Title: Scalable, Extensible, and Safe Monitoring of GENI Clusters
Project Number: 1723

Authors:
Sonia Fahmy, Department of Computer Science, Purdue University
Address: 305 N. University St., West Lafayette, IN 47907–2107; E-mail: fahmy@cs.purdue.edu
Puneet Sharma, HP Labs
Address: 1501 Page Mill Rd., Palo Alto, CA 94304; E-mail: puneet.sharma@hp.com

January 2010
Version 0.2

This material is based upon work partly supported by BBN Technologies.

1 Introduction

1.1 About This Document

This document describes the design of the *Scalable Monitoring* service for GENI, and how it will be integrated with the ProtoGENI control framework. The document will be used as a guideline for the implementation.

The document will be reviewed by the ProtoGENI cluster, GENI security team, GPO and any other interested parties.

1.2 Overview

Network measurements such as delay, loss, and available bandwidth are essential for optimizing and managing network applications and services, especially in a GENI-like shared infrastructure. Sharing a network measurement service across multiple applications can significantly reduce measurement overhead, increase accuracy, and remove the burden of performing network measurements from individual applications. A shared measurement service can also aid the management and operation of slices on GENI clusters. Network measurement data is most useful if measurements can be requested on-demand by users/operators at desired *times* and *frequencies*. Clearly, in a federated environment like GENI, it is difficult to have the same level of trust in all users making measurement requests. Therefore, we propose to build and deploy a framework on GENI clusters to provide network measurement information to non-expert or untrusted users such that *active* measurements can be taken in a safe manner. By safety, we mean that users cannot trigger arbitrarily large measurement probes that exceed a specified budget. The careful admission control of measurement requests prevents the use of the measurement infrastructure as a denial of service tool, and allows the administrators of measurement hosts to prioritize the measurements they admit according to their *policies*.

We will design and implement a *scalable* and *safe* measurement service and integrate it with the ProtoGENI control framework. The measurement service will be *extensible* in that it can answer an evolving set of queries using an evolving set of measurement tools. In Section 2, we describe the architecture of S^3 – our scalable sensing service for ProtoGENI. S^3 comprises three components: (i) sensor pods that are a web-service-enabled collection of sensors exposing interfaces for various types of sensor invocations, (ii) sensing information manager for aggregating and disseminating the sensor data, and (ii) engines that provide scalable inference services.

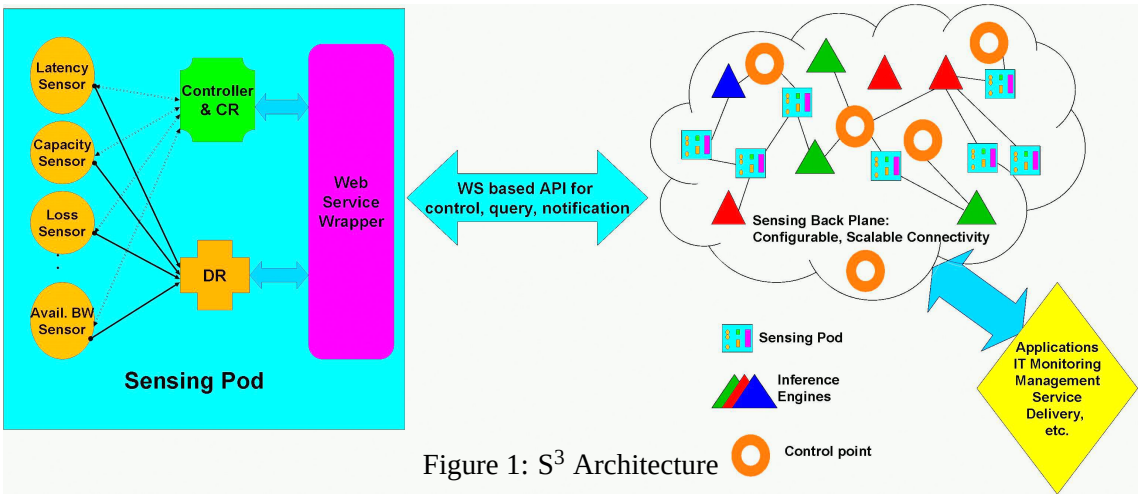

Figure 1: S^3 Architecture

2 S^3 Architecture

The S^3 architecture (see Figure 1) comprises three components: Sensor Pods, Sensing Information Manager, and Scalable Inference Engines. We describe these components in detail in this section.

2.1 Sensor Pods

A sensor pod is a web-service-enabled collection of light-weight measurement and monitoring sensors that collect information at a machine. This information spans network properties such as connectivity to the Internet, latency to some other machine in the system, and bandwidth to another machine. It can also be extended to include machine attributes such as current CPU load, free memory, and number of processes. These sensors gather information actively (e.g., send some packets on a network link to detect available bandwidth) and possibly passively (e.g., infer the current RTTs of a link from communication patterns of a TCP connection on the same link). Simple sensors can also be created to extract already existing SNMP MIB data from different network elements.

As shown in Figure 1, along with sensors, each sensor pod has a Data Repository (DR) to store information measured by different sensors, and a Configuration Repository (CR) to store sensing configurations that are used to determine which sensor to invoke, how frequently to invoke, how long to invoke, what parameters to supply on sensor invocation, and how to store the information returned by the sensor. Each sensor pod has a controller that coordinates the invocation of sensors based on the configuration information provided to the sensor pod. Sensor pods expose interfaces for both querying the data in DR and for setting new configurations in CR through the web service API. Exposing sensor pod interfaces as a web service in our architecture enables sensor composition — new sensors can be easily built by composing information generated by some existing sensors.

A sensor pod optimizes the number of sensor invocations (to minimize network communication and computation costs) by analyzing sensing configurations supplied by different applications/experiments for a unique sensor. For example, if two applications need to measure the latency to the same destination machine but at different periodic rates, say once in 6 seconds and once in 10 seconds, then our subsystem invokes the sensor at the higher rate (once in 6 seconds) and stores that as an answer for both sensing configurations. If multiple applications/experiments invoke the same sensor for one-shot measurement and if those invocations overlap, then the sensor pod performs only one measurement and returns the same value to all requesting applications. Thus, our sensor pod design effectively eliminates redundant monitoring traffic.

2.2 Sensing Information Manager

The sensing information manager is a data aggregation and management system that collects measurement data from the individual sensor pods on physical machines in the network. In a sensor pod, either sensors can feed the measurement data directly to the sensing information manager, or the controller can insert data from DR based on the configurations stored at the sensor pod. Clients of the sensing infrastructure can either specify how to aggregate the data or subscribe to existing aggregate feeds of the data.

In the planned deployment on ProtoGENI, we will build a central manager as a sensing backplane. This central manager will act as the portal for:

- Monitoring and deployment on sensor pods on various nodes,
- Collecting the sensing data,
- Publishing the sensing data to clients.

2.3 Scalable Inference Engines

The task of collecting the complete information about network metrics is an immense task both in terms of the infrastructure requirements as well as the measurement traffic. Scalable inference engines estimate complete information about the relevant network metrics based on partial information measured using the sensing pods. The main idea behind inference algorithms is to measure various metrics on a small number of network paths and use the information to infer the properties of all the paths.

While scalable inference of all network properties is a challenge, a large body of research efforts (e.g., [6, 11–13] successfully tackled latency estimation. Though these efforts take different approaches, they all involve periodic measurements from each node to few other nodes in the system to answer for proximity or latency queries accurately reflecting the current status of the network. In S^3, inference engines will use sensor pods to perform measurements and the sensing information manager to gather the data in an efficient manner. In the third year of the project, we will dynamically invoke inference mechanisms based on the measurement request load [3].

3 Implementation Plan

Sensor pods are being implemented as `cgi` scripts accessible through any web-server that supports `cgi`. We currently use Boa (`http://www.boa.org`), a light-weight open source web-server. This framework enables third party measurements, that is, measurements between two nodes can be initiated by a third node. We plan to have a wide variety of sensors, some of which are listed in Table 3, that leverage several open source network monitoring tools for measuring various network path metrics (latency, number of hops, available bandwidth, bottleneck capacity, and loss rate). To enable large-scale concurrent measurements, we are modifying some of the tools. We leverage the web-services-based sensing pod architecture to deploy various sensors measuring different metrics and also to configure the periodic measurements.

The central manager will ensure the liveness of our service running on ProtoGENI using *vxargs* [10] script. It will also collect the measurements from the sensor pods and store it in a datastore. The client queries and measurements requests can be made using the RSPEC format. We plan to initially leverage a single ProtoGENI slice for S^3 measurements to avoid the overhead of slice creation whenever a new measurement request is submitted, and also to share the measurement probe traffic over multiple similar requests. However, if the experimenter requires a view of the network as seen by a particular slice, we plan to support that as well.

Sensors	Purpose
PING	Measures latency to a specified destination
TRACEROUTE	Collects number of hops and latency on the network path to a destination
SPROBE [8], PATHRATE [4]	Measure capacity of the network path to a destination
TULIP [5]	Measures error rate of the network path to a destination
SPRUCE [9], PATHCHIRP [7]	Measure available bandwidth on the network path to a destination

Table 1: A subset of network sensors planned to be deployed

In the second year of the project, we will augment this service with admission control and scheduling algorithms for recurring measurement requests. We define a *load invariant* to ensure resource consumption of measurement tools does not exceed a given budget (e.g., 5% of link capacity), and preserve this invariant as new active measurement requests are submitted to the system [1, 2]. Flexibility in the requested measurement times and required accuracy can be leveraged to eliminate redundant measurements that can use an undue share of network resources.

The sensing information manager will pull the measurement data from the sensor pods on all nodes to a central repository to provide global views to other researchers by making this data available online, and also to archive the data for Internet behavior analysis. A snapshot of the all-pair network path metrics will be updated periodically and made available at a web portal. The same web portal will also provide the interface for invoking new measurements and also query archived measurement data.

Acknowledgments

We would like to thank Rob Ricci (University of Utah) and Vicraj Thomas (GENI project office) for helpful discussions on this design.

References

[1] E. Blanton, S. Fahmy, and S. Banerjee. A framework for an on-demand measurement service. Technical report, Purdue University, 2008. http://www.cs.purdue.edu/homes/fahmy/reports/measurement.pdf.

[2] Ethan Blanton, Sonia Fahmy, and Sujata Banerjee. Resource management in an active measurement service. In *Proceedings of the IEEE Global Internet Symposium*, April 2008.

[3] Ethan Blanton, Sonia Fahmy, and Greg N. Frederickson. On the utility of inference mechanisms. In *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, June 2009.

[4] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *Proceedings of the IEEE INFOCOM 2001*, Anchorage, AK, April 2001.

[5] Ratul Mahajan, Neil Spring, David Wetherall, and Thomas Anderson. User-level Internet Path Diagnosis. In *Proceedings of the SOSP*, Oct 2003.

[6] T. S. Eugene Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches. In *Proceedings of the IEEE INFOCOM 2002*.

[7] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient available bandwidth estimation for network paths. In *Proceedings of the PAM 2003*, La Jolla, CA, April 2003.

[8] S. Saroiu. SProbe: A Fast Tool for Measuring Bottleneck Bandwidth in Uncooperative Environments.

[9] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the ACM IMC 2003*, Miami, FL, October 2003.

[10] Vxargs. http://dharma.cis.upenn.edu/planetlab/vxargs/.

[11] B. Wong, A. Slivkins, and E.G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proceedings of the ACM SIGCOMM*, Philadelphia, PA, August 2005.

[12] Zhichen Xu, Puneet Sharma, Sung-Ju Lee, and Sujata Banerjee. Netvigator: Scalable network proximity estimation. Technical report, HP Laboratories, 2005.

[13] H. Song Y. Chen, D. Bindel and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *Proceedings of ACM Sigcomm*, 2004.