A New Approach to NGN Evaluation Integrating Simulation and Testbed Methodology

Marcial P Fernandez
Universidade Estadual do Ceará (UECE)
Fortaleza, Brazil
marcial@larces.uece.br

Sebastian Wahle
Fraunhofer FOKUS
Berlin, Germany
sebastian.wahle@fokus.fraunhofer.de

Thomas Magedanz
Technische Universität Berlin
Berlin, Germany
tm@cs.tu-berlin.de

***Abstract*—Since the beginning of Internet, network researchers have been proposing methodologies and tools to facilitate the design and development of new protocols for Internet. Analytical modeling, network simulation, network emulation, and more recently, testbeds, are being used in these researches. However, there are advantages and disadvantages in all these methodologies making difficult to decide on the ideal methodology and tool. In this paper, we propose a new methodology to evaluate Next Generation Networks that permits the integration of design, development and test of a new protocol or network service using different tools. The approach was demonstrated by designing a simple example of a network service.**

Keywords-Next Generation Networking (NGN); Network Evaluation; Network Simulation; Network Emulation.

I. INTRODUCTION

Design, development, and validation of networks protocols and services are important research issues. Generally, for analysis and comparison of different mechanisms and algorithms, five techniques are applied: analytical modeling, network simulation, network emulation, testbed and real-world experiments. Over the past 50 years, these methodologies were used to develop the protocols used on the Internet today. Although these techniques are known for many years, the predominant use of each technique over the years is dependent upon computer capacity. The potentials and limitations of these methods have been widely discussed by Jain [1].

Concerning the techniques being used currently, simulation, emulation and testbed, we can say that the first is the most distant from reality but is the easiest to work with, while the latter is the closest to real world but it is the most difficult for researchers to use. Regarding costs, simulation is cheaper than emulation and testbed. But because it is close to the real world, testbeds are hard to do, expensive, have a fixed topology, fixed environment, and it is difficult to create new impairment scenarios (broken link, routing table errors, high drops). These objectives further evolved towards refinement of experimentally-driven research as a visionary multidisciplinary research, defining the challenges for and taking advantage of experimental facilities, realized by means of iterative cycles of research, oriented towards the design and large-scale experimentation of new and innovative paradigms for the Future Internet - modeled as a complex distributed system.

A good methodology for the development of protocols and network services would be the use of both approaches: simulation and testbed. The first step, simulation, could be used to test and debug the first prototype, taking advantage of the facility of creating scenarios, traffic sources and network failures. In the second step, we could consider carrying out experiments on a testbed, now taking advantage of the reality offered by this methodology. Thus, it becomes very interesting to create a tool that permits the integration of these two methodologies to make the work of network researchers easier.

In this paper, we present a new tool to provide integrated simulation/experimentation environment to permit to develop protocols and services with four main contributions: provide a unifying approach to simulation/experimentation that makes the transition easy from simulation to network testbeds; provide a graphical interface to facilitate the topology creation and traffic definition; provide analysis tools to permit comparison of simulation and experimentation results; offer a layered and modular architecture to permit to evaluate specific parts without modification on the testbed facilities.

The rest of the paper is structured as follows. In Section II, we present some related works, Section III introduces the network research methodology and in Section IV we present our proposed methodology. Section V shows the proposal evaluation and the results and, finally, Section VI concludes the paper.

II. RELATED WORKS

Netbed/Emulab is a network testbed project, aiming to give network researchers an environment to develop, debug, and evaluate networked systems. Emulab project started as a emulation facility at the University of Utah [2], and consists of a cluster of emulation devices running an *ns-2* (Network Simulator Version 2) script [3]. One of the main contribution of this project was the ns-2 to emulation mapping [4]. The following works focused on implementing network

 ISBN: 978-1-61208-183-0

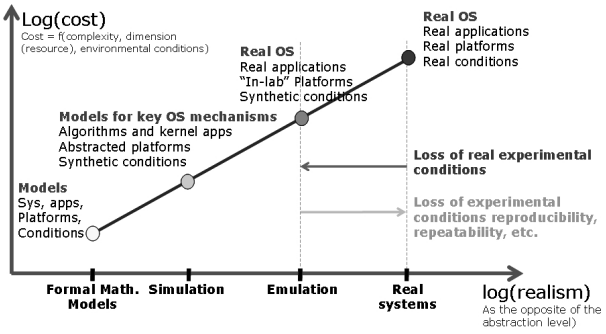

Figure 1. Network research methodology

emulation facilities on PlanetLab testbed [5] [6]. Other contribution of this project was the importance of simulation and emulation integration on network experimentation [7]. The Flexlab is a new framework that combines overlay and emulation testbeds (PlanetLab + Emulab), running an application within the emulation testbed and uses its load to measure the overlay network [8].

PL-VINI [9] is an implementation of VINI on PlanetLab. It runs on each PlanetLab slice providing network resources like link delay, link drop and routing. PL-VINI provides a realistic and controlled environment for evaluating new Internet protocols and services. Some features that could be evaluated in PL-VINI are: routing software, traffic loads and network events. To provide researchers flexibility in designing their experiments, VINI supports arbitrary network topologies on a shared physical infrastructure.

NEPI [10] is a framework proposal that makes the execution of a network experiment possible in different tools, e.g., simulation, emulation, and testbed. NEPI focuses on executing experiments using multiple tools separately and together in order to improve researchers productivity. The tool was implemented in Python and has a script and a GUI interface.

III. Network Research Methodology

The rationale was thus clear: to create a dynamic between elaboration, realization, and validation by means of iterative cycles of experimentation. Nevertheless the "validation by experimentation" objective opens a broad spectrum of experimentation tools (in large sense) ranging from simulation to real system experimentation. Our thesis is that "elaboration" requires validation by means of more abstract tools (not only because their resulting cost is lesser but because such tools produce results verifying all conditions explained here below) followed by progressive addition of realism as part of the experimented system to ultimately reach so called field trials with real systems. Thus, systematic experimentation is a continuum (Figure 1).

1) "Computer Communication/Networking" is characterized by two fundamental dimensions: distribution of a large number of dynamically interacting (non-atomic) components and the variation of their inner properties that in its turn influences these interactions. Thus compared to computer science, the distribution/interaction and the large number of elements composing the system add two fundamental dimensions to computer science "paradigms".

2) On the other hand, one shall characterize the output of experimentation: in order to ensure verifiability, reliability, repeatability, and reproducibility of the experimental results. Ensure these properties implies in provide strict control to the experimental conditions (parametrization, i/o, and running). Verifying the repeatability, reproducibility, and reliability conditions ensures generalization of experimental results, and verifiability of their credibility.

3) Different experimental tools can be used. As stated above their selection is neither arbitrary nor religious: it depends on the experimental objective and maturity of the experimented corpus. Nevertheless, each of them needs to ensure that the conditions defined here above are verified. However it is clear that fulfilling these conditions does not come at the same cost for the same level of abstraction. Validation of a new algorithm would be better conducted on a simulation platform (after formal verification) not only because their resulting cost is lesser but because such tools produce results verifying all conditions explained here above. Emulation experiments can lead to reproducible and repeatable results but only if "conditions" and "executions" can be controlled. Realism can thus be improved compared to simulation (in particular for time-controlled executions of protocol components on real operation system).

A. Simulation

Network simulation is a technique in which a software simulates the behavior of a network and its components (routers, hosts, links, protocols, etc) by calculating the interaction between them using mathematical models. Most network simulators use discrete-event simulation, in which a list of events are processed according to a virtual time, independently of the computer's clock where the simulator software is running. Then, a simulation produces the same result in different computers. Since the beginning of Internet, network simulators have been an invaluable tool for network researchers.

The ns-3 is a discrete-event network simulator, intended to replace the traditional ns-2 simulator [3]. The first release of ns-3 was published in July of 2008 and it has been improved and extended since then. Since it was proposed, ns-3 concept was to be a simulator capable to interact with real world. Some improvements pointed in this direction, e.g., the ns-3 API is a Unix socket-like API, to permit easy migration from simulated code to real-system code, and the Network Simulation Cradle (NSC) allows ns-3 using the Linux TCP/IP stack.

 ISBN: 978-1-61208-183-0

B. Emulation

The Emulation is the technique where a network is simulated in a real hardware and software. The emulation platform implements virtual network topologies and scenarios over real hardware and protocols, i.e., that experiments can be executed in real hardware, use real operating systems and protocols, run their real applications, and obtain actual (not simulated) performance measures. Although emulation is much closer to real environment than simulation, the links should be simulated in order to create delay and communication impairments (noise, drops, etc). Sanaga et.al. [11] shows the difficulties to emulate a network link.

IV. A New Methodology for Network Research and Experimentation

The development of new protocols and services for Internet requires a series of procedures before it can be used in the real world. The first one is to verify whether it works, i.e., the protocol or service performs what we want. Then we must explore the parameter space to find the best configuration to achieve the best trade-off. Thus, if the protocol or service is working, we must verify that it will not kill the network. Finally, we need to perform a couple of experiments to see the overall performance, and scalability. The network research environment must provide:

Reality: the proposal should be tested in real environment.

Configuration: large scale experiments require a lot of configuration.

Instrumentation: need to gather data about the behavior of the experiment to figure out what happened.

Fidelity: did the experiment really capture the effects you are really interested in?

Reproducibility: scientific methodology means that you must publish reproducible results.

The methods currently used to develop protocols and services for Internet are the simulation and the emulation testbed. However, there are advantages in both of them, not found in the other, so instead of comparing both methodologies, we associate them. The first step, simulation, allows easy creation of different scenarios, as well as different types of traffics. As the prototype is running on a computer, we can create a series of experiments that will allow evaluation of the prototype operation in many different situations. The simulation makes easy the creation of impairment in environment, such as link break, link degradation (increased of drop rate), very long delays, routing instability, evaluating the prototype in very adverse situations.

Simulation provides facility to change the source code (we do not need to change any code in multiple remote machines, only on the simulation server) also facilitating the prototype development. The code can be changed and tested very quickly. Another advantage of simulation is the possibility of taking execution snapshots, e.g., we can simply put a *printf* in simulated code. In a testbed, it is often difficult to change the code and create mechanisms for collecting information, making the code debug hard. Since the simulator environment is controlled, it is possible to obtain the repeatability necessary to validate a scientific work. The creation of traffic sources statistically distributed in a controlled environment allows repeatability, which is very difficult to obtain in a testbed due to the difficulty to reproduce the same situation and events in a certain moment. However, simulations tend to be unrealistic. The packets do not pass through a real network, so even if it used sophisticated simulation tools, it continues far from reality in some scenarios. Emulation can provide a more realistic environment because it uses real machines and real operation system. But it is also difficult to emulate the reality, e.g, Sanaga et al [11] shows the difficulties to emulate an Internet path.

Nowadays testbeds are presented as the ideal methodology to develop and test protocols and services for the Internet. As it is an extract from a real network, tests are performed in an actual infrastructure. The possibility of setting up paths and the monitoring tools offer a control degree that allows the creation of specific test situations. But as it is a separated experimental environment, there is no risk to damage the production network. Despite its proximity to the real world, Testbeds are not the ideal tool. Setting up a testbed is complex and may require individual configuration of each resource. The repeatability of an experiment is difficult to be achieved, given the environment unpredictability. Another difficulty is to perform measurements on a testbed compared to simulation, usually performed through measurement points placed on routing devices. Most of testbed provides the collecting of statistics information (e.g., sFlow) or collecting flow traces (e.g., *pcap* format).

As Testbeds could be categorized as an Overlay Emulation, i.e., Testbeds run over production networks. It is difficult to evaluate routing mechanisms, because we use the real network routing environment. This imposes limit to evaluate tests of low level protocols (layers 2 and 3). The implementation of the Click routing engine in testbed performs poorly [9].

Our proposal is to offer an integrated tool to evaluate Internet protocols and services joining simulation and emulation in a testbed. Figure 2 shows a diagram of the proposed solution. We can visualize the two prototype development steps, the first running in a simulation and the second running in a testbed. In the first step, the designer can debug the code in different environments. By the end of this step the code has been debugged and probably, it does not have serious flaws. As we run the experiment in simulation, we can test in a vast range of topologies and network scenarios in order to validate the proposal and explore the parameter space. In the second step, we need to test the code on a testbed, closer to reality. At this point we can make a

 ISBN: 978-1-61208-183-0

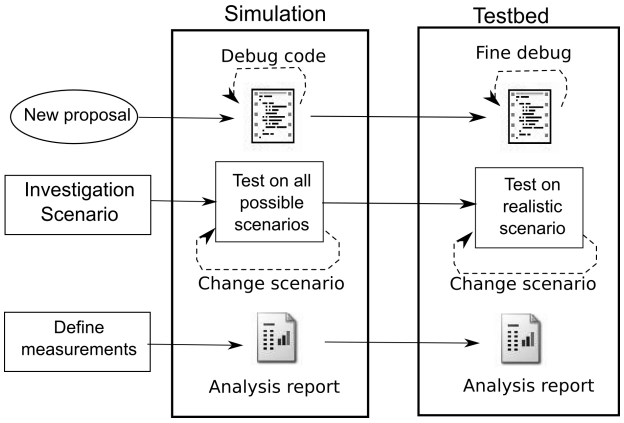

Figure 2. Proposed methodology

fine debug and fine parameters tuning in an (almost) real environment. As we saw, the testbed has less resources to capture information, but now we should be satisfied with some statistical traffic information.

A. Teagle Framework

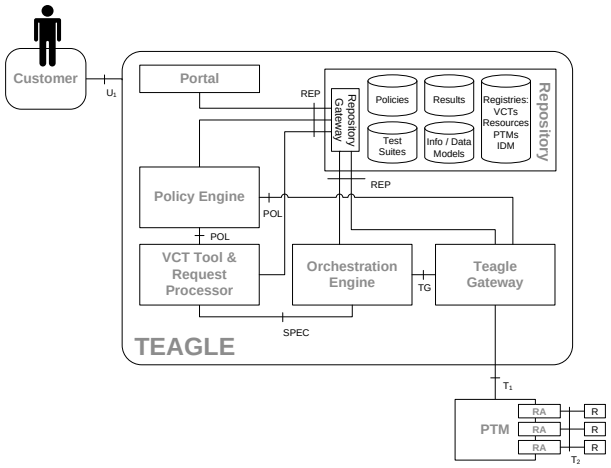

Figure 3. Teagle architecture

Reference [12] defines a federation model and framework that allows users to get access to distributed resources and group them into a virtual environment, which is called Virtual Customer Testbed (VCT). Teagle as a control framework and collection of central federation services helps the user in configuring and reserving a desired set of resources. Resources are offered by participating organizations across Europe.

On the federation layer, our Teagle framework implementation offers several services to the user and other framework entities, such as the registry and a common information model. Teagle allows browsing through the federation offerings, enables the definition of VCTs, and executes their provisioning. Figure 3 shows the Teagle architecture that is detailed in [12].

B. Teagle VCT Tool

The Teagle VCT offers a graphical tool (GUI) that permits the user describe the network topology and parameters to simulation and testbed evaluation. Before starting the experiment description, the user needs to define the kind of experiment: simulation or testbed. Then, Teagle framework will create a simulation script or a testbed setup file. The user can design the test topology using a set of graphical objects interconnected by arrows. Three different components to design an experiment were defined in Teagle VCT: Resource, Connector, and Monitor.

1) Resource: Resource represents a functional unit in a network system. A Resource could be a hardware device, like a Node or Link, or a software entity, like a protocol or a traffic generator. A Resource has Attributes and Events.

- *Attribute* is the configuration parameter of a Resource, which can be defined before the experiment and can be changed while the experiment is running. An example of attribute is the node IP address or the link bandwidth. In some Resource is possible to define the experiment planning, i.e., the sequence of values in experiments that should be performed, e.g., packet length of 100, 500 and 1000 bytes.
- *Events* is the set of timed events that will happen during the experiment. The Event time is based on virtual time independent from the real time. An example of Event is the start and stop time to transmit a traffic or to interrupt a link.

The Figure 4 shows the *Resource Application* configuration and the *Event* definition of a specific Application. The *Application Client* box has a ***cfg*** button that permits the configuration of its attributes like ClientType (Sink or Echo), Packet Length, Data Rate, and Port. The ***cfg*** windows also defines the *Experiment Planning*, i.e., the definition of various experiments will be executed. In this example, we use five different Random Seeds, five different Data Rates, from 10 pkt/s to 50 pkt/s, and three different packet lengths, 500, 1000, and 1500 bytes. It is important to notice that the total quantity of experiments, simulation or testbed, will be the combination of all different attributes, in our example $5 * 5 * 3 = 75$ experiments. The ***rules*** button defines the Events that will happen in this Application, e.g., the data stream starts at 2.0 sec and stops at 14.0 sec.

2) Connector: Connector is a representation of an interconnection from a Resource to another Resource. For example, the Resource Node is connected to Resource Link to build the topology. It is important to notice that the Connector is an abstract component only to permit joining different Resources, i.e., a Link is a Resource not a Connector although we use a Link to connect routers.

 ISBN: 978-1-61208-183-0

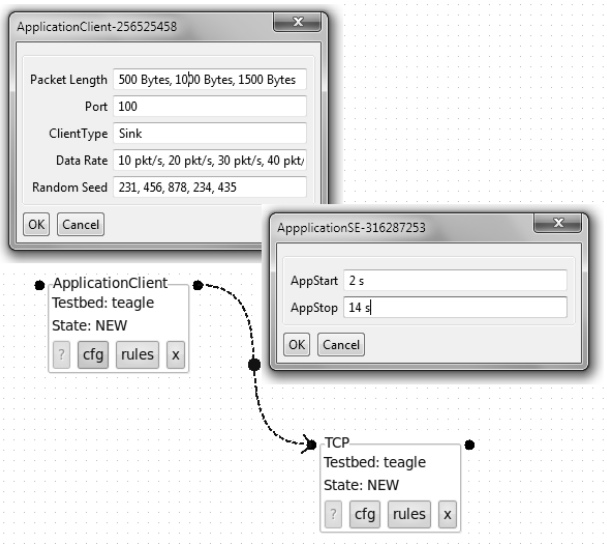

Figure 4. Teagle VCT Application configuration

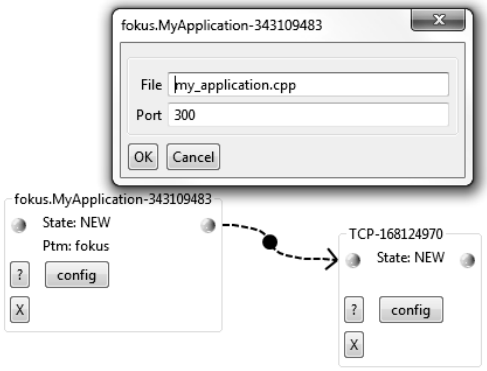

Figure 5. Creating a new application on Teagle VCT

3) Monitor: Monitor is a special Resource used to monitor the experiment and collect information about the experiment. However, Monitor does not interfere in network experiment. We consider using a *pcap* collector that collects useful information at simulation environment and also at testbed environment.

C. Creating a new protocol or service

The key feature of Teagle is the development and test of new protocols and services. For that, you can create a new module that will implement the desired protocol or service. Suppose that the user will design an application level module.

Figure 5 shows the definition of the *MyApplication* module that uses TCP protocol. The configuration sets the protocol port and the filename that contains the C code of the test protocol. The interface with Teagle is based on the Unix Socket, so the protocol implementation should be very similar to actual interface. To facilitate the development, a code template with the interface definition and the suggestion of the most common methods is provided.

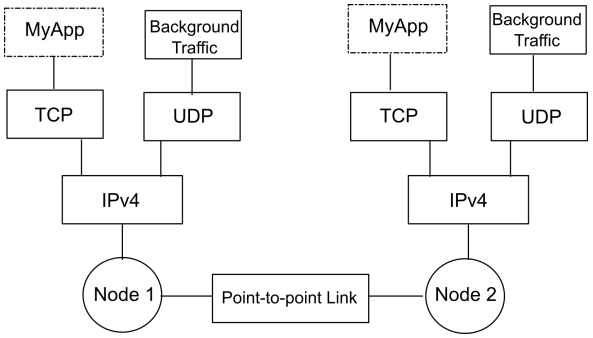

Figure 6. Test topology to validate the proposal

D. Mapping Teagle to ns-3

The Network Simulator version 3 – *ns-3* – was chosen as simulation framework. This version uses a network interface similar to Unix Socket, making easy the Teagle VCT translation to simulation script and Testbed configuration specification. However, the use of ns-3 as a standard tool to Teagle does not invalidate the creation of simulation models in other simulators, if it provides an abstract interface based on Sockets. The Teagle components are similar to ns-3 modules, then the translation is almost direct.

E. Mapping Teagle to Testbed

The Teagle platform aims to coordinate the execution of experiments in a Testbed federation. So naturally, an experiment specification in Teagle can be converted directly to Testbed configuration. However, an application performance validation tests requires more functionality than the standard Testbed platform can offer. Teagle offers the opportunity to create new functionality in a testbed in order to expand the scope of testing to be performed.

F. Analysis of results

Carrying out experiments on two methodologies and tools from a unique specification is not difficult because the components used are similar (nodes, protocols, applications). The great problem is the analysis of the results produced in different environments. However, the ns-3 simulator can produce a file in *pcap* format. In the testbed, it is possible to capture traffic in *pcap* format using tools like *tcpdump*. Comparing both *pcap* files is possible to analyze the results and the conclusions.

V. Proposal Evaluation and Results

Aiming to validate the proposed model, we created a simple test scenario that would allow to run a prototype testing. This experiment does not intend to validate a protocol or service, but it only intends to demonstrate the proposed methodology validation. The testing scenario is

 ISBN: 978-1-61208-183-0

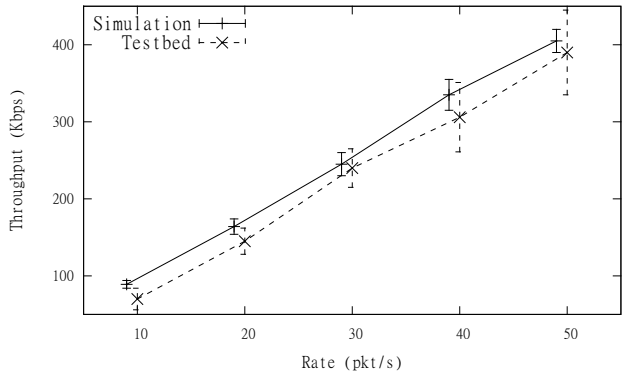

Figure 7. Bandwidth test result

shown in Figure 6, demonstrating a simple application with a background traffic. We have two nodes connected with a link and two applications: one is the proposed protocol running over TCP and a background application over UDP protocol. We run 10 experiments on both environment, simulation and testbed, and confidence interval is calculated.

The result graph is shown in Figure 7. The simulation environment is more controlled than the testbed environment and results tend to be different even with the same parameters. However, although we might have expected that the testbed results are not identical to the simulation, the results are very similar, mostly inside the confidence interval.

VI. Conclusion and Future Works

This paper presented a new methodology for developing and testing protocols and services using simulation and testbed. A single interface for the end user is the Teagle tool, Teagle Simulation and Emulation, enabling it to perform a test in simulation and emulation from the same specification in Teagle-VCT GUI. In both experiments, the technique chosen to collect and evaluate the results of the protocol under test performance and operation was the *pcap* files. They were generated by ns-3 and collected in the testbed using tcpdump software. To demonstrate the feasibility, a prototype model was developed using the Network Simulator ns-3 and the PANLAB testbed. The results were analyzed by comparing the *pcap* files generated in both experiments, which demonstrated the feasibility of the proposed model.

As future work, we wish to improve the collection of information in *pcap* files, which produces large files that require much processing capacity to analyze. One possibility is defining a filter to choose the specific information before the test that we want to collect in Teagle-VCT. It will reduce the amount of information stored. The Teagle-VCT specification translation considered only basic objects, so it becomes necessary to increase the amount of new objects to allow more functionality to the researcher.

References

[1] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. Wiley New York, 1991.

[2] 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*. Boston, MA: USENIX Association, Dec. 2002, pp. 255–270.

[3] T. Henderson, S. Roy, S. Floyd, and G. Riley, "ns-3 project goals," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*. ACM, 2006, p. 13.

[4] R. Ricci, C. Alfeld, and J. Lepreau, "A solver for the network testbed mapping problem," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, p. 81, 2003.

[5] K. Webb, M. Hibler, R. Ricci, A. Clements, and J. Lepreau, "Implementing the Emulab-PlanetLab portal: Experience and lessons learned," in *Proc. WORLDS*, 2004.

[6] M. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau, "Large-scale virtualization in the emulab network testbed," in *USENIX Annual Technical Conference, Boston, MA*, 2008.

[7] S. Guruprasad, R. Ricci, and J. Lepreau, "Integrated network experimentation using simulation and emulation," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2005. Tridentcom 2005. First International Conference on*, 2005, pp. 204–212.

[8] R. Ricci, J. Duerig, P. Sanaga, D. Gebhardt, M. Hibler, K. Atkinson, J. Zhang, S. Kasera, and J. Lepreau, "The Flexlab approach to realistic evaluation of networked systems," in *Proc. of the Fourth Symposium on Networked Systems Design and Implementation (NSDI 2007)*, Cambridge, MA, 2007.

[9] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: realistic and controlled network experimentation," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2006, p. 14.

[10] M. Lacage, M. Ferrari, M. Hansen, T. Turletti, and W. Dabbous, "Nepi: using independent simulators, emulators, and testbeds for easy experimentation," *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 4, pp. 60–65, 2010.

[11] P. Sanaga, J. Duerig, R. Ricci, and J. Lepreau, "Modeling and emulation of Internet paths," in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*. USENIX Association, 2009, pp. 199–212.

[12] S. Wahle, B. Harjoc, K. Campowsky, and T. Magedanz, "Pan-European testbed and experimental facility federation–architecture refinement and implementation," *International Journal of Communication Networks and Distributed Systems*, vol. 5, no. 1, pp. 67–87, 2010.

 ISBN: 978-1-61208-183-0