DiSSOmniaG Distributed Simulation Service with OMNeT++ and Git

Sebastian Wallat, Martin Becke
University of Duisburg-Essen
Institute for Experimental Mathematics
Ellernstrasse 29, 45326 Essen, Germany
sebastian.wallat@uni-due.de, martin.becke@iem.uni-due.de

I. Introduction

Within the area of protocol design, multiple methods exist in order to make it possible to test and evaluate the behavior and the deployment of a protocol.

One of the important methods, especially in early stages of the protocol design, is the use of simulations. It makes it possible to test protocol ideas in a sandboxed manner. In addition to it, with simulations, it is possible to perform highly scalable experiments. In this context, some discrete event simulation tools such as OMNeT++ [1] and NS-2 [2] has become more and more important for the developer of protocol stacks.

However simulation is just one part. Before deploying a protocol, it is inevitable to test the specification within real life environments. Here it could be useful to implement the specification for example in the user land or in the kernel depending on the performance requirements. The test and evaluation of the implementation have to be performed in a controllable and reproducible environment. For this purpose, multiple solutions exist such as Planetlab [3] or G-Lab [4], with ToMaTo [5] and provide tools which allow network researchers to setup of even complex network experiments, distributed over several dedicated physical node systems. In general they are based on a central configuration unit which controls physical host machines. This virtual approach is used to provide a working environment which allows a homogenous and scalable network user structure. However, both Planetlab and G-Lab show severe drawbacks. Planetlab for example uses the internet as a hardware base and this involves a certain unpredictability of the network conditions and makes it difficult to reproduce experiments. In contradiction to Planetlab, with G-Lab, experiments are performed in an isolated network setup which makes it easier to eliminate interferences. However, with G-Lab the scalability is limited to the hardware used.

Based on the experience made with the existing systems, we present in this demo a new tool we call *Distributed Simulation Service with OMNeT++ and Git (DiSSOmniaG)*, which combines the use of a full controllable test bed with a simulation. This allows the deployment of high scalable and reproducible experiments for current and as well as future protocol development.

II. Concept

DiSSOmniaG adopts several mechanisms used in Planetlab and G-Lab ToMaTo and integrates in addition to it simulations within the testbed system. It provides beside a topology based configuration system the ability to distribute applications, we denote as *Apps*, among the virtualized DiSSOmniaG node systems. These so called Apps can be fully remote controlled by the researcher and allow a node specific configuration for a testbed scenario. To provide a fully configurable App environment, DiSSOmniaG relies on a Live CD [6] system, which can be configured by the researcher to fit to the scenario needs. Here customized kernel images can be used within the DiSSOmniaG system. One of the most usefull usecases of the App concept is the possibility to use simulations within real life testbeds by using OMNeT++ and INET [7]. In addition to it, OMNeT++ gives also the possibility to communicate with physical networks; here packages can be captured on a network interface and injected into the simulation environment and vice versa. This leads to some interesting aspects. The link emulation concept, which is used within G-Lab ToMaTo and Planetlab, can be replaced. It also allows a kind of abstraction level. In fact, with the use of this App, it becomes possible to make experiments with early implementations — e.g. a simulative model — of a protocol specification within a real life environment.

The main perspectives of DiSSOmniaG are described in the following:

- The topology based configuration improves the reproducibility of the considered scenarios. A scenario can be reconfigured by only knowing the topology.
- The controlled environment is achieved with the use of live CDs and the possibility to select which connection should be used as a virtual or an internet connection with uncontrolled cross traffic.
- Often the scalability of a scenario is a big challenge. Especially limited resources lead to restricted scenarios. The use of simulation for e.g. router networks provides a partial improvement to these restrictions.

A. DiSSOmniaG components

The DiSSOmniaG system consists of several components which will be described in the following section.

- *The DiSSOmniaG-backend* is the main controller of the DiSSOmniaG system. It manages the host resources and creates the corresponding Live CDs and their configurations. It also deploys and controls the testbed application on the DiSSOmniaG-node systems.

- *The DiSSOmniaG-hosts* are the physical machines where DiSSOmniaG-nodes run on.
- *The DiSSOmniaG-nodes* are the virtual machines which are booted with a customized Live CD.
- *The DiSSOmniaG-gui* acts as the configuration interface for the researcher. Here hosts can be configured, topologies can be created and deployed and applications can be remote controlled.
- *The DiSSOmniaG Live CD pattern creator* is a tool which is used to create customized Live CD images. Although a pre-configured image is already provided, this tool can be used to apply custom modifications.
- *The DiSSOmniaG App management system* is a git repository where all Apps are stored. When a modification from a user is pushed to the repository the App management system informs the DiSSOmniaG-backend to update the applications on the node systems.

III. Demonstration

In order to show the main functionalities of DiSSOmniaG, we will demonstrate the recommended working process. First of all we will introduce the DiSSOmniaG-gui system and the maintenance abilities of DiSSOmniaG. After setting up the connection to the backend system and the initial user setup, we will show how to upload the user SSH key, which is needed to access the nodes and the app system.

In order to create a testbed topology, the first step is to create a Live CD pattern with a customized set of pre-installed tools. In this context the inclusion of the OMNeT++ simulation system and the INET simulation model is of interest.

A. Live CD pattern creation

At the beginning of the testbed setup, a researcher may want to setup a customized Live CD pattern. The pattern is a compressed archive with all files needed from the backend system to build a node Live CD with all node specific configuration details. Here we will show how to configure a Live CD preinstalled with OMNeT++ and INET. Additionally we will then install some typical measurement tools like Netperfmeter [8].

B. DiSSOmniaG-gui and user management

The DiSSOmniaG-gui is used in order to get in contact with the backend. After configuring the connection to the backend system, we will upload a SSH public key, which is used to log in to the node systems via SSH or to clone the git repositories from the App management system.

C. Topology creation and deployment

In this step we will configure a sample testbed setup. According to the G-Lab ToMaTo terminology, we call such a setup a topology. Here we will configure two node systems, acting as endpoints of our simulation scenario setup. This simulation setup will be configured in the next step as an App. Finally the topology will be deployed. This will trigger the backend system to create the live CDs and to start the nodes with it.

D. Creating a simulation App

After adding an App to the DiSSOmniaG system, a checkout for the App repository is performed and the simulation setups are configured. The needed start and stop scripts to remote control the App are introduced in addition to the environment parameters used to run an OMNeT++ setup are configured. After making final modifications, the App is pushed to the App management system.

E. Remote control the testbed setup

At this point we log in to the node systems via VNC. We show how to start and stop the App via the DiSSOmniaG-gui.

F. Gathering results

The final key feature of DiSSOmniaG is the automatic result gathering functionality. After stopping a simulation, the result files are committed to the git repository and pushed to the App management system. Here the researcher can checkout the results, and analyze them from the desktop workstation.

IV. Conclusion

After an overview of existing testbed management systems, we introduced the basic principles of DiSSOmniaG. Here the application management and the use of simulations within real life testbeds are two major aspects. In a second step, we introduced the workflow of DiSSOmniaG and explained the key features by an example.

References

[1] A. Varga, "OMNeT++ Discrete Event Simulation System," 2011. [Online]. Available: http://www.omnetpp.org

[2] NS-2, "The Network Simulator NS-2," 2011. [Online]. Available: http://www.isi.edu/nsnam/ns/

[3] L. Peterson and T. Roscoe, "The Design Principles of PlanetLab," *Operating Systems Review*, vol. 40, no. 1, pp. 11–16, Jan. 2006, ISSN 0163-5980. [Online]. Available: http://www.planet-lab.org/files/pdn/PDN-04-021/pdn-04-021.pdf

[4] P. Tran-Gia, A. Feldmann, R. Steinmetz, J. Eberspcher, M. Zitterbart, P. Mller, and H. Schotten, *G-Lab White Paper Phase 1 – Studien und Experimentalplattform fr das Internet der Zukunft*, Jan. 2009. [Online]. Available: https://www.german-lab.de/fileadmin/Press/G-Lab_White_Paper_Phase1.pdf

[5] D. Schwerdel and D. Hock, "ToMaTo-a network experimentation tool," 2011. [Online]. Available: http://dspace.icsy.de:12000/dspace/handle/123456789/324http://dspace.icsy.de:12000/dspace/handle/123456789/309

[6] C. Boehme, T. Ehlers, J. Engelhardt, A. Felix, O. Haan, T. Kalman, B. Neumair, U. Schwardmann, and D. Sommerfeld, "Instant-grid: Fully automated middleware-deployment using a live-cd," in *Networking and Services, 2006. ICNS '06. International conference on*, july 2006, p. 70.

[7] M. Tüxen, I. Rüngeler, and E. P. Rathgeb, "Interface Connecting the INET Simulation Framework with the Real World," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools)*, Marseille/France, Mar. 2008, pp. 1–6, ISBN 978-963-9799-20-2. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1416222.1416267

[8] T. Dreibholz, *NetPerfMeter Homepage*, 2011. [Online]. Available: http://www.iem.uni-due.de/~dreibh/netperfmeter/

[9] M. Becke, T. Dreibholz, and E. Rathgeb, "Link Emulation on the Data Link Layer in a Linux-based Future Internet Testbed Environment," *ICN 2011, The Tenth*, 2011. [Online]. Available: http://www.tdr.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/ICN2011.pdf