TWO-STAGE WIRELESS NETWORK EMULATION

Tanguy Pérennou,[1,2] Emmanuel Conchon,[1,2] Laurent Dairaine[1,2] and Michel Diaz[2]

[1] *ENSICA, 1 place Émile Blouin, 31056 Toulouse Cedex 5, France*
{tanguy.perennou,emmanuel.conchon,laurent.dairaine}@ensica.fr

[2] *LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4, France*
michel.diaz@laas.fr

Abstract Testing and deploying mobile wireless networks and applications are very challenging tasks, due to the network size and administration as well as node mobility management. Well known simulation tools provide a more flexible environment but they do not run in real time and they rely on models of the developed system rather than on the system itself. Emulation is a hybrid approach allowing real application and traffic to be run over a simulated network, at the expense of accuracy when the number of nodes is too important. In this paper, emulation is split in two stages: first, the simulation of network conditions is precomputed so that it does not undergo real-time constraints that decrease its accuracy; second, real applications and traffic are run on an emulation platform where the precomputed events are scheduled in soft real-time. This allows the use of accurate models for node mobility, radio signal propagation and communication stacks. An example shows that a simple situation can be simply tested with real applications and traffic while relying on accurate models. The consistency between the simulation results and the emulated conditions is also illustrated.

1. Introduction

Developing and testing wireless communication protocols and distributed applications is a big challenge. Developers must face a potentially high dynamism of the network topology due to terminal mobility and environment-specific radio propagation issues, in addition to well-known difficulties related to the general networking aspects in a distributed environment.

A first solution is to set up an experimental testbed with a sufficient amount of nodes and links, including routers, network interface hardware, full communication stacks and applications. Even in a wired context, such testbeds are expensive, impractical to use or configure, unable to repeat the same experiments or to set up limit conditions. In a wireless context, making the nodes move is an additional difficulty. A more practical solution is to use a discrete event simulation tool such as *ns-2* [1] or GloMoSim [18]. In this case, developers must provide models for all the communication stacks of the tested system (including applications), the environment and the generated traffic. Models can be very accurate because there are no real-time constraints. However, entirely relying on models may imply other drawbacks: when the system is under development, providing appropriate models leads to a potentially expensive redundant development; when the system is already implemented and is too complex, developing the associated models may be too expensive or simply impossible.

Emulation is a solution offering a compromise between real life experiments and simulation. It simulates in real-time the service of a given layer on the basis of models of the underlying layers, allowing real software to be executed on top of that layer. In a network-level emulator for instance, true applications and transport-level protocols can send and receive regular IP datagrams. The operational part of the emulation (i.e. from the emulated service up to the applications) can be composed of off-the-shelf elements, like TCP, UDP or probing tools like `tcpdump`, as well as software under development. The delivered IP datagrams are delayed or lost in real-time so as to reproduce some user-defined experimental conditions. Nevertheless, the operational part of the emulation can cause scalability problems, e.g. when many nodes are involved. Classical examples of IP-level emulators are Dummynet [15] and NIST Net [2].

The present work introduces W-NINE, an emulation platform aiming at the emulation of a wide range of wireless networks. It is based on a preliminary off line non real-time simulation stage associated with a regular IP-level emulation stage. This preliminary stage computes the dynamic topology of a wireless mobile network. The simulation models can range from very simple probabilistic models to accurate ones since this stage is not run in real-time. This stage produces an emulation scenario that is subsequently scheduled in soft real-time (i.e. deadlines may be missed, but should then be reported to the user) on the experimentation platform, while real applications and protocols are tested. Thus W-NINE combines models accuracy with real-time IP-level emulation.

This paper provides a brief overview of related work and then focuses on the architecture of the W-NINE platform. A detailed description of the offline simulation stage is then given. Then the usage of W-NINE is illustrated in a WiFi environment through a simple mobile application.

2. Related Work

The main goal of emulation is to reproduce in real time the behavior of some target network. As real distributed nodes must communicate, an experimentation network is used during the emulation. A major aspect of emulation is the need for an over-provisioned physical experimentation network. Typically a classical research environment cannot emulate core networks but is suitable for most access networks such as satellite, xDSL, end-to-end WAN communications [10], ad-hoc environments, etc.

In network-level emulation, only a few parameters need to be manipulated to mimic specific network conditions for an end-to-end communication: bandwidth, delay and packet losses. Traffic shapers such as Dummynet [15] or NIST Net [2] can be used as basic tools allowing the manipulation of these parameters and can constrain the traffic of the experimentation network. A number of large emulation testbeds use such traffic shapers, e.g. Netbed/Emulab [17]. Various approaches allow the dynamic configuration of a traffic shaper during an emulation.

Emulators built on a trace-based approach [11] reproduce the behavior of a real network (wired or wireless) on the experimentation network, according to previously captured traces. Those traces have to be processed to produce an emulation model that will be easily interpreted by the traffic shaping part of the emulator (i.e. a model made of bandwidth, delays and losses). This processing phase is called the distillation phase. It results in an accurate emulation of what happened in the real network. However, capturing traces can be difficult and expensive in time, for instance in large-scale networks. Moreover, traces are a snapshot of specific network conditions at some moment in time. For a user, it is nearly impossible to set up some unforeseen condition: a new experiment should be set up so that the expected situation really happens, thus allowing to capture an adequate set of traces.

A second approach consists in using a simulation stage. As an example, *ns-2* provides an emulation mode [3], although it is widely used by researchers as a discrete event simulation platform. Contrary to this classical use, the emulation mode operates with a real time scheduler and is able to process real packets. The *ns-2* emulator can work in two complementary modes: an opaque mode where real packets are not modified but undergo semantic or temporal modifications, and a protocol mode

which, in addition, allows real packets to be interpreted and modified by *ns-2* network models. An extension for wireless networks emulation has been presented in [8]. It uses the new features added by wireless extensions of the *ns-2* simulator to emulate ad-hoc environments. In this emulator, wireless and ad-hoc conditions are reproduced by *ns-2*, so its accuracy depends on the modeling level used. Another exemple of this type of emulation platform is Seawind [9], which is used in the context of mobile wireless WANs like GPRS.

The main difficulty with this fully centralized type of emulation is that the discrete-event simulator has to process all relevant events in real-time. When the density of events becomes too important, the simulator drifts and does not meet the timing contraints, thus invalidating the emulation results. The main causes of the increase of event density are the number of nodes involved and the accuracy of the implemented models. For wireless networks, these models handle not only communication layers, but also node mobility and radio signal propagation. The latter may involve complex random number generation or ray-tracing algorithms [16].

Several ways of preventing the simulation drift have been proposed. A first distribution scheme consists in parallelizing the simulation task across several hosts, each being responsible for only a subset of emulated nodes. This is done in e.g. PDNS [14] and Netbed/Emulab [17]; the latter makes use of a grid and provides an integrated access to three disparate experimental environments: simulated, emulated, and wide-area network testbeds. Another distribution scheme consists in delegating part of the simulation task to each emulated node. The central simulation node(s) then periodically synchronize its computation with that of emulated nodes. This approach has been used in several wireless network emulation platforms, such as EMWIN [19] and JEmu [4]. Each emulated node computes its own propagation capacities, typically neighbor tables, while the central simulation periodically broadcasts new positions. This approach is rather intrusive in that each emulated node must host special software. In addition none of these platforms currently implement accurate propagation models.

An alternative approach consists in using a precomputed scenario to control a traffic shaper, as in the Network Emulation Testbed [5], which allows the use of XML-formatted scenarii to emulate wireline networks. This paper extends this approach in the context of wireless networks, so as to allow the use of accurate mobility, propagation and communication models. These models are run at simulation-time, in a preliminary stage that does not undergo real-time constraints. The real-time constraints have to be met only at emulation-time, when applications and traffic get

operational. The topic of real-time traffic shaping is outside the scope of this paper.

3. The W-NINE Platform

The W-NINE platform is designed to emulate a large spectrum of mobile wireless networks based on *high-level experiment descriptions* written by developers for tests. A high-level experiment description allows the combination of accurate simulation models at the user level, so that tested situations can approximate real conditions. W-NINE consists of a network emulation platform, called NINE (Nine Is a Network Emulator), which was already used in various emulation experiments [10], and a Simulator for Wireless Networks Emulation, called SWINE (see Figure 1).

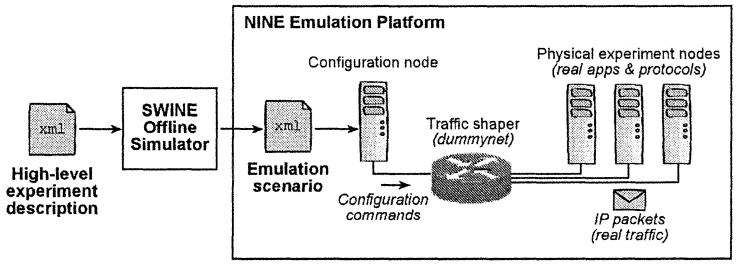

Figure 1. The W-NINE Platform Architecture

The NINE network emulator is a platform of interconnected nodes where the IP traffic is shaped with Dummynet [15]. The experimental network is a wired Ethernet gigabit network. NINE also includes a *configuration node* that uses an *emulation scenario* to schedule in (soft) real-time Dummynet *configuration commands* corresponding to precomputed QoS events, such as bandwidth or delay change on a given link. Real applications and transport protocols are deployed on the *physical experiment nodes* and the IP datagrams that they exchange are systematically routed to the *traffic shaper*, that delays or loses them according to the current emulation state.

The SWINE simulator is a discrete event simulator that runs offline before the emulation stage. Its purpose is to generate the NINE *emulation scenario* corresponding to a *high-level experiment description.* This description fully describes an experiment, i.e. the movement of mobile nodes, their energy consumption, the positions of obstacles, the radio propagation conditions and the communication stack from the physical medium to the network level on each node. A model toolbox is used

for helping the end-user to provide a high-level experiment description, for example to express node mobility (e.g. Random Waypoint or Pursue mobility model [6]), propagation of the radio signal between end systems (e.g. Path Loss Exponent propagation model, associated with Rayleigh or Rice fading model [13]), energy management or the wireless technology used. The parameters of these models can be changed for each new experiment. Moreover, SWINE is defined as an open architecture and allows the easy addition of new models. A more detailed description of SWINE can be found in Section 4.

To sum up, the roles of SWINE and NINE are quite simple: SWINE precomputes a scenario composed of IP traffic constraints, and NINE provides a physical network interconnecting real node, where the IP traffic is shaped in soft real-time according to this scenario, as if the underlying network was a mobile wireless network.

The limited role of SWINE makes it a much simpler simulator than *ns-2* or GloMoSim. SWINE does not have to model neither network layers above IP nor the application or the traffic itself, while in *ns-2*, every packet is simulated and passed across all the modeled layers, starting from the application to the physical channel. The principle behind SWINE is not to simulate packets, but to compute the effects of the target network on the bandwidth, delay and losses of each potential link, according to models of lower communication layers (physical, MAC and IP), propagation issues and node mobility.

4. Simulator Architecture

The SWINE simulator computes the evolution over a given period of time of a mobile wireless network in terms of network-level parameters, i.e. data throughput, delay and packet losses. The simulation must take into account the node movements, radio propagation issues and the communication stack used for delivering IP packets to the transport layer. Moreover, SWINE takes its input from the end-user (high-level experiment description) and delivers its output to the NINE configuration software.

The high-level experiment description describes the nodes (their movements, their communication stacks up to IP, including ad-hoc routing protocols) and the environment (obstacles and radio propagation conditions). This description is analysed by SWINE, which consequently creates both the domain objects and the model objects. *Domain objects* include nodes, links and routes, as well as obstacles. *Model objects* realize the equations corresponding to the user-specified models. For instance, a path loss exponent model object computes the pathloss in dB,

i.e. $PL(d) = \overline{PL(d_0)} + 10n \log(d/d_0)$ [13]. The simulation block then discretely computes the dynamic topology: the positions of each node over time, then the received power, data throughput, packet loss and delay for each node and link, and finally available routes and IP parameters on those routes. This simulation stage delegates a great amount of the computation to the model objects. Finally, the results are synthesized into an emulation scenario for the NINE platform.

At the time of writing, a prototype version of SWINE is implemented and supports the following models: Random Walk, Random Waypoint, Rectilinear Uniform, Still mobility models; Friis, Two-ray, Path-Loss Exponent large-scale pathloss models; Per-Obstacle Absorption, Lognormal medium-scale shadowing model; Rayleigh short-scale fading model; Thermal noise model; Table-based IP throughput model. This first version has been set up to demonstrate the usefulness of a simple IP-level simulator, as opposed to more complex simulators like *ns-2* that encompass additional models for applications, transport protocols and network traffic. A more mature version will be developed with a full support for mobile ad-hoc networks in mind. This implies group-based mobility models and a more explicit simulation of the concurrent access to the radio resource.

5. Experimenting with W-NINE

In this section we describe a simple W-NINE session. The emulated network is a 802.11b infrastructure network in an office, where a fixed station F1 wired to the access point continuously transmits UDP data to a mobile host M1. The purpose of this experiment is to demonstrate that W-NINE can emulate wireless network with a sufficient amount of realism.

The emulated network and experimental conditions are described in the high-level experiment description, which contains the models for mobility, propagation and IP communication. The mobile host moves rectilinearly back and forth with a constant speed of 0.5 m/s over 40 m (this model is easier to interpret than a classical random waypoint model). Among other curves, Figure 2 shows the F1-M1 distance computed by SWINE with a time granularity of 500 ms. The large-scale propagation model used is a path loss exponent model, where $n = 4.6$, $d_0 = 1$ m and $\overline{PL(d_0)}$ is estimated using the Friis free-space path loss model. Two obstacles absorb respectively 50% and 100% of the received signal. A second simulation adds a Rayleigh fading model. Figure 2 also shows the received power P_r for both faded and not faded experiments. As expected, Rayleigh fading causes fast variations of P_r. Additionally, the

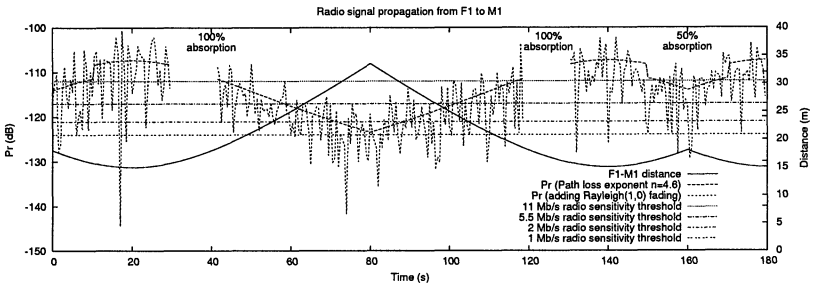

Figure 2. Plot of SWINE's Raw Output (Distance and Received Power)

P_r brutal drops and holes are due to signal absorption by the obstacles when the M1 passes in front of them. The horizontal lines show the minimal power needed to receive data transmitted at 1, 2, 5.5 and 11 Mb/s.

The communication model is based on a simple table associating an IP throughput with the received signal: the receiver's radio sensitivity levels allows the selection of the appropriate transmission speed and the 802.11b PHY and MAC protocols then allow the computation of the theoretical maximum data throughput [7]. The packet loss rate is either 0 or 1, and delays due to auto-rate fallback were ignored. The computed throughput is illustrated in Figure 3 for both faded and not faded experiments. The steps-like curve represents the computed throughput and changes according to the P_r level, with much more frequent changes when Rayleigh fading is taken into account. The throughput changes for each link compose the emulation scenario computed by SWINE.

Until now, no real traffic has been sent from F1 to M1. During the emulation stage, two tasks are run in parallel: applications are run on NINE physical nodes while the Dummynet traffic shaper is configured in real-time according to the scenario computed by SWINE. Here, the application used is MGEN [12]: a MGEN/UDP transmitter runs on the Linux PC hosting F1 and a MGEN receiver runs on M1's host. MGEN is configured to measure the throughput from F1 to M1, now with real UDP/IP packets going through the Dummynet traffic shaper. Figure 3 shows that as soon as QoS events are scheduled on the shaper, the measured throughput changes accordingly.

This experiment shows that a high-level, i.e. user-friendly, description of an experiment can be used to dynamically configure an Ethernet-based emulation platform where real applications can run in real-time as if they were deployed over a wireless network.

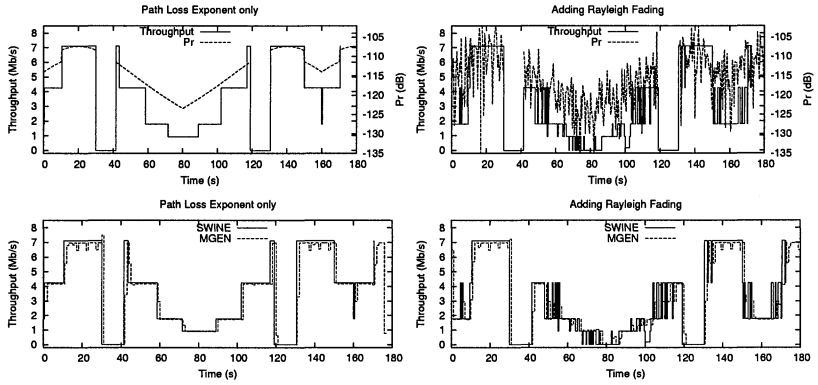

Figure 3. Throughput Computed by SWINE and Measured by MGEN on NINE

6. Conclusion

This paper introduces W-NINE, a platform allowing the test of distributed applications and/or transport protocols over an emulated mobile wireless network. It is based on a two-stage process that allow both the use of accurate simulation models for lower layers and testing real application and higher layers protocols in real time.

W-NINE is based on the SWINE offline simulator which produces an emulation scenario for NINE, a real-time IP-level emulation platform. NINE is currently deployed in our labs and a first version of the SWINE simulator has been developed, focusing on the integration of well-known models for node mobility, radio propagation and medium access control. More models can be easily integrated to SWINE.

Acknowledgments

We wish to thank Emmanuel Goua de Baix, Jérôme Fimes and Hervé Thalmensy for their help in the development and administration of the W-NINE platform.

References

[1] 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*, 33(5), May 2000.

[2] M. Carson and D. Santay. NIST Net: A Linux-based Network Emulation Tool. *ACM SIGCOMM Computer Commununications Review*, 33(3):111–126, 2003.

[3] K. Fall. Network Emulation in the VINT/NS Simulator. In *Proceedings of the fourth IEEE Symposium on Computers and Communications*, 1999.

[4] J. Flynn, H. Tewari, and D. O'Mahony. JEmu: A Real Time Emulation System for Mobile Ad Hoc Networks. In *Proceedings of the First Joint IEI/IEE Symposium on Telecommunications Systems Research*, November 2001.

[5] D. Herrscher and K. Rothermel. A Dynamic Network Scenario Emulation Tool. In *Proceedings of ICCCN 2002*, pages 262–267, October 2002.

[6] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Towards Realistic Mobility Models for Mobile Ad hoc Networks. In *Proceedings of MobiCom'03*, September 2003.

[7] J. Jun, P. Peddachagari, and M. Sichitiu. Theoretical Maximum Throughput of IEEE 802.11 and its Applications. In *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA-03)*, 2003.

[8] Q. Ke, D. Maltz, and D. Johnson. Emulation of Multi-Hop Wireless Ad Hoc Networks. In *The 7th International Workshop on Mobile Multimedia Communications (MoMuC 2000)*, October 2000.

[9] M. Kojo, A. Gurtov, J. Mannner, P. Sarolahti, T. Alanko, and K. Raatikainen. Seawind: a Wireless Network Emulator. In *Proceedings of the 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB 2001)*, 2001.

[10] L. Lancerica, L. Dairaine, F. de Belleville, H. Thalmensy, and C. Fraboul. MITV, A solution for Interactive TV Based on IP Multicast over Satellite. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, June 2004.

[11] B. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz. Trace-Based Mobile Network Emulation. In *Proceedings of ACM SIGCOMM'97*, September 1997.

[12] NRL/PROTEAN. MGEN: The Multi-Generator Toolset. `http://mgen.pf.itd.nrl.navy.mil`.

[13] T. Rappaport. *Wireless Communications Principles and Practice, 2nd Edition.* Prentice Hall, 2002.

[14] G. F. Riley, R. Fujimoto, and M. H. Ammar. A Generic Framework for Parallelization of Network Simulations. In *Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 1999.

[15] L. Rizzo. Dummynet: A Simple Approach to the Evaluation of Network Protocols. *ACM Computer Communication Review*, 27(1), January 1997.

[16] A. Schoonen. Designing Wireless Indoor Radio Systems with Ray Tracing Simulators. Technical report, Eindhoven University of Technology, Dec. 2003. Available at `http://people.spacelabs.nl/~admar/SimpleCSD/report.pdf`.

[17] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of OSDI02*, 2002.

[18] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks. In *Proceedings of PADS'98*, May 1998.

[19] P. Zheng and L. Ni. EMWin: Emulating a Mobile Wireless Network using a Wired Network. In *Proceedings of the 5th ACM International Workshop on Wireless Mobile Multimedia*, 2002.