Synchronized Network Emulation

Elias Weingärtner
Distributed Systems Group
RWTH Aachen University
elias.weingaertner@cs.rwth-aachen.de

1. INTRODUCTION

In the area of computer networks and distributed systems, people nowadays often face the challenge of investigating protocols and communication systems of high complexity. In order to evaluate new protocols, many researchers rely on network simulation to investigate all kinds of performance metrics, such as throughput or end-to-end delays. The main reason for the widespread use of network simulation is its flexibility: Network simulators, such as ns-2 [6] or OMNet++ [7] facilitate the simulation of large networks with thousands of virtual nodes. All parameters of the simulated network can be changed in a quick and convenient way, and the provided abstractions enable the rapid development of simulation models. In addition, many simulation frameworks already provide a comprehensive collection of simulation models for all kinds of protocols and networked systems. However, a major drawback of network simulation remains to be the negligence or even the complete disregard of the execution context: The real world performance of a network protocol particularly hinges on the underlying implementation. System interrupts, caching and of course the system design itself influence the overall performance, and the network simulators' pure functional models naturally cannot take such effects into account.

In order to investigate the resource requirements and the performance impact of a particular protocol, it is usually implemented as a prototype and evaluated in a testbed that consists of real physical machines. However, the set-up and maintenance of larger testbeds is usually complex and often very costly. Although public testbeds such as PlanetLab [2] enable the evaluation of protocol in a larger setting, their flexibility is limited due to the inability of changing the network's topology or its nodes' underlying configuration in a fundamental way.

A hybrid approach which combines the flexibility of network simulations with the benefits of real-world prototypes is network emulation: A prototype is connected to a network simulator, which models the network the prototype interacts with. This concept has surfaced almost ten years ago [3]. However, network emulation up to now suffers from the constraint that the network simulation is expected to be real-time capable. This means that the virtual nodes which reside inside a simulation are expected to respond as timely as real systems would. On the contrary, one can easily think of a simulation which can not cope with this real-time requirement: If a simulated network consists of thousands of nodes or if very complex channel models, e.g. for wireless links, are in place, such simulations may in fact execute much slower than the time in the real world progresses. When a prototype is connected to such a "slow" simulation, the time drift between the simulation and the prototype finally may lead to connection time-outs or retransmissions as the simulated hosts are not able respond in time. Thus, corrupted results would be the straight consequence. Obviously, this restrains the applicability of network emulation to cases where the complexity of the simulated network is limited.

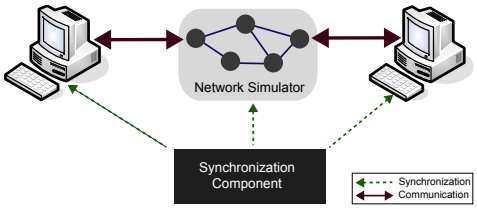

Figure 1: Synchronized Network Emulation

2. SYNCHRONIZED NETWORK EMULATION

With the goal of facilitating the combination of simulations with an arbitrary degree of complexity and real-world prototypes, we're currently investigating a concept we refer to as *synchronized network emulation*. The idea is sketched in Figure 1: A central synchronization component controls the run-time behavior of both the simulation and the real systems attached. In the following, we shortly outline the requirements regarding the three building blocks which constitute a synchronized network emulation set-up:

2.1 Synchronization Component

The synchronization component is in charge of controlling the progress in time, both at the real system and the simulations side. Hence, it needs to implement a suitable synchronization algorithm. Currently, we rely on conservative

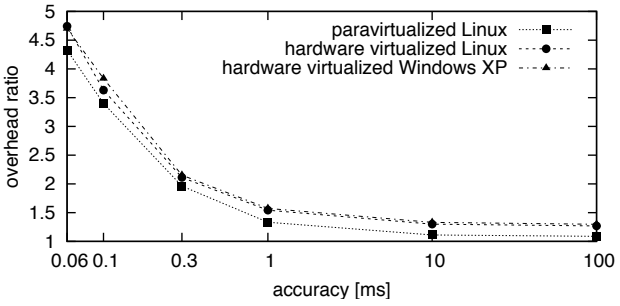

Figure 2: Synchronization overhead vs. accuracy

algorithms borrowed from the domain of parallel discrete event-based simulations [4] for this purpose.

2.2 Real System Integration

In order to synchronize the real systems' execution with the simulation, we need to stall their execution regularly in order to prevent them from drifting away in time. Hence, the real systems must be placed in an execution environment which provides full control both over their run-time execution as well as over internal state variables such as clocks: As the synchronization introduces artificial gaps during their execution, we must provide the real systems with a virtual, continuous flow of time that is in fact aligned to the network simulations' progress. Currently, we investigate the usability of virtualization techniques, such as hypervisors like Xen [1] and full-system simulators as Simics [5] for this purpose.

2.3 Network Simulation

As illustrated in Fig. 1, the network simulator provides a virtual network consisting of virtual hosts and virtual channels. We rely on common discrete-event based simulators such as ns-2 or OMNet++ for this task. In order to achieve a synchronization with the real systems or rather their virtualized counterparts, a modified simulation core executes the simulation events following the used synchronization scheme. Another issue is that network simulations typically use simplified protocol models, which need to be extended for real-world compatibility. Moreover, the message formats in both worlds differ as well. Therefore, an adequate message translation has to be carried out as soon as a packet trespasses from the simulation to a real system or vice versa.

3. RESEARCH STATUS

So far, we have implemented a working research prototype with the goal of investigating the applicability of synchronized network emulation. The research prototype comprises an extensively modified Xen hypervisor, add-ons for the OMNet++ network simulator and a custom implementation of the synchronization component. It allows one to build up synchronized network emulation scenarios which contain virtualized hosts running an arbitrary x86 operating system and OMNet++ network simulations of any complexity. We have evaluated among other aspects the possible degree of synchronization accuracy as well as the overhead which is introduced by the synchronization itself: Our system facilitates synchronous execution of our Xen-based virtualized systems and the network simulation with an accuracy up to 60μs. The synchronization overhead, depicted in Fig. 2, reaches a value between 4.4 and 4.8 at this accuracy level, which means that a synchronized host runs between 4.4 and 4.8 slower than an unsynchronized one. However, our evaluation also shows that the synchronization overhead decreases quickly if the accuracy is diminished. For example, if the synchronization accuracy is set to 0.3ms, the synchronization overhead remains below 2.5. Considering the fact that we are interested in combining highly complex network simulations with our virtualized systems, these results suggest that the integration of real systems will not be the performance bottleneck in synchronized network emulation scenarios. In addition, the achieved level of possible accuracy is sufficient in many cases, e.g. if one aims at the investigation of application-level protocols used in wide-area networks. Further experiences with this implementation and more details regarding the concept of synchronized network emulation are elaborated in [8].

4. FUTURE DIRECTIONS

We consider the extension of the framework to other application domains such as wireless sensor networks and embedded systems in general. Another issue we look into is the development of frameworks, based on synchronized network emulation, for automated performance evaluation and the calibration of models within the network simulation.

5. 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)*, pages 164–177, Bolton Landing, NY, USA, Oct. 2003. ACM.

[2] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[3] K. R. Fall. Network emulation in the Vint/NS simulator. In *Proceedings of the 4th IEEE Symposium on Computers and Communication*, pages 244–250. IEEE Computer Society, 1999.

[4] R. M. Fujimoto. Parallel discrete event simulation. *Communcations of the ACM*, 33(10):30–53, 1990.

[5] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *IEEE Computer*, 35(2):50–58, 2002.

[6] The network simulator ns-2. `http://www.isi.edu/nsnam/ns/`.

[7] A. Varga. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, June 2001.

[8] 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 and Modeling of Computer Systems (HotMetrics 2008) (to appear)*, June 2008.