Improved IP-level Emulation
for Mobile and Wireless Systems

Emmanuel Conchon∗†, Johan Garcia‡, Tanguy Pérennou∗†, Michel Diaz∗

∗LAAS-CNRS, 7 avenue du Colonel Roche, F-31077 Toulouse Cedex 4, France
†ENSICA, 1 place Emile Blouin, F-31056 Toulouse Cedex 5, France
‡Karlstad University, Dept. of Computer Science, SE-65188 Karlstad, Sweden
Email: econchon@ensica.fr, johan.garcia@kau.se, perennou@ensica.fr, diaz@laas.fr

Abstract—More and more applications and protocols are now running on wireless networks. Testing such applications and protocols is a real challenge as the position of the mobile terminals and environmental effects strongly affect the overall performance. Network emulation is often perceived as a good trade-off between experiments on operational wireless networks and discrete-event simulations on Opnet or ns-2. However, ensuring repeatability and realism in network emulation while taking into account mobility in a wireless environment is very difficult. This paper proposes a network emulation architecture based on off-line computations preceding online pattern-based traffic shaping. The underlying concepts of repeatability, dynamicity and accuracy are defined in the emulation context. Three different simple case studies illustrate the validity of our approach with respect to these concepts.

I. INTRODUCTION

Testing and evaluating a transport protocol or a distributed application is a challenging task for researchers. In networking and in wireless networking in particular, several difficult points have to be considered. This paper focus on accuracy, dynamicity and repeatability. Accuracy is a necessary condition to obtain realism, dynamicity addresses the ability to have conditions evolving over time, and repeatability allows fine-tuned evaluation of the protocol or the application.

To deal with these three points, several solutions can be used. First of them is the use of actual tests, where real applications or communication stacks under test are deployed on the real operating system. The accuracy of this method is high, but it has drawbacks since all of the environment parameters cannot be controlled.

The second solution which is widely used by researchers is simulation which provides a fully controllable environment ensuring repeatability. A major drawback of simulation is the need for many models: application models, protocol models and traffic models. While some of these may be available, other ones can be hard or even impossible to develop (closed source applications). Obtaining the traffic model can also be very difficult especially for multimedia or distributed applications evaluation.

Network emulation is a compromise between real test and simulation that allows the testing of real applications or transport protocols on an adequate experimentation network while in real time mimicking the behavior of another network.

A key interest of network emulation (also called IP-level emulation in this paper) is that only a few parameters need to be manipulated (typically bandwidth, delays and losses). The major problem is to use accurate models to produce accurate emulation parameters while respecting real time constraints. Some traffic shapers such as Dummynet [1] allow the manipulation of bandwidth, delays and losses, but they do not compute these parameters in real time. Moreover, most of these traffic shapers do not allow the exact repetition of an experiment.

A possible solution to deal with accuracy is to add a simulation step before emulation time in order to be able to use accurate models to produce the emulation parameters (bandwidth, delays and losses) that will then be used during emulation. Based on accurate models, the proposed simulator generates an emulation scenario that synthesizes all the simulated emulation parameters, providing a dynamic behavior of the network to emulate. Moreover, this emulation scenario can be played several times ensuring part of the repeatability. To further enhance repeatability, a traffic shaper supporting precise loss positioning as well as bit-error insertion can be used to reproduce losses and bit-errors.

The paper is organized as follows. Section II provides a brief overview of related work. Section III discusses the importance of accuracy, dynamicity and repeatability in network emulation. Section IV provides an overview of the proposed architecture and section V illustrates the usefulness of this emulation solution with three different case studies. Finally, Section VI provides the conclusions.

II. RELATED WORK

In this work we focus on IP level emulation which allows the evaluation of transport protocols or distributed applications. The main advantage of network level emulation is that only three parameters need to be manipulated to mimic specific network conditions: bandwidth, delays and packet losses. Traffic shapers such as Dummynet [1] or NISTNet [2] can be used as basic tools to constrain the experimentation traffic according to user specified rules. These rules are expressed in terms of bandwidth, delays and losses. Seawind [3] is another traffic shaper that has been designed to emulate wireless networks focusing on GPRS networks. It provides a more accurate delay
support specifying allocation, transmission and propagation delays. By using many instances of traffic shapers it is possible to build large emulation testbeds, e.g. Netbed/Emulab [4]. In these large testbeds, various approaches are used to allow the dynamic configuration of a traffic shaper during emulation.

Emulators built on a trace-based approach [5] mimic a real network on an experimentation one according to previously captured traces. It results in an accurate and reproducible 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.

EMWin [6] is a fully distributed wireless emulator focusing on the impact of mobility. For this purpose, an emulated MAC layer has been developed to mimic the medium access on a wired experimentation testbed. Based on neighbor tables, conditions and paths can evolve over time reproducing the dynamic topology of a wireless network. This solution is quite accurate for the mobility purpose but due to the MAC emulation and the distributed aspect the reproducibility is decreased.

Another type of emulator is based on real-time simulation such as NSE [7] or NCTUns [8]. The real traffic is intercepted by the simulator which in real time computes the impact of the wireless network on this traffic. The main drawback of this solution is the difficulty to respect real time constraints because the more complex the models are, the more computation time is needed. Even with recent optimizations such as those proposed in [9], this fully centralized solution is still not scalable and can only support a small set of wireless emulated nodes. To further increase performances, it is planned to distribute the real time simulation process but at the time of writing these solutions have not been evaluated.

III. PROBLEM STATEMENT

This section highlights three identified aspects that are challenging for emulators.

A. Accuracy

When performing emulation the accuracy of the emulated conditions should always be considered. The emulated conditions are often meant to reflect actual conditions in a network. The transition from actual conditions to emulated conditions is typically made either by the use of analytical models, simulations or based on traces captured in real networks.

In wireless networks the environment and the behavior of the end user have a strong impact on the observed QoS at the IP level: the current location of the communicating entities and the propagation of the radio signal directly affect delays, losses and the available throughput that IP packets will experience. To emulate wireless networks it is necessary to reproduce these effects as accurately as possible. Dealing with movements of terminal nodes may lead to the use of rather complex models to provide a realistic behavior. Similarly, the propagation model often will need to be complex in order to provide results that are sufficiently accurate.

If the propagation model is too simple, the behavior will not be realistic and this can in turn lead to misinterpretation of experimental results. For instance, relying on an uniform independent packet loss model often leads to false results as shown in [10]. The use of accurate propagation models such as those based on Ray Tracing or on probabilistic models such as Rayleigh or Ricean distributions [11] will help to reduce this kind of misinterpretations, but at the expense of large amounts of computation time.

The fact that the most accurate models are typically also the most computationally intensive leads to a trade-off between accuracy and scalability when the models are used in a real-time emulation setting. To avoid this trade-off problem we developed the use of off-line precomputation of emulation scenarios, which allows the use of accurate but resource demanding analytical and simulation models.

B. Dynamicity

There are several sources that contribute to the changing network conditions between a pair of nodes in a wireless network: fast and slow fading, node movement and varying cross traffic. The timescale of dynamicity is different among these sources, and so is the nature and complexity of the underlying process. The handling of dynamicity in an emulated environment needs to consider both the need for realistic models and the constraints of emulation.

Although off-line precomputation allows the use of computationally intensive and accurate propagation and mobility models, this can in many cases cover only a part of the dynamicity in the system. Wireless behaviors such as the hidden or the exposed terminal are related to the traffic that crosses the wireless network, and cannot be computed off-line without a traffic model. One of the major advantages of emulation is that it does not need a traffic model. An off-line simulation can however produce a dynamic scenario of emulation commands describing the evolution of the wireless network topology. This dynamic scenario can then be played in real time by a traffic shaper on an emulation testbed.

C. Repeatability

The third issue that is highlighted in this paper is repeatability. Repeatability can be viewed from multiple viewpoints. On the highest abstraction level repeatability is the ability to identically reproduce any given experiment setup in order to validate the results of an experiment performed earlier. The factors that may cause differences between different runs of an identical experimental setup should be minimized. On a lower abstraction level this translates to the amount of control over the experimental environment that is possible to achieve. From a repeatability standpoint the maximum possible amount of control of the emulated network is desirable. An analysis of the cause for indeterminism in current network emulators gives at hand that one major contributor is the stochastic insertion
of packet losses. When an experimental setup contains a non-congestion-related packet loss element, the random positions of these losses will vary between runs and this variance can have considerable impact on the individual experimental results as is shown in [12]. For instance, if a user wants to compare two error control mechanisms such as FEC or hybrid ARQ, it is necessary to ensure that packet losses will be exactly the same in both cases. If losses do not occur on the same packets, it is hard to examine the behavior of the two error control mechanisms in a fully repeatable fashion and it is also harder to conclude which one has the best performance. Similar arguments hold for bit-errors, for instance in the context of audio/video codec development.

To ensure a maximum amount of control over losses and bit errors, deterministic error positioning is needed.

IV. A SOLUTION ARCHITECTURE

This section describes W-NINE, an emulation platform that addresses the issues of Section III. As depicted in Figure 1, W-NINE integrates and enhances various existing tools. This integration work was done within the framework of the NEWCOM Network of Excellence.

W-NINE is mainly based on an off-line simulator called SWINE and an emulation platform. The current version on W-NINE has evolved from the earlier version, described in [13], mainly by employing the KauNet emulator extensions instead of plain Dummynet. KauNet is used in the central router of a physical platform called NINE. When a user wants to test a protocol or an application, he first writes a high-level description of the experiment which is processed by SWINE to produce an emulation scenario. Then the experiment can be run in real-time on the NINE platform: the applications and protocols to be tested are deployed on NINE terminals, while the emulation scenario is played in real-time by KauNet, which constrains all the IP traffic issued by the terminals in real-time. The behavior of these tools is detailed throughout the remainder of this section.

a) SWINE: The Simulator for WIreless Network Emulation [13], [14] is a model-based Java simulator that produces an emulation scenario based on a high-level description. The emulation scenario and high-level description are both XML files that must be validated against schemas. In this high-level description file, researchers describe the wireless environment to emulate as well as the Java models used for this purpose.

SWINE’s architecture is split in three main steps: Mobility, Propagation and Communication, each step hosting a number of models with an open architecture that allows the easy addition of new and/or more realistic models. The Mobility step computes all positions of all nodes at every time step of an experiment, using models ranging from the classical random waypoint mobility model to group mobility models such as the Pursue model. The Propagation step uses the positions computed by the mobility step to compute the power of the radio signal received by each node from every other node at every time step of the experiment. Classical propagation models are implemented, including the Rayleigh and Rice fading as well as Pathloss Exponent models. The Communication step uses the propagation information to compute the QoS parameters on each link and route at every time step of the experiment. These QoS parameters form an emulation scenario that can be repeatedly played by KauNet. The Communication step currently computes the maximum available bandwidth on a link with a time step of typically 100ms, as well as KauNet loss patterns with a granularity of 1 ms. These QoS parameters are then used to form the emulation scenario, which consists of events describing changes in QoS conditions on each link.

b) The Emulation Manager: It is a standalone application that reads an XML emulation scenario and sends ipfw update commands to KauNet at the appropriate time. When SWINE is configured with a 100 ms time step, updates may be sent up to every 100 ms. An update may then trigger the loading of a 1 ms-grained loss pattern into KauNet and/or the application of regular Dummynet rules. In order to minimize the memory consumption of loss patterns in the emulator, several loss patterns are used during the experiment and loaded and unloaded under the control of the emulation manager. Updates are sent through the administration network so as to not interfere with the experimentation traffic.

c) KauNet: KauNet is an extension of Dummynet that provides the ability to precisely place both packet losses and bit-errors at specific locations in a data transfer to examine transport layer protocol implementations and also application layer effects. The losses and errors can be placed either as a function of the amount of data transferred (data-driven mode) or as a function of time passing (time-driven mode).

When performing evaluation of implementation correctness it is important to be able to have a large degree of repeatability to recreate experimental runs that produce anomalous effects, so that these can be studied in greater detail and then hopefully corrected. As mentioned earlier, the placement of packet losses can have a large impact on the behavior of both transport layer protocols and also application behavior. While the basic protocol mechanisms can be studied by injecting losses in a controlled way using a simulator, this does not help to verify the behavior of actual implementations. KauNet was developed to provide a tight control over the placement of losses during a live experiment involving real implementations and real traffic.
V. CASE STUDIES

In this section, three different case studies illustrate how W-NINE addresses the repeatability, dynamicity and accuracy issues by applying the principles discussed in Section III. First, the KauNet extension implementing pattern-based packet losses is used to improve experiment repeatability. Second, dynamic QoS for a mobile user on a wireless link is enforced by the use of an emulation manager. Third, packet losses for a mobile user are computed by the SWINE off-line simulator to enhance the accuracy.

A. Case Study #1: Improving repeatability

1) Validation of a Protocol Implementation: In this experiment we examine the recent FreeBSD 6.0 TCP implementation, performing a validation of a limited set of protocol functionality. This is a production release which is used in a large number of computers around the world in a variety of conditions, and inaccuracies in the implementation can potentially have an impact on a large number of users.

There are several approaches to validating an implementation. Since the KauNet extension allows us to precisely place packet losses this is the approach that we focus on here. In order to evaluate the loss handling and congestion control code for inaccuracies and anomalies, a short TCP transfer of 25 Kbytes is performed. One packet loss is inserted at a specific position by the KauNet extension, and this position is then shifted around at all possible positions in subsequent runs. The emulated experiment was performed for a large number of bandwidth and delay combinations. In this section we show only one result from this evaluation, for a case with a low bandwidth and low delay link, as could for example be used for wireless communication in sensor networks. The bandwidth is 40 kbps and the link delay is 10ms. Since these networks are often battery-constrained, an efficient transfer and minimization of retransmissions are important to maximize network lifetime.

2) Emulation Results: When looking at the behavior of the FreeBSD TCP an anomaly is clearly visible for this case. The result for a 40 kbps link with 10 ms delay is shown in Figure 2. The figure shows both the values that could be expected from a well behaved TCP-implementation and the problematic results of FreeBSD 6.0. For a well-behaved TCP implementation the line evens out after loss position 20 since all packet containing data have then been transferred. For FreeBSD, losses after position 20 still have an impact providing a first indication that FreeBSD transfers more packets than necessary. Also, there are some notable peaks at loss positions 13 and 19. Using the ability to place losses at specific positions this anomaly was investigated. Additional experimental runs and examination of the implementation code revealed that this anomalous behavior can be traced to the TCP sender’s improper use of the SYNACK-ACK pair for RTT estimation. Since these early packets do not contain any data their transmission times are much lower than for the later regular data packets. For the examined case this large time difference causes the RTT estimation to be too low, so that the additional transfer time incurred by a later packet containing data actually leads to a timeout. This timeout causes unnecessary retransmissions and additional follow-up problems. A more detailed discussion about this specific case is outside the scope of this paper, but this experiment serves to illustrate how a protocol implementation can be verified and examined with precisely placed losses in a repeatable manner.

B. Case Study #2: Improving Dynamicity

1) Losses on a wireless 802.11 link: On a wireless 802.11 link, packet losses occur much more often than in wired networks. In unicast communications, the MAC level recovers a number of lost data frames by a retransmission mechanism based on acknowledgements. When consecutive losses occur in spite of retransmissions, an additional mechanism is triggered: the auto rate fallback algorithm, which switches to a more robust radio signal modulation at the physical layer, thus reducing losses but offering a lower transmission rate. The combination of MAC-level retransmissions and the auto rate fallback makes unicast communications much more reliable, however at the expense of decreased data throughput and increased jitter and latency.

In multicast mode, MAC data frames are not acknowledged and therefore neither packet retransmissions nor the auto rate fallback algorithm are activated. Consequently, packet losses are frequent. These losses can be characterized by a packet loss rate (PLR). The PLR depends on the location of mobile nodes as well as on propagation parameters and it can increase very quickly due to radio signal attenuation with distance.

This case study illustrates the ability of W-NINE to set up experiments with dynamic changes of the PLR over time.

2) Emulated Experiment: In the experiment illustrated by Figure 3, a mobile user receives a multicast UDP data flow while going through different areas of our offices in ENSICA. At the beginning of the experiment, the user starts close to a stationary sender (F1). Then, he follows the path $1 \rightarrow 2 \rightarrow$
3 → 2 → 4 → 5 with a speed of 1.5 m/s. Along this path he crosses areas with different packet loss rates (PLR): at $t = 0$ it starts in his office (1) with PLR $= 0\%$; at $t = 2.3$, he enters the corridor (2) with PLR $= 10\%$; then goes into the lecture room (3) with PLR $= 50\%$ at $t = 12.9$; after that he leaves the lecture room at $t = 15.7$ and goes to the secretariat (4) with PLR $= 30\%$ at $t = 21.9$; finally he leaves the secretariat to go to the exit (5) with PLR $= 60\%$.

These PLR values were chosen in a purely intuitive way as a function of the distance and the number of walls between the sender and the receiver. We will see in case study #3 how to compute the PLR according to propagation models. Here, SWINE computes time-driven loss patterns according to these PLR changes. These patterns are loaded by KauNet at emulation time under the control of the emulation manager.

While the user is moving, the sender generates a multicast UDP flow with 1472-byte long packets in CBR (constant bit rate) mode. The CBR throughput is 4.19 Mb/s which is the theoretical maximum throughput for physical transmission rate of 5.5 Mb/s in multicast mode. This 4.19 Mb/s throughput was used so that packet losses can only result from the emulation of wireless losses and not from buffer overflow on the emulation node.

Before emulation time, the PLR values are used to generate small-sized KauNet loss patterns. At emulation time, the emulation manager updates the pattern to be played every 100 ms. All patterns are used in a data driven way so that the same packet can still be lost from one run to another, as in case study #1. The MGEN [15] traffic generator is used to send the desired CBR traffic (4 Mb/s multicast UDP) and to collect the packets received by the user.

3) Emulation results: As expected, the effective PLR measured against the MGEN reception traces dynamically changes over time, and these changes arise at the expected time (see Figure 4). The high loss rates between $t = 12.9$ and $t = 15.7$ correspond to the position of the node in the lecture room and the loss rate after 21.9 s occur when the user is in secretariat. Finally, the higher loss rate is reached after $t = 32.6$ when the user reaches the exit. The measured PLR in each of these areas is close to the expected 50%, 30% and 60%. Variations that are observed on the curve are due to TRPR, the tool used to generate the curve based on MGEN traces, which provides an average PLR on a specified time interval (here 500 ms), and to the limited number of packets that are transmitted during this time interval.

C. Case Study #3: Improving Accuracy

1) Modelling Packet Losses on a Wireless Link: In case study #2, packet loss rates have been set up arbitrarily. This kind of solution can be used to see the behavior of an application or a protocol under evolving conditions but cannot be considered realistic: for instance, the attenuation of the radio signal in function of the sender-receiver distance is not realistically modelled. A solution to this problem consists in using a propagation model to compute the evolution of the PLR over time. Fortunately, researchers have developed a number of propagation models that match many different environments and radio technologies [11]. We use a combination of a pathloss exponent model and a Rayleigh fading model to provide a more realistic model for an indoor environment.

The following experiment shows that a combination of mobility and propagation models can be used to provide more realism when emulating a mobile wireless LAN than in case study #2. Moderately complex models are used here, but much more complex ones could be used to further enhance realism without compromising real-time constraints, since computations based on these models are entirely done off-line before emulation time.

2) Emulated Experiment: We use the same location and mobility model (a predefined path) as in case study #2, i.e. ENSICA’s offices illustrated in Figure 3. However, the radio signal propagation conditions are described by a combination of pathloss exponent and Rayleigh fading models.

The parameters of the pathloss exponent model are $n = 5.68$, $d_0 = 1$ m and $PL(d_0) = 19.97$ dB (estimated with the Friis free-space pathloss model). Those parameters were computed in our labs through measurements of the signal/noise along the same path as the mobile node M1.

3) Emulation Results: Figure 5 shows that the propagation effects computed by SWINE emulate the evolution of the PLR over time in a much more realistic way than in case study #2: the overall shape of the PLR curve is a function of distance,
and the small variations of the curve are a consequence of the Rayleigh fast fading model, which takes into account the radio signal attenuation and reflections on walls and obstacles. The maximum PLR is now reached after 26.2 s when the user leaves the secretariat and goes to the exit, while in case study #2 the maximum PLR was located in the lecture room.

VI. Conclusion

In this paper we have presented an emulation framework that improves accuracy, dynamicity and repeatability of the emulated network conditions of a given experiment. Three simple distinct case studies have illustrated these improvements to diagnose new classes of problems (case study #1) or to reproduce realistic conditions over time (case studies #2 and #3). Globally, this approach allows the designer to test real protocols, applications and traffic in real time without forcing the user to sacrifice modelling accuracy. This provides a unique opportunity to test “black-box” applications and/or protocols (such as the new FreeBSD 6.0 TCP implementation of case study #1) under both realistic conditions and limit conditions. Such black-box implementations are at least hard or even impossible (e.g. for copyright reasons) to model using classical simulation platforms.

The proposed framework is based on a two-stage process. In the first stage, IP-level emulation conditions (i.e. bandwidth, delay and loss patterns for all links) are simulated off-line, which allows the use of the most appropriate models to describe the mobile wireless network to emulate, no matter how complex these models are. The SWINE offline simulator produces an emulation scenario that is played in the second stage by an emulation manager, and it also produces loss patterns for the KauNet extension of the Dummynet traffic shaper. The emulation manager is in charge of periodically sending update commands to KauNet which reproduces the corresponding conditions. KauNet shapes the incoming IP traffic in terms of bandwidth, delay and packet loss rate, and additionally provides support for loss and bit-errors patterns which ensures the precise placement of losses/bit-errors needed for repeatability.

In future work we plan to integrate the ability to have also delay and bandwidth changes controlled by patterns and not only by the emulation manager. This can improve scalability, and the enhanced capabilities can for example be applied in cellular networks to emulate handover delay and bandwidth evolution in a very detailed manner. Furthermore, we plan to continue to evaluate the performance limitations of our platform with regards to the number of concurrent connections and the maximum bandwidth that can be emulated.

REFERENCES