

# A PARAVIRTUALIZED SCALABLE EMULATION TESTBED FOR MOBILE AD-HOC NETWORKS \*

PATRICK REINHARDT, OLIVER BATTENFELD, MICHAEL ENGEL, BERND FREISLEBEN

Department of Mathematics and Computer Science, University of Marburg

Hans-Meerwein-Str., D-35032 Marburg, Germany

{reinhardp, odb, engel, freisleb}@informatik.uni-marburg.de

## Abstract

*The evaluation and development of routing algorithms as well as applications for mobile ad-hoc networks (MANETs) often entails the use of a simulated or emulated testbed to run reproducible scenarios or to obtain otherwise infeasible measurements. While simulators are useful for optimizing protocols and algorithms using precise physical models, emulators permit the execution and interaction of unmodified software in a realistic environment. In this paper, we present a distributed emulation environment capable of creating a virtual MANET of hundreds or even thousands of nodes using only a small number of standard PCs. Each of the nodes runs an (almost) unmodified instance of the Linux operating system which allows us to use and develop standard operating system kernel extensions for ad-hoc support features, such as protocol and routing algorithm implementations as well as cross-layer channels. In addition, any Linux application software can be executed in a MANET environment.*

**Keywords:** mobile ad hoc networks, simulation, emulation, evaluation, virtualization

## 1 Introduction

Research in the field of mobile ad-hoc networks often involves the formulation of hypotheses about the characteristics or dynamics of the environment, such as hardware devices, software implementations, mobility patterns or even user behaviour. A tool commonly used in the process of verifying such hypotheses is a simulation or emulation system, since the use of real hardware and software or real users proves to be difficult in terms of both effort and costs as well as the possibility to obtain and reproduce certain measurements, such as the propagation of radiowaves.

Both approaches have their merits: simulators (such as ns-2 [3], GloMoSim [1], SSFNet [21] or OPNET [20]) are closed systems by nature, aggregating the software logic in a single component and thus allowing a global view on all aspects of the simulated scenario at any point of time. Conversely, this requires the adaptation of the test subject, such as

an implementation of a routing algorithm to a simulator's programming interface. Emulators (such as MobiEmu [23], ns-e [12], EMPOWER[19] or MINT [10]), on the other hand, have the ability to run unmodified software in a real environment, thereby permitting studies of actual live code both as a part of the operating system or user space which could also be executed on real hardware. It should be noted that this property creates the opportunity to investigate the efficiency and feasibility of exchanging information between various levels of a network stack. This so called cross-layer approach [13] often requires live interaction between the components involved.

An example of cross-layer interaction is the use of signal strength information or the number of re-transmissions or collisions etc. available on the data link layer to estimate the distance and speed of two nodes, to gain information about neighboring nodes or to divert traffic to other routers. Accordingly, there are a number of design challenges, such as the minimization of side effects of the emulation layer and scalability issues. Additionally, the development of software components, such as routing algorithms for MANETs, often involves modifications of the operating system. To be able to evaluate such implementations, it is desirable to have an isolated operating system instance per emulated node.

In this paper, we present *MarNET*, the Marburg Ad-Hoc Networking Emulation Testbed as a scalable testbed for the evaluation of application software and extensions for Linux-based operating systems running in a MANET using radio hardware conforming to the IEEE 802.11-standard. The evaluated software can be run on a real system without the need for code modifications which makes it especially suitable for cross-layer studies.

The paper is organized as follows. Section 2 gives an overview of the architecture of the emulation testbed. Section 3 presents the results of conducted experiments. In Section, 4 related work is discussed. Section 5 concludes the paper and outlines areas of future research.

## 2 MarNET Architecture

An overview of the MarNET testbed is shown in figure 1. The principal component is the Topology Manager, a distributed system to manage communi-

\*This work is financially supported by the Deutsche Forschungsgemeinschaft (DFG, SSP 1140, FR-791/7-2).

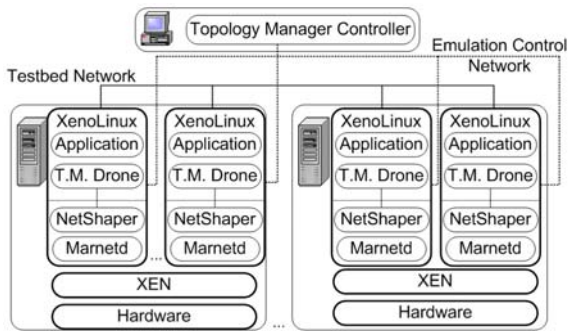


Figure 1: Overview of the MarNET

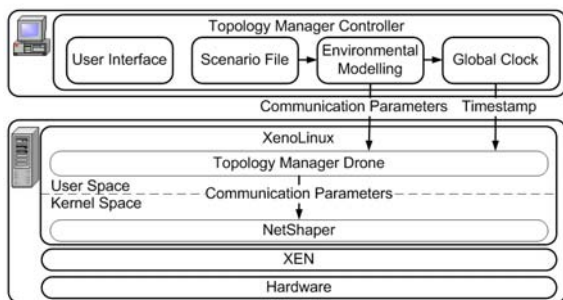


Figure 2: MarNET Topology Manager

cation links in the emulated network scenario. The Topology Manager Controller runs on a dedicated control PC communicating with the corresponding Topology Manager Drone on each emulated node through the emulation control network. Additionally, the NetShaper kernel module used to influence communication parameters, the MarNETd routing daemon and the application under test run on each emulated node. Using the Xen virtualization technology [6], several emulated nodes are run on a single physical computer.

## Topology Manager

The Topology Manager's main task is to govern the communication parameters of mobile ad-hoc networks on a wired testbed network. Additionally, it offers a global view of the emulated network during the emulation run with a visualization of node positions, movements and communication parameters. An architectural overview is shown in figure 2, a visualization view is illustrated in figure 3.

The Topology Manager consists of two components, the *Controller* and the *Drone*. The Controller parses a scenario file containing locations and movement patterns of the participating nodes. It has a global view on the emulated network and offers the global emulation time to the connected Drones which are started on every system used as a network node. Using the Environmental Engine, the communication parameters are calculated and announced to the

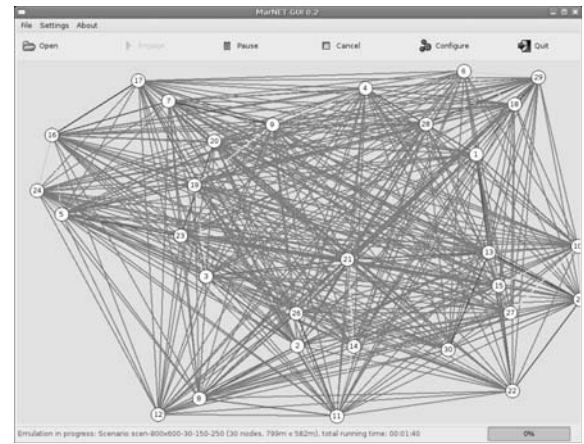


Figure 3: Visualization in the Topology Manager

Drones: Due to the limited range and the high mobility of network nodes, the transmission quality in MANETs is highly fluctuating when compared with other network types [9, 18]. This dynamic behavior is emulated by triggering actions through scenario timestamps sent by the Controller. After receiving a timestamp, each virtual node – respectively each Drone – has to set the communication parameters calculated by the MarNET Environmental Modelling Engine. To achieve this, the Topology Manager uses NetShaper, a tool developed by Herscher and Rothermel [15] designed to influence communication parameters, such as packet loss and delay of a network link.

In comparison to emulation testbeds that model wireless links with binary states (*reachable* vs. *not reachable*), MarNET's Environmental Engine supports communication parameters with a higher granularity. This enables us to analyze the effect of lost and corrupted packets on the performance and stability of a given application.

It is possible to switch between the different link state models at runtime. Currently, two models are implemented using the MarNET API. The *Pisa Model* is based on measurements conducted at the University of Pisa and the Istituto IIT [5]. The second model, the *Marburg Model*, is the result of experiments conducted at the University of Marburg.

The measurements leading to the Marburg Model were conducted using two notebooks running Debian GNU/Linux. One of the notebooks was set up to send groups of 1000 UDP broadcast packets of 1500 byte size. Since packet loss in broadcast UDP packages is not handled, it was possible to count the received packages and to calculate the loss ratio. Since distance is only one of the possible values influencing packet loss, these measurements are possibly affected by side effects, such as (a) interferences of other devices especially 802.11 networks and bluetooth appliances emitting electromagnetic waves, and (b) the multi-path problem where deferred data fragments arrive at the receiver and possibly destroy data. For a detailed discussion of these

side-effects see [4].

When using the Marburg Model, the Environmental Engine calculates the communication parameters as follows: After parsing the scenario file, the positions of all nodes and the distances between all node pairs are calculated at regular timestamps. Based on both distances and the signal attenuation caused by obstructions on the playing field, the signal levels for each node pair are computed. Finally, the packet loss ratio is derived from the signal levels.

To avoid interference between control messages and the emulation network, the messages are sent using a separate emulation control network. Scalability is achieved by multicasting the control messages, the generated traffic is thus independent of the number of listening Drones. Hence, the number of nodes is not limited by the bandwidth.

## Xen

Our testbed uses of the the Xen hypervisor [6] to execute several independent virtual machines (VMs) in parallel on a single computer.

The paravirtualization approach of Xen requires the guest operating system to be slightly modified to enable the VMs to communicate directly with the physical hardware of the host system. This reduces the virtualization overhead and significantly improves the performance.

Xen was chosen due to multiple reasons: Measurements conducted by Barham et al. and Clark et al. show that the performance of XenLinux is virtually identical to the performance of a standard Linux system [6, 8]. Additionally, the application binary interface does not require to be changed. Hence, the applications developed and researched on the emulation testbed can be used on real computers (*live code*). We also consider isolation between two virtual nodes to be a major concern which is met by Xen since communication is only possible through the emulated network. Furthermore, Xen is soon to be integrated into the standard Linux kernel.

According to [6], Xen is designed to run about 100 virtual machines. The measurements conducted to verify this statement are addressed in section 3.

## Network Topology

Two separate networks are used in the testbed. The control network is basically used as the Topology Manager control channel. Additionally, virtual nodes utilize this network to access a common directory on a dedicated file server.

The testbed network – being the actual emulated network – is exclusively utilized by the test subjects. Its communication parameters are adjusted during the emulation by the Controller. Using a Linux software bridge, a network is created containing all MarNET nodes which then share a single virtual Ethernet medium. The testbed is easily extensible by adding additional (physical) machines to the

medium, i.e. the bridge. Emulated nodes may be migrated to other physical hosts at runtime, thus providing a scalable and flexible solution.

## Routing Daemon

A routing algorithm or strategy is crucial to the operation of a MANET since it provides local information about working communication paths and reachable nodes. Users of the emulation testbed are thus either developers or users of an implementation of such an algorithm. The latter is the case when the behavior of user level application software communicating over multiple hops in the virtual ad-hoc network is examined. Consequently, we have developed an extensible routing daemon<sup>1</sup> software [7] which specifically addresses the needs of either roles. Contrary to other routing software designed for MANETs, it encapsulates the algorithm implementation in a module, thus allowing adaptive switches between multiple implementations during runtime. This capability is especially useful when evaluating scenarios in which the participating nodes have different or multiple roles either simultaneously or over time: A network between various mobile devices of walking pedestrians, for instance, differs greatly from a network of moving cars in terms of mobility patterns, energy constraints, CPU power or radio equipment. Furthermore, there are interdependencies between these factors: For example, increasing the transmission range of a device has an adverse effect on its energy supply and possibly limits the overall network bandwidth since it now interferes with the medium access of nodes previously not in range [11, 22].

Our routing daemon provides a tool to evaluate the efficiency and feasibility of runtime algorithm changes, either adaptively or through user interaction as well as concurrent instances of multiple different algorithm implementations. In addition, it provides a common interface for any application software to access and manipulate algorithm parameters (such as beacon intervals or route expiration intervals) of any active routing module. This also provides the means to inject or extract information from the routing layer to effectively support cross-layer software development using a defined interface.

Routing modules developed for our routing daemon can be used on real hardware without any additional development effort (i.e. program code changes<sup>2</sup>).

## 3 Experimental Results

A series of experiments has been conducted to evaluate the emulation testbed. As one test system, a 2

<sup>1</sup>Daemon is the "Unix term" for a persistent system service.

<sup>2</sup>Recompiling the source code might be necessary, however, depending on the target platform.

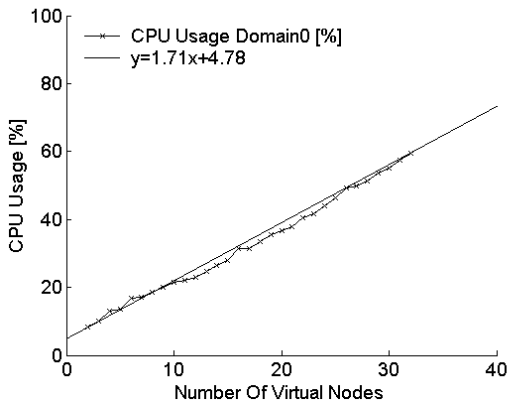


Figure 4: CPU usage as a function of the number of virtual xen machines.

GHz AMD Opteron 246 with two Broadcom NetXtreme BCM5704 Gigabit Ethernet adapters and 2 GB main memory was used. 1 GB of memory is reserved for the Xen host domain (“Domain-0”) and 16 MB are allocated for each virtual node. The second test system is an Intel Pentium IV 3 GHz with Intel 82547EI Gigabit Ethernet controllers and 1 GB main memory. 400 MB of the memory is allocated for Domain-0, each virtual machine uses 16 MB. The bandwidth of the testbed links was limited to 11 Mbps being the nominal data rate for IEEE 802.11b.

## CPU Usage

An important scalability criterion of an emulation testbed is the host system’s CPU usage as a function of the number of virtual machines. These measurements were conducted on the Opteron system. The acquired values are mean values of 100 measurements sampled once per second.

The readings shown in figure 4 show the following correlation between the host CPU usage and the number of virtual machines:

$$CPU_{host} = 0.0171n + 0.0478 \quad (1)$$

Thus,  $CPU_{host}$  is linear in the number of virtual nodes. More precisely, each virtual node uses about 1.71% of the host CPU. The applications running on the virtual node include the MarNETd [7] routing daemon, the Drone and a ssh daemon.

## Round Trip Time

In emulation environments, the round trip time (RTT) is an important performance concern when evaluating the time a single packet takes to be transmitted over the network. The measurements were conducted in two phases: During the first test run, the RTT was measured by sending packets from one node to another whereas both nodes were located on the same host, see figure 5. The setup of the second run put the nodes on different physical machines.

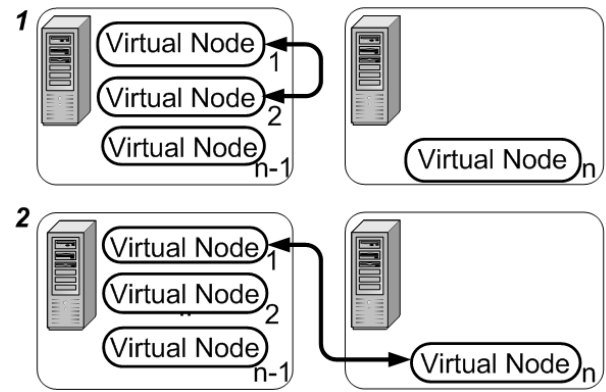


Figure 5: Measurement setups for round trip time and throughput evaluation.

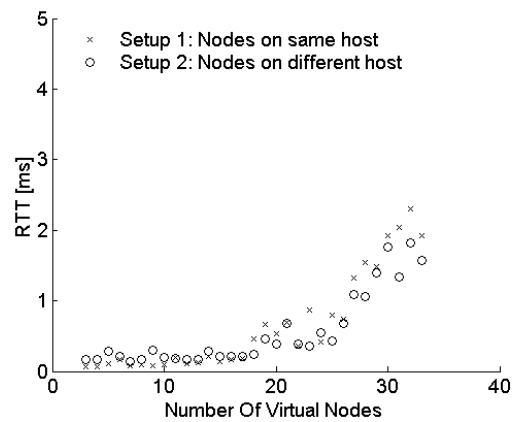


Figure 6: Round trip time as a function of the number of virtual nodes.

Figure 6 shows the round trip times versus the number of nodes. In both setups, the RTT grows with the number of virtual nodes, whereas the maximum value is at 32 nodes using setup 1. In this case, the RTT is 2.298 ms. The average difference between setup 1 and setup 2 is 0.098 ms. We conclude that in the MarNET topology, the influence of the wired network between physical nodes is minimal with respect to the round trip time.

## Throughput

Apart from the round trip time, the achievable throughput is a major concern in network performance evaluation. Hence, the achievable bandwidth between two virtual nodes in the testbed was evaluated. Iperf [2] was used to take 10 TCP throughput samples with a total duration of 100s.

Figure 7 shows the measurement results. Since the bandwidth was limited to 11 Mbps, the values are near this maximum. Again, the difference between setup 1 and setup 2 is minimal, averaging at 0.010 Mbps.

In 2005, Menon et al. conducted measurements

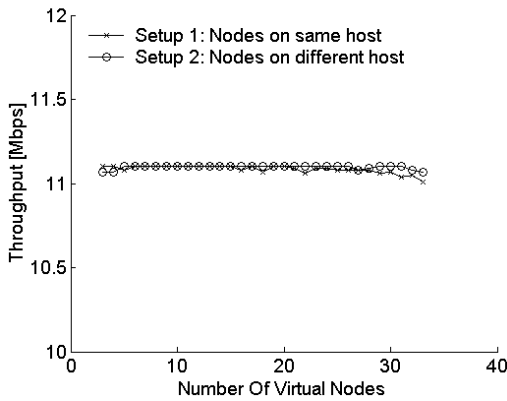


Figure 7: TCP Throughput as a function of the number of virtual nodes.

investigating the network performance in Xen [17]. It was discovered that Xen suffers from the degradation of network throughput due to the CPU being the bottleneck. Our measurements indicate that MarNET does not suffer from these problems since it limits the network bandwidth to the nominal data rates of IEEE 802.11b/g which can be easily handled by modern commodity hardware.

## 4 Related Work

MobiEmu by Zhang and Li is an emulation tool which emulates a MANET on a set of computers connected through a wired network [23]. A master component calculates node movements based on scenario files, deciding whether two virtual nodes are able to communicate with each other. Ethernet communication is governed by Netfilter firewall rules set to block or allow packets originating from a node not currently in range. A drawback is its emulation approach which only knows binary link states, i.e. loss-free communication or no communication at all.

Herrscher et al. [14] proposed an emulation environment based on a cluster consisting of 64 computers connected over two independent networks, the emulation network and the administration network. A dedicated control PC manages communication between the nodes using an XML control file which includes network parameters such as delay, bandwidth limitation and packet loss. Meier et al. [16] describe the environment's extension by a virtualization solution. According to the paper, a virtualization factor of 30 is possible. The drawback of this solution is that several instances of an application are running on a single physical node each using a different virtual network interface without any isolation from each other. Furthermore, the testbed cannot be used to evaluate implementations running in the operating system kernel such as kernel routing extensions.

Fall [12] suggested an emulation facility for the NS Simulator. This extension offers the ability to pipe traffic originating from physical nodes through

the simulator. Hence, it is possible to evaluate code developed as a NS module using real world traffic. One of the drawbacks of this solution is the performance. According to the authors, round trip times average at 2000 ms. Additionally, the centralized NS simulator is likely to become a scalability bottleneck.

EMPOWER [19] is an emulation system developed by Ni and Zheng. In this testbed, emulator nodes are the main components. These nodes use several 4-port network devices connected to a hardware switch and virtual devices enabling to interfere with packet loss and introduce transmission delays. The need for one physical network interface per emulated node limits the number of virtualized nodes to a maximum of 24 nodes on commodity hardware, thus limiting scalability.

In [10], MINT, a testbed using 802.11 compatible network devices, has been proposed. Using attenuators, the transmission power is significantly reduced which allows the testbed to be set up inside a room. Due to the presence of real wireless communication, the emulation is very accurate. However, the drawback of MINT is the need for one physical computer as well as a robot per emulated node making large-scale testbeds very expensive.

## 5 Conclusions

In this paper, we have presented MarNET, a test environment for mobile ad-hoc networks where scalability and resource isolation are major concerns. MarNET emulates a mobile ad-hoc network on a wired network by computing and influencing the communication parameters on the testbed nodes. The use of XenLinux allows to run multiple virtual nodes on a physical one with little side effects. Each virtual node uses its own operating system instance allowing researchers to use existing Linux code without the need to modify the software. Since Xen isolates the virtual nodes, the only way to communicate with other nodes is the emulation network. The measurements that have been conducted to evaluate the performance of MarNET indicate that on the given hardware, it is possible to run about 30 virtual nodes on a single physical system. At 30 virtual nodes, the round trip time inside the network is under 2 ms. The adjusted bandwidth limit of 11 Mbps for 802.11b remained constant during the whole experiment.

MarNET is currently tested on the hardware of the 84 node cluster of dual core dual Opteron systems at the University of Marburg. Assuming the values determined on the Opteron core, a testbed on this system will be able to support at least 2550 virtual network nodes. Enhancements of the current physical model especially with respect to the shared medium will be analyzed to improve the emulation quality. The development of the shared medium model will be evaluated by creating a centralized dispatcher capable of recreating the wireless collision domains at runtime using a clustering algorithm to

maintain the current scalability.

## References

- [1] GloMoSim Scalable Network Simulator. <http://pcl.cs.ucla.edu/projects/gloMosim/>.
- [2] Iperf Bandwidth Measurement Tool. <http://dast.nlanr.net/Projects/Iperf/>.
- [3] Network Simulator NS2. <http://www.isi.edu/nsnam/ns/>.
- [4] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. *SIGCOMM Comput. Commun. Rev.*, 34(4):121–132, 2004.
- [5] G. Anastasi, E. Borgia, M. Conti, and E. Gregori. IEEE 802.11 Ad Hoc Networks: Performance Measurements. *Cluster Computing Journal, Special Issue on Ad Hoc Networks*, 8:135–145, 2005.
- [6] 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 *SOSP '03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.
- [7] O. Battenfeld, M. Smith, P. Reinhardt, T. Friese, and B. Freisleben. A Modular Routing Architecture for Hot Swappable Mobile Ad hoc Routing Algorithms. In *Proc. of the Second International Conference on Embedded Software and Systems, Xian, China*, pages 359–366. Springer-Verlag, 2005.
- [8] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J. N. Matthews. Xen and The Art of Repeated Research. In *In Proceedings of the Usenix Annual Technical Conference, Freenix Track.*, pages 135–144, 2004.
- [9] S. Corson and J. Macker. RFC 2501: MANET Routing Protocol Performance Issues and Evaluation Considerations. <http://www.ietf.org/rfc/rfc2501.txt>.
- [10] P. De, A. Raniwala, S. Sharma, and T. Chiu. MiNT: A Miniaturized Network Testbed for Mobile Wireless Research. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2731–2742. IEEE, 2005.
- [11] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris. Effects of Loss Rate on Ad Hoc Wireless Routing. Technical Report MIT-LCS-TR-836, MIT Laboratory for Computer Science, 2002.
- [12] K. Fall. Network Emulation in the VINT/NS Simulator. *Proceedings of the Fourth IEEE Symposium on Computers and Communications*, pages 244–250, 1999.
- [13] A. J. Goldsmith and S. B. Wicker. Design Challenges for Energy-Constrained Ad Hoc Wireless Networks. In *IEEE Wireless Communications*, volume 9, pages 8–27, 2002.
- [14] D. Herrscher, A. Leonhardi, and K. Rothermel. Modeling Computer Networks for Emulation. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1725–1731, 2002.
- [15] D. Herrscher and K. Rothermel. A Dynamic Network Scenario Emulation Tool. In *Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN 2002)*, pages 262–267, Miami, October 2002.
- [16] S. Maier, D. Herrscher, and K. Rothermel. On Node Virtualization for Scalable Network Emulation. In *Proceedings of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pages 24–28, 2005.
- [17] A. Menon, J. R. Santos, Y. Turner, G. Janakiraman, and W. Zwaenepoel. Diagnosing Performance Overheads in the Xen Virtual Machine Environment. In *VEE '05: Proceedings of the First ACM/USENIX International Conference on Virtual Execution Environments*, pages 13–23, New York, NY, USA, 2005. ACM Press.
- [18] P. Mohapatra and S. Krishnamurthy. *Ad Hoc Networks - Technologies and Protocols*. Springer, 2005.
- [19] L. Ni and P. Zheng. EMPOWER: A Network Emulator for Wireline and Wireless Networks. *IEEE InfoCom 2003. San Francisco.*, pages 1933–1942, 2003.
- [20] OPNET - Optimal Network Performance. The OPNET MODELER. <http://www.opnet.com/products/modeler>.
- [21] Scalable simulation framework. <http://www.ssfnet.org>.
- [22] S. Toumpis and A. J. Goldsmith. Performance, Optimization, and Cross-Layer Design of Media Access Protocols for Wireless Ad Hoc Networks. In *International Conference on Communication (ICC) 2003*, pages 2234–2240. IEEE, May 2003.
- [23] Y. Zhang and W. Li. An integrated environment for testing mobile ad-hoc networks. In *MobiHoc '02: Proceedings of the Third ACM International Symposium on Mobile Ad-Hoc Networking & Computing*, pages 104–111, New York, NY, USA, 2002. ACM Press.