S^3: A Scalable Sensing Service for Monitoring Large Networked Systems[*]

Praveen Yalagandula, Puneet Sharma, Sujata Banerjee, Sujoy Basu and Sung-Ju Lee
HP Labs, Palo Alto, CA

ABSTRACT

Efficiently operating and managing large scale distributed and federated systems is an extremely challenging problem. Current solutions are a combination of centralized management and significant over-provisioning of the infrastructure. With the explosion of new resource-intensive media applications and services, over provisioning of the infrastructure is no longer a viable option. Timely and accurate knowledge of the global environment (particularly the highly dynamic network path properties) is necessary for management of performance SLAs, just-in-time resource provisioning, near-optimal dynamic service placement and reuse, construction of network service overlays, and fast detection of failures and malicious attacks. Further, different applications require information about different aspects of the environment at different timescales. We propose S^3, a Scalable Sensing Service, that achieves the above requirements and enables personalized sensing of the environment as dictated by applications.

1. INTRODUCTION AND CHALLENGES

Efficient operation of large-scale distributed networked systems such as enterprise networks, grid systems or sensor systems remains a challenge for a number of reasons. One of the crucial reasons is the lack of current and accurate knowledge of the global state of the various components in the system, including network attributes as well as individual machine attributes. *Sensing* infrastructure refers to such a sub-system that measures, estimates, and enables querying of the global state. The sensing capabilities of most commercial management systems today suffer from the following drawbacks: 1) monitor only a subset of application/system metrics, 2) measure at coarse grain timescales, and 3) perform centralized processing of the measurement data.

Though such sensing might suffice for long term capacity planning and management, it impedes real-time control and decision making, which is important for several emerging multi-media services. What is needed is a scalable sensing service (S^3) that can securely provide customized information at appropriate periodicity as required by different applications and services. S^3 has been designed for scalable and robust operation and can be used for a number of management tasks:

- Detection of failures and anomalous behavior: Link failures, node failures, service failures, root cause diagnosis for poor application/service performance, and suspicious activity such as DoS attacks, viruses, worms, and spam mail.
- Resource placement and location: Adaptive placement based on observed performance (e.g., choosing a host with better availability for running a key service), placement based on observed loads (replicate services based on usage conserving key resources such as power).
- Better performance: Fast network path selection for quick content transfers.

Design of a scalable sensing service need to tackle several challenges. First, there are a large number of attributes (network related and machine related) that need to be tracked. Some attributes are highly dynamic (e.g., available bandwidth on an end-to-end path or current load on a machine) while others are slowly varying or static (bottleneck capacity on a network path or processor type/OS version on a machine). The total number of individual metrics to be tracked increases exponentially as the overall system grows. Second, measuring everything at the smallest possible timescale is impossible and smart estimation techniques are needed that give an accurate approximation of the global state with low measurement overhead. Third, multiple applications conducting their own measurements could lead to high communication overheads. Due to the lack of end-to-end QoS in existing infrastructures, many applications/services are forced to conduct their own measurements and make operational decisions based on these end-to-end measurements. Further, the individual application level measurements may be at different timescales ranging from seconds to even days. For example, an end client could ping multiple web proxy machines to identify the one that has the least latency, or a media client player could measure the available bandwidth to multiple servers and pick one that has the highest current bandwidth. Other examples include building efficient service overlays where the overlay nodes maintain node and path properties to their neighbors. This causes much duplication of measurements leading to a high system overhead.

In Section 2, we describe the architecture of S^3 - our scalable sensing service that tackles the above challenges. S^3

[*]Supported in part by DARPA Contract N66001-05-9-8904.

comprises of three components: (i) sensor pods that are web-service enabled collection of sensors and exposes interfaces for various types of sensor invocations, (ii) sensing information backplane that provides a programmable middleware for aggregating and disseminating the sensor data, and (ii) analysis engines that leverage the above two components for providing scalable inference services.

We have built a prototype of S^3 modules that we have deployed on the PlanetLab. We are using this system in a DARPA Internet Control Plane project called CHART (Control for High-Throughput Adaptive Resilient Transport). CHART combines a novel adaptive routing infrastructure and a distributed network sensing infrastructure to improve end-to-end performance across an unreliable network. The sensing infrastructure monitors the state of underlying network and conveys state information to the routing infrastructure. The combined system adaptively routes around failed or congested links under fine real-time control to maintain high end-to-end throughput. To achieve this, constant, pervasive sensing of network conditions including delay, loss, bandwidth information is needed every few seconds to build high throughput network overlay paths. We are using S^3 as the sensing infrastructure for the CHART project.

2. S^3 ARCHITECTURE

The S^3 architecture (see Figure 1) comprises of three components: Sensor Pods, Sensing Information Management Backplane, and Scalable Inference Engines. We describe these components in detail along with a discussion of applications leveraging S^3 in the following sections.

2.1 Sensor Pods

A sensor pod is a web-service enabled collection of lightweight measurement and monitoring sensors that collect information at a machine. This information spans both network properties such as connectivity to the Internet, latency to some other machine in the system, bandwidth to another machine, and machine attributes such as machine's 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) or passively (e.g., infer the current RTTs of a link from communication pattern 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, 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. Also 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 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.

Sensor pod allows authenticated and authorized applications and users to install new sensors, supply sensing configurations, and query information gathered using sensors. Mechanisms such as thresholding are built into each sensing pod to reduce communication costs and ensure that metrics that exhibit very low variability, as specified by an application, are not disseminated by the sensing pod. Also, network metric values that become stale for whatever reason are reported along with a timestamp denoting when they were measured to ensure that they are not used in a real-time decision.

A sensor pod optimizes the number of sensor invocations (to minimize network communication and computation costs) by analyzing sensing configurations supplied by different applications 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. Also if multiple applications 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 the redundant monitoring traffic incurred by applications if each conducts its own measurements. We are developing a measurement scheduler to limit the measurement overhead under a specific fraction of the network capacity (e.g. 1% of the link capacity).

2.2 Sensing Information Management Backplane

The sensing backplane is a data aggregation and management middleware that collects measurement data from the individual sensor pods on physical machines in the system and provides a programmable substrate for application services to aggregate the data in an efficient and scalable manner. In a sensor pod, either sensors can insert the measured data directly into the sensing backplane 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.

Scalability both in terms of management of large amounts of sensing information as well as fast query-response time is a key goal of this component. We leverage SDIMS [31] (Scalable Distributed Information Management System), which is a data aggregation framework that scales to a large number of machines and large number of data attributes and supports flexible interfaces to support wide-range of distributed applications. We employ SDIMS to scalably aggregate and disseminate collected network and node state information. In the following, we first briefly describe SDIMS and present details on how the sensing backplane uses SDIMS.

SDIMS is a distributed system that runs on all machines in the system and leverages Distributed Hash Table algorithms (DHTs) (for example, [21]) to expose their internal routing mesh as a set of multiple aggregation tree overlays on top of the physical machines. As Figure 2 illustrates, each physical node in the system is a leaf and each subtree represents a logical group of nodes. An internal *virtual node* is simulated by one or more physical nodes at the leaves of the subtree rooted at the virtual node. SDIMS exposes flexible interfaces for applications to *install* arbitrary aggregation functions (java objects) that are used to perform in-network aggregation at the intermediate virtual nodes of data at the

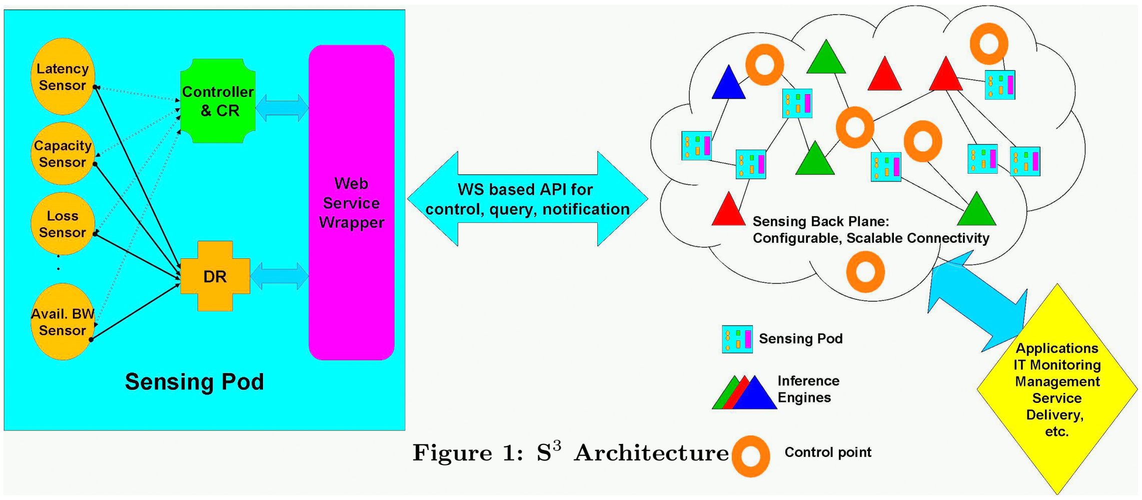

Latency Sensor
Capacity Sensor
Loss Sensor
Avail. BW Sensor
Controller & CR
DR
Web Service Wrapper
Sensing Pod
WS based API for control, query, notification
Sensing Back Plane: Configurable, Scalable Connectivity
Sensing Pod
Inference Engines
Control point
Applications IT Monitoring Management Service Delivery, etc.

Figure 1: S^3 Architecture

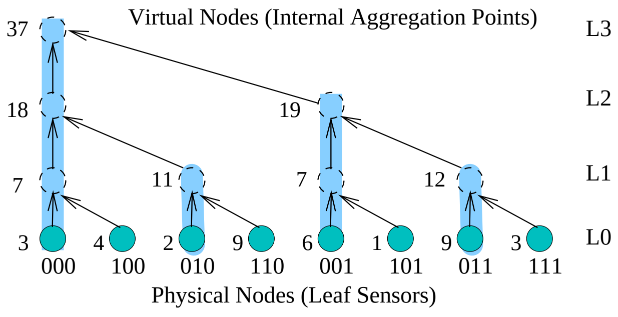

Virtual Nodes (Internal Aggregation Points)
37
18
19
7
11
7
12
3
4
2
9
6
1
9
3
L3
L2
L1
L0
000
100
010
110
001
101
011
111
Physical Nodes (Leaf Sensors)

Figure 2: An example aggregation tree in an eight node SDIMS. Also shown are the aggregate values for a simple SUM() aggregation function.

leaves in an aggregation tree, *update* the data at the leaves, and *probe* for the aggregated values at different levels in the tree. The example in Figure 2 also illustrates how a simple SUM operation is performed on an aggregation tree in an eight node SDIMS.

The sensing backplane in the S^3 architecture comprises of SDIMS running on all backplane machines in the system and extends SDIMS to support several functionalities common in wide-scale network oriented monitoring we are targeting. One particular extension is to support efficient publish-subscribe service. While publish is supported through *update* interface in SDIMS, we need to extend the probe interface of SDIMS to efficiently support subscribe requests. In SDIMS, applications can perform a continuous probe to receive aggregated values at specified levels in the tree but there is no functionality for subscribers to specify that they be notified only when the aggregated value satisfies a custom predicate. In current SDIMS with continuous probe, subscribers have to get any changes to the aggregated value and will have to filter out any changes that does not satisfy their predicates. In S^3, we extend SDIMS with *probe functions* that allow applications to specify a subscription predicate thus effectively filtering out unnecessary messages.

2.3 Scalable Inference Engines

Scalable Inference engines leverage configuration interfaces of sensor pods to perform periodic measurements at the nodes in the system and leverage the scalable sensing backplane to aggregate the measured data. 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 the 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., [30, 13, 29, 28] 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 use sensor pods to perform these periodic measurements and the sensing backplane to gather the data in an efficient manner using thresholding and in-network aggregation. Below we briefly describe how Netvigator [29] and its distributed version [5], a scalable proximity/latency estimation algorithm, can be plugged into our architecture.

Landmark clustering is a popular scheme used for network distance estimation that uses a node's distances to a set of special nodes (referred to as landmark nodes) to estimate the node position. In Netvigator, the sensing pod at each node measures distances to a given set of landmarks, similar to other landmark clustering techniques. Netvigator additionally records the distances to the milestones that are encountered while probing the landmarks. Instead of attempting to embed all the nodes in a global Cartesian-space based on RTT measurements, Netvigator performs local clustering for proximity estimation. Particularly, one scheme Netvigator [29] uses to estimate latency from a node X to another node Y is based on MIN-SUM formulation:

$$\text{latency}(X, Y) = \min_{l \in L}\{d(X, l) + d(Y, l)\},$$

where L is the set of landmarks and milestones and $d(X, l)$ denotes the measured latency from node X to l.

In the S^3 architecture, distributed Netvigator uses web service interfaces of sensor pods to configure periodic invocations of traceroute sensor from each machine to chosen landmark nodes. These measurements are fed into the sensing backplane and distance information to different landmark nodes or milestones are aggregated along different aggregation trees exposed by the SDIMS middleware using an aggregation function that tracks Top-k minimum distant nodes from a given landmark or milestone. To answer the proximity queries quickly, nodes subscribe to global aggregate values in the aggregation trees corresponding to their Top-

K nearest landmarks or milestones. Nodes use the publish-subscribe feature of the sensing backplane to filter out most of the changes in the latency values that does not affect their proximity information.

2.4 Applications

While existing Network Management Systems (NMS) extract MIB data from SNMP compliant routing elements that provide device centric information, S^3 enables applications to obtain a fine grained flow-centric view and and to monitor and control the performance of their flows (for e.g., by switching to a better server or using a overlay path if the current connection is not meeting required latency or bandwidth constraints). Also, the flexibility and scalability of the sensing backplane enables application to perform real-time aggregation and analysis of information about flows at thousands of end machines such as to detect and control heavy hitters that use a significant fraction of the network bandwidth, and to detect and control Internet worms.

As described in the introduction, the CHART system depends on a sensing infrastructure to monitor the network state and feed that information to other components of that system including a DHT based routing substrate. A monitoring and control component of CHART uses sensor pods to perform periodic measurement of available bandwidth and latency metrics from each participating node to the neighboring nodes in the CHART routing substrate. Other components of the CHART system, such as a TCP Jumpstart driver that uses the available bandwidth measurements to decide the congestion window size instead of observed packet losses, subscribe to these measurement streams.

A distributed malicious activity monitor can leverage the sensing information backplane to detect misbehavior in large scale networks such as Distributed Denial-of-Service attacks (DDoS), massive port scanning to locate exploitable services, Internet worms, and e-mail spam, that affect the performance of legitimate applications. Consider a simple example of IP traffic monitoring where the network administrator of a large enterprise wants to query for heavy hitters, i.e., monitor all network flows that account for a significant fraction (say 0.1%) of the volume of ongoing traffic. A centralized approach of logging the entire data at a single coordinator would either generate monitoring traffic whose volume is proportional to the total traffic, or be too late in detecting network anomalies, both of which are clearly unacceptable. Instead, our SDIMS based sensing backplane can scalably collect monitored data generated by sensing pods at the individual nodes and analyze the data in an efficient manner [11].

3. PROTOTYPE AND DEPLOYMENT

We have built a prototype of the S^3 modules that is deployed on on the Planet-Lab testbed. Currently, we deploy and ensure the liveness of our service on Planet-Lab nodes using *vxargs* [24] script run from a central manager. In near future, we plan to switch to one of the distributed frameworks like AppManager [1] to deploy and run our service. Sensor pods are 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. Our current implementation has 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 had to modify some of the tools. We are currently measuring all-pair network metrics periodically. We leverage the web-services based sensing pod architecture to deploy various sensors measuring different metrics and also to configure the periodic measurements. Figure 3 shows an example of accessing a web-enabled sensing-pod deployed on the Planetlab. The sensing backplane is not yet integrated with the sensing pods or analysis engines. This first version of S^3 service for PlanetLab was announced to the PlanetLab mailing list on January 25, 2006. During the first month of the service, it has received approximately 500 unique visitors downloading about 12GB of data.

Sensors	Purpose
PING	Measures latency to a specified destination
TRACEROUTE	Collects number of hops and latency on the network path to a destination
SPROBE [19], PATHRATE [3]	Measure capacity of the network path to a destination
TULIP [12]	Measures error rate of the network path to a destination
SPRUCE [22], PATHCHIRP [18]	Measure available bandwidth on the network path to a destination

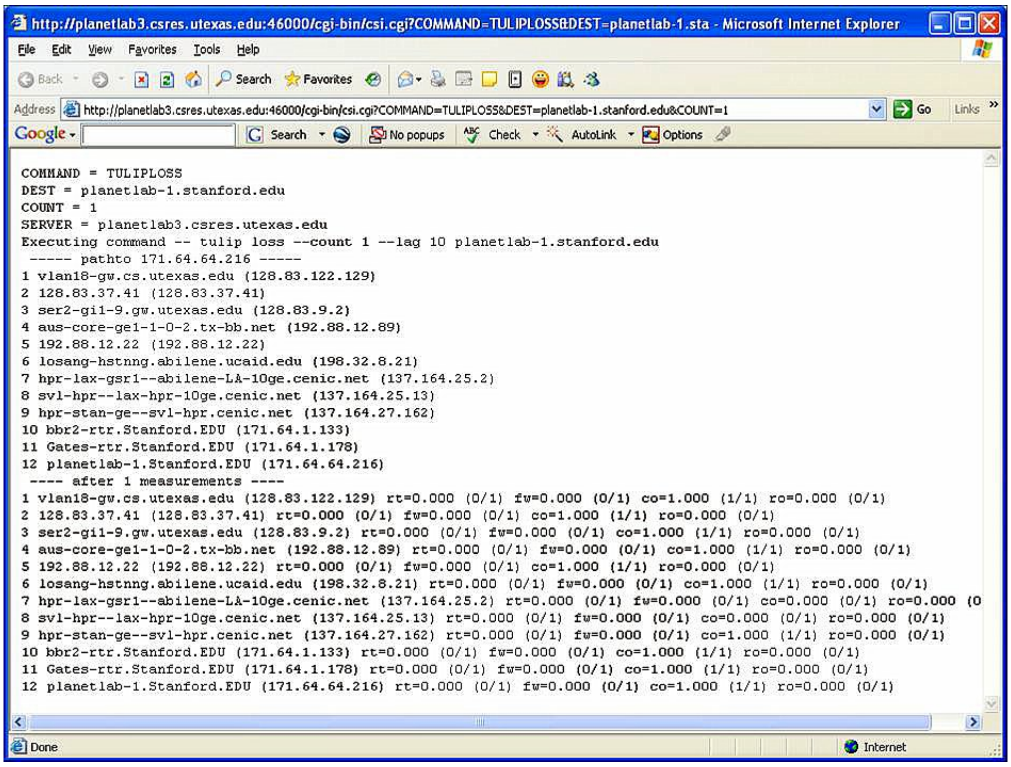
Table 1: A subset of network sensors currently deployed

We also pull the measurement data from the sensor pods on all nodes to a central node to provide the 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 capacity and available bandwidth metrics updated about every 4 hours is available at the following website: `http://networking.hpl.hp.com/s-cube/PL`. We use the Pathrate tool [3] to measure bottleneck capacity of a path and the Spruce tool [22] to measure available bandwidth on a path.

We also provide estimated latencies between all planetlab nodes as estimated by Netvigator. For every snapshot of data collected, we compute the estimation error over a small number of paths (about 5% of the total number of paths) for which we have the actual measured latency. Figure 4(a) plots the estimated delay versus the actual measured delay for Netvigator for a single snapshot of Planet-Lab data. The units for the axes in these plots are in microseconds. In these scatter plots, the closer the points plotted are to the diagonal, the better is the estimation. In our Planetlab experiments, the delay estimation with Netvigator was the best, with a mean absolute estimation error of 23 msec, followed by Vivaldi and GNP in this order.

In Figure 4(b), we present Netvigator latency estimation results on the snapshots generated over a 7-day period between March 23 15:49:37 PST 2006 and March 30 19:12:05 PST 2006 and compute various statistics of the absolute estimation error. These statistics are the mean, the 25th, 50th, 75th and 90th percentile of the absolute error. The main observation is that the 25th, 50th and 75th percentile absolute error is fairly low and stable across the entire time period, with 75% of the measured latency having estimation error less than 25 msec.

4. RELATED WORK


http://planetlab3.csres.utexas.edu:46000/cgi-bin/csi.cgi?COMMAND=TULIPLOSS&DEST=planetlab-1.sta - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://planetlab3.csres.utexas.edu:46000/cgi-bin/csi.cgi?COMMAND=TULIPLOSS&DEST=planetlab-1.stanford.edu&COUNT=1 Go Links
Google Search No popups Check AutoLink Options
Done Internet

```
COMMAND = TULIPLOSS
DEST = planetlab-1.stanford.edu
COUNT = 1
SERVER = planetlab3.csres.utexas.edu
Executing command -- tulip loss --count 1 --lag 10 planetlab-1.stanford.edu
 ----- pathto 171.64.64.216 -----
1 vlan18-gw.cs.utexas.edu (128.83.122.129)
2 128.83.37.41 (128.83.37.41)
3 ser2-gi1-9.gw.utexas.edu (128.83.9.2)
4 aus-core-ge1-1-0-2.tx-bb.net (192.88.12.89)
5 192.88.12.22 (192.88.12.22)
6 losang-hstnng.abilene.ucaid.edu (198.32.8.21)
7 hpr-lax-gsr1--abilene-LA-10ge.cenic.net (137.164.25.2)
8 svl-hpr--lax-hpr-10ge.cenic.net (137.164.25.13)
9 hpr-stan-ge--svl-hpr.cenic.net (137.164.27.162)
10 bbr2-rtr.Stanford.EDU (171.64.1.133)
11 Gates-rtr.Stanford.EDU (171.64.1.178)
12 planetlab-1.Stanford.EDU (171.64.64.216)
 ---- after 1 measurements ----
1 vlan18-gw.cs.utexas.edu (128.83.122.129) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
2 128.83.37.41 (128.83.37.41) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
3 ser2-gi1-9.gw.utexas.edu (128.83.9.2) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
4 aus-core-ge1-1-0-2.tx-bb.net (192.88.12.89) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
5 192.88.12.22 (192.88.12.22) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
6 losang-hstnng.abilene.ucaid.edu (198.32.8.21) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
7 hpr-lax-gsr1--abilene-LA-10ge.cenic.net (137.164.25.2) rt=0.000 (0/1) fw=0.000 (0/1) co=0.000 (0/1) ro=0.000 (0
8 svl-hpr--lax-hpr-10ge.cenic.net (137.164.25.13) rt=0.000 (0/1) fw=0.000 (0/1) co=0.000 (0/1) ro=0.000 (0/1)
9 hpr-stan-ge--svl-hpr.cenic.net (137.164.27.162) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
10 bbr2-rtr.Stanford.EDU (171.64.1.133) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
11 Gates-rtr.Stanford.EDU (171.64.1.178) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
12 planetlab-1.Stanford.EDU (171.64.64.216) rt=0.000 (0/1) fw=0.000 (0/1) co=1.000 (1/1) ro=0.000 (0/1)
```

Figure 3: Sensor Pod: Tulip hop-by-hop loss measurement

A large body of research has focused on developing systems for distributed monitoring, aggregation, and querying in large networked systems. Examples of such systems include both commercial systems such as HP OpenView, IBM Tivoli's Total Monitoring Environment and CA's Unicenter as well as academic efforts such as Ganglia [7], IrisNet [15], PIER [10], Sophia [27], SDIMS [31], and Astrolabe [23]. While any of these can potentially be leveraged for our sensing information backplane, we chose SDIMS because of its scalability and flexibility features.

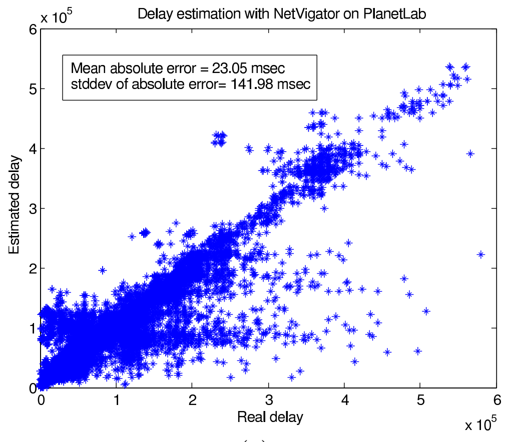
Systems such as NetProfiler [14], STRIDER [26], and PeerPressure [25] employ peer-to-peer information exchange for trouble-shooting of different network or system configuration problems. We believe NetProfiler can be implemented as an analysis engine in our S^3 architecture with NetProfiler's data acquistion sensors—TcpScope, WebScope—deployed in our sensor pods. Boutaba et al. citeboutaba01fcaps describe how Active Networks can enhance existing Network Management and examine the impact on five areas of ISO FCAPS framework [4]. We can view S^3 as an active network framework (note that sensor pods allow custom installation) and leverage S^3 system to deploy the techniques described in this paper to improve current network management.

Our system shares many of its goals with the Network Oracle [9] proposal and the Knowledge Plane [2] proposal. In our work, we focus on designing and building a system that can achieve a major subset of those goals and progress toward understanding the issues involved in achieving the grand goal of global pervasive monitoring and information management infrastructure.

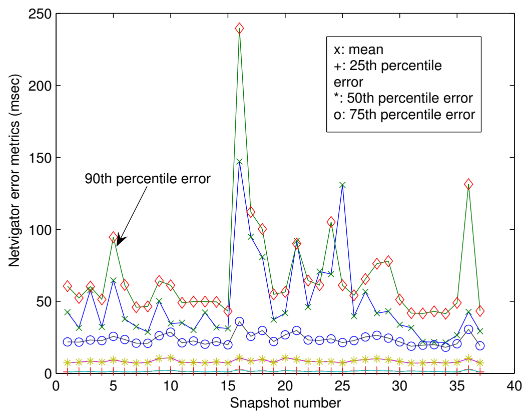
The scalable inferencing (e.g., [30]) of network properties is still in research stages. The latency metric has received the most attention, followed by the loss metric. IDMaps [6], King [8], and M-coop [20] answer latency estimation queries for any source and destination nodes by composing other measured data and they need infrastructure support. Landmark based approaches such as Netvigator [29], Landmark ordering [17], GNP [13], and Lighthouse [16] use landmark nodes for network proximity estimation. While we have described a distributed version of NetVigator in our architecture section, the other inference algorithms can also be potentially cast as analysis engines in our system.

5. CONCLUSIONS AND FUTURE WORK

We have presented S^3, a scalable sensing service for real-time and configurable monitoring and management system for large networked systems. Our architecture comprises of web-service enabled sensor pods with a variety of sensors that measure or monitor both network metrics and node metrics and expose flexible interfaces to satisfy a wide spectrum of sensor invocation policies. The sensing information backplane provides a substrate to aggregate the measured data from the individual machines in a scalable manner. Finally, analysis engines leverage the other two components to provide inference and operation control services. We have prototyped a subset of S^3 modules and deployed them on Planetlab to perform and aggregate all-pair measurements of several network metrics. We currently provide a four-hour period snapshot of inferred latency and bandwidth on our project webpage. We plan to add data for more metrics (e.g., loss rate) to this web site. We are currently working on the admission control policies, security, and privacy



(a)



(b)

Figure 4: Estimated vs. actual delay on PlanetLab using Netvigator (b) NetVigator error statistics

framework for the S^3 architecture.

6. REFERENCES

[1] Appmanager: Planetlab application manager. http://appmanager.berkeley.intel-research.net/.

[2] D. Clark, C. Partridge, J. Ramming, and J. Wroclawski. A Knowledge Plane for the Internet. In *SIGCOMM*, Aug 2003.

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

[4] FCAPS Overview: White paper. http://www.futsoft.com/pdf/fcapswp.pdf.

[5] R. Fonseca, P. Sharma, S. Banerjee, S.-J. Lee, and S. Basu. Distributed querying of internet distance information. In *Proceedings of the IEEE Global Internet 2005*, Miami, FL, March 2005.

[6] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A global internet host distance estimation service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, October 2001.

[7] `http://ganglia.sourceforge.net`.

[8] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the ACM IMW 2002*, pages 5–18.

[9] J. M. Hellerstein, V. Paxson, L. Peterson, T. Roscoe, S. Shenker, and D. Wetherall. The Network Oracle. In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*. 2005.

[10] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *Proceedings of the VLDB Conference*, May 2003.

[11] N. Jain, P. Yalagandula, M. Dahlin, and Y. Zhang. INSIGHT: A Distributed Monitoring System for Tracking Continuous Queries. Work-in-Progress Session at SOSP 2005.

[12] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet Path Diagnosis. In *Proceedings of the SOSP*, Oct 2003.

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

[14] V. N. Padmanabhan, S. Ramabhadran, and J. Padhye. NetProfiler: Profiling Wide-Area Networks Using Peer Cooperation. In *IPTPS*, Feb 2005.

[15] G. Phillip B, B. Karp, Y. Ke, S. Nath, and S. Seshan. Irisnet: An architecture for a world-wide sensor web. In *IEEE Pervasive Computing*, Oct 2003.

[16] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti. Lighthouses for scalable distributed location. In *Proceedings of the IPTPS 2003*.

[17] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of the IEEE INFOCOM 2002*, June 2002.

[18] 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.

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

[20] S. Srinivasan and E. Zegura. M-coop: A scalable infrastructure for network measurement. In *Proceedings of the IEEE WIAPP 2003*.

[21] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *ACM SIGCOMM*, 2001.

[22] 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.

[23] R. VanRenesse, K. P. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *TOCS*, 2003.

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

[25] H. Wang, J. Platt, Y. Chen, R. Zhang, and Y. Wang. Automatic Misconfiguration Troubleshooting with PeerPressure. In *OSDI*, Dec 2004.

[26] Y. Wang, C. Verbowski, J. Dunagan, Y. Chen, Y. Chun, H. Wang, and Z. Zhang. STRIDER: A Black-box, State-based Approach to Change and Configuration Management and Support. In *Usenix LISA*, Oct 2003.

[27] M. Wawrzoniak, L. Peterson, and T. Roscoe. Sophia: An Information Plane for Networked Systems. In *HotNets-II*, 2003.

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

[29] Z. Xu, P. Sharma, S.-J. Lee, and S. Banerjee. Netvigator: Scalable network proximity estimation. Technical report, HP Laboratories, 2005.

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

[31] P. Yalagandula and M. Dahlin. A scalable distributed information management system. In *ACM SIGCOMM 2004*, Aug. 2004.