A Feedback Based Solution to Emulate Hidden Terminals in Wireless Networks

Emmanuel Conchon*†, Tanguy Pérennou*†, Michel Diaz†
*ENSICA, 1 place Emile Blouin, F-31056 Toulouse Cedex 5, France
†LAAS-CNRS, 7 avenue du Colonel Roche, F-31077 Toulouse Cedex 4, France
E-mail: {econchon, perennou}@ensica.fr, diaz@laas.fr

***Abstract*— Mobile wireless emulation allows the test of real applications and transport protocols over a wired network mimicking the behavior of a mobile wireless network (nodes mobility, radio signal propagation and specific communication protocols). Two-stage IP-level network emulation consists in using a dedicated offline simulation stage to compute an IP-level emulation scenario, which is played subsequently in the emulation stage. While this type of emulation allows the use of accurate computation models together with a large number of nodes, it currently does not allow to deal with dynamic changes of the real traffic. This lack of reactivity makes it impossible to emulate specific wireless behaviors such as hidden terminals in a realistic way. In this paper we address the need to take into account the real traffic during the emulation stage and we introduce a feedback mechanism. During the simulation several emulation scenarios are computed, each scenario corresponding to alternative traffic conditions related to e.g. occurrence or not of hidden terminals. During the emulation stage, the traffic is observed and the currently played emulation scenario can be changed according to specific network conditions. We propose a solution based on multiple scenarios generation, traffic observers and a feedback mechanism to add a traffic-based dynamic behavior to a two-stage emulation platform. The solution will be illustrated with a simple experiment based on hidden terminals.**

I. Introduction

Test and evaluation of applications and transport protocols on mobile wireless networks is a challenging task. In wireless networks developers of applications and protocols must deal with mobility and propagation aspects in addition to classical difficulties encountered in the development of networking aspects in a distributed environment. Furthermore, developers must face problems due to traffic-based behavior and to topological aspects such as hidden terminals in MANETs.

A first solution is to set up a real wireless network in order to evaluate the application and/or the protocol under test. In practice, this solution is not suitable when the number of nodes increase or to evaluate the application/protocol under limit conditions (e.g. with a high speed mobility model). Moreover, this solution forbids the reproducibility of an experiment. A more practical solution is to use a discrete event simulation tool such as *ns-2* [1] or GloMoSim [2]. Despite its usefulness and its ability to deal with traffic-based aspects, this solution requires the use of a model of the application/protocol under evaluation instead of the real application/protocol itself. Providing appropriate models may lead to a potentially expensive redundant development or may even be impossible when the system to test is too complex.

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: real software can be executed on top of that layer while the behavior of the underlying layers is reproduced. In an IP-level emulator for instance, true applications and transport-level protocols can send and receive regular IP packets. These packets are then delayed and lost according to user-specified conditions in order to mimic the behavior of the target network. However, when those delays and losses are computed in real-time as well as the dynamic wireless network topology (as in NSE, the emulation part of *ns-2*), scalability issues arise even when only a few nodes are involved.

A more scalable solution is to split the emulation process in two stages: an *offline* simulation stage computes an emulation scenario using complex and accurate models without real-time constraints, followed by an emulation stage that reproduces in soft real-time the emulation scenario. As existing implementations of this solution do not appropriately take traffic-based behaviors into account, we investigate an enhancing feedback mechanism based on the observation of the real-time traffic on the emulation platform to switch among precomputed possible scenarios.

This paper provides a brief overview of the related work (Section II) and reminds the general architecture of W-NINE (Section III), our emulation platform, then focusing on multiple possibilities and on the feedback mechanism (Section IV). Finally a simple experiment dealing with a classical hidden terminals situation will then highlight the use of these mechanisms (Section V).

II. State of the Art

The main goal of emulation is to reproduce in real time the behavior of some target network on an experimentation network (typically Ethernet), which of course must be physically over-provisioned. 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 [3], wireless networks and so on.

In this paper we will focus on a network level emulation where 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 [4] or NIST Net [5] 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 [6]. Various approaches allow the dynamic configuration of a traffic shaper during an emulation.

Emulators built on a trace-based approach [7] reproduce the behavior of a real network (wired or wireless) on the experimentation network, according to previously captured traces. This approach provides a very accurate emulation but prevents user from stepping away from existing traces.

A second approach consists in using real-time simulation. As an example, *ns-2* provides an emulation mode [8], although it is widely used by researchers as a discrete event simulator. Contrary to this classical use, the emulation mode operates with a real time scheduler and is able to process real packets. An extension for wireless networks emulation has been presented in [9]. However it seems that the simulator scheduler has difficulties to deal with the real-time constraints leading to false simulation results in the 802.11 protocol implementation. An improvement of the simulator scheduler has been proposed in [10]. 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 constraints, thus invalidating the emulation results. The increase of event density can be due to the number of nodes involved or the accuracy of the implemented models, which are numerous in wireless networks (node mobility, radio signal propagation, and of course communication, including traffic-based behaviors).

Several ways of preventing the simulation drift have been proposed. A first distribution scheme consists in parallelizing the simulation task across several hosts as in PDNS [11] and Netbed/Emulab [6]. Another distribution scheme consists in delegating part of the simulation task to each emulated node as in EMWIN [12] and JEmu [13]. This approach is rather intrusive in that each emulated node must host special software. None of these platforms currently implement accurate propagation models nor support traffic-based behaviors.

An alternative approach consists in using a precomputed scenario to control a traffic shaper, as in the Network Emulation Testbed [14] for wired networks, or in W-NINE [15], which 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.

III. W-NINE

A. Architecture of W-NINE

The W-NINE platform (see Figure 1) 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 by the user, 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 [3], and a Simulator for Wireless Networks Emulation, called SWINE.

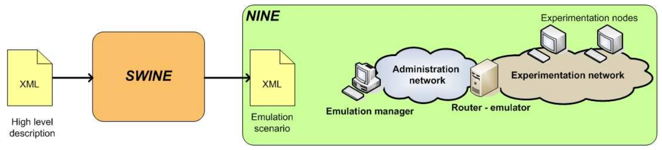

Fig. 1: The W-NINE Platform Architecture

The roles of SWINE and NINE are quite simple: SWINE is an offline discrete events simulator that precomputes *emulation scenarios* composed of IP traffic constraints corresponding to a high-level experiment description, and NINE provides a physical network interconnecting real nodes, where the IP traffic is shaped in soft real-time according to this scenario, as if the underlying network was a mobile wireless network.

B. The SWINE Simulator

The limited role of SWINE makes it a much simpler simulator than *ns-2* or GloMoSim: neither network layers above IP nor the application nor the traffic itself must be modelled, while in *ns-2*, every packet is simulated and passed across all the modelled 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. The architecture of SWINE is presented on Figure 2.

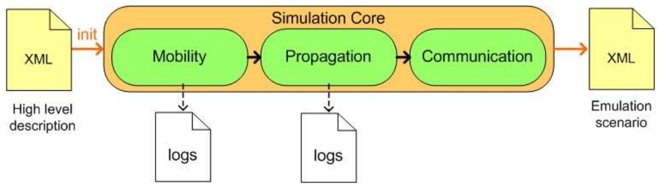

Fig. 2: SWINE Architecture.

The high-level description file describes the experiment to set up: the nodes (their movements, their communication stacks up to IP, including ad-hoc routing protocols) and the environment (obstacles and radio propagation conditions). This XML description is parsed by SWINE, which consequently creates the objects corresponding to nodes, links and routes, as well as obstacles, and realizes the equations corresponding to the user-specified models. For instance, a path loss exponent model object computes the pathloss in dB, i.e. $PL(d) = \overline{PL(d_0)} + 10n\log(d/d_0)$, where d is the transmitter-receiver distance, d_0 a reference distance, $\overline{PL(d_0)}$ an estimate of the pathloss at d_0 and n an empirical exponent [16]. At least d_0 and n should be specified in the high-level description file.

The simulation core discretely computes the dynamic topology: node positions over time in the mobility part, then the potentially received power in the propagation part, and finally parameters of the communication stack up to IP for each link and route are computed in the communication part to establish the different QoS events (i.e. evolution of available IP data throughput, losses and delays on each end-to-end link) characterizing the mobile wireless network behavior at the IP level. The communication part will be detailed in Section IV-C. Finally, all of the QoS events are synthesized into emulation scenarios for the NINE platform.

C. *The NINE emulation platform*

The NINE network emulator is a platform of interconnected nodes where the IP traffic is shaped by a central *router-emulator* with Dummynet [4] on a wired Ethernet gigabit network. An *emulation manager* is in charge of the platform dynamic configuration based on emulation scenarios provided by SWINE. The emulation manager schedules in soft real-time Dummynet *configuration commands* corresponding to precomputed QoS events. Real applications and transport protocols are deployed on the *physical experimentation nodes* and the IP packets that they exchange are systematically routed to the router-emulator, which delays or loses them according to the last configuration command.

IV. Enhancing W-NINE with a traffic-based feedback mechanism

A. *Impact of real traffic on the behavior of a wireless network*

In wired networks, instantaneous network traffic has a rather simple impact on the QoS at the IP level. In a 10Mb/s Ethernet network, if a node transmits data at 2Mb/s (IP throughput), another node can simply transmit IP data with the remaining 8Mb/s.

In wireless networks more complicated behaviors are observed, such as the 802.11 performance anomaly [17], the exposed and hidden [18] terminal problems. Under the same topological conditions, the performance will dramatically change with traffic conditions. We briefly present those behaviors.

In the 802.11 infrastructure mode, [17] has shown that the maximum available IP data throughput for one node is that of the slowest emitting node. If a node uses the 11Mb/s transmission rate (which corresponds to a maximum IP data throughput of 7.74Mb/s) while another node belonging to the same cell has to use the 2Mb/s transmission rate (i.e an IP data throughput of 1.4Mb/s), the first node will be slowed down until it observes an IP data throughput smaller than 1.4Mb/s.

Figure 3 presents the hidden terminals problem [18] that can occur in 802.11 Ad-Hoc mode. In this situation the node M2 lies in between the transmission range of the nodes M1 and M3. So, as M1 and M3 are out of range, they cannot sense each other's transmissions. They are said to be mutually hidden. In this case the node M3 can start sending its packets to the node M2 while M2 is still receiving packets from M1, which may lead to severe interferences. All of these interferences will result in losses at the IP level.

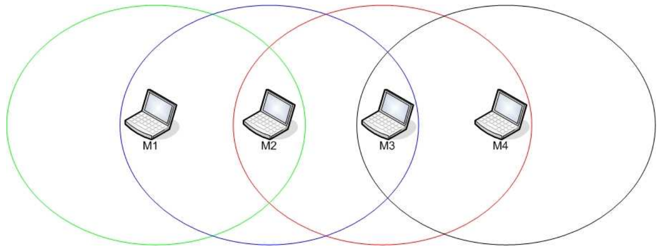

Fig. 3: Hidden and exposed terminals situation.

The exposed terminals problem is also presented in Figure 3. This situation occurs when the node M1 starts a communication with the node M2 while there is a communication between M3 and M4. As the node M3 communicates with M4 and is in range of M2's communication range, this communication will cause interferences on M2 leading to packet losses at the IP level.

B. *Possible feedback mechanisms*

To reproduce such mechanisms, a model of the network traffic can be used in order to establish if a collision occurs at a time *t*. However, one of the main interests of emulation is that it works without any traffic model. To keep working without a traffic model, a solution consists in observing the experimentation network during an experiment and to react according to the real network traffic and to collisions which would occur in the real wireless network. For this purpose, several solutions were investigated. The most intuitive one is to modify Dummynet, but this solution increases the Dummynet complexity which may bring about problems to provide a correct emulation when the traffic load increases on the router-emulator. Another similar solution is to develop an additional module to Dummynet which will be in charge of observing the network traffic, computing its effects on the emulated wireless link and modifying Dummynet emulation accordingly. This solution is very close to real time simulation solutions such as NSE which are not well suited when the traffic load or the models complexity increase.

Finally, according to W-NINE architecture, the process was split in two stages. The first stage computes the different network traffic possibilities and their impact on wireless communications within SWINE. For this purpose, SWINE has to search for all potential traffic cases and has to generate a particular possibility of the emulation scenario for each case. Then, in the second stage (emulation), it is necessary to observe in real time the network traffic on NINE and to react according to the precomputed possibility which corresponds to the traffic behavior that occurs.

C. The multiple possibilities approach

As presented in section III, SWINE is split in three parts. After the mobility and the propagation part, the complete network topology and available communication links among nodes at each time step are known. During the communication part SWINE computes the IP parameters that will be reproduced at the emulation stage. This communication part is modified to investigate the topology to search all potential traffic-based problems. For example in Ad-Hoc networks, it will search every group of at least two nodes that can be hidden from each other. If there is no potential hidden terminal an emulation scenario with a single possibility at each time step is produced. But if there are potential hidden terminals, SWINE produces an emulation scenario with two possibilities at each time step. The first possibility represents the "normal" case when no interference occurs, and the second represents the case when two hidden nodes try to send packets simultaneously. This second scenario is computed according to the user-specified hidden terminals model in the high-level description file (e.g. a 100% packet loss rate). All of these possibilities provide a multiple scenarios framework.

D. Traffic-based emulation decision

Modules that will be in charge of observing the NINE experimentation network during an experiment are called the *traffic observers*. A traffic observer takes no decision. It is hosted on the router-emulator and observes in real time the network traffic crossing the experimentation network. At the beginning of the experiment, it is configured by the emulation manager in order to know which nodes need to be observed. Then, using feedback mechanisms, it sends information relative to this traffic to the emulation manager which will choose the corresponding precomputed possibility of the emulation scenario. For example, if it has to observe a couple of nodes in order to determine if they are hidden from each other at a time *t*, it will check the traffic emitted by the two potentially hidden nodes and warn the network observer when these two traffics reach the router-emulator during the same time δt. This time interval is fixed by the emulation manager at the beginning of the experiment, just as the nodes that need to be observed.

The decision process is centralized in the emulation manager because it's the only node of the administration network that has a complete view of the experiment and of different scenarios. This solution is far simpler than distributing the decision process between the emulation manager and traffic observers: Dummynet receives update commands only from the emulation manager whereas in a distributed solution a priority scheme between the emulation manager and traffic observers should be set up.

A limitation of this solution is that the detection of a situation by traffic observers and the reaction of the emulation manager are not simultaneous. The emulation manager sends Dummynet configuration commands according to the time granularity of the precomputed scenario. That means that in the worst case it will react a full time step after the traffic observer detection.

V. A SIMPLE EXPERIMENT OF FEEDBACK BASED EMULATION

In this experiment, three nodes are used and stay still to simplify the interpretation of results. Nodes M1 and M3 are hidden from each other by a wall which totally absorbs the radio signal. Node M2 is located at the wall level and directly sees M1 and M3. M1 sends UDP datagrams to M2 during the whole experiment. M3 sends UDP datagrams to M2 from $t = 11$ to $t = 22$ and from $t = 38$ to $t = 59$. The chosen hidden terminals model consists in simply losing all IP packets on both links.

An excerpt of the XML emulation scenario produced by SWINE is presented on Figure 4. Only the M1 to M2 link is presented, M2 to M3 being exactly the same. At $t = 0$, SWINE has precomputed two possibilities. The first possibility (`hidden="0"`) is used when M1 is the only node sending data to M2: no problem occur and the packet loss rate (plr) is 0%. The second possibility (`hidden="1"`) is used when M1 and M3 simultaneously send data to M2: according to the chosen hidden terminal model all packets are lost (100% plr). In both cases the IP throughput is 7.11Mb/s. As nodes do not move both possibilities are reproduced at each time step (here set to 100ms, which is sufficient for e.g. a multimedia application) until the end.

```
<emulation_experiment>
 ...
 <scenario>
  <date ident="t0.0" start="0" unit="ms">
   <link_update hidden="0" hidden_id="M1-M2-M3">
       <on_link>M1 to M2</on_link>
       <bandwidth unit="Mb/s">7.11</bandwidth>
       <plr>0%</plr>
       <delay>0</delay>
   </link_update>
   <link_update hidden="1" hidden_id="M1-M2-M3">
       <on_link>M1 to M2</on_link>
       <bandwidth unit="Mb/s">7.11</bandwidth>
       <plr>100%</plr>
       <delay>0</delay>
   </link_update>
   ...
```

Fig. 4: Emulation possibilities precomputed by SWINE.

During the emulation stage, two MGEN [19] transmitters are running on the physical nodes M1 and M3, and one MGEN receiver is running on the physical node M2. Meanwhile, the Dummynet traffic shaper is configured in real-time by the emulation manager according to the pre-computed emulation scenario. The MGEN applications have been configured to use the UDP transport protocol and to measure the data throughput from M1 to M2 and from M3 to M2, i.e. the *real* UDP/IP traffic observed on the platform. Figure 5 shows that when M3 starts its first transmission at $t = 11$ the measured throughput on the M1-M2 link drops down to 0, which corresponds to the hidden terminal model implemented. A few packets sent by M3 reach M2 because of the latency between the traffic observer detection and the reaction of the emulation manager. The M1-M2 connection comes back at $t = 22$ when M3 stops its transmission. The same connection breakdown is observed between $t = 38$ and $t = 59$ during M3's second transmission to M2. The little variations observed are due to the MGEN data throughput measurement which is averaged every 25ms.

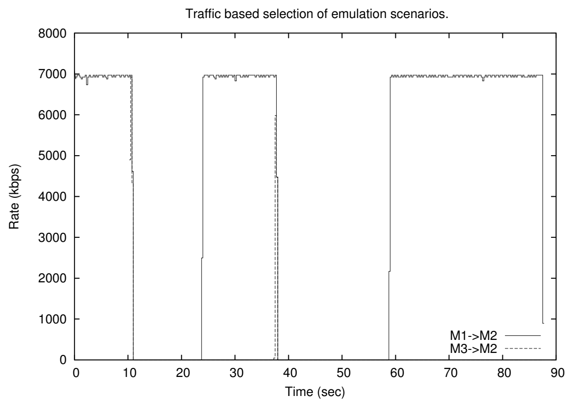

Fig. 5: Throughput Measured by MGEN on NINE.

This experiment shows that our approach based on a scenario with multiple possibilities and real traffic observation can deal with traffic-based behaviors such as hidden terminals in a two-stage emulation framework.

VI. Conclusion and future work

This paper proposes a new solution to deal with the reproduction of traffic-based behaviors in wireless networks on a two-stage emulation platform. It is based on the generation of multiple possibilities during the offline simulation stage and feedback from traffic observers to switch among possibilities during the emulation stage. A simple experiment successfully reproduced a hidden terminals situation.

Future work will essentially consist in enhancing the realism of wireless networks emulation. Other traffic-based behaviors such as exposed terminals or the 802.11 infrastructure-mode performance anomaly should be integrated. Also we need to control the delay introduced by the feedback mechanism. The big question is: does this delay unacceptably decrease realism? Of course only comparison with experiments on real wireless networks could give a definitive answer. However, we feel that a trade-off is possible between an acceptable loss of realism and the possibility to experiment real applications and protocols on a wired emulation platform rather than on an operational wireless network.

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*, vol. 33, no. 5, May 2000.

[2] 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.

[3] 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.

[4] L. Rizzo, "Dummynet: A Simple Approach to the Evaluation of Network Protocols," *ACM Computer Communication Review*, vol. 27, no. 1, January 1997.

[5] M. Carson and D. Santay, "NIST Net: A Linux-based Network Emulation Tool," *ACM SIGCOMM Computer Commununications Review*, vol. 33, no. 3, pp. 111–126, 2003.

[6] 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.

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

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

[9] 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.

[10] D. Mahrenholz and S. Ivanov, "Real-Time Network Emulation with ns-2," in *In proceedings of The 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications*, 2004.

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

[12] 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.

[13] 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.

[14] D. Herrscher and K. Rothermel, "A Dynamic Network Scenario Emulation Tool," in *Proceedings of ICCCN 2002*, October 2002, pp. 262–267.

[15] T. Pérennou, E. Conchon, L. Dairaine, and M. Diaz, "Two-stage Wireless Network Emulation," in *In proceedings of the 2004 Workshop on Challenges of Mobility (WCM 2004)*, 2004.

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

[17] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, "Performance Anomaly in 802.11b," in *Proceedings of IEEE INFOCOM 2003*, 2003.

[18] M. Gast, *802.11 Wireless Networks: The Definitive Guide*. O'Reilly, 2002.

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