1. INTRODUCTION

A common method for analyzing and developing new network topologies and applications is to use simulation. With the help of simulators, the development of new communication architectures and network protocols can be considerably simplified and improved. However, many different software simulators exist, each of which supports its own particular features and differs in computation performance, tuning accuracy and hardware requirements. In cases where simulation of networks is not sufficient, the possibility to emulate network infrastructure is also needed. Tools which support both methods, simulation and emulation, are classified as hybrid virtualization (HV) tools. This allows network simulators to extend real network testbeds.

A major argument why computer network analysts and developers are opting for HV-tools is that complex and long-term experiments on real hardware are costly, expensive and greedy for resources. Instead of buying expensive and large hardware components which are only used temporarily for testing purposes, the use of an HV-tool is cheaper and also environmentally friendly. Furthermore, altering the parameters or topology of an experiment can be done virtually by software and does not need any time-consuming and error prone cable patching or similar physical changes.

This paper is structured as follows: section 2 motivates and explains the aim of the presented experimental work. Section 3 gives a short overview of related work. Section 4 describes the features, advantages and disadvantages of the tools that are being considered. In section 5, the comparison criteria used and the experimental setup are defined. Furthermore the results of the measurements are graphically presented and evaluated. Finally, the paper concludes with a summary and some considerations for future work.

2. MOTIVATION

Consider a scenario where a network architect intends to build up a hybrid virtual network environment. Sooner or later he will need to select a simulation program that fits his requirements and constraints the best. Here, we investigate which tool best satisfies the criteria, experimental requirements and constraints defined in the context of such a scenario.

Every HV-tool has a different quality of service which, in the context of this paper, is the quantitative difference between the desired value of the tunable parameters and the actual measured values. This paper focuses on the accuracy of three parameters: packet delay, data transfer rate and packet loss. With these three parameters, the most important and most common characteristics of a given network infrastructure can be expressed.

For emulation purposes a network device is essential as it offers the possibility to attach the simulated environment to a real end system. The end system itself cannot (and should not) be able to distinguish between the emulated network interface and a real one. Both devices are treated exactly the same. Thus, to behave like its real equivalent, the emulated device needs to imitate the required functionalities of the real device and to mimic physical network attributes like latency, jitter, packet loss, etc. In the next section, related work is briefly presented.

3. RELATED WORK

Landsiedel describes a model of protocol development where experiments are first conducted in network simulators and then applied to the real world [5]. While network simulators reproduce an entire network, they can also simulate only a portion of a network, thus creating a synthetic network environment.

Herscher proposed a Network Emulation Testbed which creates a synthetic network environment to create reproducible tests for wireless network experimentation [4]. Besides creating a consistent environment, the use of network simulators also enables creation of environments that are larger, more complicated, or more capable than a specific network testbed.
To work well, a network simulator must also perform well. Weingärtner has compared the performance of network simulators in terms of run-time performance and memory usage [8]. This comparison was restricted to simulation, and did not consider emulation.

This paper is concerned with how well the simulator can match real-world performance. To answer this question we connect a simulated network environment to a real-world testbed.

4. ANALYSED TOOLS

This section gives a detailed look on both HV-tools that are compared, namely ns-3 and OMNeT++. Additionally, the pros and the cons of each tool are presented in a short listing.

4.1. HV-tool ns-3

The ns-3 is a discrete event based network simulator which is used for scientific simulations in the field of network research [2]. The software is licensed under the terms of GNU GPLv2 and thus freely available as open source software. It is written in C++ and offers the option to bind Python scripts to the simulator framework. The first steps in the development of ns-3 were taken in August 2005.

The configuration of ns-3 is based on an hierarchical structure. Basic elements called Node, NetDevice, Queue, etc. can be arbitrarily combined and configured in any desirable context. Additionally, individual communication elements (e.g. Devices or Nodes) can be combined in groups, called containers (e.g. NodeContainer). Inside these containers, interfaces (e.g. NetDevice) with tunable parameters can be defined.

Within ns-3, simple and complex network simulations can be realized in different ways. With the aid of helper classes like InternetStackHelper, complex simulations can be easily realized. Furthermore, these classes can be used to simplify the usage of standardized methods by reducing configuration overhead.

Also, ns-3 supports application modules which can be integrated in simulated nodes. These application modules can be started or stopped at any time by scheduling an appropriate event. It includes powerful logging mechanisms while running a simulation by creating textual log files or logs to read with other tools (e.g. tcpdump) for any simulated node interface.

Additionally, the ns-3 simulation results can be graphically visualized by NAM, the Network Animator [3], for evaluation purposes. A further feature of ns-3 is the EmuNetDevice which permits connecting the simulation environment to real world networks and nodes. Thus data packets from the simulation can be sent to the real world and data packets from real networks can be received by the simulation for further handling.

ns-3 has the following major advantages:

- configurations are built in a modular way
- supports real-time scheduling
- provides an easy way to create complex configurations through helper classes
- outputs simulated and emulated network traffic in the tcpdump file format, readable by Wireshark

ns-3 has the following major disadvantages:

- configuration and development is only text-based
- the documentation of ns-3 is partially incomplete and difficult to handle
- root privileges are required when using the emulation feature, as it accesses system-level functions, which are not allowed to normal users

4.2. HV-tool OMNeT++

OMNeT++ is an open source discrete event simulation environment. It has a modular architecture and supports many operating systems, including Linux, Unix (derivates like OS X, FreeBSD) and Windows. It is written in C++ and a simulation can be created in both C++ and Java programming languages. Furthermore, a simulation can be configured by using a description language called NED (NEtwork Description). OMNeT++ separates simulation from configuration.

A simulation is based on message modules. The message modules are interconnected via gates, with gates being communication channels between the modules. More complex simulations can be created by using one of many available frameworks which provide preconfigured modules. Furthermore, complex networks and network topologies can be easily built up by combining the modules provided by an appropriate framework.

OMNeT++ uses its real-time-scheduler when interacting with the real world [7]. To set up a communication bridge connecting the simulation to a physical environment, an ExtInterface has to be configured. Possible use cases are traffic generators, simulated server farms or complex network topologies for any supported protocols.

OMNeT++ offers the following major advantages:

- support of real-time scheduling
- many frameworks available (for example INET Framework,...)
- excellent documentation
OMNeT++ has the following major disadvantages:

- not developed only for network simulations
- root privileges are required when using the emulation feature, as it accesses system-level functions, which are not allowed to normal users

5. EVALUATION

In the following section, criteria for the evaluation are defined. Afterwards, the experimental setup is presented and validated. The section ends with an evaluation of the measurement series.

5.1. Evaluation criteria

This section defines three objective criteria used to compare both HV-tools: packet delay, data transfer rate, and packet loss.

- Packet delay is the time that a packet needs to travel through the environment to the destination node and back. Note that in a simulated environment the delay can indeed be set to 0 ms, but, in a real network, delay cannot be zero.

- Data transfer rate is the average number of bits transferred per second (bps). To simulate a given data transfer rate, the simulation tool counts the bits that are being processed during a fixed time period.

- Packet loss means that a packet, that has been sent, is not received at its destination. In the experimental measurements of section 5, packet loss is expressed by a probability. For example, a packet loss probability of 10% means that every tenth packet is lost on average.

5.2. Experimental setup

The experimental setup for the test environment is presented in figure 1. We chose a simple simulated component to focus on the combination of simulated and real components.

A and B are two real server nodes which are interconnected through the simulation/emulation environment. This environment is running on a third server node which is providing the two network interfaces eth0 and eth1. Both interfaces are emulated endpoints of the environment. In figure 1 the simulated/emulated environment is depicted in the shaded box. Inside the virtual setup, packets are forwarded by three routers. These routers are configured with an IP address on each interface and an adequate routing table. The links between the routers are point to point connections. The experiments measure the characteristic parameters packet loss, data transfer rate, and packet delay which are arbitrarily configured on the simulated links.

Figure 1. Experimental setup

5.3. Experimental Measurements

In the next three subsections, the accuracy of the packet delay, data transfer rate, and packet loss settings are graphically represented and analyzed for both HV-tools ns-3 and OMNeT++.

5.3.1. Packet delay

For measuring delay, the commonly available Ping tool is used. Ping sends an ICMP packet to a remote network node and measures the elapsed time until the remote node replies. The round trip time is given in ms. As preconditions for measuring delay, the data transfer rate has to be set to the maximum possible value (the maximum value depends on the rate of the links, here 100 Mbit/s is used) and packet loss has to be set to zero.

For the measurement procedure, four different delay settings (regarding the logical data transfer link, see figure 1) are used: 50 ms, 100 ms, 200 ms, and 300 ms. For each experimental measurement, ICMP packets with a size of 100 bytes

Figure 2. Comparison of 50 ms packet delay
For each of the four settings, the measured values are represented by cumulative distribution functions (CDF). The CDFs of both HV-tools for a same setting are drawn in one diagram to easily compare and evaluate the accuracy of both tools (see figures 2, 3, 4, and 5). As the discretely measured values are not interpolated, the CDF is not a continuous graph but looks rather like a step function. The CDF representation is a method to visualize the cumulated probability that a measured value is found less than or equal to the set value. Furthermore, the probability that a measured value is located between two fixed bounds can be calculated.

It can be observed that for the four series of measurement ns-3 is very accurate. The results of OMNeT++, however, are increasing.

5.3.2. Packet Loss

An inherent problem of packet switching networks is packet loss. Reasons for packet loss include side effects of algorithmic methods, for instance traffic shaping to provide QoS characteristics, and more rarely electromagnetic interference.

Depending on the degree of packet loss, adequate methods for compensating for the data loss have to be developed. Realistic, verifiable and reproducible scenarios with defined packet loss characteristics can be provided by simulation and emulation. To guarantee precise and accurate measurements for a defined loss rate in the experimental emulation and simulation setup, the two configuration parameters – data transfer rate and packet delay – are set to an optimal value that is not negatively influencing the packet loss rate.

In this experimental setup, the data transfer rate was set to 100 Mbit/s and the packet delay to 0 ms. This configuration prevents a small data transfer rate combined with a long packet delay creating packet loss by exceeding timeout thresholds; which, in this case, would be an unwanted side-effect as it is not controlled by the actual packet loss parameter of the HV-tools.

### Table 1. Comparison of packet loss probability between ns-3 and OMNeT++

<table>
<thead>
<tr>
<th>configured $p_E$</th>
<th>ns-3</th>
<th>OMNeT++</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>0.10</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
</tr>
<tr>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
</tr>
</tbody>
</table>

In order to measure the configuration accuracy of the tools for a given series of measurements, which is the arithmetical difference between the set loss rate and the actual measured loss rate, the packet loss probabilities of each sub-link ($p_L$), which is proportional to the overall packet loss probabil-
ity ($p_E$), is calculated by the following model where $n$ is the number of links the packet traverses in the proposed environment:

$$p_l(n) = 1 - \sqrt[3]{1 - p_E} \quad \text{where} \quad n \in \mathbb{N} \setminus \{0\}$$  \hspace{1cm} (1)

For measuring the packet loss probabilities, 10,000 packets with a size of 100 bytes have been sent and then counted on the receiver’s side in order to calculate the packet loss probability. Again, the measuring tool is Ping. The chosen series of measurements are six defined overall packet loss probabilities ($p_E$). For both HV-tools (ns-3 and OMNeT++), no deviation between the set packet loss probability and the measured probability can be observed. The determined values are summarized in table 1.

### 5.3.3. Data transfer rate

In the current Internet, different data transfer rates exist between computer nodes such as routers or switches. In particular, the data rate on the last mile of the Internet varies greatly.

It can even be observed that Internet connections are generally slower than links between nodes inside a network testbed [6]. This is caused by the evolutionary development of the Internet. In compliance with this, six data link rates are defined in this paper. This is necessary to evaluate typical network circumstances.

These are a standard modem line with 56 kbit/s, an ISDN connection with two speeds, 64 kbit/s, and 128 kbit/s, and three DSL connections with 1,000, 2,000, and 3,000 kbit/s respectively. These data rates will be emulated on each link in the test environment.

The data transfer rate is measured between node A and node B. Iperf [1] is used for the measurements. Iperf provides two basic protocols for data rate measurements, TCP and UDP. Here, UDP is used to avoid protocol dependent rate adjustments like TCP congestion control. The defined pre-conditions for the evaluations are packet loss = 0% and delay = 0 ms. Negative side effects are avoided by this configuration.

Data rate is measured for 300 seconds. The chosen data rates are configured for both the ns-3 and OMNeT++ environments.

The evaluation of the data rate emulation accuracy is shown in figures 6 and 7. It can be noticed that ns-3 differs more from the configured data rate than OMNeT++. In case of an emulated modem line with 56 kbit/s, the measured value provided by ns-3 was 53.9 kbit/s. For the same configuration 54.7 kbit/s, were measured in OMNeT++. It can also be observed that the difference between the measured value and the configured value increases when the emulated data rate gets higher. This is the case for both tools. This fact is shown in figure 7 for typical DSL data rates.

### 6. FUTURE WORK

In this paper, the accuracy of three parameters were measured independently. In the future, the correlation between these three parameters will be evaluated. Furthermore, the experimental setup will be significantly enhanced to measure the simulation and emulation performance of the tools when using larger numbers of simulated nodes, such as would be used when simulating peer to peer networks.

Other interesting simulated extensions to network testbeds include simulating jitter for applications such as audio and video streaming, integrating wireless environment simulations including simulated interference, and generating network traffic and workloads.

### 7. CONCLUSION

In this paper, two well-known simulation and emulation tools, ns-3 and OMNeT++, have been compared in the context of extending network testbeds. The objective comparison of both tools is based on the evaluation of experimental measurements. In summary, ns-3 offers a very precise realization of the set values, especially the recorded delay values remained fixed in contrast to OMNeT++ which, in this case, showed fluctuations during the delay measurements.

OMNeT++ is better able to match specified data rates.
While both tools are able to extend a network testbed, OMNet++ has better documentation and more intuitive tools. Both tools have their strengths, so a testbed would need to allow for both.

8. ACKNOWLEDGEMENTS

We would like to thank Martin Becke for setting up the OMNeT++ environment and for support during development; and the anonymous reviewers for their comments.

REFERENCES


