Wireless Ad-Hoc Network Emulation Using Microkernel-Based Virtual Linux Systems

Michael Engel, Matthew Smith, Sven Hanemann, and Bernd Freisleben

Dept. of Mathematics and Computer Science,
University of Marburg,
Hans-Meerwein-Str., D-35032 Marburg, Germany
{engel,matthew,hanemann,freisleb}@informatik.uni-marburg.de

ABSTRACT

To explore the behavior of new algorithms or applications for mobile wireless ad-hoc networks, dedicated simulation systems are widely used. Since such simulation systems can only approximate the situation existing in real networks, significant efforts have to be spent to adapt algorithms or applications running in a simulation environment to run in a real network environment. In this paper, we present an approach to provide an emulation environment for mobile ad-hoc networks that is as similar as possible to the real networking conditions and in fact can be interfaced to real-world devices. The proposed system is based on multiple Linux instances running on top of a microkernel. A special multiplexer component provides interconnection between the Linux instances and optionally a link to physical networks. By controlling network connections between the different instances, an emulation of node movement as well as error conditions in the network is possible.

KEYWORDS

Wireless Ad-Hoc Networks, Network Emulation, Microkernel

1 Introduction

Mobile wireless ad hoc networks consist of a number of mobile nodes operating autonomously in a particular area without using any pre-existing infrastructure. Each mobile node acts as an intermediate router forwarding messages received by other nodes. Thus, a mobile wireless ad hoc network is only operational if nodes offer their forwarding capabilities to other nodes. Due to the drop in hardware prices it has become feasible to field a large number of mobile nodes to form large ad hoc wireless networks.

Building testbeds for real life systems to efficiently test and diagnose wireless ad hoc networks consisting of 100 nodes or more is non-trivial due to very high costs and the requirement to test them under a wide range of mobility scenarios. Hence, network simulators implemented in software are valuable tools for researchers in developing, testing and diagnosing such ad hoc networks. Simulation is economical because experiments can be carried out on a single powerful computer without buying the actual wireless computing devices. Furthermore, it is flexible because different scenarios concerning link quality, topology chances and node configuration can be implemented and tested in a reproduceable manner. Simulation results are easier to analyze than experimental results because important information can be specifically logged to help researchers diagnosing network components. The benefits of simulation in the field of wireless ad hoc networks are the ability to rapidly prototype systems, reduce the initial costs of the project, the ability to test the scalability of the system and to be able to reproduce the test conditions through the use of scenarios.

However, simulations are only an approximation of real networks, since a simulator model is necessarily a simplification of the real world system. To constrain their complexity, most existing ad hoc network simulators like NS2[1], GloMoSim[2] and OpNet[3] can only simulate real-world network protocol implementations with limited detail. Thus, testing application specific network protocol modifications or the efficiency of applications in such an environment requires an adaptation or redesign of the simulation. Furthermore, the code produced to run on the simulator is usually not transferable to a real world system as it contains simulation specific commands. A further critical problem with simulation is that due to the fact that only a certain portion of the real world system is modeled, *cross-layer design* issues can not be examined. Cross-layer design for wireless networks is an important field of study and has been examined in a number of papers [4, 5, 6, 7, 8].

To be able to examine such cross-layer issues, a system is required which adequately models all relevant parts of the system. With network emulators this is possible, since a real machine running a full operating system is used for testing. The drawback of emulation systems is that for each node to be emulated, a dedicated machine is needed, making large scale tests difficult. To bypass this shortcoming, multiple operating system instances can be launched on a single machine using User Mode Linux (UML) [9]. The draw back of such systems is that they are more resource intensive than simulations.

In this paper, we present a mobile wireless ad hoc network emulation system capable of running multiple nodes on a single PC based system, using significantly less resources than comparable approaches based on UML. Based on a variant of the L4 [10] microkernel and an improved communication model, we are able to implement a scalable model of network behavior ranging from the simple binary decision matrix like the one used in MobiEmu [11] to a complex modeling of the 802.11b range based signal degradation, channel contention and interference, e.g. by employing Markov-chain based models [12]. This offers a new level of detail to wireless network emulation, allowing cross-layer design while at the same time reducing the performance overhead of traditional emulation systems.

The paper is organized as follows. Section 2 describes related work in mobile ad hoc network simulation and emulation systems. Section 3 gives an emulation-oriented overview of the system. Section 4 presents details of the system architecture. Section 5 concludes the paper and outlines areas for future research.

2 Related Work

2.1 User Mode Linux

A system that can be used for emulation of wireless networks is User Mode Linux (UML) [9]. It allows to use several independent Linux instances each running an adapted version of a complete Linux kernel in user mode, providing a shared network environment through the underlying base Linux system. During the creation of an ad hoc wireless network emulation system, we discovered two fundamental drawbacks of UML. Firstly, emulation performance is severely reduced due to the fact that UML runs in user mode and privileged operating system functions must be emulated by the underlying kernel running on the real hardware. This means that, for instance, context switches within the virtual UML take up to a factor 100 times longer than on a standard Linux system. The second drawback is that UML only offers a virtual ethernet interface and not a virtual WLAN wireless interface, such as e.g. a 802.11b interface. In addition, the structure of the underlying common network emulation only permitted us to implement a binary model of wireless connections allowing a single connection to either function fully or not at all.

2.2 MobiEmu

UML itself only emulates a single wireless node. To emulate an entire network, a central controlling instance is required which monitors and controls the network state of all UML instances. MobiEmu[11] is one such central control solution which emulates a rudimentary wireless network based on UML. It uses the same scenario files as the mostly used network simulator NS2 [**?**] to calculate the location of the virtual wireless nodes, and based on this information, disallows communication between nodes which are out of range of each other. Communication is either allowed or forbidden based solely on distance information. No form of communication errors are emulated.

3 Wireless Ad Hoc Network Emulation

The basic idea of mobile wireless ad hoc network emulation is to test networking protocols, such as novel routing algorithms, as well as application software for wireless ad hoc networks by creating a software environment that permits unaltered deployment of basic kernel networking code (e.g. modified routing algorithms) and networked applications on each of the emulated nodes. An added advantage is that cross-layer interaction with the application layer (or any other layer above the MAC layer) can be examined and utilized. For example, figure 3 shows the layers of a network protocol stack and lists aspects in each layer for which cross-layer interaction could be required. A further benefit of emulator is that code written for the emulator can be run on real systems without the need for redesign.

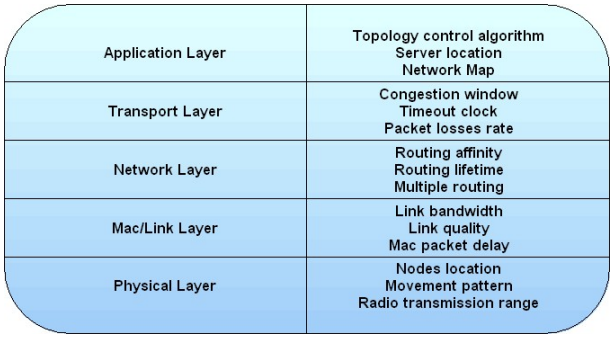

Figure 1. Cross-layer emulation possibilities

Packet flow between the emulated nodes can be controlled in various levels of granularity. The simplest form is to create or remove connections between pairs of nodes, thereby simulating mobile stations that are out of reception range of each other. A more elaborate mode extends this by modeling asymmetrical connections – i.e. packets flowing only in one direction due to differences in the emulated transmitting power available in the nodes involved.

These methods, however, only permit for a relatively coarse grade of emulation – a more elaborate mode of emulation includes not only the binary decision to send or withhold complete network packets, but also enables the emulation user to employ models of packet corruption, loss as well as duplication.

Traditional wireless network emulators use a number of real nodes connected via a wired network. The wireless aspect of the network is then approximated by allowing or disallowing the communication between certain nodes. An obvious drawback of these network emulators are the hardware requirements, as one emulator machine is required for every node. However, recent developments and extensions to operating systems like UML or microkernel-based systems allow the deployment of multiple virtual system instances on a single physical system.

4 System Architecture

In our system proposed for wireless ad hoc network emulation, we utilize the Fiasco [13] microkernel system which is a descendant of the L4 microkernel development [10]. Fiasco is a second-generation microkernel that concentrates on only implementing the most basic operating system functionality, leaving advanced aspects to server modules running in a non-privileged system mode. Thus, performance of the Fiasco-based system is greatly improved compared to previous microkernel-based systems like Mach [14].

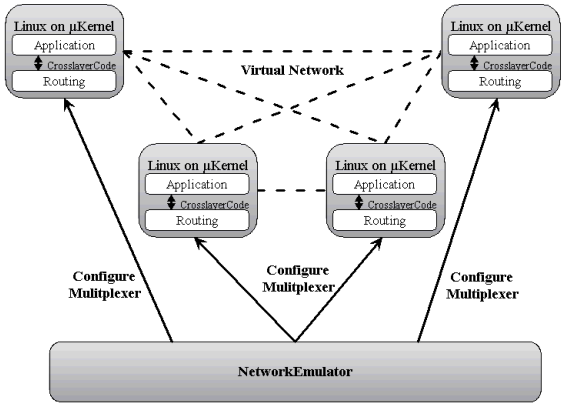

Figure 2. Microkernel-based network emulation

The structure of the system is shown in figure 4. On top of Fiasco, we run multiple instances of L4Linux [15], a special adaptation of the Linux 2.4 kernel that delegates hardware-related functionality to the microkernel and runs completely in user mode, permitting us to employ a standard Linux distribution as a basis for the emulation.

Since this approach leaves the implementation of device drivers to the Linux instances, we are able to create a wireless ethernet driver as a virtual device by replacing all low-level network related code of a standard WLAN driver with methods interfacing to the underlying Fiasco microkernel. Thus, for applications running on top of each Linux instance, the system looks exactly like a real Linux system connected to a 802.11b network. This is a great improvement compared to other simulation and even emulation systems, since the client code does not have to distinguish between real world and emulation. In addition, we are now able to implement a more realistic model of wireless communication by extending the simple binary decision matrix used in UML for switching connections on and off.

The interconnection of all virtual Linux instances is handled by a newly developed multiplexer component. The multiplexer connects to the virtual network driver in each Linux instance and can thus in turn control the packet flow between the instances. Packet flow can be controlled in various levels of granularity.

To connect the emulation system to a wired or wireless network the machine running the simulation is connected to, we employ a separate Linux instance. In contrast to the other running instances that each simulate one mobile device, this instance takes care of forwarding packets between the multiplexer and the physical network. As a consequence, that instance makes use of a driver for a physical (ethernet or 802.11) network card in addition to the virtual network adapter connecting it to the multiplexer.

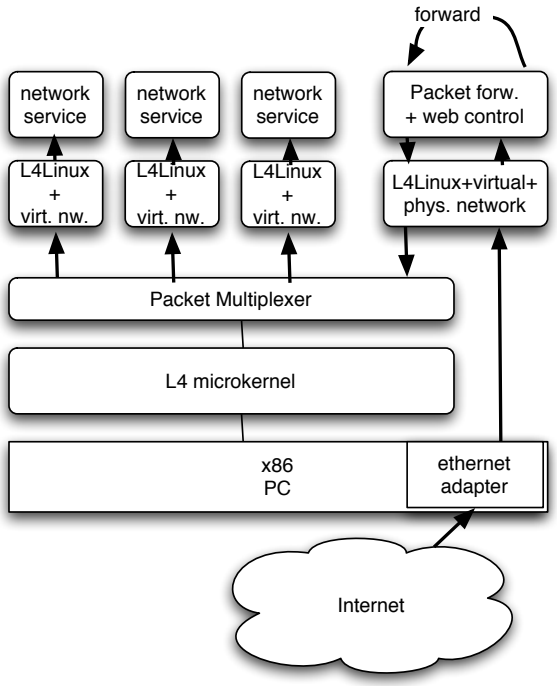

Figure 3. System structure

5 Experimental Results

The selection of the L4 microkernel as a basis for our emulation system has several advantages over alternative solutions. By using a microkernel-based virtual Linux system, emulation performance shows a significant gain compared to systems like UML. Compared to Linux instances running as UML processes, virtual Linuxes on top of L4 enjoy a greatly reduced context switching overhead and a greater flexibility in implementing virtual network devices by utilizing microkernel communication methods between running Linux instances on top of the microkernel. While the context switching overhead of UML instances compared to Linux running on the bare hardware involves a factor of up to 100 when system calls are involved [16], a typical L4-based virtual Linux instance only suffers from a context switch overhead factor of approximately 2 to 4.

This additional available computing time allows the special conditions of wireless ad hoc networks, like limited transmission range of the nodes, limited bandwidth, possible interference, limited computational and energy resources to be modeled more realistically than with UML. In addition, it is now possible to deploy a larger number of virtual Linux instances on a single machine, thereby increasing the number of nodes that can be simulated. While a UML system running on our test platform (a 1.8 GHz Pentium 4 system with 512 MB of RAM) could only sustain 3–4 UML instances at the same time, the L4-based system is capable of running about 8–10 instances simultaneously.

6 Conclusions

In this paper, we presented a novel method of emulating mobile wireless ad hoc networks by utilizing a set of Linux instances on top of a L4 microkernel. The central part of the emulation

system is a flexible multiplexer component that allows a modeling of wireless network connection conditions as well as error injection with varying levels of detail while permitting a greater scalability compared to a solution based on User Mode Linux.

Future work will include replacing the MobiEmu emulation master with a system employing dynamic temporal aspect-oriented technologies [17] that enable the user to create more flexible and precise definitions of timing dependencies in the emulation.

REFERENCES

[1] "NS2." Network Simulator NS2 `http://www.isi.edu/nsnam/ns/`.

[2] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: a Library for Parallel Simulation of Large-Scale Wireless Networks," in *Proceedings of the 12th Workshop on Parallel and Distributed Simulations, 1998*.

[3] "The OPNET Modeler," http://www.opnet.com/products/modeler/home.html.

[4] S. Toumpis and A. Goldsmith, "Performance, Optimization, and Cross-Layer Design of Media Access Protocols for Wireless Ad Hoc Networks," in *International Conference on Communication (ICC) 2003*, pp. 2234–2240, IEEE, 2003.

[5] W. Yuen, H. Lee, and T. Andersen, "A Simple and Effective Cross Layer Networking System For Mobile Ad Hoc Networks," *Proceedings of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 2002, pp. 1–5, 2002.

[6] A. K. Singh, V. Raisinghani, and S. Iyer, "Improving TCP Performance over Mobile Wireless Enviromentd using Cross Layer Feedback," *Proceedings on the IEEE International Conference on Personal Wireless Communications*, pp. 81–85, 2002.

[7] O. MacDonald and Y. Fang, "Cross Layer Performance Effects of Path Coupling in Wireless Ad Hoc Networks: Power Throughput Implications of IEEE 802.11 Mac," *Proceedings of IEEE International Performance, Computing and Communications Conference 2003*, pp. 281–290, 2003.

[8] S. Shakkottai, T. S. Rappaport, and P. Karlson, "Cross Layer Design for Wireless Networks," *IEEE Communications Magazine*, pp. 1–14, 2003.

[9] J. Dike, "A User-Mode Port of the Linux Lernel," *5th Annual Linux Showcase & Conference, Oakland, California*, 2001.

[10] J. Liedtke, "μ-Kernels Must and Can Be Small," *5th IEEE International Workshop on Object-Orientation in Operating Systems (IWOOOS), Seattle, WA*, October 1996.

[11] Y. Zhang and W. Li, "An Integrated Eviroment for Testing Mobile Ad Hoc Networks," *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 104–111, 2002.

[12] J. J. Lemmon, "Wireless Link Statistical Bit Error Model," *NTIA Report 02-394, U.S. Department of Commerce*, June 2002.

[13] M. Hohmuth, "The Fiasco Kernel: Requirements Definition," *TU Dresden Technical Report TUD-FI98-12*, 1998.

[14] M. Accetta, R. Baron, W. Bolosky, D. Golub, R. Rashid, A. Tevanian, and M. Young, "Mach: A New Kernel Foundation for UNIX Development," *Proceedings of the USENIX 1986 Summer Conference*, July 1986.

[15] H. Härtig, M. Hohmuth, and J. Wolter, "Taming Linux," *Proceedings of the 5th Australasian Conference on Parallel and Real-Time Systems, Adelaide*, 1998.

[16] S. T. King, G. W. Dunlap, and P. M. Chen, "Operating System Support for Virtual Machines," *Proceedings of the 2003 USENIX Technical Conference*, 2003.

[17] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-Oriented Programming," *Proceedings of the European Conference on Object-Oriented Programming, Springer-Verlag*, pp. 220–242, 1997.