Dasu - ISP Characterization from the Edge

A BitTorrent Implementation

Mario A. Sánchez
Northwestern University

John S. Otto
Northwestern University

Zachary S. Bischof
Northwestern University

Fabián E. Bustamante
Northwestern University

ABSTRACT

Evaluating and characterizing access ISPs is critical to consumers shopping for alternative services and governments surveying the availability of broadband services to their citizens. We present *Dasu*, a service for crowdsourcing ISP characterization to the edge of the network. *Dasu* is implemented as an extension to a popular BitTorrent client [5] and has been available since July 2010. While the prototype uses BitTorrent as its host application, its design is agnostic to the particular host application. The demo showcases our current implementation using both a prerecorded execution trace and a live run.

Categories and Subject Descriptors

C.2.3 [**Computer Systems Organization**]: Computer Communication Networks—*Network Operations*; C.2.5 [**Computer Communication Networks**]: Local and Wide-Area Networks—*Internet*; C.4 [**Performance of Systems**]: Measurement techniques

General Terms

Experimentation, Performance, Measurement

Keywords

Broadband access networks, ISP, Characterization

1. INTRODUCTION

Evaluating and characterizing access ISPs is critical to subscribers shopping for alternative ISPs, companies providing reliable Internet services, and governments surveying the availability of high-speed Internet services to their citizens [3,4].

Given its importance and the lack of publicly available information, several recent projects have started to evaluate alternative approaches for profiling edge network services. The approaches being explored range from using Web-based technologies to run tests against centralized or cloud servers, to actively measuring end hosts from a set of dedicated servers, to installing special networking devices for active monitoring from inside PoPs or home networks. While these efforts have begun to shed some much needed clarity on network conditions at the edge, their approaches present an apparently unavoidable tradeoff between extensibility, vantage point coverage and continuous monitoring.

Measurement from dedicated servers in the core of the network, including platforms such as Archipelago and PlanetLab, can be done at scale, but cannot capture the view of end users, typically located behind middle boxes and not welcoming unsolicited active measurements. Web-based approaches can capture the user's perspective and have low barriers to adoption, but are susceptible to white-listing from ISPs and limited by infrastructure and bandwidth costs that scale with the number of adopters. In addition, both of these approaches capture performance *only* when the test is run, missing dynamic changes to service levels. Last, while deploying devices inside home networks provides continuous monitoring at the network edge, the dependence on hardware devices severely limits these approaches' geographic coverage and flexibility, hampering the representativeness of the captured view.

We have argued [1] that hosting ISP characterization on end systems running network-intensive applications can avoid these tradeoffs. We have presented a new approach for crowdsourced ISP characterization that leverages the detailed views of Internet-wide ISP performance offered by these applications and evaluate its feasibility through a large-scale study of data gathered from BitTorrent users. By passively monitoring user-generated traffic within these applications, our approach is able to capture the end user's view in a scalable manner. By extending existing applications, it has the flexibility and low-barrier to adoption of other software-based models. By combining passive monitoring with dynamically extensible active measurements, it can achieve the effectiveness of hardware-based solutions without their associated costs.

In this demo, we present a prototype implementation of these ideas as an extension to a popular BitTorrent client [5]. Our extension – *Dasu* – was first released to Beta testers in June 2010. At the time of submission and *without any advertisement*, Dasu has been adopted by more than 37,000 users in over 100 countries. While the prototype uses BitTorrent as its host application, its design is agnostic to the particular host application.

Additional information about the project and the current Dasu implementation can be found at: `http://www.aqualab.cs.northwestern.edu/projects/Dasu.html`.

2. DASU

Dasu is implemented as an extension to a popular BitTorrent client [5]. The extension passively monitors user-generated BitTorrent traffic, which allows it to scale while monitoring the performance experienced by end users, as well as relevant signals from the host application and OS in real time. Examples of the collected information include per torrent statistics (such as the number of RSTs received, upload rate and download rate), application-wide statistics (such as total upload and download rates, number of active

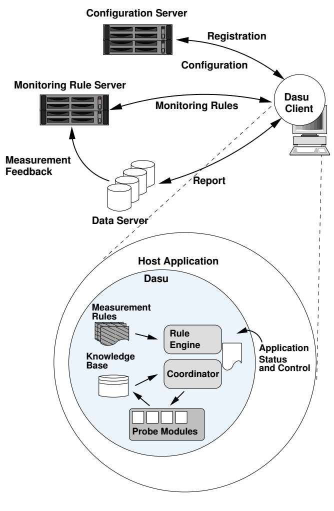

Figure 1: Dasu System components.

torrents) and system-wide statistics (including connection-related statistics reported by the operating system such as number of active, current and passive connections, number of connections torn by TCP RST, and number of failed connections).

In addition, each Dasu client (Fig.1) includes an easy to extend set of *Probe Modules* for active measurements such as traceroute, ping, and the MLab's Network Diagnostic Tool (NDT).

Launched probe modules add measurement results to the client's *Working memory* or *Knowledge Base* – a repository of objects known as *facts*. The set of facts in the working memory drives the firing of monitoring rules by the *Rule Engine*. As new facts are added to the working memory and new monitoring rules are added to the rule base, the rule engine fires rules whose actions can result on the launching of new monitoring probes. The *Measurement Coordinator* is responsible for launching the necessary active probes as required by the rule engine. The coordinator receives measurement requests and schedules them in such a way as to guarantee maximum responsiveness (in terms of time taken to launch the requested probe) while minimizing overhead and interference between probes and the host's application activity. The coordinator is also responsible for limiting the number of active probes that can be executed on the system over a period of time.

2.1 Dasu System

Each client, upon initialization, contacts the *Dasu Configuration Server* to register itself and retrieve a configuration file. Each client's configuration file specifies the set of servers to use (to report collected measurements or retrieve new monitoring rules, for instance), and the frequency with which it should contact them. Other parameters in the configuration file include the size of measurement logs and the maximum frequency of periodic events, for instance. After registering, clients contact the *Monitoring Rule Servers* to retrieve the measurement rules they would run for both passive and active monitoring. Clients submit their collected results to the *Database* server, at a given rate specified by the configuration file, and can access experiment descriptions from the *Web Server* as needed.

All monitoring rule files served by the active or continuous monitoring rule servers are digitally signed for authenticity. The servers use public-key cryptography to sign every rule file served to clients. Clients verify the authenticity of the file using the public-key distributed as part of the Dasu client.

Limits on the number, type and rate of measurements that a client can issue together with the secure assignments of tasks ensure the fail-safe property of the platform.

Rules Specification. The basic structure of monitoring rules include a rule name followed by the condition and consequence sections. Rules have the following general form:

```
rule <name>
when {<condition>}
then {<consequence>}
```

Actions can result in the inclusion or removal of facts to/from the knowledge base, and the launching of new monitoring probes.

As a concrete example, the following simple rule triggers the Vuze BitTorrent Speed Test when this has been requested to validate potential interference. The request appears as a fact `FactFireAction( action == "launchBTSpeedTest")` in the client knowledge base. The consequence section issues a request for the probe and retracts the triggering fact from the knowledge base.

```
rule "Launch BT speed test to
      corroborate interference"
  when
    $fact : FactFireAction( action ==
            "launchBTSpeedTest");
  then
    addPriorityProbe("download_non_encrypted",
       ProbeType.BTTest);
    sendToLog("Launching BT download Test");
    retract($fact);
end
```

Monitoring rules support a straightforward implementation of a number of useful functions for ISP characterization including interference detection (as in [2]), and evaluation of the users' DNS resolver.

2.2 Demonstration

Our demonstration will showcase our current implementation of *Dasu* using both a prerecorded execution trace and a live run. We will show the information available to users through the plugin interface and demonstrate the behind the scenes execution of monitoring rules.

3. REFERENCES

[1] BISCHOF, Z. S., OTTO, J. S., SÁNCHEZ, M. A., RULA, J. P., CHOFFNES, D. R., AND BUSTAMANTE, F. E. Crowdsourcing ISP characterization to the network edge. In *Proc. of ACM SIGCOMM W-MUST* (August 2011).

[2] DISCHINGER, M., MARCON, M., GUHA, S., GUMMADI, K. P., MAHAJAN, R., AND SAROIU, S. Glasnost: Enabling end users to detect traffic differentiation. In *Proc. of USENIX NSDI* (2010).

[3] SAMKNOWS. Samknows & Ofcom UK broadband performance testing. `http://www.samknows.com/broadband/ofcom_and_samknows`, June 2009.

[4] SAMKNOWS. Samknows & the FCC American broadband performance measurement. `http://www.samknows.com/broadband/fcc_and_samknows`, June 2009.

[5] VUZE, INC. Vuze. `http://www.vuze.com/`.