Scalable Network Emulation: A Comparison of Virtual Routing and Virtual Machines

Steffen Maier, Andreas Grau, Harald Weinschrott, Kurt Rothermel

University of Stuttgart, Institute of Parallel and Distributed Systems (IPVS)
Universitätsstr. 38, D-70569 Stuttgart, Germany
Phone: +49 711 7816–226, Fax: –424
E-mail: {maier|grauas|weinschd|rothermel}@ipvs.uni-stuttgart.de

Abstract

Performance analysis is a necessary step during the development of distributed applications and communication protocols. Network emulation testbeds provide synthetic, configurable environments for comparative performance measurements of real implementations. However, realistic scenarios require more communicating nodes than usual testbeds are able to provide. In order to enable scalable network emulation, various concepts for the virtualization of nodes have been proposed. The overhead of virtualization strongly impacts the total size of a scenario, that can be emulated on a given testbed. However, the overhead of different virtualization approaches in the context of network emulation has not been compared directly so far. In this paper, we present a comparison of different virtual machine implementations (Xen, User Mode Linux) and our own virtual routing approach (NET). We discuss qualitative evaluation criteria and present a quantitative evaluation showing the efficiency of each approach in a traditional wired infrastructure-based and in a wireless ad hoc network emulation scenario. Our results give insights on which virtualization approach is best suited for which kind of network emulation.

1. Introduction

When implementing distributed applications and communication protocols it is essential to analyze the impact of various network environments on their performance. The required measurements usually compare the performance of one implementation in different network environments or of different implementations in the same network environment. Setting up a real live testing environment is difficult due to administrative or economical limitations. In real live measurements, it is also hard to obtain repeatable results especially in scenarios with mobile nodes and wireless communication. Therefore, there is strong demand for synthetic network environments that can be parametrized in order to reproduce the properties of an existing or yet to be built network.

Network emulation allows unmodified real implementations to be subjected to a synthetic network environment. For that purpose, a piece of software called *network emulation tool* reproduces specified network properties while communicating over flexible networking hardware possibly having different properties. A facility consisting of a combination of flexible networking hardware and suitable emulation tools is called *network emulation testbed*. Distributed applications or communication protocols that are subject to performance measurements in a network emulation testbed are called *software under test*.

Various research projects have published different approaches to network emulation testbeds [1, 7, 14, 15, 16, 17]. They have recognized the need for emulating scenarios with a large number of nodes. Evaluations in mobile computing scenarios, e.g. of an ad hoc routing protocol implementation, typically require scenarios with hundreds of nodes. Evaluations in traditional infrastructure-based scenarios require the modeling of the communicating end systems and all intermediate systems to obtain realistic measurement results. In order to support the scalable emulation of scenarios with more nodes than the number of computers in a testbed, different projects have proposed to execute multiple instances of the software under test on a single testbed computer by means of node virtualization.

Hosting too many instances on the same computer leads to resource contention, which influences performance evaluation results and hence is to be prevented.

Published in 12th IEEE Symposium on Computers and Communications (ISCC'07), pages 395 - 402, Aveiro, 2007.
© IEEE 2007
http://dx.doi.org/10.1109/ISCC.2007.4381529

The overhead of virtualization has a strong impact on the possible degree of virtualization and hence the total scenario size. However, the overhead of different node virtualization approaches in the context of scalable network emulation has not been compared directly so far.

In this paper, we compare the virtual machine implementations Xen [2] and User Mode Linux [3] as well as our own virtual routing approach NET [14]. For each candidate, we discuss qualitative evaluation criteria. In order to allow a quantitative comparison with respect to scalability, we evaluate the efficiency of each approach in both a traditional wired infrastructure-based and in a wireless ad hoc network emulation scenario. Our results give insights on which virtualization approach is best suited for which kind of network emulation.

The remainder of this paper is structured as follows. In Section 2, we introduce the architecture of our network emulation testbed, which provides the basis for our comparison of virtualization approaches. In Section 3, we state evaluation criteria for node virtualization. We give an overview of the candidate virtualization approaches and evaluate the qualitative criteria in Section 4. In Section 5, we present measurement results showing the efficiency of each approach for two important kinds of scenarios: emulation of infrastructure-based networks and wireless ad hoc networks. In Section 6, we summarize the results from the previous two evaluation sections. We discuss related work in Section 7. Finally, we conclude the paper with a brief summary.

2. Network emulation

The Network Emulation Testbed (NET) [14] developed at the University of Stuttgart provides the basis for our evaluation of node virtualization approaches. It consists of 64 PCs connected by a monolithic, programmable gigabit switch, and a separate administration network for setup and control (Fig. 1). Using IEEE 802.1Q VLAN (virtual LAN) technology, the gigabit switch is able to create an arbitrary connection topology between the computers. Each point-to-point link or shared media network segment in an emulation scenario, such as a WLAN (wireless LAN) channel, is mapped to a uniquely tagged VLAN.

We call a testbed computer a *pnode* (physical node). On a pnode, several tagged VLANs represent several virtual network interfaces, each of which is assigned a separate instance of our emulation tool called NETshaper. Each NETshaper instance emulates the specified network properties. It enables the configuration of arbitrary bandwidth limitations, delays, and frame error loss ratios. NETshaper provides the service level ab-

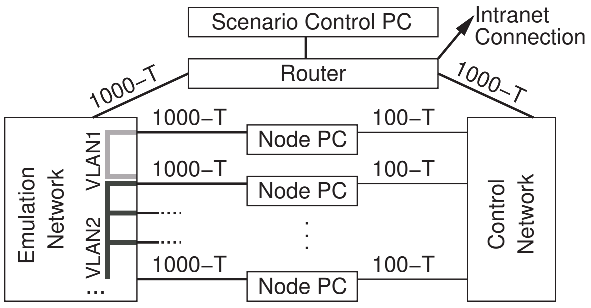

Figure 1. Network emulation testbed.

straction of an unreliable datagram service to the software under test. This is the lowest possible emulation abstraction feasible to be implemented in software. NETshaper is implemented as a virtual network device driver, and thus completely transparent to protocol implementations on the network layer. As a result, the network and transport layer of the protocol stack as well as the application layer can be considered as software under test. Each node in a specified network emulation scenario corresponds to an instance of the software under test which we call a *vnode* (virtual node).

In order to control the distributed network emulation tools during an experiment, we pursue a hybrid architecture including a central scenario controller. The controller dynamically updates the parameters of the emulation tools. For mobile ad hoc network emulation, this includes changing connection quality and thus frame error rates between communicating nodes. The connection quality is automatically derived from the simulated node mobility and the application of different possible radio propagation models.

Without node virtualization one pnode is required for each vnode. Therefore, the size of the emulated scenarios would be limited to 64 vnodes in NET. In the following, we first define evaluation criteria for node virtualization and then evaluate candidate virtualization approaches.

3. Evaluation criteria for node virtualization

In general, node virtualization provides a way to schedule access of a number of consumers to otherwise exclusively used hardware resources. With respect to network emulation, each consumer is a vnode, i.e. an instance of the software under test. In this section, we define three evaluation criteria for node virtualization in the context of network emulation.

Node virtualization should be *transparent* to the software under test, to allow the execution of unmodified code of an implementation. Heterogeneous scenarios involve different implementations of the software under test which require different execution environments, such as a TCP implementation in Linux and another one in Windows. Since more execution environments than pnodes may be necessary, node virtualization should be *flexible* to support different execution environments on the same pnode. The main reason for the introduction of node virtualization to network emulation is scalability with respect to the maximum size of emulation scenarios. As already mentioned before, the overhead of node virtualization strongly impacts the possible degree of virtualization per pnode and hence the size of an emulation scenario. Thus, node virtualization should be *efficient* to maximize the scenario size in terms of vnodes for a given number of pnodes.

The criteria transparency and flexibility can only be discussed on a qualitative basis whereas efficiency can be evaluated quantitatively by means of experiments. Therefore, we split the evaluation discussion over the following two sections. We treat the qualitative criteria first, then the quantitative criterion.

4. Evaluation of qualitative criteria

We classify the node virtualization approaches into two main categories: virtual machines and virtual routing.

4.1. Virtual machines

A straightforward way to introduce node virtualization to network emulation is using a virtual machine (VM) approach. Instead of running an operating system (OS) directly on the hardware, a virtual machine monitor (VMM) schedules access of multiple guest operating systems to exclusive hardware resources (Fig. 2). From the different solutions for integrating an emulation tool into such an architecture, we choose to insert our unmodified existing emulation tool on top of network interface drivers inside each guest OS. This allows to focus on the comparison of the virtualization approaches by keeping the surrounding environment fixed. For communication with other vnodes on the same and other pnodes, a software switch forwards frames correspondingly.

We restrict our selection of evaluation candidates to open source implementations, that are easily available and work well with our existing NET environment, i.e. Linux as OS and NETshaper implemented as a Linux kernel module. Consequently, we consider Xen and User Mode Linux.

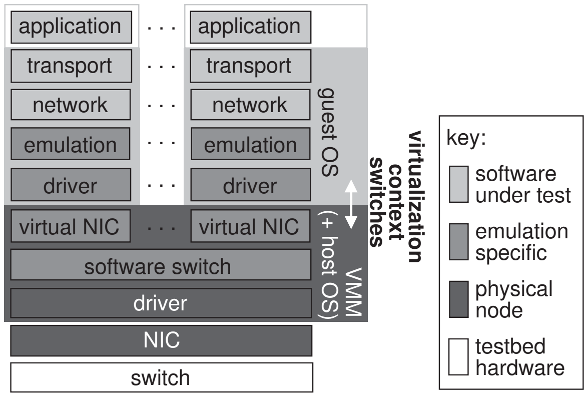

Figure 2. Virtual machine approach.

Xen [2] provides an unhosted VMM, that runs directly on the hardware without any host OS. It offers two different ways of virtualizing the hardware resources. On modern processors with hardware virtualization support, Xen supports the execution of unmodified guest OSs. However, the necessary exact emulation of hardware devices implies significant virtualization overhead. Therefore, we focus on the second option called para-virtualization, where the virtual instruction set architecture and virtual device interfaces are designed for low virtualization overhead. This requires an adaptation of the hardware abstraction layer and device drivers of the guest OS. However, there is no need to modify the software under test inside the guest OS which makes Xen a *transparent* node virtualization approach. Since each provided VM can execute an arbitrary guest OS, it fulfills the criterion of *flexibility*.

User Mode Linux (UML) [3] is a hosted VMM, where the guest OS is of the same type as the host OS (though the versions may differ) which *restricts flexibility*. UML is a port of Linux to its own system call interface, and requires a special UML hardware abstraction layer in form of a Linux architecture backend. However, those modifications do not concern the software under test which makes UML a *transparent* approach.

4.2. Virtual routing

The VM approaches described in the previous subsection virtualize more than is actually needed for network emulation. It would be sufficient to provide virtual execution environments for just the software under test, i.e. for exactly those layers above the emulation tool. This can be accomplished with virtual routing [11] (Fig. 3). Consequently, a vnode consists of the following entities: a set of processes on the application layer, a set of sockets on the transport layer, a routing table on the network layer, and a set of network devices on the data link layer. In contrast to virtual machines, there is no need for separate virtual network devices and their drivers anymore. Additionally, there are no redundant context switches and copy operations.

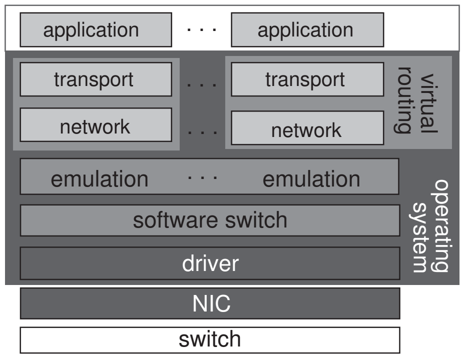

Figure 3. Virtual routing approach.

Virtual routing has *limited flexibility* since there is only one host OS, which is at the same time the execution environment for the vnodes. The integration of virtual routing into the protocol stack of an existing OS requires modifications to the network and transport layer. Hence, virtual routing has *limited transparency* for the software under test.

5. Evaluation of quantitative criterion

In the following, we evaluate the efficiency of each of the previously discussed three virtualization approaches by experiment. All measurements are performed on identical pnodes in our testbed equipped with an Intel Pentium 4 2.4 GHz processor, 512 MB RAM, and a Gigabit Ethernet adapter in a 32 bit, 33 MHz PCI bus. For more details on our experiment configurations, we refer to [14]. Here, we use the Linux distribution Fedora Core 4 (FC4) with the following software version combinations: Linux 2.6.11-1.1369_FC4 with Xen-2-20050522 (both from FC4); UML backend of Linux 2.6.16-bs2 on Linux 2.6.11 patched with SKAS3-v8.2 [5] to enable performance optimizations similar to [10]. In our experiments, we focus on the network and transport layer protocol implementations. We consider two types of network: a wired network and a wireless ad hoc network which apply different routing protocols.

5.1. Wired network emulation

The network topology of the emulated wired scenario consists of a linear chain with a varying number of router vnodes using static routing. The point-to-point links connecting the routers are full duplex and have an emulated limited bandwidth of 100 MBit/s in each direction. We obtain reference measurements once by executing each vnode on its own pnode. For each virtualization approach, we conduct the same experiment and execute all vnodes on a single pnode except for the last vnode, which resides on a separate pnode without virtualization (Fig. 4). Note that the protocol stack of each vnode is involved while a message is passed from source to sink.

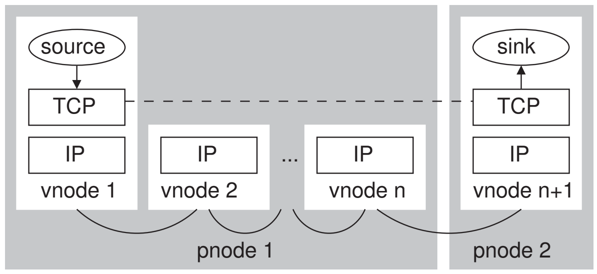

Figure 4. Wired infrastructure emulation scenario, virtualized case.

On network layer, we measure maximum ICMP round trip time (RTT) delays between vnode 1 and vnode $n+1$. In Fig. 5, the reference values without virtualization show linear increase with an increasing number of hops in the routing chain. Both VM implementations cause delays that are larger than without virtualization. Due to time slices for the scheduling of guests by the VMM and the necessary virtualization context switches each router introduces additional delay which cannot be compensated. The delay with Xen even increases unrealistically starting with 6 vnodes. Virtual routing of NET causes a lower delay than without virtualization. This is because the software switch has lower communication delay than the hardware emulation switch and there are no redundant virtualization context switches. Of course, the emulation tool could compensate for that, if a particular scenario requires inter-node delays to be exactly the same. We did not increase the degree of virtualization beyond 8 vnodes for the VM approaches, since main memory would become a bottleneck. Also their maximum degree lies below 8 vnodes for higher network loads as shown in the next paragraph.

On transport layer, we measure the throughput of one TCP flow over the router chain (Fig. 6). The source is located on vnode 1 and the sink on vnode $n + 1$ (flow 1). In a subsequent experiment we interchange the source and sink locations (flow 2). Again we measure the throughput of one TCP flow. The earliest undesirable deviations from the reference case happen at

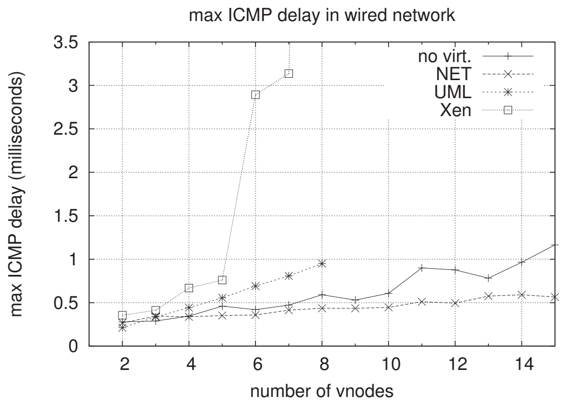

Figure 5. Maximum round trip time delay versus number of vnodes in wired network.

a number of 3 vnodes for UML, 5 vnodes for Xen, and 13 vnodes for NET. For all candidates the deviations are due to contention of the CPU resource only. When the TCP source is on vnode $n + 1$ for flow 2, it does not have to compete for resources with other vnodes on the same pnode and is able to achieve a realistic throughput with more vnodes than flow 1 in the opposite direction. UML and NET are affected by this asymmetry, where the remaining resources on pnode 1 suffice TCP receive processing for 3 vnodes with UML or 50 vnodes with NET (the latter is not shown in Fig. 6). We assume that the scheduling of the Xen VMM and its CPU resource isolation between guests prevents such a behavior.

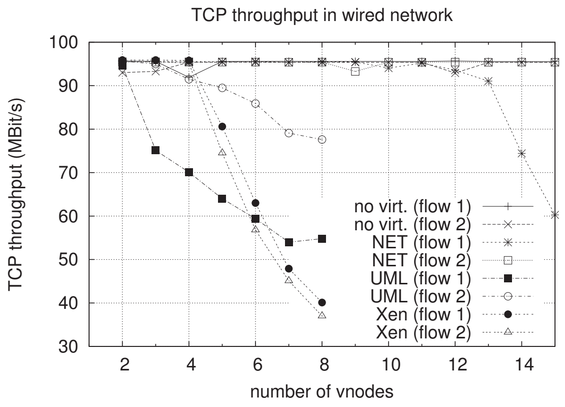

Figure 6. Throughput versus number of vnodes in wired network.

5.2. Wireless ad hoc network emulation

In the wireless scenario, we arrange the vnodes to also build a linear topology to make the results of our experiments comparable. The scenario is shown in Fig. 7, where neighbors can communicate directly over a wireless communication link. This is accomplished by a frame loss ratio for ingress traffic of zero for frames from reachable neighbors, and one for all others. Although our emulation system fully supports dynamic node mobility for comprehensive mobile ad hoc network scenarios, we use stationary nodes here for the sake of comparison with the wired scenario. The wireless links between vnodes are full duplex and have a limited bandwidth of 11 MBit/s. Here, we do not emulate the effects of a MAC layer, i.e. there are no frame collisions. We use an implementation of the Ad-hoc On-Demand Distance Vector Routing protocol called AODV-UU [12] in version 0.8 for dynamic routing on each vnode. Similar to the wired network scenario, we perform measurements once without virtualization and for each virtualization approach separately. The measurements are conducted analogous to the wired scenario.

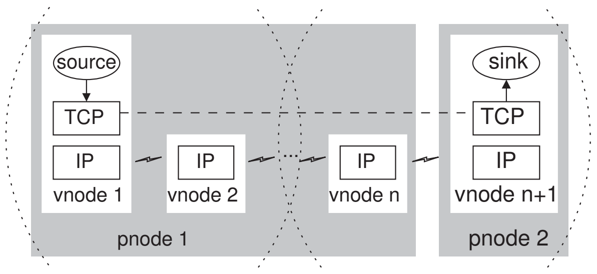

Figure 7. Wireless ad hoc network emulation scenario, virtualized case.

Fig. 8 shows the typical behavior of maximum ICMP RTT delays for AODV with different hop counts. Starting with one hop, we observe expanding ring search in combination with binary exponential backoff for outgoing route requests. Beyond the default time to live threshold, route requests work without expanding ring search leading to delays with linear increase starting at 10 hops, i.e. 11 vnodes. Virtual routing of NET matches the reference values without virtualization closely up to 10 vnodes. Starting with 11 vnodes, the load implied by the AODV routing daemon on each vnode causes increasingly larger delays with each additional vnode. UML has a relatively high cost for context switches and hence for each packet transmission. Each additional hop causes an increase in the RTT delay compared to the reference values. The delay with Xen is lower than with UML but still larger than the reference values. As in the wired scenario the delays with the VM approaches are too large and cannot be compensated.

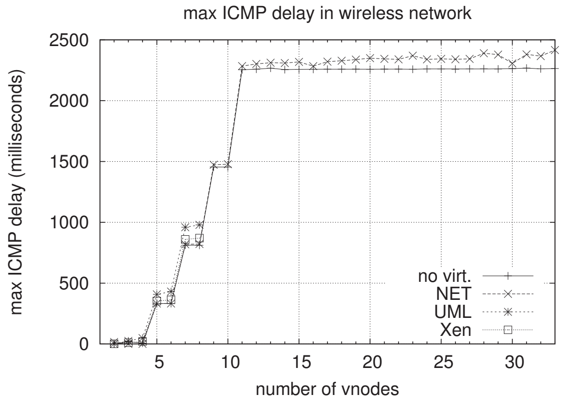

Figure 8. Maximum round trip time delay versus number of vnodes in wireless scenario.

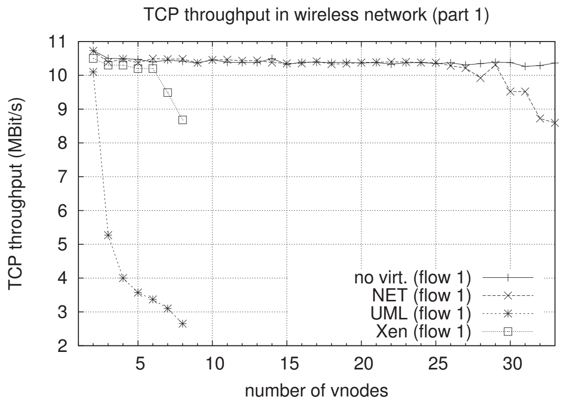

Figure 9. Throughput versus number of vnodes in wireless scenario (part 1).

Measurement results for the transport layer are depicted in Fig. 9 and Fig. 10. TCP throughput starts deviating from the reference values without virtualization at 2 vnodes for UML, 7 vnodes for Xen, and 30 vnodes for NET. The emulated bandwidth and thus the network load is lower compared to the wired scenario. However, each AODV routing daemon overhears all IP packets from its two respective neighbors. This causes many guest OS context switches in order to pass received packets from the kernel to the daemon running in user space. Those context switches also involve the

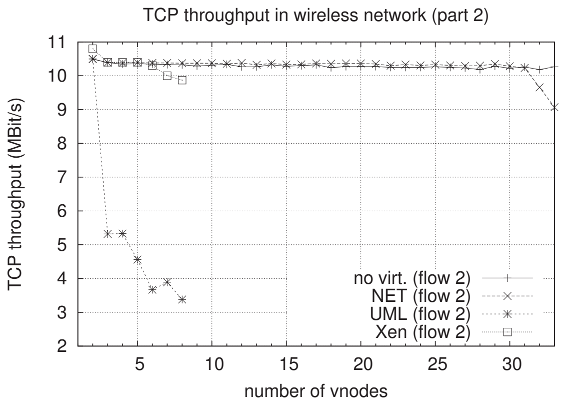

Figure 10. Throughput versus number of vnodes in wireless scenario (part 2).

VMM and are thus expensive for the VM approaches. Therefore, they only support about the same number of vnodes per pnode as in the wired scenario. Virtual routing of NET however is not affected as strongly by the additional context switches and is able to take advantage of the comparatively low emulated bandwidth.

6. Discussion

CPU is the limiting factor for scalability, i.e. the maximum scenario size, with all virtualization approaches in the evaluation scenarios from the previous section. UML's high context switch costs allow hardly more than one vnode per pnode. Hence, UML is not suitable for scalable network emulation in realtime – not even with applied optimizations similar to [10]. This confirms results reported in [8] stating that such a virtualization approach is more suitable for qualitative analysis than for performance measurements. Xen supports 4 to 6 vnodes per pnode on our testbed hardware depending on the scenario. On the one hand, Xen is transparent and flexible allowing the performance evaluation of arbitrary mixes of software under test such as TCP implementations of Linux, BSD, and Windows. On the other hand, Xen suffers from virtualization context switches that are disadvantageous if software under test on the application layer is involved. Virtual routing of NET shows to be the most efficient alternative and should be the choice if scalability is the main concern of network emulation. The reduced context switch costs of virtual routing are especially advantageous in emulation scenarios, where software under test on the application layer generates significant traffic.

7. Related work

In this paper, we are mainly interested in the influence of different node virtualization approaches on the scalability of network emulation in general. We discuss other individual scalable network emulation approaches in this section.

Ns-e [4] is an emulation extension of the well-known network simulator ns-2. The scalability is limited by the amount of traffic, that can be processed by the centralized simulator. For a typical mobile ad hoc network (MANET) experiment, a total scenario size of about 50 nodes is possible [9].

ModelNet [15] is a parallel network emulator for point-to-point link based scenarios without support for the emulation of wireless communication. It is not entirely transparent for the software under test since it interposes socket calls. MobiNet [13] is an extension for wireless communication but due its centralized architecture, it constitutes a bottleneck as with ns-e. While the authors report successful emulation of 100 MANETs with 2 nodes each, this result can hardly be compared to a realistic MANET scenario with 200 nodes.

vBET [8] is designed to emulate a network scenario on a single computer by using UML for virtualizing nodes. While this centralized architecture would constitute a bottleneck, multiple computers can be combined to a scalable distributed testbed. The overall performance is reported to be more suitable for qualitative analysis than for performance measurements. Our results confirm that, since UML hardly supports more than one vnode per pnode in our scenarios even with applied optimizations for hosted VMs similar to [10].

Empower [17] allows the emulation of multiple routing instances on one computer. Each connection to the emulated network is mapped to a physical link of an existing hardware network interface. The scalability, i.e. the number of vnodes per pnode is limited by the number of physical network interfaces per pnode.

Imunes [16] virtualizes the entire protocol stack in the kernel. The required extensive modifications limit the transparency for the software under test. In combination with the network emulation tool dummynet, link-based scenarios can be emulated in a scalable way. The authors report TCP throughput of 420 MBytes/s over 15 routing hops on a single machine with a slightly faster processor than used in our evaluation. Though scaling significantly better than a VMware based virtualization implementation, the throughput was measured in a best case without any introduction of emulated network properties such as bandwidth limitation and is thus hardly comparable to our results.

Netbed [7] is an emulation testbed, that supports scalable network emulation by using virtual routing. The testbed supports the emulation of scenarios with wired links. While it is possible to link real wireless nodes to an emulated scenario, there is no support for the reproducible emulation of wireless networks.

V-eM [1] supports scalable network emulation by integrating the emulation tool NISTnet into the Xen VMM. It forces traffic between vnodes on the same pnode to pass externally through physical network interfaces and thus scalability is severely limited for wireless scenarios with shared media. The authors report successful emulation of a scenario with 10 routers connected by 100 MBit/s links. Unfortunately these results are not comparable to ours, since they used two processors each of which is faster than our hardware.

8. Summary

Network emulation testbeds provide a synthetic, configurable network environment for comparative performance measurements of distributed applications and communication protocols. The number of communicating nodes in meaningful evaluation scenarios (hundreds) is often larger than the number of available computers in a testbed (tens). In order to enable scalable network emulation for such large scenarios, various node virtualization approaches have been proposed in the literature.

In this paper, we compared different virtual machine implementations (Xen, User Mode Linux) as well as our own virtual routing approach (NET). Based on a discussion of qualitative evaluation criteria and on quantitative evaluation results, we provided insight into which kinds of network emulation are best suited for the different node virtualization approaches. Xen is more suitable for scenarios with arbitrary mixes of software under test such as TCP implementations of Linux, BSD, and Windows. Virtual routing of NET is advantageous if scalability is the main concern and if software under test on the application layer generates significant traffic.

We have shown that a performance analyst faces a trade-off between scalability on the one side and transparency or flexibility on the other side. Those limitations could be mitigated by virtual machine monitors, that provide software under test with virtual time to enable non-realtime network emulation [6]. Thus, experiment execution time may be traded for scalability, transparency, *and* flexibility in the future.

9. Acknowledgment

This work is partially funded by the Deutsche Forschungsgemeinschaft (German Research Foundation) under grant DFG-GZ RO 1086/9-1. We thank Christopher Schwarzer for helping with the initial experiment setup for different virtual machine implementations.

References

[1] G. Apostolopoulos and C. Hasapis. V-eM: A Cluster of Virtual Machines for Robust, Detailed, and High-Performance Network Emulation. In *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS'06)*, pages 117–126, Monterey, CA, USA, Sept. 11–14 2006.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 164–177, Bolton Landing, NY, USA, Oct. 19–22 2003.

[3] J. Dike. A user-mode port of the Linux kernel. In *Proceedings of the 4th Annual Linux Showcase & Conference*, Atlanta, GA, USA, Oct. 10–14 2000.

[4] K. Fall. Network Emulation in the Vint/NS Simulator. In *Proceedings of the Fourth IEEE Symposium on Computers and Communications (ISCC'99)*, pages 244–250, Red Sea, Egypt, July 6–8 1999.

[5] P. Giarrusso. SKAS3-v8.2 for UML-hosts. http://www.user-mode-linux.org/~blaisorblade/patches/skas3-2.6/skas-2.6.11-v8.2/ visited Nov. 2006.

[6] D. Gupta, K. Yocum, M. McNett, A. C. Snoeren, A. Vahdat, and G. M. Voelker. To Infinity and Beyond: Time-Warped Network Emulation. In *Proceedings of the 3rd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI 06)*, pages 87–100, San Jose, CA, USA, May 8–10 2006.

[7] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau. Feedback-directed Virtualization Techniques for Scalable Network Experimentation. Flux Group Technical Note FTN-2004-02, School of Computing, University of Utah, May 2004.

[8] X. Jiang and D. Xu. vBET: a VM-Based Emulation Testbed. In *Proceedings of the ACM SIGCOMM 2003 Workshops*, pages 95–104, Karlsruhe, Germany, Aug. 25&27 2003.

[9] Q. Ke, D. A. Maltz, and D. B. Johnson. Emulation of Multi-Hop Wireless Ad Hoc Networks. In *Proceedings of the Seventh International Workshop on Mobile Multimedia Communications (MoMuC 2000)*, Tokyo, Japan, Oct. 2000.

[10] S. T. King, G. W. Dunlap, and P. M. Chen. Operating System Support for Virtual Machines. In *Proceedings of the 2003 USENIX Annual Technical Conference*, pages 71–84, San Antonio, TX, USA, June 9–14 2003.

[11] K. Kourai, T. Hirotsu, K. Sato, O. Akashi, K. Fukuda, T. Sugawara, and S. Chiba. Secure and Manageable Virtual Private Networks for End-users. In *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN'03)*, pages 385–394, Bonn/Königswinter, Germany, Oct. 20–24 2003.

[12] H. Lundgren, E. Nordström, and C. Tschudin. Coping with Communication Gray Zones in IEEE 802.11b based Ad hoc Networks. In *Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia (WoWMoM'02)*, pages 49–55, Atlanta, GA, USA, Sept. 28 2002.

[13] P. Mahadevan, A. Rodriguez, D. Becker, and A. Vahdat. MobiNet: A Scalable Emulation Infrastructure for Ad Hoc and Wireless Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 10(2):26–37, Apr. 2006.

[14] S. Maier, D. Herrscher, and K. Rothermel. Experiences with node virtualization for scalable network emulation. *Computer Communications*, 30(5):943–956, Mar. 8 2007.

[15] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, MA, USA, Dec. 9–11 2002.

[16] M. Zec and M. Mikuc. Operating System Support for Integrated Network Emulation in IMUNES. In *Proceedings of the 1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure (2004 OASIS)*, pages 3–12, Boston, MA, USA, Oct. 9–13 2004.

[17] P. Zheng and L. M. Ni. EMPOWER: A Network Emulator for Wireless and Wired Networks. In *Proceedings of the Conference on Computer Communications (INFOCOM 2003)*, volume 3, pages 1933–1942, San Francisco, CA, USA, Mar. 30 – Apr. 3 2003.