A measurement experimentation platform at the Internet’s edge

Mario A Sanchez, John S. Otto Zachary S. Bischof, David R. Choffnes, Fabian E. Bustamante, Balachander Krishnamurthy, Walter Willinger

Presented By: Dipendra Jha
Motivation/Related Works

- Poor visibility into the network hampers research progress
- Numerous existing Internet Measurement Platforms rely on dedicated infrastructure
- Cooperative model- PlanetLab, RIPE Atlas, DipZoom)
- Altruistic model- Ark, RON, Scriptroute)
- Altruistic, but relying on end users (DIMES, SatelliteLab, NETI@home, Seattle)
- ‘Lack of representativeness’ to generalize network measurements
- Capturing the edge of the network is hard goal
- Experimental platform deployed at end user’s devices at scale, with continuous availability
Complex Edge-Network Environment

- Over 65% users have at least 1 UPnP device, and more than 16% have 3 or more devices
- UPnP-enabled gateways increasing from 45% to 81% as the number of extra devices goes from one to four
- Over 60% of users see no other traffic in the network
- For 60% of the clients, the traffic in their access-link never exceeds 50% of their link capacity
Dasu

- Dasu: a measurement experimentation platform
- Support network measurement experimentation and broadband characterization
- Wide network coverage to capture network and broadband service diversity
- Captures network and broadband service diversity
- Publicly available since June 2010
- 100,118 users spread over 2,431 Ases and 166 countries
Dasu System Components

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.

When we 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.

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.
Dasu Client Architecture

The image illustrates the Dasu client architecture, which includes components like Measurement Rules, Knowledge Base, Rule Engine, Measurement Coordinator, Probe Modules, Application Control, Application Status, and Host Application. The architecture is designed to facilitate Internet measurement experimentation while controlling the impact on hosts' resources.

Dasu leverages passively collected traffic statistics from the host's naturally-generated traffic as passive measurements. In addition to these active measurements, Dasu leverages passive monitoring, which includes DNS resolution and JavaScript testing support (through PhantomJS [25]).

Dasu is designed to facilitate Internet measurement experimentation while controlling the impact on hosts' resources. It supports a wide range of experiments and is safe (i.e., does not permit run-away programs). Dasu leverages passively collected traffic statistics to request the execution of an experiment.

Experimental rules are defined in a declarative model, using a rule-based programming language, as in Seattle and Fathom [27]. These rules can be chained together to schedule measurements and handle results. The API includes some basic accessor methods (e.g., getClientIps) and states the conditions to be matched. The right-hand side of a rule's left-hand side is the consequential part of the rule, which is executed when certain activation conditions hold. A rule's left-hand side is the condition part, and the right-hand side is the consequence or action.

Dasu is a versatile framework that supports a variety of measurement platforms and models. It provides a clean separation between experiment specification and run-time execution. Dasu's modular design allows for the creation of new modules and a programmable API to combine them. Rules related to the programs define an experiment.

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. The number of measurement primitives supported is relatively small, with an average of 24 lines of code per rule. Different research (PlanetLab) networks can be specified using only 3 different rules.

Experimentation while controlling the impact on hosts' resources is part of future work. Dasu supports active measurements as well as passive monitoring, making it applicable in a variety of contexts, including broadband characterization [26]. Dasu's architecture allows for the implementation of complex measurement experiments, with the ability to chain results through a shared working memory, multiplexing resources across experiments. The architecture also supports the updating of the base client itself without end-user interaction.

Dasu is a flexible and lightweight approach for specifying measurement experiments. Although the number of measurement primitives is relatively small, the high level of flexibility is maintained for the underlying network.

### Example

To illustrate the application of rules, let's consider a simple example of detecting cross-traffic. The following rules are defined to schedule measurements and handle results:

1. **Rule 1:**
   - **Fact 1:** Requests a DNS resolution for a domain name using the client's IP.
   - **Fact 2:** Extracts the IP address from the DNS result.
   - **Fact 3:** Schedules a traceroute.
   - **Fact 4:** Measures the ping latency to the IP address.

2. **Rule 2:**
   - **Fact 5:** Requests a DNS resolution for a domain name.
   - **Fact 6:** Extracts the IP address from the DNS result.
   - **Fact 7:** Schedules a traceroute.
   - **Fact 8:** Measures the ping latency to the IP address.

3. **Rule 3:**
   - **Fact 9:** Checks if the ping measurement is larger than 50 ms.
   - **Fact 10:** Schedules a traceroute if this is larger than 50 ms.

### Execution Environment

The rule-based execution environment consists of a rule-based program execution engine, a measurement coordinator, and a working memory. The engine rules are responsible for collecting relevant statistics from the host (as described in III-B) as well as continuously monitoring the end-host Internet connection.

The execution engine processes the rules and triggers the measurement mechanisms as per the defined conditions. The measurement coordinator ensures that resources are multiplexed across experiments and that progress and results are communicated through a shared working memory. The working memory serves to request the execution of a rule-based program to update the base client itself.

### Diagram

The diagram shows the architecture with the following components:

- **Rule Engine**
- **Measurement Coordinator**
- **Working Memory**
- **Probe Modules** (Traceroute, Ping, NDT, DNS, Passive)
- **Application Control**
- **Application Status**
- **Dasu**
- **Host Application**

The diagram illustrates the flow of data and control between these components, highlighting the integration of measurement rules, knowledge base, and measurement modules.
Rule-based program execution environment

Inference Engine
- Pattern Matcher
- Agenda

Execution Engine
- Fire-Rule 3

Rule Base
- Rule 1
- Rule 2
- Rule 3

Working Memory
- Fact 1
- Fact 2
- Fact 3
- Fact 4
Delegating Code Execution to Client
Detecting cross-traffic

Local cross-traffic

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

![Traffic upload activity graph](image)

- BitTorrent - local
- Netstat - local
- UPnP - gateway

**TABLE I: Dasu API – Methods.**

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

**TABLE II: Dasu API – Measurement modules currently supported.**

<table>
<thead>
<tr>
<th>Probe</th>
<th>Params.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PING</td>
<td>&lt;dest-list (IP/name)&gt;</td>
<td>Use the local host ping implementation to send ECHO REQUEST packets to a host.</td>
</tr>
<tr>
<td>TRACEROUTE</td>
<td>&lt;dest-list (IP/name)&gt;</td>
<td>Print the route packets take to a network host.</td>
</tr>
<tr>
<td>NDT</td>
<td>[&lt;server&gt;]</td>
<td>Run the M-Lab Network Diagnostic Tool [24].</td>
</tr>
<tr>
<td>DNS</td>
<td>[&lt;server&gt;]</td>
<td>Submit DNS resolution request to a set of servers.</td>
</tr>
<tr>
<td>HTTP</td>
<td>[&lt;server&gt;]</td>
<td>Submit HTTP request to a host associated with the given (host, port) pair.</td>
</tr>
<tr>
<td>PhantomJS</td>
<td>[url]</td>
<td>Run functional javaScript tests against a given URL.</td>
</tr>
</tbody>
</table>

**Rule:**

\[ \text{rule } (1) \text{ Resolve IP address through local DNS} \]

\[ \text{when} \quad \$\text{fact} : \text{FactFireAction(action==\text{resolveIp})} \]

\[ \text{then} \quad \text{addProbeTask(ProbeType.DNS, \text{\textquotesingle example.com\textquotesingle})} \]

\[ \text{end} \]

**Rule:**

\[ \text{rule } (2) \text{ Handle DNS lookup result} \]

\[ \text{when} \quad \$\text{dnsResult} : \text{FactDnsResult(toLookup==\text{\textquotesingle example.com\textquotesingle})} \]

\[ \text{then} \quad \text{String ip = \$\text{dnsResult.getSimpleResponse()}}; \]

\[ \text{addProbeTask(ProbeType.PING, \text{ip})}; \]

\[ \text{end} \]

**Rule:**

\[ \text{rule } (3) \text{ Handle ping measurement result} \]

\[ \text{when} \quad \$\text{pingResult} : \text{FactPingResult()} \]

\[ \text{then} \quad \text{if ( \$\text{pingResult.getRtt()} > 50 )} \]

\[ \text{addProbeTask(ProbeType.TRACEROUTE, \$\text{pingResult.ip})}; \]

\[ \text{end} \]

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.
Detecting cross-traffic
Cross-traffic (down)

**TABLE I: Dasu API – Methods.**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addProbeTask</td>
<td>Submit measurement request of the specified type.</td>
</tr>
<tr>
<td>commitResult</td>
<td>Submit completed experiment results to data server.</td>
</tr>
<tr>
<td>getClientIPs</td>
<td>Return network information about the client including the list of IP addresses assigned (both public and private).</td>
</tr>
<tr>
<td>getDnsServers</td>
<td>Return the list of DNS servers configured at the client.</td>
</tr>
<tr>
<td>getEnvInfo</td>
<td>Return information about the plugin and the host node, including OS information and types of measurement probes available to the experimenter.</td>
</tr>
</tbody>
</table>

**TABLE II: Dasu API – Measurement modules currently supported.**

<table>
<thead>
<tr>
<th>Module</th>
<th>Params.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PING</td>
<td>&lt;dest-list (IP/name)&gt;</td>
<td>Use the local host ping implementation to send ECHO REQUEST packets to a host.</td>
</tr>
<tr>
<td>TRACEROUTE</td>
<td>&lt;dest-list (IP/name)&gt;</td>
<td>Print the route packets take to a network host.</td>
</tr>
<tr>
<td>NDT</td>
<td>[&lt;server&gt;]</td>
<td>Run the M-Lab Network Diagnostic Tool [24].</td>
</tr>
<tr>
<td>DNS</td>
<td>[&lt;server&gt;]</td>
<td>Submit DNS resolution request to a set of servers.</td>
</tr>
<tr>
<td>HTTP</td>
<td>[&lt;server&gt;]</td>
<td>Submit HTTP request to a given host, port pair.</td>
</tr>
<tr>
<td>PhantomJS</td>
<td>[url]</td>
<td>Run functional javaScript tests against a given url.</td>
</tr>
</tbody>
</table>

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

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 (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.**

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.
Security and Safety

- Security and Safety
  - Experiment Sandbox
  - Secure communication
  - Limits on resource consumption

- Coordination
  - Experiment Leases
  - Elastic Budgets

- Synchronization
Current Deployment

- An extension to BitTorrent client (June 2010)
- Standalone client (June 2013)
- Provides information about services they receive from ISP

Coverage:
- 100,118 users spread over 2,431 Ases and 166 countries
- More than 57% countries of different continents
- 93% of Dasu peers are located in small transit providers and eyeball ASes
Dasu Dynamics

- For 90% of the clients, the median task (75 seconds) completes < 150 seconds
- 60% of Dasu clients complete 80% or more of the tasks assigned to them
- For 30% of the clients, median probe is launched < 1 sec, 60% < 30 seconds
- Every scheduled probe can be launched for 85% of peers (capping at a download utilization of 80%)
Third-party Experimenters

- Secondary EA responsible for task parameterization and client allocation
- Secondary EA_dataCollector responsible for collecting and storing experiments
- Secondary EA interacts with primary EA
- Primary EA responsible for allocating clients, authenticating and signing assigned experiments
Summary

- Characterize the challenging setting of edge-network experimentation platforms
- Design and implementation of Dasu – an extensible platform for measurement experimentation for the Internet’s edge.
- Characterize Dasu’s current deployment
Moving Forward/ Open Issues

- Extending platform beyond single incentive for adoption

- Integrating with stable experimental platforms like PlanetLab

- Determining feasible but inappropriate for end users
Thank You!!!!
Queries???