Workload Generation and Instrumentation for Reproducible Networking Experiments

Tilman Wolf*, Ellen Zegura† and Santosh Vempala†
*Department of Electrical and Computer Engineering University of Massachusetts, Amherst, MA, USA
wolf@ecs.umass.edu
†School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA
{ewz,vempala}@cc.gatech.edu

***Abstract*—We propose the design of a benchmarking and instrumentation component for the GENI experimental networking testbed. Our design uses a workload generation system to create reproducible network conditions and an instrumentation component to collect measurement data. We discuss the research challenges related to this design.**

I. Introduction

Research on the design and implementation of computer networks requires experimental validation. As with any complex engineered system, theoretical models and simulation are useful in the early stages of the design process, but experimentation is necessary to obtain results on actual system performance under realistic conditions. Therefore, the networking community has put much emphasis on experimentation on real, deployed network and the measurement of network performance.

When performing experiments on actual, deployed networking infrastructure, results are highly realistic. However, it is very difficult to experiment with novel network protocols unless they are compatible with those already deployed. Therefore, it is necessary to develop networking testbeds, which allow experimentation with entirely new protocols and applications that are not backward compatible. A key aspect of such testbeds is the need scalability to hundreds or thousands of nodes. Only larger systems can produce the operating conditions encountered in typical networks. This scale as well as the distributed nature of network applications and protocols makes it particularly challenging to perform experiments under reproducible conditions. Thus, benchmarking has been difficult in the networking domain.

Our work proposes a benchmarking and instrumentation system for the GENI (Global Environment for Network Innovation) testbed [10], which is a large experimental infrastructure. The ability to control many components of the testbed makes it feasible to attempt the design of a benchmarking system. The well-designed control infrastructure of GENI provides users with unprecedented ability to set up experiments. In this context, we describe our benchmarking and instrumentation system. Specifically, we discuss

- How to set up automated experiments and control the topology of the network as well as traffic generation to provide basic capabilities for reproducible experiments and
- How to design flexible instrumentation systems to collect measurement data.

Our work discusses the general design issues of such a system and raises some of the research problems encountered in this space. Results from a specific implementation are left to future work.

The remainder of the paper is organized as follows. Section II discusses related work. Issues related to reproducibility of experiments are discussed in Section III. The overall system design is presented in Section IV. Details on workload generation are presented in Section V and on instrumentation in Section VI. Section VII summarized and concludes this paper.

II. Related Work

Recent efforts in the networking community have aimed at designing a next-generation network architecture. This architecture design is based on a clean-slate approach to incorporate novel networking technologies and paradigms [5]. Novel research results not only aim at improving the networking infrastructure, but also novel application use.

Testbeds for research on networking infrastructure include Emulab [16], Planetlab [2], the Open Network Laboratory (ONL) [4]. Emulab and ONL are based on dedicated end-system and networking resources, whereas Planetlab uses an Internet overlay for experimentation. Recently, an initiative by the United States National

Science Foundation has launched an effort to develop a Global Environment for Network Innovation (GENI) [10]. This program aims at developing a large-scale experimental infrastructure that can be used for network-level and application-level experimentation.

Performance evaluation and measurement has a long tradition in computer systems [8], [9]. Particularly important has been the development of benchmarks (e.g., Whetstone [3], Dhrystone [15], and SPEC CPU [12]). In particular, the SPEC suite of benchmarks is developed through community effort and thus widely accepted. More recently, similar benchmarks have been developed in specialized areas (e.g., embedded systems (MiBench) [6], transactions-level processing (TPC) [14], and web servers [11]). Our work aims at providing a basis for developing similar benchmarks on the GENI testbed. An early specification of the GENI Instrumentation and Measurement System is specified in [1]. Our work fits into the concepts laid out in this work.

III. Reproducible Research

Reproducible research is an essential criterion for science. The networking community has struggled with achieving reproducibility mostly due to the complexity of the distributed nature of networks and the lack of a common experimental facility. GENI promises to overcome these obstacles and allow such reproducibility. A common benchmarking and instrumentation platform is essential in this process.

The ultimate goal of our work is to achieve a level of reproducibility and community acceptance that has been achieved in other fields: the SPEC CPU benchmarks in computer architecture [12], the TPC benchmark for transaction processing [14], the MiBench suite for embedded systems [6], etc. A networking researcher should be able to propose new ideas for protocols or systems and have the ability to easily obtain performance results with standardized topologies and workloads. A key criteria is that these standardized scenarios are contributed and selected by the community and thus become widely accepted. Our benchmarking system is designed specifically to be extensible in the sense that the GENI community can and should help develop the benchmarking scenarios that will be used. The goal of this design is to provide the infrastructure, but not to prescribe the usage scenarios.

Our benchmarking and instrumentation system aims at solving a fundamental requirement for GENI, which is to allow for scientific research through reproducible experimentation and data collection. We envision that

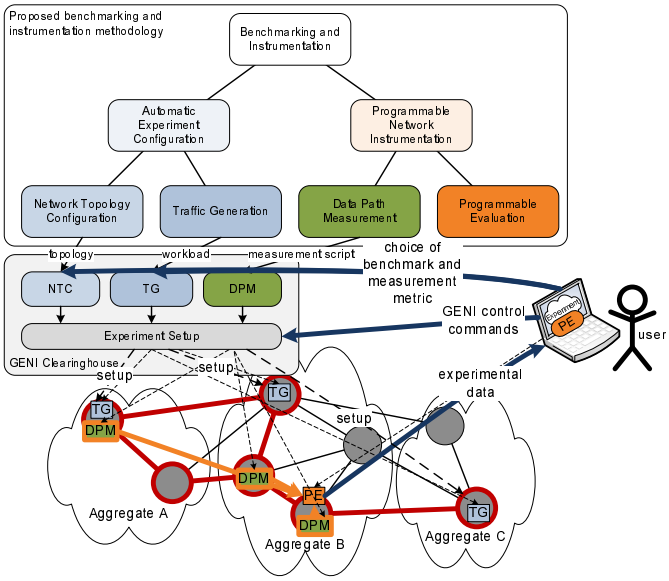

Fig. 1. System Architecture of Benchmarking and Instrumentation System.

benchmarking and measurement can be used widely in a number of scenarios:

- During testing of GENI platform prototypes to compare design alternatives;
- During experimentation with network systems where networking researchers explore system performance;
- During experimentation with applications running on the GENI infrastructure where users need to validate the correct operation of the network, identify performance bottlenecks, etc.;
- During operation of GENI by administrators.

Over the life of GENI, we expect that many different types of hardware platforms will be used as network nodes. In our visions, the workload generation and programmable measurement concepts will also be applicable to these emerging platforms. Thus, new GENI nodes could implement the design discussed in this paper on their specific system architecture (e.g., network processor, FPGA).

IV. System Design

The system architecture for our proposed Benchmarking and Instrumentation System (BIS) is shown in Figure 1. The figure also illustrates how it fits into the overall GENI architecture. We envision that a user can specify the benchmark and measurement metrics when setting up an experiment with the GENI Clearinghouse. The clearinghouse is equipped with our topology configuration, workload generation, and data path measurement

components, which are installed accordingly on the GENI substrate when the experiment is set up. Then, the user can control the experiment and obtain measurement data via the programmable evaluation component.

Our main focus is on the design of the following components, which are described in more detail in the following sections:

- Experimentation Configuration Subsystem: The goal of this benchmarking subsystem is to provide a GENI component that manages the configuration of network topologies and generation of traffic during experiments. This component ensures that networking researchers can use topologies and traffic workloads that are representative and that measurement results are reproducible. Our design aims at controlling experiment configurations and generating synthetic traffic on network nodes. The GENI community can extend the benchmarking system by adding network configurations and workload generation tools as networking research evolves.
- Instrumentation Subsystem: The goal of the instrumentation subsystem is to provide a general, programmable measurement system that can be used across GENI systems. This system simplifies the process of obtaining measurement data while GENI experiments are in progress. These data can be used by networking researcher to evaluate new architectures, by GENI operators to monitor system performance, and by GENI application users to monitor progress of experiments. The programmable data path measurement component allows the collection of passive measurement data. Processing them locally avoids the collection of large trace files that need to be harvested later. The processed measurement data are collected in a centralized evaluation system for correlation, logging, and presentation to users.

Both subsystem areas are tightly coupled as they mutually depend on each other. In combination, they provide a platform for reproducible experiments.

V. Automated Experiment Configuration

Researchers can benefit tremendously from tools to automatically configure experimental topologies and traffic. Such tools can reduce researcher effort, minimize researcher error at configuration time, and allow for reproducibility and standardized comparisons across different experimental instances.

Clearly, different experiments desire different levels of control over configuration. Some experiments run on long-lived virtual networks with real user traffic, and hence exert no additional control over experiment configuration. Beyond these piggyback experiments, any experiment that configures a new network minimally wants to specify something about the topology (connectivity) of the network elements. At the next level, a researcher may want to control network link parameters, including bandwidth and latency (fixed delay). Experiments that attract real opt-in users may not need or want to specify any artificial workload. However, experiments that use generated traffic may also desire to specify something about the workload carried by the topology.

We further posit that *benchmarks*, standards across these types of configuration, will prove invaluable to researchers by supporting community agreement and investment in a set of "good" experimental configurations that many experiments can utilize. Our Automated Experiment Configuration (AEC) supports benchmark topologies, with link characteristics, and generated traffic. We expect to realize our benchmarks in libraries that can be easily selected by a user for a particular experiment.

A. Network Topology Configuration

What matters to *users* of topologies? Fundamentally, a researcher using topologies wants to evaluate a proposed solution to understand how well it works. As in algorithmic analysis, both worst-case and expected-case understanding can be valuable. A rigorous approach to both types of analysis, however, requires describing the space of topologies under consideration ("bounding" the universe).

Our goal is to develop benchmark suites of topologies for studying algorithms and protocols in networks. A "good" benchmark suite should have the following characteristics:

- Coverage: The suite should sufficiently represent different parts of the space of possible topologies, allowing testing under different conditions.
- Scale: The suite should be capable of generating topologies of many different sizes.
- User control: A user testing an algorithm or protocol may have some intuition or evidence about what topological characteristics are most relevant for performance. The suite should allow a user some control over which dimensions are fixed and which are variable.
- Minimality: A minimal suite is preferred, one that produces as few topologies as necessary and provides as few knobs as possible while providing

sufficient coverage and user control.

We have made research progress in this area by mathematically characterizing what it means for a benchmark suite to be good, using the concept of coverage of the target space provided by a topology. We have developed a method to generate a good suite, as mathematically defined, that also allows some user control based on characteristics of interest. The key idea behind our method is to use a *skeleton* that defines the structured part of a graph, followed by randomness to fill out the skeleton. A graph is in the space if it can be generated in accordance with the skeleton and random details. We have two motivations for this skeleton approach. The first is that the form of our skeleton is intuitively natural for networking research. We believe it accommodates high-level topology characteristics that researchers and network designers tend to find useful for consideration. The second motivation comes from the regularity lemma from graph theory, which says, roughly, that any graph can be approximated by a graph with a small structured part.

Random generation from a skeleton produces an arbitrary number of topologies. To capture the notion of representation we allow the user to specify one or more graph functions that guide our system in reducing the large number of generated topologies to a small number for the benchmark set. We call these functions *filters*.

We use two types of coverage. The first, *uniform coverage*, identifies a subset of topologies that are nearly uniformly spaced in the parameter range. The second type of coverage, *extreme coverage*, identifies topologies that are extreme (minimum or maximum) for at least one of the parameters. We believe researchers may find each type of coverage useful in different settings.

Beyond connectivity, some experiments may want to specify link characteristics, such as bandwidth capacity and latency (fixed delay). We can augment our topology generation with bandwidth and latency associations for links. This is a simple extension of our generation method so that link probabilities also include distributions for bandwidth and latency. We can extend our filters to allow graph functions on the bandwidth and latency of the links, not just on connectivity. We can provide extensibility for any semantic associated with links.

Thus, we know it is technically feasible to generate a benchmark suite of topologies. In next steps, we may take that technical core and create the tools necessary to give researchers access to well-grounded suites.

There are, of course, open questions in this area. These include: how should specified topologies, especially with bandwidth and delay characteristics, be mapped efficiently onto the available resources? How will Aggregates express their ability to support different types of topology characteristics? Will users have visibility into Aggregates, or will tools such as ours be responsible for mediating between high level user desires and available Aggregate resources?

B. Traffic Generation

In traffic generation, our design supports two types of basic workload generation: generation from a parameterized traffic distribution and generation that replays a trace. We may include all standard distributions for traffic generation. On top of parameterized traffic distributions, we may construct higher-level specifications of network load that may be more natural for researchers to use, e.g., peer-to-peer file sharing, interactive web surfing, massive multi-player game. We can also make use of the best known models for specific application workloads to realize these sorts of specifications.

Our replay capability allows the capture of trace information from a prior GENI experiment or from another measurement system. We can adopt or develop a standard format for trace representation to support replay. We may support the chosen format in our data path measurement tools so that we can capture traces in GENI experiments and replay them later for reproducibility or to support a controlled comparison.

Open questions in this area include: what level of detail for "traffic" will GENI experiments find useful? Will some experiments desire bit-level generation to experiment with alternative packet formats or even circuits? How should alternate packet formats interact with traffic generation? For example, could a user can supply a packet header format or rule that is automatically attached to packets generated at a specified source? These are important considerations that will need to be settled through a combination of prototyping experience and community discussions.

VI. Programmable Network Instrumentation

In conjunction with the ability to set up experiments automatically, researchers want to be able to observe what is happening in the network. Such instrumentation capabilities are essential and integral components of any large experimental facility. For GENI, we propose a programmable measurement infrastructure that provides easy access to network measurement data while at the same time providing the flexibility to customize what information is recorded and how it is preprocessed.

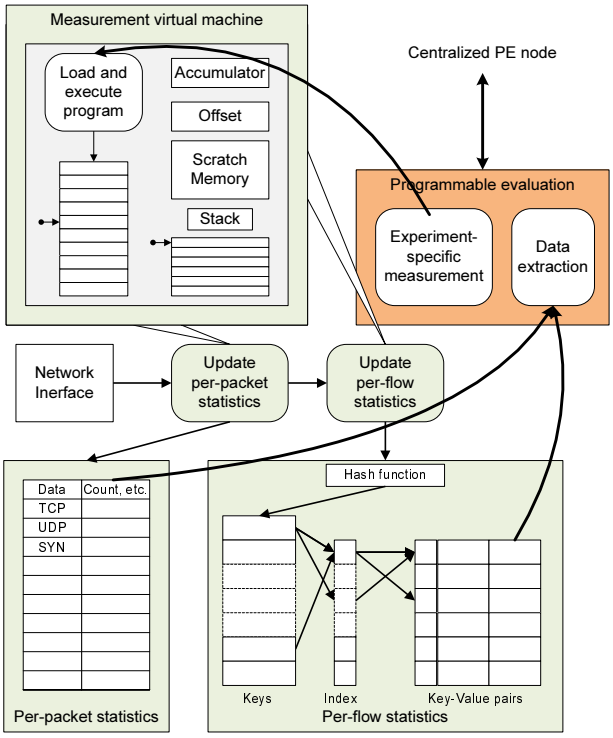

Fig. 2. Data Path Measurement Component.

Programmability in the network instrumentation component is a fundamental aspect of our design. Since we cannot predict what measurement results GENI researchers need to obtain for their experiment, we make as few assumptions as possible. Instead of prescribing a fixed set of measurement options, a programmable instrumentation component, as described in this proposal, allows researchers to define what data to record, how to preprocess data, and how to report them. This programmability can be provided through a simple script interface to the measurement system.

As illustrated in Figure 1, there are two subcomponents that make up the Programmable Network Instrumentation (PNI) system: Data Path Measurement (DPM) and Programmable Evaluation (PE). The first handles the per-packet processing that is necessary to obtain passive measurement data on a network node. The latter handles the processing of these data on the node, collection of multi-node data in a centralized PE system, and presentation of results to the user. We describe each component in more detail below.

It is important to note that we focus on *passive network measurement* in this design. Active network measurement can be seen as an application-layer function and thus does not require as much infrastructure support as passive measurement does.

A. Data Path Measurement

The Data Path Measurement component of our architecture performs all operations that are necessary to capture per-packet measurement data. The key characteristic of DPM is that it performs *online measurement*. Unlike many traditional passive measurement systems, DPM does not collect large packet traces that need to be processed later to harvest information. (Optional trace collection for archiving purposes or for later playback in traffic generation is a possible feature.) Instead, DPM processes packets as they traverse the system and collects the information that is relevant to a particular experiment.

An example that highlights these differences is an experiment where the packet size distribution of a particular protocol stack and network configuration needs to be obtained. In a trace-based measurement system, all packet headers are collected in a huge file. After completion of the experiment, the file is parsed to extract packet length fields and the resulting distribution is reported. In online measurement, the DPM component parses every packet as it traverses the node and collects the distribution locally. Not only does this require no off-node storage, but the results can also be reported *live* while the experiment is in progress.

The ability to collect and process data online (and based on instructions provided by the PE system) can be provided by a system that is illustrated in Figure 2. The key idea is to use a virtual machine that is programmable with a simple, measurement-specific instruction set to extract packet and flow-specific data. Conceptually, this is a similar approach to how the `pcap` library [13] provides filtering capabilities via a virtual machine. The instruction sequence is provided by the programmable evaluation component that is controlled by the user.

We have developed a prototype of such a DPM system on an Intel IXP2400 network processor [7]. Details and performance results are provided in a recently published paper [17]. The prototype does not support flow-specific measurement and virtual machine capabilities are limited to filtering specific packets and selecting from a fixed set of aggregation statistics. Nevertheless, our prior work indicates that such a system is feasible for use in GENI.

Once measurement statistics have been collected, they are extracted by the programmable evaluation component as described below. Note that the DPM system is illustrated for a single experiment. When using a substrate with multiple active slices, then DPM (and PE) is replicated to each slice.

B. Programmable Evaluation

The Programmable Evaluation component consists of a centralized component that aggregates the measurement data collected for the PE components on each node that runs a DPM. The user controls what data are retrieved from each DPM node and how it is stored and presented to the user. The PE component also provides an interface where the user can specify DPM operations that are executed on the measurement virtual machine.

One important functionality of the PE system is to monitor resource consumption by ongoing measurements on the DPM. If a user requires too many different per-packet or per-flow statistics or creates virtual machine code that is too complex, then the DPM component is not able to process all packets at line speed. To avoid this problem, the PE system can limit the number and complexity of measurement programs that are installed by a user.

It can be expected that GENI will consist of many different hardware platforms ranging from high-performance wired router platforms to potentially mobile wireless nodes. We believe that the proposed measurement architecture is suitable for implementation on a range of different hardware architectures. For high-end routers, network processors or even field-programmable gate arrays (FPGA) could be considered. For lower-end systems general-purpose processing platforms may suffice.

As indicated above, we have previously implemented a system similar to DPM but with somewhat less functionality. While this presents a good starting point for the PNI subsystem, there are still several open questions: How can we balance a general-purpose instruction set in the measurement virtual machine with the need for predictable high-throughput operation? What aggregation statistics will be typically used by researchers? What flow classification mechanism will be typically used by researchers? How much content inspection can be provided given performance constraints? Should the DPM system be shared between slices using the same substrate?

VII. Conclusions

Reproducible network experimentation is a crucial aspect of network testbeds. Our design of a benchmarking and instrumentation component could contribute an important capability to the GENI testbed. Our system can make it easy for researchers to set up experiments and obtain measurement results without having to understand the details of the underlying infrastructure. Using broadly accepted benchmarking scenarios further increases the attractiveness of GENI as an experimental platform as it allows researchers to test their network systems and applications against a broad set of standard workloads.

References

[1] P. Barford. Geni instrumentation and measurement systems (GIMS) specification. Technical Report GDD-06-12, GENI: Global Environment for Network Innovations, Dec. 2006.

[2] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating system support for planetary-scale network services. In *Proceedings of the 1st Symposium on Network System Design and Implementation (NSDI '04)*, San Francisco, CA, Mar. 2004.

[3] H. J. Curnow and B. A. Wichmann. A synthetic benchmark. *The Computer Journal*, 19(1):43–49, Feb. 1976.

[4] J. DeHart, F. Kuhns, J. Parwatikar, J. Turner, C. Wiseman, and K. Wong. The open network laboratory: a resource for networking research and education. *ACM SIGCOMM Computer Communication Review*, 35(5):75–78, Oct. 2005.

[5] A. Feldmann. Internet clean-slate design: what and why? *SIGCOMM Computer Communication Review*, 37(3):59–64, July 2007.

[6] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *Proc. of IEEE 4th Annual Workshop on Workload Characterization*, Austin, TX, Dec. 2001.

[7] Intel Corporation. *Intel Second Generation Network Processor*, 2005. http://www.intel.com/design/network/products/npfamily/.

[8] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley-Interscience, New York, NY, Apr. 1991.

[9] D. J. Lilja. *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, July 2000.

[10] National Science Foundation, http://www.geni.net/. *Global Environment for Network Innovation*.

[11] Standard Performance Evaluation Corporation, http://www.spec.org/. *SPECweb 2005 - Version 1.20*, Dec. 2007.

[12] Standard Performance Evaluation Corporation, http://www.spec.org/. *SPEC CPU2006 - Version 1.1*, Aug. 2008.

[13] TCPDUMP. *TCPDUMP Public Repository*, 2003. http://www.tcpdump.org.

[14] Transaction Processing Performance Council. *TPC Benchmark C, Revision 5.1*, Dec. 2002.

[15] R. Weicker. Dhrystone: A synthetic systems programming benchmark. *Comm. of the ACM*, 27(10):1013–1030, Oct. 1984.

[16] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.

[17] T. Wolf, R. Ramaswamy, S. Bunga, and N. Yang. An architecture for distributed real-time passive network measurement. In *Proc. of 14th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 335–344, Monterey, CA, Sept. 2006.