A STANCE ON EMULATION AND TESTBEDS, AND A SURVEY OF NETWORK EMULATORS AND TESTBEDS

Erek Göktürk
Department of Informatics
University of Oslo
Blindern, N-0371 Oslo, Norway
Email: erek@ifi.uio.no

Keywords— **emulation, testbeds, network emulation, network testbeds**

Abstract— **Emulations and testbeds are popular tools in networking research. However, what differentiates testbeds from emulations is not terminologically clear. In this article, we first discuss the terms emulation and testbed, and present the reflexivity principle for identifying testbeds. We then present a survey of forty two network emulators and testbeds reported in the literature, categorizing them according to the approaches used. Another eighteen that can be added to this survey as future work are also identified.**

I. Introduction

Simulation is an important tool for networking research. However, concerns about accuracy or computational load of models used in simulation have lead researchers to search for ways of combining real and virtual, or to conduct experiments directly on the system under test (SUT), in those cases where it appears feasible.

While there are considerable number of systems that involve real elements of SUT in the experiment setup in one way or another, the terms testbed and emulation appear to be used by different researchers with different meanings. In this paper we first attempt to discuss what testbeds and emulators are in Section II. In that section, we introduce the reflexivity principle for identifying testbeds, and elaborate on what it means to use some entities as real in an emulation-based experiment. This discussion prepare the grounds for our survey.

Then in Section III, we present our survey of forty two network emulators reported in literature, and identify the main approaches used by these systems. Finally, we conclude in Section IV.

II. STANCE

A. Testbeds

Before we move on to discuss what emulation is, we should first define testbeds as they lie on the opposite end of the spectrum from the simulation, in which everything is virtual.

The distinguishing property of a testbed is that it satisfies what we will call the *reflexivity property*[1]: a testbed itself is a perfectly normal instance of the system that is under study (SUT) in a particular experiment, which is used for meeting various experimental objectives such as collecting data to be interpreted for obtaining indicative results for the SUT. The testbed might be a smaller scale example of the same kind of system as SUT, but it might not be an all-aspects-scaled-down version of it. For example, a token ring network of five nodes can be used for developing an application targeted for token ring networks. Then, the application would later be deployed onto a token ring network of hundred nodes.

An important point that should always be paid considerable attention in any testbed based experiment is careful evaluation of in which dimensions the test results can be regarded as indicative for the properties of the SUT. The results derived from testbed based experiments are not always indicative in all dimensions. Scalability is a good example of the dimensions testbeds are not very useful for deriving indicative results about. Following the example of five host token ring network in the previous paragraph, such a testbed can rarely be indicative of whether the protocols used in the application under test will scale to networks of bigger size, unless one can replicate the situations of interest in the big target network using just five nodes.

B. Emulation

In an emulation-based experiment, the system under test (SUT) as defined in the experiment is represented by a combination of one or more surrogate systems, or in other words simulators, and parts of the SUT used as real in the experiment setup. Therefore, the level of abstraction of representation of SUT in the experiment setup differs between different parts of the experiment setup, depending on whether the part is used as real, or simulated, and even among the different parts that are represented as being simulated.

Providing an exact definition of what it means to be used as real in the experiment setup is not always straightforward. We are going to consider composite SUTs composed of entities, which can be partitioned into parts, which are sets of entities in SUT. We will refer to the system in a particular experiment setup that represents the SUT of that particular experiment as a *stand-in for the SUT* (SIFSUT). Then the property of being used as real can be defined as follows: An entity E that is a part of SUT in a certain experiment is considered to be used as real, if E is used in the SIFSUT as it appears in SUT, for providing the exactly same function it would provide in SUT. This definition can easily be extended to parts, which are sets of entities.

Therefore another question to ask in order to find out whether an experiment setup is emulation-based or testbed-based is this: what is simulated in the SIFSUT

[1]Olivier Dalle has been the one to put a name to the property when the author was explaining the concept to him in private conversations.

Proceedings 21st European Conference on Modelling and Simulation
Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007
ISBN 978-0-9553018-2-7 / ISBN 978-0-9553018-3-4 (CD)

for this experiment? The answer depends not only on the properties of the experimental setup, but also on the definition of the SUT for the particular experiment. Therefore the same setup may be considered as a testbed for one experiment, while it might be more proper to call it an emulation for some other experiment. This creates a considerable confusion about the difference between testbeds and emulation-based experiment setups.

For a simulator to be used in emulation, it has to support the same modes of interaction that the original system uses in order to interact with other systems in the situation of interest. Such a simulator that sufficiently supports the modes of interaction of the original system for the purposes of a particular emulation-based experiment, is generally called an emulator.

To give an example, take a flight training simulator. A flight simulator is mainly an emulator, since it act as the surrogate for a plane, and interacts with the pilot in many ways in the same mode as a plane would interact with in a real flight situation. That is, it "interacts realistically", except perhaps for the gravitational effects.

Flight simulation is also a good example that demonstrates the role of the key-phrase "sufficiently" in our definition of an emulator. For example, the Microsoft's Flight Simulator interacts with the player realistically using the same mode of information a plane would, such as showing a graphical representation of an altitude-meter that matches the plane whose flight is being simulated. If a situation of interest can be defined over such items that the Flight Simulator provides a realistic mode of interaction for, the Flight Simulator could be said to provide sufficient support for the modes of interaction of interest in the experiment, and so can be referred to as an emulator in that specific experiment.

Emulation is perhaps most popular in training applications such as medical training especially for surgeons, pilot training in avionics, or training of the nuclear power plant personnel. This provides realistic training environments without risking expensive equipment, or more importantly lives.

Games are another another example of popular application of emulation. As an example, consider the 1^{st} person shoot'em'up games try to realistically interact with the user by providing a realistic simulation of the point of view from the character's eyes.

Another popular example is so called "emulators", which are software applications that act as if they are a particular type of machine or operating systems. To give examples, there exists "emulators" of Commodore Amiga, ZX-Spectrum, and Atari platforms, and for operating systems like Palm Os.

Time is an interesting entity that is part of SUTs in many experiments. Although running in real-time is considered to be a characteristic property of emulators by many researchers, time is only another part of the reality that in theory can be simulated in an emulation based experiment setup. A simple example is provided by the various computer system emulators mentioned in the previous paragraph, where time appears already abstracted as timing signals, and they can be made to run slower or faster than real-time, while still carrying the properties of an emulator since they continue to interact with (run) real programs.

III. SURVEY OF NETWORK TESTBEDS AND EMULATORS

With respect to the discussion in the previous section, network emulation is characterized by use of real parts of the network defined as the system under test (SUT), along with simulators for constructing the stand-in for the system under test (SIFSUT).

We should point out that discussions about network emulation that can be found in the literature agrees with our stance, in that it is generally recognized that emulation involves real elements used along with simulators. However, additional properties are usually required. For example, Guruprasad et al. presents running in real time also as a defining property of emulators [1]. As discussed at the end of the previous section, we do not agree that this should be taken as a defining property. Nicol et al. also takes running in real time as a defining property [2]. They further classify emulators used for networking research into network emulators and real-time simulators, based on whether time-stepped (which they call time-based) or event-driven approach to time management is used for the simulation in the SIFSUT. Although we agree that time-stepped and event-driven approaches have their own set of advantages and disadvantages when used in an emulation-based experiment setting, which justifies differentiating between the two, their naming is quite confusing: both network emulators and real-time simulators as they define, end up used in emulation-based experiment setups.

In the survey below, we will start with the very few systems that we consider as testbeds according to our stance. Then we will look at the emulation systems reported in the literature, starting from systems that have mostly real elements, and then move more or less gradually towards systems that are mostly simulated. This division is not introduced as a proper categorization, but just to make presentation a bit more tidy: it is easier to discuss what is being simulated for the mostly real emulation systems, and what is being used as real for the mostly simulated ones.

A. Network Testbeds

The systems that would satisfy the reflexivity property presented in Section II-A, therefore which would be called testbeds, are very rare. Ionescu et al. describes NIST*net2, which provides network resources available to the researchers [3]. It is built by connecting dedicated networks in four institutions in Canada. CREATE-NET is a network installation in a rural part of Italy in the autonomous province of Trento. It has both wired and wireless parts. The goal of CREATE-NET is providing a network where researchers can experiment with networked communities. Another testbed is MIT RON (Resilient Overlay Network) [4]. It is a set of hosts distributed over the Internet (36 in 31 different cites as reported in [4]), available to the researchers to use by acquiring an account. It is similar to PlanetLab, however it is not open to every-

one, therefore security is not actively addressed. All hosts run a normal FreeBSD installation.

B. Network Emulation – Mostly Real Systems

In certain emulation environments reported in the literature, almost nothing is explicitly simulated, therefore being considered as a testbed or an emulation environment depends on the definition of the SUT in a given experiment to be conducted using them. MIT Roofnet is a good example, where real changing conditions for the wireless channel between the nodes distributed on the roofs of some buildings near the MIT campus is used for testing routing in wireless networks [5]. PlanetLab is very similar to MIT RON in the sense that it is made up of some number of hosts distributed over various sites all over the world, while the definition of the concept of slides introduces virtuality for the hosts as used in the experiments, which prevents its categorization as a testbed according to our stance [6].

In the CMU DSR experiment [7], Maltz et al. uses the changes of the topology due to mobility of the nodes which are mounted on cars in the campus, as a surrogate for topology changes due to mobility in mobile ad hoc networks (MANETs). In Ad hoc Protocol Evaluation testbed (APE), which is not a testbed according to our definition of it but an emulation, the style in CMU DSR in simulating topology and mobility is taken one step further and the movements of the people carrying laptops are explicitly choreographed and controlled [8], [9].

There is a good reason why the emulations related to wireless networks mentioned up to now are trying to use real physical conditions as surrogates for the physical conditions in the SUT of the experiments: the physical channel is hard and computationally intensive to simulate accurately. However, it turns out that there is also a need for simulating physical channel using the physical conditions in the emulation in a more controlled way. In the Illinois Wireless Wind Tunnel (iWWT), an anechoic chamber is set up which provides isolation of the experiment environment from outside RF interference, at the same time providing an anechoic enclosure [10]. The topology is then scaled down by adjusting transmitting power of the devices put into the iWWT, and background noise is added explicitly as needed. Similar methods for creation of wireless channel effects are also used in ORBIT [11], [12], [13]. However, the positions of the nodes are fixed in ORBIT on a 20x20 matrix, while in iWWT the researchers use small mobile robots.

MiNT (Miniaturized Wireless Network Testbed), which is again not a testbed according to our definition of the term, is another system that uses the miniaturization, or scaling down, in order to make use of physical radio communications in emulating MANETs. In MiNT, radio signals attenuators are attached to the wireless devices, thereby reducing their transmission range [14]. Then the nodes are mounted on small mobile robots that are remotely controlled, as in iWWT. An interesting property of the MiNT is that it can work with NS [15], in such a way that the physical layer is simulated using MiNT. However in that case, MiNT becomes a complete simulator, and ceases to have any entities used as real unless the SUT involves mobile robots with signal attenuators attached.

In addition to iWWT and MiNT, other researchers also have looked into using attenuators for controlling the wireless channel effects. Kaba and Reichle works with unmodified computers and network interface cards (NICs), and attempt to build an environment where wireless signal propagation is either attenuated, or guided through cables between the communicating NICs [16]. Judd and Steenkiste also capture the signals at the antenna, and uses an FPGA based digital signal processor to attenuate signals between the transmitting station and the receiving stations [17], [18]. Their method makes it more possible to be able to repeat wireless physical layer effects across simulation runs, while still using unmodified NICs.

Another focus in building systems to be used for emulation-based experiments is the experiment control in order to allow multiple researchers to share the resources available. This problem have been targeted in many different systems, such as ORBIT, PlanetLab, and MIT RON. Some of the projects only aim for providing a set of resources for experimenters to build their own emulation-based experiment setup, such as Embedded Wireless Modules (EWM) [19], or the UCLA HNT [20].

C. Network Emulation – Mostly Simulated Systems

Some protocols have complex implementations, which makes re-implementing them for a simulation prone to errors and inaccuracies. For this reason, various researchers have taken the real protocol implementations in open source operating systems, and packed them in a way that they can be incorporated in simulators. In ENTRAPID, network stack from the FreeBSD is packed in a way to work in the user level, so that the protocol developers can experiment with their own protocols incorporated into the stack [21]. The topology and the physical layer in ENTRAPID is simulated, and the processes running on the simulated nodes need slight modification. Ely et al. also have worked on the same problem, and they have converted the FreeBSD 3.3 protocol stack to work as a library in the user space [22]. Zec and Mikuc modifies the protocol stack of 4.4BSD operating system in order to allow for multiple independent instances of the stack to exist in the kernel, connected via simulated links [23]. Jansen and McGregor have packed network protocol stacks in Linux, FreeBSD, and OpenBSD as shared libraries, and implemented an NS agent that is capable of using these stacks [24]. They call their approach the Network Simulation Cradle, and say that their approach can be used with other simulators as well. Bless and Doll uses OMNeT++ [25] instead of NS, and incorporate the TCP/IP stack from FreeBSD as a simple model in OMNeT++ [26]. They address synchronization of the kernel timers that are used by the protocol stack, with the virtual time in OMNeT++. Furthermore, the function calls to the socket library are represented by messages to be received by the simple model they have developed. Bavier et al. uses a different approach from the ones mentioned, where they implement Click modular routers in slices on PlanetLab hosts, and construct the

network that is the stand-in for the SUT as an overlay on PlanetLab [27].

In some of the systems that allow use of implementations of protocols as real, it is more difficult to decide whether the experiment setup constructed using these systems are emulation-based, or pure simulation. The reason is that these systems use unusual protocol implementations, at the same time pointing out that they can be used in real systems as well. The JEmu system builds on a four layer protocol stack for MANETs, and at the lowest layer which corresponds to the radio communications, the frames are forwarded to a physical layer simulator running on a different host [28]. In [29], Karrer et al. incorporates into NS protocols that are implemented as Click protocol graphs used by the Click modular router [30].

A different approach is using real hosts whose traffic is routed through virtual networks. The emulation extension of NS is a typical example [31]. NS is monolithic, but since NS emulation extension simulates whole networks, it is possible to partition the SUT into different networks and assign them to a set of simulators running on different hosts, as done in EmuLab [1]. While NS is monolithic, there are also distributed simulators used in emulation-based experiment setups, such as IP-TNE. IP-TNE is built on IP-TN, which uses CCTKit that implements the Critical Channel Traversing (CCT) algorithm for parallel discrete event simulation [32], [33], [34]. In [35], Bradford et al. discusses different methods for reading packets from and writing packets to real networks for network emulators such as IP-TNE. Another system that uses a discrete event based simulator is RINSE [36], which is built on iSSF (formerly known as DaSSF). RINSE uses what Liljenstam et al. call "multiresolution modeling", which means that background traffic is simulated using fluid models that require less resources to simulate, while the traffic of interest —the foreground traffic— is simulated at the packet level. The target application area of RINSE is network attack preparedness exercises, and therefore it includes some models that are not normally found in network simulators, such as CPU models. In ModelNet, the environment is divided into two sets of hosts called core nodes and edge nodes [37]. The network in the SUT is modeled as a set of pipes, which are assigned to the core nodes. The core nodes then cooperate to subject the traffic to the bandwidth, congestion constraints, latency, and loss profile of the target network topology. The edge nodes are the real hosts whose traffic is routed through the virtual network. While ModelNet is targeted for wired IP networks, MobiNet is an extension of the same approach but it targets MANETs [38]. In addition, MobiNet allows for multiplexing of virtual nodes on the edge nodes.

Another popular approach is the use of traffic shapers, which are placed between the protocol stack and the network device driver in kernels. This way, the protocol stack and the programs running on it are used as real, while the rest of the network is simulated. For example, in NETShaper, flow parameters such as bandwidth and delay are controlled by a user-space program [39], [40]. A similar approach is taken in EMPOWER [41], which targets wired IP networks, and EMWIN [42], which is based on EMPOWER but it targets MANETs. At the extreme case of traffic shaping, it is possible to simulate presence of connection between nodes in the network with only existence and non-existence of links. As an example, MNE (Mobile Network Emulator) is a distributed system which abstracts away physical layer effects and mobility behind topological changes simulated by IPTABLES based packet filtering controlled from a central controller [43].

An alternative to using traffic shaper modules in the kernel is the use of the universal TUN/TAP driver, which is designed for implementing tunneling using user level programs. NCTUns [44], EmuNet [45], and DINEMO [46] use the TUN virtual network interfaces, which intercept packets after the IP protocol implementation. NEMAN [47] is a similar system, but it uses the TAP virtual network interfaces, which intercepts frames before they are handled to the network driver to be sent to the network. Of these systems, NCTUns and NEMAN are monolithic, while EmuNet and DINEMO are capable of being distributed. In all of these systems, the protocol layers above and including IP, along with the programs communicating over the network, are used as real. Considered from software engineering perspective, DINEMO has the added advantage of being supported by a simple component model designed for simulators and emulators.

With the developments that allow multiple operating systems running on one base operating system, another approach have recently become possible. User Mode Linux (UML) provides a virtual Linux kernel running in user mode. UML have been used for implementing virtual nodes that are then connected by a network simulated below the network driver layer. vBET [48], which targets wired networks, and the system developed by Guffens and Bastin [49], which targets MANETs, are examples to using UML. Using micro-kernel based approaches have also been attempted, as exemplified by the work done by Engel et al. [50], which targets wireless networks.

While it may not be feasible for MANETs or wired networks, simulation of all hardware including the CPU so that unmodified programs can be run, appears to be a feasible technique for emulation-based experiments for sensor networks. ATEMU is one such system which allows different hardware configurations [51]. MEADOWS VMN (Virtual Mote Network) allows multiple virtual motes per physical host participating in the emulation [52]. The virtual motes can tun TinyOS, and TinyDB or other applications on top, while hardware of the mote, and sensor and wireless channels are simulated.

Another original direction is explored by Haeberlen et al. in their system called Monarch [53]. In Monarch, the idea is to use the latency observed at the moment between the host on which the virtual sender and the virtual receiver resides, and a remote host on the Internet. For this purpose, for every packet the virtual sender wants to send, Monarch captures it and sends a probe packet of the same size to a remote host associated with the virtual receiver. When a reply is received, the virtual receiver is allowed to receive the packet. In the direction from virtual receiver to virtual

sender, Monarch passes the packets without delay. This way, both the sender and the receiver observes the round-trip-time obtained from the probe packet. Their approach targets transport layer studies only, and can be used with unmodified implementations of transport layer protocols in the Linux kernel.

IV. CONCLUSION

We have presented in this article a discussion of the terms emulator and testbeds, where we defined the reflexivity principle as a test for identifying testbeds, elaborated on what it means for entities in the SUT to be used as real in the emulation-based experiment setup. We have also presented our survey of forty two network emulators and testbeds reported in the literature.

Our survey mainly focuses on the techniques used for building emulators for various kinds of networks. Other surveys exist in the literature, such as that by Kropff et al. [54], which focuses on MANETs. A small-scale comparative survey also appears in [14].

The survey presented is the most encompassing one reported in the literature known to the author. However, no claim is made that it is complete. Examples of systems that can be added to this survey include Netbed [55], Dummynet [56], MobiEmu [57], MASSIVE [58], NIST Net [59], hitbox [60], Delayline [61], SensorSim [62], W-NINE [63], ONE [64], ENDE [65], REAL, NEST, PacketStorm, UML-Sim, RAMON, TOSSIM, EMStar, and possibly some others. Future work on this survey should focus on new ways of categorizing the emulators, as well as extending the number of emulators being covered in the survey.

References

[1] S. Guruprasad, R. Ricci, and J. Lepreau, "Integrated network experimentation using simulation and emulation," in *Proc. of the First Int. Conf. on Testbeds and Research Infrastructures for Development of Networks and Communities (TRIDENTCOM'05)*, 2005.

[2] D. M. Nicol, M. Liljenstam, and J. Liu, "Advanced concepts in large-scale network simulation," in *Proc. of the 37th Winter Simulation Conf. (WSC'05)*, December 2005.

[3] B. Ionescu, M. Ionescu, S. Veres, D. Ionescu, F. Cuervo, and M. Luiken-Miller, "A testbed and research network for next generation services over next generation networks," in *Proc. of TRIDENTCOM'05*, 2005.

[4] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Experience with an evolving overlay network testbed," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 13–19, 2003.

[5] B. A. Chambers, "The grid roofnet: a rooftop ad hoc wireless network," MIT Master's thesis, June 2002.

[6] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, 2003.

[7] D. A. Maltz, J. Broch, and D. B. Johnson, "Experiences designing and building a multi-hop wireless ad hoc network testbed," School of Computer Science, Carnegie Mellon University, Tech. Report CMU-CS-99-116, March 1999.

[8] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin, "A large-scale testbed for reproducible ad hoc protocol evaluations," in *Proc. of IEEE Wireless Communications and Networking Conf. 2002 (WCNC'02)*, March 2002.

[9] E. Nordström, P. Gunningberg, and H. Lundgren, "A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks," in *Proc. of TRIDENTCOM'05*, 2005.

[10] N. H. Vaidya, J. Bernhard, V. V. Veeravalli, P. R. Kumar, and R. K. Iyer, "Illinois wireless wind tunnel: a testbed for experimental evaluation of wireless networks."

[11] M. Ott, I. Seskar, R. Siraccusa, and M. Singh, "ORBIT testbed software architecture: Supporting experiments as a service," in *Proc. of TRIDENTCOM'05*, 2005.

[12] J. Lei, R. Yates, L. Greenstein, and H. Liu, "Wireless link snr mapping onto an indoor testbed," in *Proc. of TRIDENTCOM'05*, 2005.

[13] S. Ganu, H. Kremo, R. Howard, and I. Seskar, "Addressing repeatability in wireless experiments using ORBIT testbed," in *Proc. of TRIDENTCOM'05*, 2005.

[14] P. De, A. Raniwala, S. Sharma, and T.-C. Chiueh, "Design considerations for a multihop wireless network testbed," *IEEE Commun.*, vol. 43, no. 10, pp. 102–109, October 2005.

[15] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in network simulation," *IEEE Computer*, vol. 33, no. 5, pp. 59–67, May 2000.

[16] J. T. Kaba and D. R. Raichle, "Testbed on a desktop: strategies and techniques to support multi-hop MANET routing protocol development," in *Proc. of the 2nd ACM int. symp. on Mobile ad hoc networking & computing (MobiHoc'01)*. New York, NY, USA: ACM Press, 2001, pp. 164–172.

[17] G. Judd and P. Steenkiste, "Repeatable and realistic wireless experimentation through physical emulation," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 63–68, 2004.

[18] ——, "A software architecture for physical layer wireless network emulation," in *Proc. of the 1st int. workshop on Wireless network testbeds, experimental evaluation & characterization (WiNTECH'06)*. New York, NY, USA: ACM Press, 2006, pp. 2–9.

[19] H. Ritter, M. Tian, T. Voigt, and J. Schiller, "A highly flexible testbed for studies of ad-hoc network behaviour," in *Proc. of the 28th Annual IEEE Int. Conf. on Local Computer Networks*, October 2003.

[20] M. Takai, R. Bagrodia, M. Gerla, B. Daneshrad, M. Fitz, M. Srivastava, E. Belding-Royer, S. Krishnamurty, M. Molle, P. Mohapatra, R. Rao, U. Mitra, C.-C. Shen, and J. Evans, "Scalable testbed for next-generation wireless networking technologies," in *Proc. of TRIDENTCOM'05*, 2005.

[21] X. W. Huang, R. Sharma, and S. Keshav, "The ENTRAPID protocol development environment," in *Proc. of the Eighteenth Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'99)*, vol. 3, March 1999, pp. 1107–1115.

[22] D. Ely, S. Savage, and D. Wetherall, "Alpine: A user-level infrastructure for network protocol development," in *Proc. of the 3rd USENIX Symp. on Internet Technologies and Systems*, 2001, pp. 171–184.

[23] M. Zec and M. Mikuc, "Real-time IP network simulation at gigabit data rates," in *Proc. of the 7th Int. Conf. on Telecommunications (ConTEL 2003)*, 2003, pp. 235–242.

[24] S. Jansen and A. McGregor, "Simulation with real world network stacks," in *Proc. of WSC'05*, 2005, pp. 2454–2463.

[25] A. Varga, "The OMNET++ discrete event simulation system," in *Proc. of the European Simulation Multiconference (ESM'01)*, June 2001.

[26] R. Bless and M. Doll, "Integration of the FreeBSD TCP/IP-stack into the discrete event simulator OMNet++," in *Proc. of WSC'04*, 2004, pp. 1556–1561.

[27] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: realistic and controlled network experimentation," in *Proc. of the 2006 conf. on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'06)*. New York, NY, USA: ACM Press, 2006, pp. 3–14.

[28] J. Flynn, H. Tewari, and D. O'Mahony, "JEMU: A real time emulation system for mobile ad hoc networks," in *Proc. of the First Joint IEI/IEE Symp. on Telecommunication Systems Research*, November 2001.

[29] R. Karrer, A. Sabharwal, and E. Knightly, "Enabling large-scale wireless broadband: the case for TAPs," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 27–32, 2004.

[30] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[31] K. Fall, "Network emulation in the Vint/NS simulator," in *Proc. of the The Fourth IEEE Symp. on Computers and Communications (ISCC'99)*. Washington, DC, USA: IEEE Computer Society, 1999, p. 244.

[32] R. Simmonds, R. Bradford, and B. Unger, "Applying parallel discrete event simulation to network emulation," in *Proc. of the fourteenth workshop on Parallel and distributed simulation*

(PADS'00). Washington, DC, USA: IEEE Computer Society, 2000, pp. 15–22.

[33] R. Simmonds, C. Williamson, R. Bradford, M. Arlitt, and B. Unger, "Web server benchmarking using parallel wan emulation," in *Proc. of the 2002 ACM SIGMETRICS int. conf. on Measurement and modeling of computer systems (SIGMETRICS'02)*. New York, NY, USA: ACM Press, 2002, pp. 286–287.

[34] C. Kiddle, R. Simmonds, and B. Unger, "Improving scalability of network emulation through parallelism and abstraction," in *Proc. of the 38th annual Symp. on Simulation (ANSS'05)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 119–129.

[35] R. Bradford, R. Simmonds, and B. Unger, "Packet reading for network emulation," in *Proc. of the 9th Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, August 2001, pp. 15–18.

[36] M. Liljenstam, J. Liu, D. Nicol, Y. Yuan, G. Yan, and C. Grier, "RINSE: The real-time immersive network simulation environment for network security exercises," in *Proc. of PADS'05*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 119–128.

[37] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker, "Scalability and accuracy in a large-scale network emulator," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 271–284, 2002.

[38] P. Mahadevan, A. Rodriguez, D. Becker, and A. Vahdat, "MobiNet: a scalable emulation infrastructure for ad hoc and wireless networks," in *WiTMeMo '05: Papers presented at the 2005 workshop on Wireless traffic measurements and modeling*. Berkeley, CA, USA: USENIX Association, 2005, pp. 7–12.

[39] D. Herrscher and K. Rothermel, "A dynamic network scenario emulation tool," in *Proc. of the 11th Int. Conf. on Computer Communications and Networks (ICCCN'02)*, 2002, pp. 262–267.

[40] D. Herrscher, S. Maier, and K. Rothermel, "Distributed emulation of shared media networks," in *Proc. of the 2003 Int. Symp. on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2003)*, 2003, pp. 226–233.

[41] P. Zheng and L. M. Ni, "EMPOWER: A scalable framework for network emulation," in *Proc. of the Int. Conf. on Parallel Processing 2002*, 2002, pp. 185–192.

[42] ——, "EMWIN: emulating a mobile wireless network using a wired network," in *Proc. of the 5th ACM int. workshop on wireless mobile multimedia (WOWMOM'02)*. New York, NY, USA: ACM Press, 2002, pp. 64–71.

[43] J. P. Macker, W. Chao, and J. W. Weston, "A low-cost, IP-based mobile network emulator (MNE)," in *Proc. of the Military Communications Conf. (MILCOM 2003)*, October 2003.

[44] S. Y. Wang, C. L. Chou, C. H. Huang, C. C. Hwang, Z. M. Yang, C. C. Chiou, and C. C. Lin, "The design and implementation of the NCTUns 1.0 network simulator," *Computer Networks*, vol. 42, pp. 175–197, 2003.

[45] A. Kayssi and A. El-Haj-Mahmoud, "EmuNET: a real-time network emulator," in *Proc. of the 2004 ACM symp. on Applied computing (SAC'04)*. New York, NY, USA: ACM Press, 2004, pp. 357–362.

[46] E. Göktürk, M. Pužar, and M. N. Akkøk, "Distributing NEMAN network emulator using MICA component architecture," in *Proc. of the AI, Simulation and Planning in High Autonomy Systems (AIS), and Conceptual Modeling and Simulation (CMS) Conf. (AIS-CMS 2007) (co-located with the Int. Modeling and Simulation Multiconference (IMSM 2007))*, F. Barros, C. Frydman, N. Giambiasi, and B. Zeigler, Eds. SCS, February 2007, pp. 199–205.

[47] M. Pužar and T. Plagemann, "NEMAN: A network emulator for mobile ad-hoc networks," in *Proc. of ConTEL 2005*, 2005.

[48] X. Jiang and D. Xu, "vbet: a vm-based emulation testbed," in *Proc. of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research (MoMeTools'03)*. New York, NY, USA: ACM Press, 2003, pp. 95–104.

[49] V. Guffens and G. Bastin, "Running virtualized native drivers in user mode linux," in *Proc. of USENIX 2005 Annual Technical Conf.*, 2005.

[50] M. Engel, M. Smith, S. Hanemann, and B. Freisleben, "Wireless ad-hoc network emulation using microkernel-based virtual linux systems," in *Proc. of the 5th EUROSIM Congress on Modeling and Simulation*, 2004, pp. 198–203.

[51] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras, "ATEMU: a fine-grained sensor network simulator," in *Proc. of the First Annual IEEE Communications Society Conf. on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, October 2004, pp. 145–152.

[52] Q. Luo, L. M. Ni, B. He, H. Wu, and W. Xue, "MEADOWS: modeling, emulation, and analysis of data of wireless sensor networks," in *Proceeedings of the 1st int. workshop on Data management for sensor networks (DMSN'04)*. New York, NY, USA: ACM Press, 2004, pp. 58–67.

[53] A. Haeberlen, M. Dischinger, K. P. Gummadi, and S. Saroiu, "Monarch: a tool to emulate transport protocol flows over the internet at large," in *Proc. of the 6th ACM SIGCOMM on Internet measurement (IMC'06)*. New York, NY, USA: ACM Press, 2006, pp. 105–118.

[54] M. Kropff, T. Krop, M. Hollick, P. S.Mogre, and R. Steinmetz, "A survey on real world and emulation testbeds for mobile ad hoc networks," in *Proc. of TRIDENTCOM'06*. IEEE, March 2006.

[55] B. White, J. Lepreau, and S. Guruprasad, "Lowering the barrier to wireless and mobile experimentation," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 47–52, 2003.

[56] L. Rizzo, "An embedded network simulator to support network protocols' development," in *Proc. of the 9th Int. Conf. on Computer Performance Evaluation: Modeling Techniques and Tools*, ser. Lecture Notes in Computer Science (LNCS). Springer, 1997, vol. 1245, pp. 97–107.

[57] Y. Zhang and W. Li, "An integrated environment for testing mobile ad-hoc networks," in *Proc. of MobiHoc'02*. New York, NY, USA: ACM Press, 2002, pp. 104–111.

[58] M. Matthes, H. Biehl, M. Lauer, and O. Drobnik, "MASSIVE: An emulation environment for mobile ad-hoc networks," in *Proc. of the Second Annual Conf. on Wireless On-demand Network Systems and Services (WONS'05)*, 2005.

[59] M. Carson and D. Santay, "NIST Net: a Linux-based network emulation tool," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 111–126, 2003.

[60] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: emulation and experiment," in *Proc. of SIGCOMM'95*. New York, NY, USA: ACM Press, 1995, pp. 185–195.

[61] D. B. Ingham and G. D. Parrington, "Delayline: a wide-area network emulation tool," *Comput. Syst.*, vol. 7, no. 3, pp. 313–332, 1994.

[62] S. Park, A. Savvides, and M. B. Srivastava, "SensorSim: a simulation framework for sensor networks," in *Proc. of the 3rd ACM int. workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWIM'00)*. New York, NY, USA: ACM Press, 2000, pp. 104–111.

[63] T. Pérennou, E. Couchon, L. Dairaine, and M. Diaz, "Two-stage wireless network emulation," in *Proc. of the Workshop on Challenges of Mobility held in conjunction with 18th IFIP World Computer Congress (WCC)*, F. Boavida, E. Monteiro, and J. Orvalho, Eds., 2004, pp. 57–66.

[64] M. Allman, A. Caldwell, and S. Ostermann, "ONE: The Ohio network emulator," School of Electrical Engineering and Computer Science, Ohio University, Technical Report TR-19972, August 1997.

[65] I. Yeom and A. N. Reddy, "ENDE: An end-to-end network delay emulator tool for multimedia protocol development," *Multimedia Tools Appl.*, vol. 14, no. 3, pp. 269–296, 2001.

EREK GÖKTÜRK graduated with BS degree in 2000 and MS degree in 2003 from Dept. of Computer Engineering (CEng), Middle East Technical University (METU), Ankara, Turkey. He worked as an assistant in CEng@METU between 2000-2004, and participated part-time in an agent-based simulation project in the Modeling and Simulation Laboratory, a METU-TSK (Turkish Armed Forces) joint research facility. He has instructed in METU-CES (Continuing Education Center) Cisco Networking Regional Academy during 2002-2004. Since 2004 he works in Dept. of Informatics in University of Oslo, Norway, as a Doctoral Research Fellow. His research interests include component-based software engineering, engineering of simulators and emulators, network simulation and emulation, and philosophy of computing and information.