In this case, there is no BitTorrent content being downloaded (the BitTorrent signal is a flat horizontal line around 0 bytes), but there is local traffic being generated by other applications in the host device. However, for the first $\approx$ 200 minutes of the session, the traffic generated by the host devices represents only a small fraction of the total traffic present in the access link. The figure also shows the easily identifiable point at which the cross-traffic disappears.

E. 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 areas of security: 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 Experiment Administration 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 carefully controls the load its experiments impose on the local host, minimizing the impact that users' interactions (i.e., with the host and the application) can have on experiments' results. The Dasu client limits consumption of hosts' resources[6] 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. V-C). Dasu uses the 95th percentile of client's throughput rates measured by NDT to estimate the maximum bandwidth capacity of the host and continuously monitors host network activity. 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 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 predetermined quota limits have been reached.

F. 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.

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. Figure 10 illustrates the interaction between Dasu clients and the EA Service.

Submitting External Experiments. Dasu supports third-party experiments through the two-tier architecture described

[6] Currently 15% of any monitored resource.

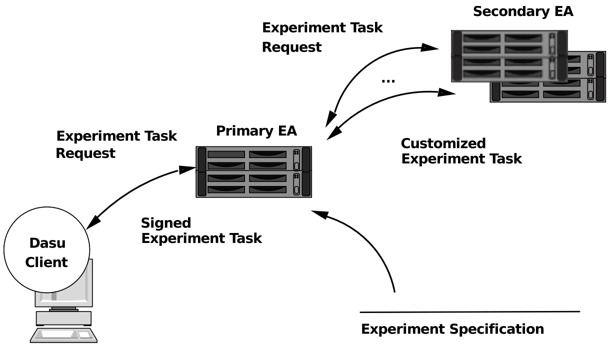


Fig. 10: Interaction between Dasu Clients and the Experiment Administration Service.

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. In Section VI we discuss our initial experience sharing the platform with external experimenters.

G. Coordination

In addition to controlling the load on and guaranteeing the safety of volunteer hosts, Dasu controls the impact that measurement experiments collectively have on the underlying network and system resources.

Most existing measurement platforms attempt to indirectly achieve this goal by limiting the number of probes each individual measurement node can perform through the use of local rate limits. This approach prevents a single node from overwhelming a specific destination, but imposes no limits on the collective actions of multiple measurement nodes –a large number of nodes can still probe the same destination at the same time, effectively causing a (DDoS) attack. The use of centralized distributed rate limits has been proposed [29] to achieve this goal where individual measurement nodes are required to contact a centralized entity before launching a probe, with the centralized server granting or denying the probe based on the aggregate system behavior. Aside from the obvious scalability constraints of such an approach, measurement nodes are left with little independence and end up becoming simple measurement extensions of the centralized server.

The challenge of coordinating the behavior of measurement nodes is compounded in platforms such as Dasu that recruit the help of ordinary users located at the edge of the network. On the one hand, such platforms enjoy an almost unlimited scalability; at the same time, the nodes suffer from reduced availability and local resource constraints. As such it is necessary to maximize their utilization while still imposing limits on the number of probes sent. For instance, a simple scheme that generates a centralized optimal probing schedule for nodes to follow is not feasible given the lack of control over client's availability. Additionally limiting the number of probes and clients assigned to a specific experiment is almost guaranteed to yield sub-optimal results as there is no assurance of when clients will launch the assigned probes (client resources are limited), or even how many of the assigned probes will actually be completed (a client might simply disappear in the middle of an experiment).

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. Our solution efficiently allows the scalable and effective coordination of measurements among potentially thousands of hosts while providing individual clients with enough flexibility to act on their own.

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 [30]. 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 [31]. 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.

The budget is computed in the following way:

Let,

d,	*destination*
M,	*aggregate max # probes per unit time to dest d*
m,	*max # of probes per unit time a client will launch*
n,	*# of clients in the experiment*
a_i,	*# of probes to dest d assigned to client i*
c_i,	*# of probes to dest d completed by client i*
p_i,	*completion rate of allowed probes in recent past*

Then,

$$Budget = \begin{cases} M/n & \text{if } M/n < ppm \\ ppm & \text{if } M/n > ppm \end{cases}$$

where,

$$ppm = \sum_{i=1}^{n} p_i * f(i)$$

$$f(i) = \begin{cases} a_i - c_i & \text{if } (a_i \text{ -} c_i) < m \\ m & \text{if } (a_i \text{ - } c_i) > m \end{cases}$$

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.

H. Synchronization

Dasu also provides support for Internet experiments that require synchronized client operation (e.g. [32], [33]). 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 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.

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 III: Internet penetration[6] and Dasu coverage (as percentage of its total population of 100,118) by January 2013.

V. Current Deployment

We have implemented Dasu as an extension to a popular BitTorrent client [34] (publicly available since June 2010) as well as a standalone client (publicly available since June 2013). The following description and analysis are based on the BitTorrent extension, as it offers a large and widespread client population.

To participating users, Dasu provides information about the service they receive from their ISP [26], [35]. Access to such information has proven sufficient incentive for widespread subscription with over 100K users who have adopted our extension with minimum advertisement.[7]

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

A. Dasu Coverage

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

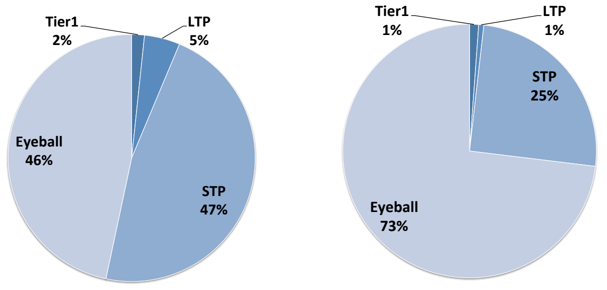
Given the high Internet penetration numbers in Europe and North America, the distribution of Dasu clients per region is not surprising. Note, however, the penetration of Dasu clients per region, measured as the percentage of countries covered. As the table shows, Dasu penetration is over 57% for most 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 July 2013, we have Dasu clients in 2,431 different ASes. We classify these ASes following a recently proposed approach [36], 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

[7] 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



(a) Peer Distribution

(b) AS Distribution

Fig. 11: Distribution of Dasu peers per AS (left). Distribution of ASes covered by Dasu peers (right).

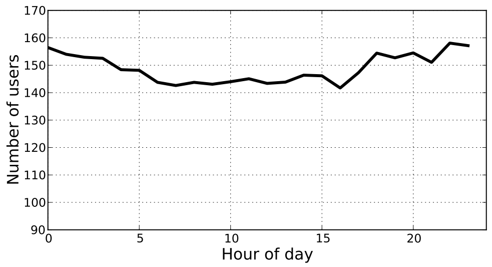


Fig. 12: Number of online Dasu clients over a 24-hour period. The fraction ranges from 39-44% of the total number of unique users, on average.

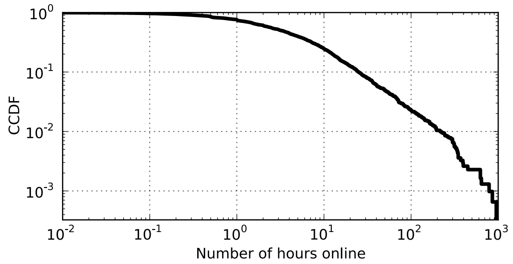


Fig. 13: Session time distribution of Dasu clients (time between their joining and leaving the system).

Figure 11a 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 11b 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.

B. 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 12 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. [7]. With respect to 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 13 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 $\approx$ 3 hours.

Given an average session time, the fraction of tasks that are able to complete depends on the duration of the task – a function of the number of actual measurements and the load at the client. To evaluate the impact of typical client load conditions on task completion times, we designed a controlled experiment in the context of the IXP mapping case study described in Sec. VII, consisting of a fixed number of traceroutes issued by clients to discover potential peerings. The experiment was designed in such a way that given ideal client conditions (sufficiently low CPU and bandwidth load) the minimum time required by any Dasu client to complete it would be $\approx$75 seconds (1.25 minutes); this time would serve as a reference point when comparing against completion times under typical client conditions. For this purpose, we selected a random set of Dasu clients over a one week period in August 2013 and assigned such tasks multiple times during that period. Figure 14 shows a cumulative distribution function of the median task completion time from 164 different clients that completed 10 or more such tasks during that period. The figure shows how for 90% of the Dasu clients the median task successfully completes in < 150 seconds with 50% of clients completing the task in less than 120 seconds (2 minutes).

Now we look at the fraction of assigned tasks to Dasu clients that are successfully completed. Given that we have no control over clients' availability or disconnection times, only a fraction of assigned tasks will be successfully completed. To perform this analysis we look at the tasks assigned to 349 different Dasu clients over a 2-week period in the month of August 2013, taking only into account clients who were assigned 5 tasks or more during that period. Figure 15 shows the complementary cumulative distribution function of the fraction of successfully completed tasks from those assigned to each client. The plot shows that a large fraction of clients are able to complete the majority of assigned tasks in the face of churn with 60% of clients completing 80% or more of the tasks assigned to them.

[7] The actual number of users changes drastically from day to day (and hour to hour). This is in part due to the nature of this new type of platforms, running on volunteer, regular Internet users' machines and in part due to the particular hosting application we rely on for our first instantiation (BitTorrent).

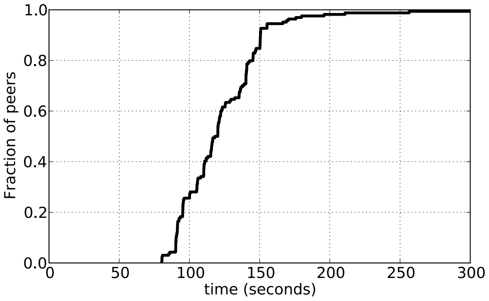


Fig. 14: CDF of median task completion times for tasks completed by Dasu clients. For 90% of the clients, the median task successfully completes in < 150 seconds (2.5 minutes).

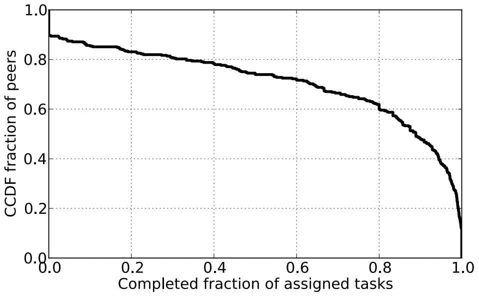


Fig. 15: CCDF of fraction of successfully completed tasks from those assigned. 60% of Dasu clients complete 80% or more of the tasks assigned to them.

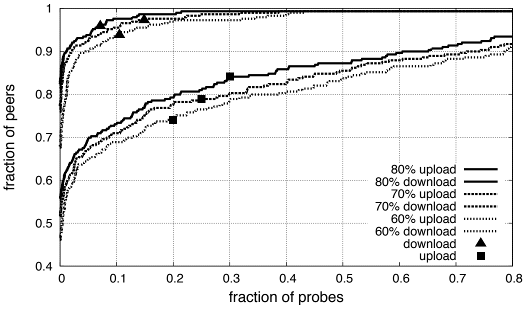


Fig. 16: Distribution of fraction of probes per peer that are delayed due to bandwidth constraints at the client.

C. 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 16 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 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.

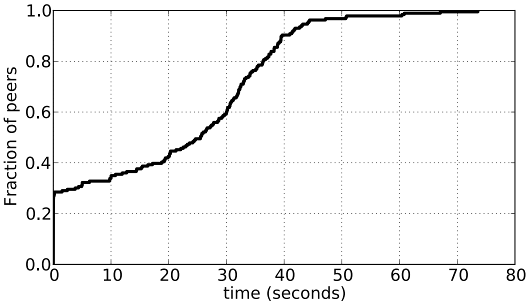


Fig. 17: CDF of median probe queue time for clients. For 30% of the clients, the median probe is launched $< 1sec.$ after being scheduled; with 60% of clients launching probes within 30 seconds of being scheduled.

Finally, we look at the amount of time probes are queued by the clients under typical load conditions from the moment they are requested until launched. Fig. 17 shows the queueing time of probes assigned to 186 Dasu clients for a given experiment over a 1-week period in July 2013. The figure plots the cumulative distribution function of the median probe-queue time on a per client basis for clients with at least 20 launched probes. The figure shows that for 30% of the clients, the median probe is launched $< 1sec.$ after being scheduled; with 60% of clients launching probes within 30 seconds of being scheduled.

D. 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. [32], 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 18 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. While Ramamurthy et al. [32] shows smaller synchronization times are achievable with this technique, Dasu's higher numbers are due to the ultra-conservative, yet not fundamental, limits we impose on the load Dasu adds to hosting nodes (which results on delayed probe scheduling depending on the host's load and activity). While we have not found our resulting synchronization granularity to be a limitation for any of the experiments deployed so far, we plan to revisit these self-imposed limits as part of future work.

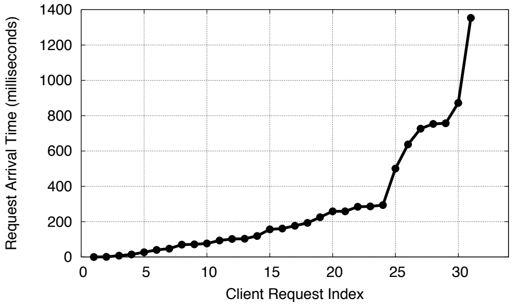


Fig. 18: Request arrival times at the target server. Approximately 80% of requests arrive within 300 ms.

VI. Third-party experimenters

Dasu has been available to external researchers since April 2013. Through the two-tiered architecture described in Sec. IV-F, we have already engaged with a handful of external groups that have or are using Dasu for their measurement experiments. Their projects, with goals that range from evaluating congestion control mechanisms in the wild, to discovering the physical connectivity between interconnected ASes in a geographic-aware manner, all share the need for a large and diverse set of vantage point locations.

To enable external researchers to engage Dasu clients in their experiments, two different secondary EA servers are shared with each group: a *Secondary EA*, responsible for task parameterization and client allocation, and a *Secondary EA_dataColector*, responsible for collecting and storing experiments' results. External experimenters modify the Secondary EA servers to implement the logic required by their particular experiment setup. Secondary EA serves interact with the *Primary EA* which is responsible for allocating clients across projects, authenticating and signing the assigned experiments.

These two-tier architecture provides experimenters sufficient flexibility and control over their experiments, while remaining simple and sufficiently scalable for the class of experiments we have deployed. The experiment curation model it implements –where each and every experiment must be approved by the system administrators– has allowed us to safely explore the sharing of our platform with external groups but has also proven to be its largest scalability roadblock. Exploring alternative models for sharing large-scale, platforms running at the Internet's edge is an interesting avenue for future work.

VII. Case study of a platform at the Internet's edge

In this section, we present a concrete case study that illustrates the unique perspective a programmable, edge-based platform with a wide-spread and diverse set of vantage points brings to Internet measurement. Additional case studies can be found in [37].

A. 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.

The value of Dasu. 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. [8]

Experiment setup. This experiment extends the work by Otto et al. [21], which examined the impact of varying the amount of 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.

Results. 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.[9] Figure 19 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

[8] To the best of our knowledge, of the existing platforms, the Seattle testbed is the only other platform one could potentially use to carry out this particular experiment, albeit with a different user footprint.

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

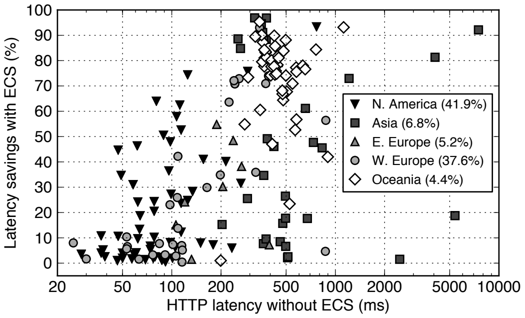


Fig. 19: HTTP latency vs. the performance benefits provided by ECS, by geographic region. Percentages in the legend indicate the geographic composition of the dataset.

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.

VIII. Related Work

Our work shares goals with and builds upon ideas from several prior large-scale platforms targeting Internet experimentation. In the following paragraphs we review some of these projects and key ideas, structuring our discussion around the platform incentive model for adoption. We then describe key differences with network measurement platforms whose vantage points are located in stub and commercial networks as these share some common characteristics and challenges as our architecture.

In platforms relying on a *cooperative model* for adoption, an experimenter interested in using the system must first become part of it. Systems built using this model largely assume that the goals of those hosting the platform and the experimenters that use it are aligned. Not surprisingly, the majority of these platforms' nodes belong to well-provisioned and supported academic or research (GREN) networks. Examples of systems that follow this model include PlanetLab [7], RIPE Atlas [8] and DipZoom [4].

Alternatively, in measurement platforms that follow an *altruistic model*, participants join the platform for the betterment of science. Ark [12], RON [13], Scriptroute [38], and public looking-glass traceroute servers (LG) are all examples of this type of platform, albeit ones hosted on nodes located in well-provisioned networks. On the other side of the spectrum are platforms that still follow the same model but rely on the help of ordinary users located at the edge of the network. Such platforms include DIMES [5], SatelliteLab [3], NETI@home [39] and Seattle [6]. We concentrate on the latter because they share similar challenges/characteristics to those of Dasu.

NETI@home for example, passively collects network performance statistics from end-systems. SatelliteLab, on the other hand, looks to improve the heterogeneity of testbeds by subjecting traffic to network conditions that would be experienced if the applications were run on the volunteers' hosts, without them having to actually contribute any resources for code execution. DIMES, probably the closest in spirit to our implementation, aims to gather topological information on the Internet. To achieve this goal, agents of the platform only require a limited subset of measurement tools (TCP/UDP/ICMP ping/traceroutes, Paris Traceroute) and don't require a high degree of programmability or fine-grain coordination capabilities. Finally, the Seattle research and educational testbed provides rapid prototyping and measurement capabilities using resources provided by end users on their existing devices. Given its original goal as an educational testing platform, Seattle supports a very flexible language for experiment specification based on a restricted subset of Python. Considering our goal of supporting measurement experimentation and our intended hosting environment, we opted for explicitly excluding the execution of arbitrary code, defining a well-structured module-based form of extensibility, a restricted rule-based language for experiment specification, and a new model for defining and deploying experiments, and controlling their aggregated impact.

IX. Conclusion

We presented Dasu, a measurement experimentation platform for the Internet's edge. Dasu reaches the network edge by explicitly aligning the objectives of the experimenters with those of the users hosting the platform, supporting both network measurement experimentation and broadband characterization. We discussed some of the challenges we faced building and using a platform for the Internet's edge, described its design and implementation, and illustrated the unique perspective its current deployment brings to Internet measurement.

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. For instance, while Dasu's two-tiered architecture has been effective for sharing system resources across several experiments and external groups, we are facing the scalability limitations of our experiment curation model and leave the investigation of alternatives (e.g., such as those described in [40]) as part of future work. Other interesting research directions include extending the platform beyond a single implementation that relies on a single incentive for adoption and integrating it with more traditional and stable experimental platforms like PlanetLab. Finally, determining the class of experiments that, while feasible, may be inappropriate for a platform hosted by end users (e.g., censorship monitoring) remains to be explored.

The Dasu client is open source and available for download from http://aqualab.cs.northwestern.edu/projects/Dasu.

X. Acknowledgments

We would like to thank Lorenzo Alvisi, Rebecca Isaacs, Aaditeshwar Seth and the anonymous reviewers for their invaluable feedback. We are always grateful to Paul Gardner for his assistance with Vuze and the many users of our software.

This research 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] 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.

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

[3] 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.

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

[5] Y. Shavitt and E. Shir, "DIMES: Let the Internet measure itself," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, October 2005.

[6] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: a platform for educational cloud computing," in *Proc. of the 40th ACM technical symposium on Computer science education*, ser. SIGCSE '09, 2009.

[7] PlanetLab, "An open platform for developing, deploying, and accessing planetary-scale services," Jul. 2013, accessed: 2013-07-26. [Online]. Available: http://www.planet-lab.org/

[8] RIPE, "RIPE atlas," May 2013, accessed: 2013-05-21. [Online]. Available: http://atlas.ripe.net/

[9] Keynote, "Internet health report," Jan. 2013, accessed: 2013-01-7. [Online]. Available: http://internetpulse.net/

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

[11] 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.

[12] Caida, "Ark," Jul. 2013, accessed: 2013-07-26. [Online]. Available: http://www.caida.org/projects/ark/

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

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] L. DiCioccio, R. Teixeira, and C. Rosenberg, "Measuring and characterizing home networks," in *Proc. ACM SIGMETRICS*, 2012.

[23] L. DiCioccio, R. Teixeira, and Rosenberg, "Characterizing home networks with HomeNet Profiler," Technicolor, Tech. Rep., 09 2011, cP-PRL-2011-09-0001.

[24] MLabs, "Network diagnostic tool," Jun. 2013, accessed: 2013-06-6. [Online]. Available: http://www.measurementlab.net/run-ndt/

[25] A. Hidayat, "Phantomjs," http://phantomjs.org.

[26] 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.

[27] M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson, "Fathom: A browser-based network measurement platform," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, ser. IMC '12. ACM, 2012.

[28] L. DiCioccio, R. Teixeira, M. May, and C. Kreibich, "Probe and pray: Using UPnP for home network measurements," in *Proc. of PAM*, 2012.

[29] Z. Wen, S. Triukose, and M. Rabinovich, "Facilitating focused internet measurements," in *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '07. New York, NY, USA: ACM, 2007, pp. 49–60. [Online]. Available: http://doi.acm.org/10.1145/1254882.1254889

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

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

[32] 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.

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

[34] Vuze, "Vuze," Jul. 2013, accessed: 2013-07-26. [Online]. Available: http://www.vuze.com/

[35] SamKnows, "Accurate broadband information for consumers, governments and ISPs," Jul. 2013, accessed: 2013-07-26. [Online]. Available: http://www.samknows.com/

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

[37] M. A. Sánchez, J. S. Otto, Z. S. Bischof, D. R. Choffnes, F. E. Bustamante, B. Krishnamurthy, and W. Willinger, "Dasu: Pushing experiments to the Internet's edge," in *Proc. of USENIX NSDI*, 2013.

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

[39] 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.

[40] J. Cappos, A. Dadgar, J. Rasley, J. Samuel, I. Beschastnikh, C. Barsan, A. Krishnamurthy, and T. Anderson, "Retaining sandbox containment despite bugs in privileged memory-safe code," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. ACM, 2010.

Mario A. Sánchez received his M.S. and Ph.D. degrees in computer science from Northwestern University in 2014 and 2011, respectively. He holds a Master of Science degree in Telecommunications from the University of Maryland at College Park and a B.E. in Telecommunications from Pontificia Universidad Católica Madre y Maestra (Dominican Republic).

Mario joined HP Labs, Palo Alto as a research scientist in the fall of 2014 . His research interests include experimental computer systems and networking, with a current focus on Internet measurements and distributed systems. He is a former recipient of the Fulbright scholarship.

John S. Otto was born in Atlanta, Georgia, in 1985. He received the B.S. degree in computer science from Northwestern University, Evanston, IL, in 2007, the M.S. in computer science from Northwestern in 2011 and the Ph.D. degree in computer science from Northwestern in 2013.

From 2007 to 2014, he was a Research Assistant at Northwestern University. In summer 2010 he interned at Telefónica I+D and in summer 2011 he interned at AT&T Labs–Research. Since 2014, he has been a Software Engineer at Google in Chicago, IL. His research interests include the economic and performance trade-offs between approaches for Internet-based content distribution.

Zachary S. Bischof received his B.S. and M.S. degrees from Northwestern University in Evanston, IL, in 2009 and 2013, respectively. He is currently a PhD candidate at Northwestern University working with Prof. Fabián Bustamante. His research interests focus on developing and evaluating large-scale distributed systems and characterizing broadband services and how they affect user behavior.

Fabián E. Bustamante received his M.S. and Ph.D. degrees in computer science from the Georgia Institute of Technology in 1997 and 2001. Before joining Georgia Tech, he studied and taught at the Universidad Nacional de La Patagonia San Juan Bosco (Argentina), from which he received both a 3-year and a 5-year-and-project degrees in computer science. He is currently a Full Professor in the Department of Electrical Engineering and Computer Science at Northwestern University. His research interests span several areas of experimental systems, with a focus on networks and distributed computing. Prof. Bustamante serves in the Editorial Board of the IEEE/ACM Transactions on Networking, the ACM SIGCOMM Computer Communication Review and the IEEE Internet Computing. He is also the chair of the steering committee of the IEEE Peer-to-Peer conference, was the co-chair for SIGCOMM 2014 and is an active member of the program committees of SIGCOMM, IMC, CoNEXT and ICDCS, among others. Prof. Bustamante is a senior ACM member, a member of USENIX and the IEEE, a recipient of the National Science Foundation CAREER award and the Science Foundation of Ireland E.T.S. Walton Visitor Award.

David Choffnes received the B.A. degree from Amherst College, Amherst, MA, in 2002, and the M.S. and Ph.D degrees from Northwestern University, Evanston, IL, in 2006 and 2010, respectively. He was a postdoc at the University of Washington as an NSF/CRA Computing Innovations Fellow and was also supported by a Google Research Award.

David Choffnes is currently an assistant professor in the College of Computer and Information Science at Northeastern University, where he started in 2013. His research interests are primarily in the areas of distributed systems and networking, with a recent focus on mobile systems.

Balachander Krishnamurthy is presently a lead member of technical staff at AT&T Labs–Research. His focus of research is in the areas of Internet privacy, Online Social Networks, and Internet measurements. He co-founded the successful ACM SIGCOMM Internet Measurement Conference and more recently the first ACM Conference on Online Social Networks (cosn.acm.org). He has been on the thesis committee of several PhD students, and has collaborated with over seventy five researchers worldwide. Bala's papers can be found at http://www.research.att.com/˜bala/papers

Walter Willinger is Chief Scientist at Niksun, Inc., Princeton, NJ. Before joining Niksun, he worked at AT&T Labs–Research in Florham Park, NJ (1996-2013) and at Bellcore Applied Research in Morristown, NJ (1986-1996). He received the Diplom (Dipl. Math.) from the ETH Zurich and his M.S. and Ph.D. degrees from the School of ORIE, Cornell University, Ithaca, NY. He is a Fellow of ACM, IEEE, and SIAM, an AT&T Fellow, and co-recipient of the 1995 IEEE Communications Society W.R. Bennett Prize Paper Award, the 1996 IEEE W.R.G. Baker Prize Award, and the 2005 ACM/Sigcomm Test-of-Time Paper Award. His paper "On the Self-Similar Nature of Ethernet Traffic" is featured in "The Best of the Best - Fifty Years of Communications and Networking Research", a 2007 IEEE Communications Society book compiling the most outstanding papers published in the communications and networking field in the last half century.