Performance Measurement of the Scalable Sensing Service (S^3) on the ProtoGENI Control Framework

Nabeel F. Butt and Ethan Blanton, Purdue University
Project PIs: Sonia Fahmy, Purdue University, and Puneet Sharma, HP Labs

January 27, 2011

Abstract

Accurate performance evaluation of any deployed system is important to its user community. The distributed nature of network measurement infrastructures escalates the complexity of this performance evaluation. In this work, we evaluate the performance of a measurement service, the S^3 system [3]. We measure the performance in terms of resource utilization and response times under increasing measurement load. By response time, we mean the time between issuing a measurement request to the time we start obtaining measurement results. Resource consumption of the measurement service is also an important attribute to measure, since it can limit the scalability of the system.

1 Overview of S^3

S^3 [1] is a service for real-time and configurable monitoring of large networked systems. Its main components are the sensor pods, the sensing information management system (SIMS), and scalable inference engines. Sensor pods execute light-weight network measurement tools for sensing network path properties between any two end-nodes. SIMS is the component responsible for initiating the sensing (measurement) commands and later collecting sensed information from the sensor pods. The main purpose of inference engines is to leverage sensing tools on the sensor pods for more efficient measurement. The S^3 system we are profiling in this work is the prototype deployed on the ProtoGENI infrastructure [2]. We evaluate S^3 performance with respect to its resource consumption on both the end-nodes and the network, i.e., we will compute the entire system footprint. We will also evaluate response times for performing different operations with S^3.

Parameter Settings		
Tool	Parameters	Values
Latency	Frequency Interval Count	10 1 10
Path	Frequency	10
PathChirp	Frequency	10
Pathrate	Frequency	10
Spruce	Frequency Cap	10 100 MB
Tulip	Lag Count	10 10
SRS	Trials Interval	10 10
Configure (ping)	Frequency Interval Count	10 1 10

Table 1: Parameter settings for the measurement tools used. Interval is seconds between probe packets for a single measurement. Frequency is minutes between measurements. Count is the number of probes generated for a single measurement. Thus, the latency configuration in this table performs a measurement every 10 minutes, sending 10 probe packets at 1 second intervals for each measurement.

2 Experimental Setup

Our instrumentation of S^3 uses light-weight processes running on all the end-nodes as well as on the management node. These instrumentation processes are used for initializing performance measurement tools like tcpdump, top, iostat, vmstat, and collectl. They are also used for book-keeping purposes including recording tasks and events. They provide an easy and effective way to remotely monitor the resources used when running the experiments, for measuring the S^3 footprint.

3 Response Time

Our first experiment studies response time, defined as the time elapsed between issuing a command or initiating a measurement request and the time we begin receiving response or output from the management node. Response time is an important attribute to test the usability of the system.

An important feature of S^3 is that it supports a web interface to the

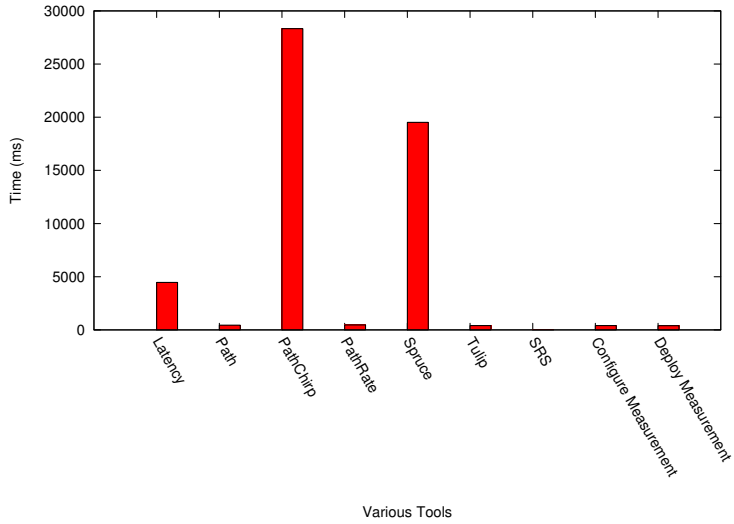

Figure 1: Response time of the different measurement tools

SIMS for operating on the slices created by the users. To reduce error in response time measurement, we subtract the round trip time (RTT) to the management node from the response time. In this experiment, we use the `ping` tool, and we send 10 packets with the interval between them set to zero. We evaluate the response time in two different sets of experiments.

In the first set of experiments, we measure the response time of all the

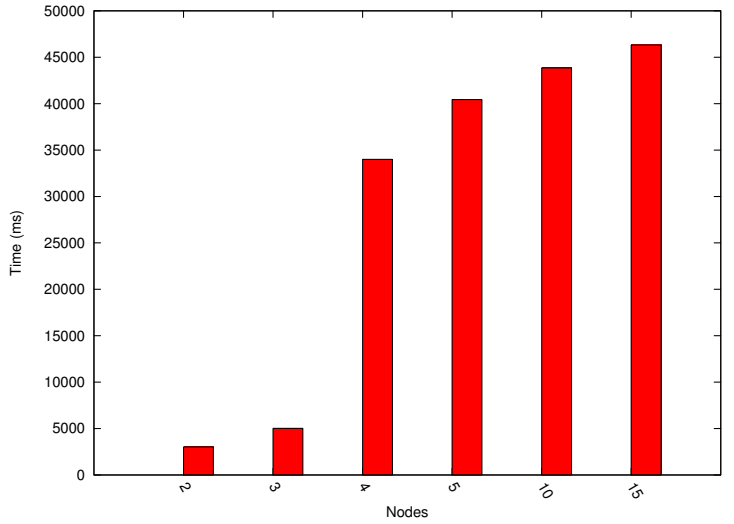

Figure 2: Increase in response time as the number of nodes increases

network measurement tools available through S^3. One thing to note is that this is not the response time of the measurement tools themselves, but rather the tools used through the S^3 interface. The S^3 interface can be operated remotely using a simple web client (web browser) or scripts. We list all the parameter settings of the all the tools used in Table 1. Note that for different settings, results could be dramatically different. We created multiple slices using the ProtoGENI interface and we evaluate the response times of all the tools. We completed 50 runs for each tool, and the results can be seen in Fig. 1.

In the second set of experiments, we measure the response times of deploying periodic measurements between every pair of nodes in the slice. The values (shown in Fig. 2) are the maximum response time of all these response times. This represents a worst-case load of all-pairs measurements. The measurement deployed for this experiment is `ping`. We repeat the experiment on different slices with different numbers of nodes, with 50 runs for each slice. The results shown in Fig. 2 are the average of the results. This experiment also provides insights into how the results vary as the number of nodes increases. As can be seen in Fig. 2, there is a sudden increase in the response times when the number of nodes is 4. The reason for this is that not all nodes can be allocated at the same physical site, and so the nodes in the slice are further apart. This is not the case when there are fewer than four nodes. As the number of nodes increases, this effect is consistent and we start seeing more stable results.

4 Resource Utilization

We now compute the S^3 footprint in terms of CPU utilization, memory utilization, disk I/O, and control traffic. These measurements are computed at the management node as well as at the end-nodes running sensor pods in the slice. CPU and memory utilization for the processes running these nodes are queried by the 'collectl' tool, and examination of the information in `/proc`. Disk I/O information is collected using the 'collectl', 'lsof', and 'df' tools. Control traffic measurements are taken with 'tcpdump.' In our case, the control traffic consists of the traffic used to initiate the experiments on sensor pods, the traffic to pull the results from the sensor pods, and finally the traffic between the sensor pods themselves. It also includes the control traffic of measurement tools (like `traceroute`, PathChirp, etc.).

We trigger periodic measurements on a slice and monitor the resource utilization parameters on all the sensor pods within that slice as well as on the management node. Before starting the experiments, we begin monitoring the operating conditions on all machines. For CPU and memory utilization we use the 'collectl' tool, and for disk I/O we use 'collectl' and information from `/proc`. We measure control traffic by using 'tcpdump' to

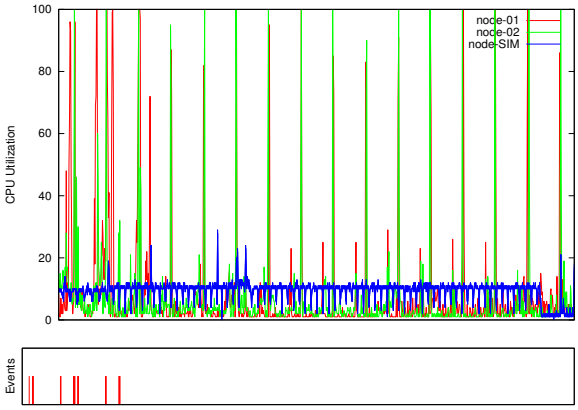

Figure 3: CPU utilization of the nodes

measure the amount of traffic that is sent to port 46000 on the sensor pods, and all traffic on port 22 (since we use 'scp' to fetch measurement results). We also capture the traffic between all the sensor pods. This instrumentation is controlled by an XML-RPC service which is deployed on the machines so that we can control and monitor them remotely. It also records events as they occur.

For this experiment, we pre-select the source and destination nodes, and run all the measurement tools on these two nodes. Meanwhile, we keep track of the CPU, disk, memory, and network resources consumed on these nodes. In figures 3, 4, 5, and 6, the graph on the bottom indicates when an event has occurred. These events are in the same order as in Table 1.

Fig. 3 plots the CPU utilization of the two sensor nodes and the management node, called node-SIM in the figure. It can be seen that the when an

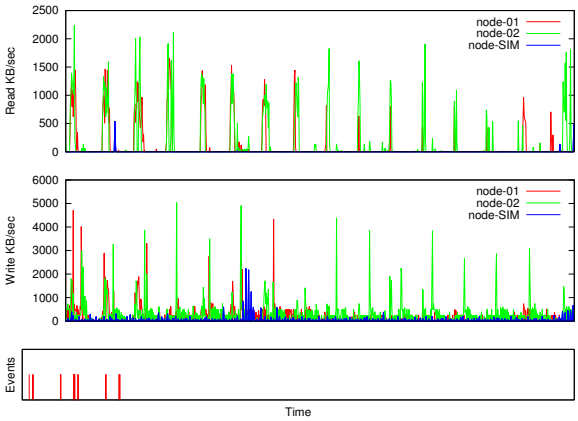

Figure 4: Disk utilization of the nodes

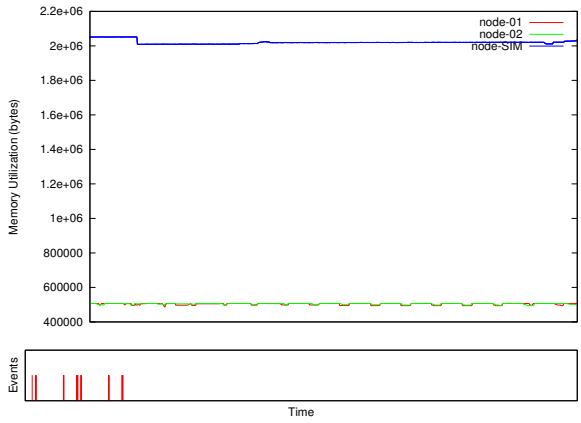

Figure 5: Memory utilization of the nodes

event occurs, there are spikes in CPU utilization. The initial spikes are due to deployment calls, and the later spikes are due to the periodic measurements. The management node, node-SIM, seems to experience little load due to measurements. On the other nodes, noticeable activity only happens when experiments are initiated or the next measurement period starts. In Fig. 4, we can see the disk usage of the nodes, including both data read and data write rates. This figure shows that when a new measurement period starts, there is a large amount of disk activity. This is because data collected by the measurement tools is read and then stored for reporting purposes. These readings are highly dependent on the measurements being performed. Memory utilization on all three nodes is shown in Fig. 5. The memory usage is only slightly affected by periodic measurements, the reason being that most of the programs running are already in memory. In Fig. 6, we can see the control traffic on the nodes. There are two iterations of this

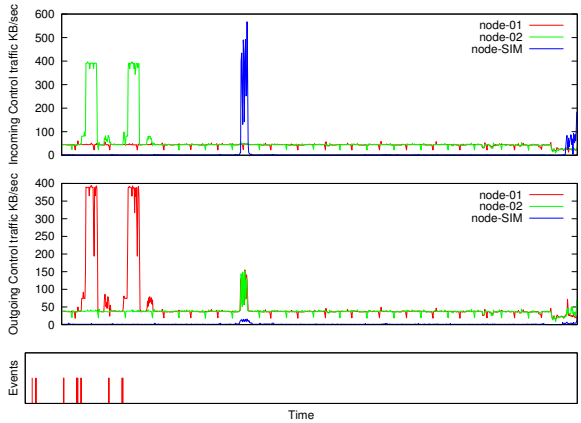

Figure 6: Control traffic on the nodes

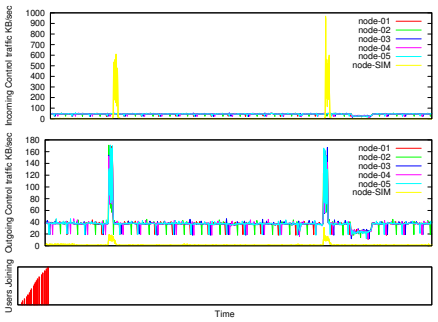

Figure 7: Control traffic for an interval of 10 minutes

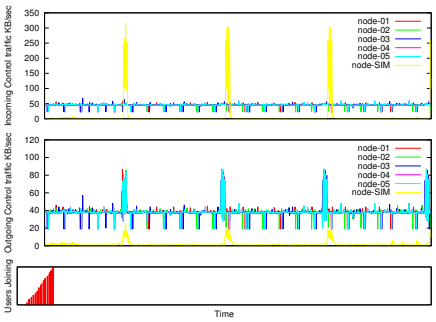

Figure 8: Control traffic for an interval of 5 minutes

experiment in the graph, and therefore there are two spikes in the figure. Note that we have assigned one node to be the source and the other the destination. This can be seen in the figure, as the outgoing traffic of one node mirrors the incoming traffic of the other. The management node later pulls the data from both of these nodes.

5 Impact of Measurement Interval on Resource Utilization

The management node pulls data from all the sensor pods after a specified interval. We investigate how resource consumption is affected by changing this interval. This interval should not be confused with the 'interval' parameter used in tools like `ping` or `traceroute`, which controls timing between individual probe packets. The resource we will examine in this experiment is control traffic, as it is the aspect affected significantly by the interval. In S^3, the default value of this interval is 10 minutes. We will see how the control traffic varies if we reduce this interval to 5 minutes. The only portion of control traffic affected is the traffic generated when the management node pulls measurement data from the sensor pods.

Control traffic bandwidth utilization can be seen in Fig. 7 and 8. The obvious result of reducing the interval is less bursty traffic. At 10 minute intervals, the highest reading for outgoing traffic from any sensor pod is slightly above 170 KB/sec, whereas at 10 minute intervals it is reduced to just under 90 KB/sec. The highest rate of incoming traffic at the SIMS is also reduced from almost 950 KB/sec to almost 300 KB/sec. With the smaller interval, however, exchange of control traffic from sensor pods to the management node is more frequent. The best configuration for this interval is highly dependent on the nature of the measurement profile. Some measurement usage and scenarios may be tolerant of large intervals, while others may be significantly affected by the collection delay these long inter-

vals induce.

6 Impact of Measurement Load on Response Time

In this experiment, we observe the change in response time as the number of users issuing periodic measurement requests to the system increases. Issuing a periodic measurement request is done in two steps. Users first have to configure the periodic measurements they wish to perform by specifying each tool to be used and the parameter settings for its invocation. Once this configuration step is complete, the next step is to deploy the measurements. After this deployment, the infrastructure starts performing the specified measurements.

This experiment consists of deployments of periodic measurements instrumented to record the response times of both the configuration and deployment steps. Results from this experiment can be seen in Fig. 9. As shown in the figure, the response time for the 'configure measurements' step varies from 450 ms to around 1400 ms. About 95% of the time, it is below 800 ms. The deployment step response time varies from 875 ms to 1400 ms. It varies quite a bit as the number of users increases, but there does not seem to be clear correlation with the number of users for this small number of users. We will be investigating larger numbers of users in our future work.

7 Conclusions

In this report, we have examined a few performance characteristics of S^3 [3]. We designed experiments to understand its impact on various resources such

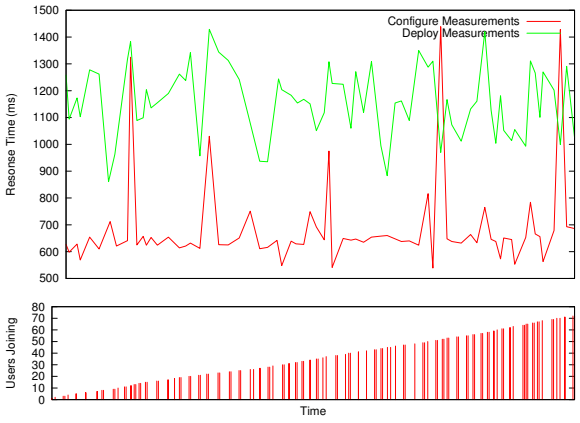

Figure 9: Variation in response time with increasing number of users

as CPU, memory, disk, and network control traffic. We also examined how the response times of different operations vary under changes in user load.

References

[1] Praveen Yalagandula, Puneet Sharma, Sujata Banerjee, Sujoy Basu, and Sung-Ju Lee. 2006. S3: a scalable sensing service for monitoring large networked systems. In Proceedings of the 2006 SIGCOMM workshop on Internet network management (INM '06). ACM, New York, NY, USA, 71-76.

[2] http://www.protogeni.net/trac/protogeni

[3] http://illusion.hpl.hp.com:8180/geni/demo