Time accurate integration of software prototypes with event-based network simulations

Elias Weingärtner, Florian Schmidt, Tobias Heer, and Klaus Wehrle
Distributed Systems Group
RWTH Aachen University
{weingaertner, florian.schmidt, heer, wehrle} @ cs.rwth-aachen.de

ABSTRACT

The concept of network emulation brings together the flexibility of network simulations and the accuracy of real-world prototype implementations. However, this approach suffers from the fundamental problem of simulation overload which occurs if the simulation is not able to execute in real-time. We tackle this problem with a concept we call *Synchronized Network Emulation*. It enables the time accurate integration of implementations with network simulations of any complexity.

Categories and Subject Descriptors: C.2.2 [Computer-Communication Networks]: Network Protocols; C.4 [Performance of Systems]: Measurement
General Terms: Experimentation, Measurement, Performance

1. INTRODUCTION

Network emulation is a hybrid methodology which combines the flexibility of network simulations with the precision of evaluations carried out on physical systems. An event-based simulation which models a computer network of choice is connected to a physical system that executes a prototype implementation. Traffic originating at the prototype is inserted into the simulated network and passed through it. The simulation provides simulated hosts that interact with the prototype implementation. This concept was first introduced by Kevin Fall [2] in 1999. Most notably, event-based simulations and software prototypes differ greatly in their underlying timing concepts. An event-based simulation consists of a series of discrete events with an associated event execution time. In contrast to that, software prototypes progress continuously through wall-clock time. For the integration of those timing domains, existing implementations of network emulation pin the execution of simulation events to the corresponding wall-clock time. In his paper, Fall already stated that this approach is only useful if the simulation executes in real time, and that there was "no simple solution to this issue". If the simulation lags behind in time, it is unable to deliver packets in a timely manner. Such *simulator overload* may arise whenever complex network simulations are used or if large amounts of packets need to be processed by the simulator. Hence, simulator overload must be prevented by all means because erroneous protocol behavior, such as connection time-outs, unwanted retransmissions, or the assumption of network congestion are straight consequences, thus rendering measured results unusable or at least questionable.

Physically speeding up the simulation hardware to make it real-time capable is the first obvious option to deal with simulation overload. This can be achieved by supplying the simulation machine with sufficient computational resources or a parallelization of the network simulation. This approach is well elaborated by Kiddle in [3], where a ns-2 network simulation is partitioned using a 128 multiprocessor machine in order to facilitate network emulations with thousands of real-time simulated hosts. However, we argue that this approach lacks generality because parallel processing cannot be applied to every kind of simulation and only scales to the degree of possible parallelism within the simulation. Moreover, the amount of hardware needed for real-time execution rapidly grows with the simulation complexity, making this option inaccessible for many research institutes and individuals.

So far, network emulation has merely been an arms race between the complexity of the simulation model and the computational power of the simulation hardware. Hence, traditional approaches result in *variable hardware requirements* and *fixed execution time* (real time). We aim at reducing the cost of precise network emulation by designing a system with *fixed hardware demands* but with *variable execution time* (real time or slower). In order to achieve this goal, we recently proposed Synchronized Network Emulation (*SNE*) [5], which uses throttling and synchronization mechanisms to match the speed of the simulation to the hosts attached.

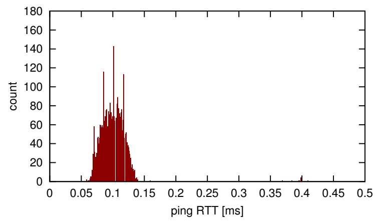
2. SYNCHRONIZED NETWORK EMULATION

A synchronized network emulation scenario contains three different kinds of components.

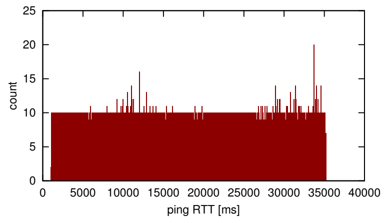
An event-based **network simulation** models an arbitrary computer network that interconnects the software prototypes. This requires an adequate message translation and bridging between the "real world" software prototypes and the network simulation model.

Instead of executing the software prototypes on physical machines, we employ **virtualized hosts** for this purpose. The use of virtualization enables us to modify the software prototypes' perception of time in order to have them execute in virtual time instead of following wall-clock time. This way, we are able to introduce artificial gaps in their continuous execution progress. This allows the network simulation to catch up with the execution of the virtualized hosts.

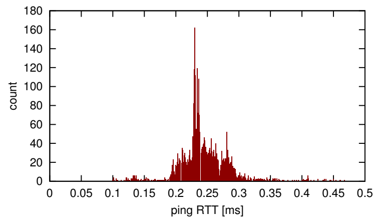
The actual synchronization is carried out by a central **synchronization component** that assigns slices of vir-



(a) RTTs for two Linux PCs



(b) Unsynchronized execution leads to corrupted results



(c) Synchronized execution of one virtualized host and a simulation

Figure 1: Synchronized network emulation produces precise timing results

tual time to the network simulation and the attached virtual hosts. Synchronized peers notify the synchronization component once they have finished the execution of the time slice. Once all virtualized hosts and the network simulations have signaled this completion, a new time slice is assigned. This way, the time drift between virtualized hosts and the network simulation is bounded by the size of a time slice.

3. IMPLEMENTATION

For the purpose of investigating the characteristics of synchronized network emulation, we have implemented a corresponding tool set. Our virtualized host implementation is based on the Xen [1] hypervisor. For the network emulation, we utilize the OMNeT++ [4] and its INET framework. The synchronization component is implemented as a standalone application. It implements a lightweight synchronization protocol based on UDP to limit the overall messaging complexity.

4. RESULTS

First, we investigated the effect of synchronization in a scenario that suffers from simulation overload if no synchronization is in place. Therefore, we measured the round trip times of 3500 ICMP echo reply packets. The results depicted in Figure 1 demonstrate that a synchronized network emulation scenario is able to reproduce a timing behavior close to one found in a real world. Second, we analyzed the overhead which is caused by synchronization of the virtualized hosts and the simulation. Figure 2 shows the total execution time for a simulation scenario with a duration of 600 seconds,

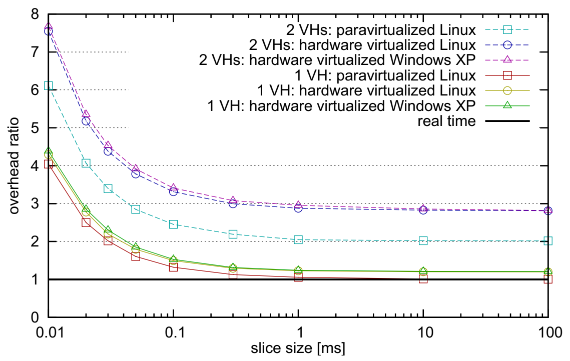


Figure 2: Run-time overhead at different levels of synchronization accuracy

given different levels of synchronization accuraccies. In addition, we measured the simulation overhead for two concurrent virtualized hosts running on one physical machine and different types of executed operating systems. Our results show that, for one synchronized virtualized host, the synchronization imposes less than 10% of additional overhead for time slices equal or greater than 1ms, which is sufficient for most wide area network scenarios. Moreover, our implementation supports the synchronization of virtualized hosts and network simulations with an accuracy of up to 10μs at higher, but still moderate cost.

5. CONCLUSION AND OUTLOOK

We conclude that synchronized network emulation is a feasible approach for the evaluation of large-scale network systems. First, even closed-source binaries (e.g., operating systems, proprietary protocols, and applications) can be analyzed with the versatility of a simulator without being limited to real-time capable simulations. Second, synchronized network emulation facilitates evaluations based on accurate end-host behavior without using static traces, statistical traffic patterns, or over-simplified models. In conclusion, synchronized network emulation enables the convenient investigation of protocol implementations in large simulated environments at low additional cost. We therefore anticipate synchronized network emulation becoming a versatile and powerful tool in network analysis.

6. REFERENCES

[1] 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)*, Bolton Landing, NY, USA, Oct. 2003. ACM.

[2] K. R. Fall. Network emulation in the Vint/NS simulator. In *Proceedings of the 4th IEEE Symposium on Computers and Communication*. IEEE Computer Society, 1999.

[3] C. Kiddle. *Scalable Network Emulation*. PhD thesis, Department of Computer Science, University of Calgary, 2004.

[4] A. Varga and R. Hornig. An overview of the OMNeT++ simulation environment. In *Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008')*, Marseille, France, March 2008.

[5] E. Weingärtner, F. Schmidt, T. Heer, and K. Wehrle. Synchronized network emulation: Matching prototypes with complex simulations. In *Proceedings of the First Workshop on Hot Topics in Measurement & Modeling of Computer Systems (HotMetrics'08)*, Annapolis, MD, June 2008.