Reproducing Network Conditions for Tests of Large-Scale Distributed Systems

Robert Lübke, Daniel Schuster, Alexander Schill
Computer Networks Group
Technische Universität Dresden
Dresden, Germany
{robert.luebke, daniel.schuster, alexander.schill}@tu-dresden.de

***Abstract*—Testing distributed systems is difficult, error-prone and time-consuming. Especially scalability tests are hard to perform because of the large number of connected devices. The heterogeneity of these devices (PCs, smartphones, tablets) and their various network access technologies (DSL, cable, WiFi, 3G/4G) even worsen the situation. We argue that the reproduction of network conditions is necessary to provide an adequate testbed for experiments with distributed systems. We therefore propose an architecture that allows performing automated tests of large scale distributed systems including the emulation of network and application behavior.**

I. Introduction

Network applications obviously show a behavior that depends on the used network technology as these technologies mostly have entirely different characteristics. Testing distributed systems is often done in laboratory environments with almost perfect network conditions. In real scenarios the distributed systems' components are situated in data centers all over the globe and have to communicate with each other overcoming thousands of kilometers and a significant number of network hops. Therefore, we argue that the network-specific characteristics have to be integrated with tests of distributed systems to reflect the real world conditions accurately enough.

The reproduction of network conditions is relevant for every application communicating over a network, but it is essential for large-scale distributed systems, for example Cloud Computing solutions. Running automated tests regularly makes it easier to achieve the general aim of fault-tolerant and reliable network computing.

Experiments with large distributed systems and reproduction of network behavior can be performed in different ways. Simulation is a synthetic approach in which the whole networked system is mapped to a model that allows various calculations. Furthermore, hardware network emulators connecting all nodes of the system can also be used for this purpose. Another approach is a distributed emulation environment, a federation of multiple test networks of various institutions that consist of thousands of nodes all over the world. Examples of these three approaches will be discussed in detail in section II.

We found limitations in all three approaches, when performing experiments with a concrete test system: a conferencing system (audio / video). Testing it requires an advanced and accurate reproduction of network conditions like in any other large real-time and multimedia application. This includes network parameters like the limitation of the available data rate, packet delay and its variation (also called jitter), packet loss, duplication and reordering. These parameters must be assignable separately to up and down link of a connection and dynamic changes during test execution are required to support varying conditions. Regarding mobile users who are connected via telecommunication technologies (3G, 4G, WiFi) advanced effects like disconnection phases and handovers between cells as well as technologies also have to be reproduced. Although all approaches support basic network parameters, only simulation can achieve the described advanced reproduction of network behavior.

A special requirement concerning scalability tests is the large number of software instances that have to be coordinated. To achieve the large number of nodes in scalability tests with a maintainable hardware effort, multiple software instances have to be executed on one machine, but their network characteristics must be emulated separately. Emulation environments can easily handle large scenarios, but simulation- and hardware-based approaches are limited.

As simulation requires the application behavior to be part of the simulation model, one cannot use the original software under test. With Hardware-based solutions and emulation environments on can handle the accurate reproduction of application behavior.

We have shown that none of the presented approaches matches all mentioned requirements. Therefore we propose another approach: integration of network emulation and experiments with distributed systems. With network emulation a slow and unreliable network is imitated on a real physical network with better characteristics. Real network traffic is exchanged between the nodes of the system, but it is shaped according to certain characteristics. These are the main research questions to be tackled with our work:

- Is it possible to use network emulation for **large-scale tests** of distributed systems with several thousand nodes covering **all necessary effects and characteristics**?
- Is it possible to emulate **application behavior** without abstraction from the original software under test when performing large-scale tests of distributed systems?
- How can one emulate the **heterogeneity** of endpoint systems, which can be found especially in mobile scenarios, when performing large-scale tests of distributed systems?

II. Related Work

Network emulation is quite an old research area and a lot of work has been done in the last two decades. This section gives a brief overview on the state of the art of network emulation and investigates its applicability for experiments with distributed systems.

Examples for the already mentioned distributed emulation environments are OneLab [11], PlanetLab [7] and G-Lab [10]. Network emulation support has been added to many emulation environments, for example in [8] for G-Lab. These environments fulfill most of the requirements, but are not flexible enough to perform large scale tests, for instance they do not support the requirement of running multiple software instances on the machines while emulating the network conditions differently. Most environments neither support complex topologies nor dynamic changes during run time. Furthermore, one can often use them only after adding nodes to the infrastructure.

There are many hardware based products, that are said to perform the emulation precisely. We found no product that was able to emulate complex network topologies and change the connection parameters dynamically. Due to these limitations of the hardware solutions, we focused our search on freely available software emulators, that are presented in the following.

NISTNet [2] is able to emulate most of the required parameters, but it cannot emulate network topologies. Another widespread tool is Netem/TC [9] that acts as traffic shaper in most common Linux distributions. It can also emulate networks as it provides means to modify bandwidth, delay, loss and other effects based on various connection parameters like IP address or port number. The wide area network emulator WANem [5] is based on Netem. It allows analyzing and measuring real network links and emulating disconnects. However, Netem and WANem are not able to represent whole network topologies. In FreeBSD Dummynet [1] is the standard traffic shaping component. Its basic concept called pipes is pretty handy to create virtual network topologies. Although Dummynet itself only emulates bandwidth, loss and delay, it has many extensions. KauNet [3] adds support of other parameters and it introduces a pattern-based approach. It further enhances precision and reproducibility of Dummynet by using time- or data-driven patterns [4]. ModelNet [12] in contrast to KauNet uses an unmodified version of Dummynet to emulate huge topologies on a predefined number of computers.

Comparison papers like [6] showed the reasonable emulation quality of Dummynet, NISTNet and Netem/TC, although some emulators are still facing problems in certain scenarios. Dummynet for example still has problems with the accurate emulation of high data rates.

Some of the presented solutions are able to emulate network topologies and link characteristics as we require it for large-scale testing of distributed systems. But these solutions focus on network development experiments and therefore do not provide any means to integrate the emulation of the network behavior with the emulation of the application behavior. To our knowledge there is no system combining the emulation of network and application behavior in an integrated way to facilitate experiments with large distributed systems.

From all presented network emulators, the combination of Dummynet and KauNet is the best solution to emulate network topologies of any size and complexity. We therefore use both as a basis for our integrated emulation platform that is presented in the following section.

III. Proposed approach: the NESSEE platform

This section covers the architecture of a platform that can perform large scalability tests of distributed systems using network emulation: NESSEE (Network Endpoint Server Scenario Emulation Environment). The general architecture and the most important components are shown in Figure 1.

As NESSEE emulates client/server-based systems, there are **test systems** dedicated to client- and server-side **software under test** (SUT). Each test system can run multiple instances of the SUT. Mobile devices like smartphones and tablets act as test systems for mobile software. The **TestNodeModule** (TNM) is a platform independent NESSEE component that also runs on each test system. It starts the SUT instances and uses the available IPC (inter process communication) mechanisms of the operating systems to control its execution. This approach allows testing the original software without further adaption than providing the IPC interface. Applications requiring special hardware or other resources can run on dedicated SUT machines. The TNM communicates with these machines via SSH (Secure Shell).

The central **NESSEE Server** acts as a coordinator for all other components. It manages all available test systems, controls the TNMs according to the executed test scenario and collects log files of all involved entities. After a test run these logs are saved in the Central Log Repository. Every time a test should be executed the test scenario has to be matched with the available resources to provide an adequate test environment. The NESSEE Server also provides a multi-tenant web front end that allows the users to manage test systems and test scenarios as well as run, control and evaluate tests.

The different test scenarios are imported to the NESSEE Server as files written in the **Test Description Language** (TDL). The TDL is a generic XML-based and modular language to describe test cases that can be executed with NESSEE. The **NESSEE Editor** is a graphical authoring tool for TDL files that facilitates creation and editing of test cases. Using this description language the testers can define the number of utilized SUT instances, their behavior and the network characteristics and topologies that should be emulated. A TDL file can also be enhanced with test-specific script files (written in JavaScript) that are later executed by the NESSEE Server's or TNMs' script engines. This script-based approach is used to flexibly emulate the application behavior.

Another important component is the **Degrader** that is responsible for the network emulation during the test. It is configured to be the standard gateway of all test systems

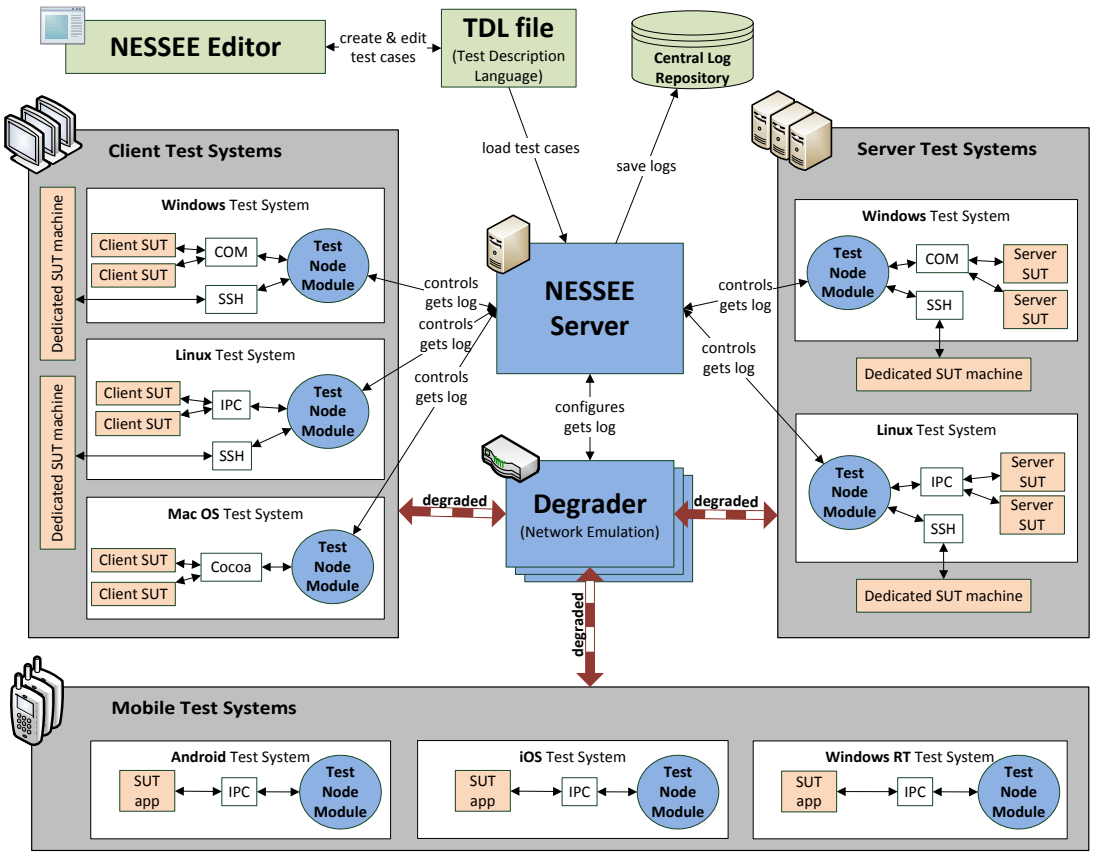

Fig. 1. General architecture of the NESSEE platform

and therefore acts as a router. To prevent scalability problems in the network emulation, multiple Degraders are supported. Before test execution the Degrader is configured by the NESSEE Server according to the loaded scenario. Filtered by Dummynet rules, which is a mechanism similar to firewall rules, the incoming traffic has to pass various Dummynet pipes dependent on the source IP addresses and ports. These pipes can be attached with certain network parameters like data rate, loss and delay, but also with the more advanced KauNet patterns. This pattern-based approach allows very accurate and reproducible emulation of network characteristics. Using these mechanisms the NESSEE Degrader supports the emulation of network topologies of any complexity, data rate limitations, packet delay and its variation, loss, reordering and duplication. Mobile clients connected with telecommunication networks can also be emulated with disconnections and handovers. With the help of traffic generators it is also possible to emulate background traffic of any kind.

IV. Preliminary Results

The evaluation of accuracy and performance of the network emulation component is done with the help of multiple tools, for example `iperf`[1] for data rate, loss and reordering as well as `ping`[2] for duplication, delay and its variation (jitter).

[1]http://sourceforge.net/projects/iperf/
[2]http://linux.die.net/man/8/ping

As the accuracy of the underlying Dummynet and KauNet have been evaluated before [6], we focus on the accuracy of our Degrader emulating complex network topologies. We therefore run scenarios in which only one parameter is configured in a chain of three network nodes. Furthermore, we run more complex scenarios as illustrated in Figure 2. The deviation of expected and measured network parameters is determined and averaged over multiple test runs with different configured values. The resulting deviations are:

- data rate: $\pm 3.50\%$
- packet delay: $\pm 0.20\%$
- packet loss: $\pm 0.10\%$
- packet duplication: $\pm 0.00\%$
- packet reordering: $\pm 0.13\%$

Except for the data rate emulation, which still has to be improved, all parameters are emulated accurately. Effects like bandwidth sharing and the variation of delay (jitter) according to a Laplace distribution can also be observed.

Although the Degrader supports multiple test scenarios by multiple testers at the same time, the scalability is limited by the capacity of the Degrader's network interfaces. Therefore, the platform has to be enhanced with multiple cooperating Degraders.

The used measurement tools have to be configured and executed separately to determine the various network parameters. To facilitate the evaluation of our approach and the comparison

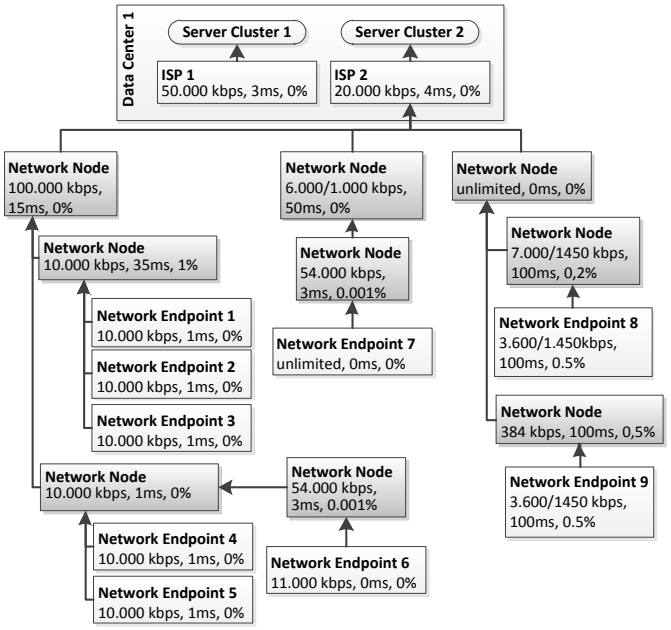

Fig. 2. One of the complex test scenarios used for the evaluation

with other network emulators we are developing an integrated measurement tool that is able to determine all relevant network parameters in one measurement. With this tool at hand we will perform an extensive comparison of the existing basic network emulators' accuracy and performance. The most challenging issue we will have to tackle is the required high-precision measurement, i.e. a time accuracy not exceeding about 50 micro seconds. The last research paper comparing multiple systems [6] is from 2009. As new versions of the emulators were published and the hardware was improved too, we are expecting very different results, which will also be an important contribution to the research community. The integrated measurement tool will also allow us to determine practice oriented values for the various parameters of different network access technologies, for example DSL, Cable and mobile networks (WiFi, 3G, 4G). This facilitates the configuration of network emulators when creating typical test scenarios.

The scalability of NESSEE's architecture itself was proofed with test executions running up to 1000 SUT instances on Windows test systems. The TestNodeModule is currently a Windows application and still has to be implemented for the other main platforms. We expect the encapsulation of OS-specific mechanisms in a platform-independent TNM to be very challenging. At the end the mobile test systems have to be integrated and the Degrader has to be enhanced to accurately emulate the mobile communication technologies as well.

In conclusion the remaining objectives of our work are:

- Development of a platform-independent TestNodeModule
- Integration of mobile test systems
- Improve the scalability of the network emulation with multiple cooperating degraders, allowing the execution of even larger scenarios
- Development of an integrated measurement tool

V. Conclusion

The main contribution of this work is an approach that integrates tests of large-scale distributed systems with the emulation of multi-level network topologies, fine-grained network parameters and complex network behavior. It facilitates performing experiments with distributed systems under real-world network conditions. To realize our approach we proposed the NESSEE architecture and the Degrader as a concept for performing the network emulation. The evaluation showed the good accuracy of the current implementation and the feasibility of emulating complex scenarios. However, scalability has to be improved for even larger scenarios and platform independence must be achieved.

We further plan to contribute an extensive comparison of current network emulators' accuracy and performance once our integrated measurement tool is implemented. The resulting parameters of typical network connections will also be valuable data sets for other researchers.

References

[1] M. Carbone and L. Rizzo, "Dummynet Revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 12–20, April 2010. [Online]. Available: http://doi.acm.org/10.1145/1764873.1764876

[2] M. Carson and D. Santay, "NIST Net: a Linux-based network emulation tool," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 111–126, July 2003. [Online]. Available: http://doi.acm.org/10.1145/956993.957007

[3] J. Garcia, E. Conchon, T. Pérennou, and A. Brunstrom, "KauNet: Improving Reproducibility for Wireless and Mobile Research," in *Proceedings of the 1st international workshop on System evaluation for mobile platforms*, ser. MobiEval '07. New York, NY, USA: ACM, 2007, pp. 21–26. [Online]. Available: http://doi.acm.org/10.1145/1247721.1247726

[4] J. Garcia, P. Hurtig, and A. Brunstrom, "KauNet: A Versatile and Flexible Emulation System," in *Proceedings of the 5th Swedish National Computer Networking Workshop (SNCNW 2008)*, Karlskrona, Sweden, April 2008.

[5] H. Kalita and M. Nambiar, "Designing WANem : A Wide Area Network emulator tool," in *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, January 2011, pp. 1 –4.

[6] L. Nussbaum and O. Richard, "A Comparative Study of Network Link Emulators," in *Proceedings of the 2009 Spring Simulation Multiconference*, ser. SpringSim '09. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 85:1–85:8. [Online]. Available: http://portal.acm.org/citation.cfm?id=1639809.1639898

[7] Princeton University, "Planetlab. an open platform for developing, deploying, and accessing planetary-scale services," http://www.planet-lab.org, 2013, accessed: 01/15/2013.

[8] D. Schwerdel, D. Hock, D. Günther, B. Reuther, P. Müller, and P. Tran-Gia, "ToMaTo - a network experimentation tool," in *7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2011)*, Shanghai, China, Apr. 2011.

[9] Stephen Hemminger, "Network Emulation with NetEm," in *linux.conf.au*, Canberra, Australia, April 2005. [Online]. Available: http://devresources.linux-foundation.org/shemminger/LCA2005_netem.pdf

[10] University of Würzburg, "G-lab," http://www.german-lab.de/, 2013, accessed: 01/15/2013.

[11] UPMC Paris Universitas, "Onelab. future internet testbeds," http://www.onelab.eu, 2013, accessed: 01/15/2013.

[12] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker, "Scalability and Accuracy in a Large-Scale Network Emulator," in *Proceedings of the 5th symposium on Operating systems design and implementation*, ser. OSDI '02. New York, NY, USA: ACM, 2002, pp. 271–284. [Online]. Available: http://doi.acm.org/10.1145/1060289.1060315