# An Overview of OpenAirInterface Wireless Network Emulation Methodology*

Hicham Anouar, Christian Bonnet, Daniel Câmara, Fethi Filali, Raymond Knopp
Eurecom
2229, route des Crêtes
06904 Sophia Antipolis, France
first.lastname@eurecom.fr, openair_tech@eurecom.fr

## ABSTRACT

The OpenAirInterface wireless network emulator, a tool with the dual objective of performing protocol and application performance evaluation, in addition to real-time layer 2/3 protocol implementation validation, is described. The current example protocol implementations closely resemble those of evolving UMTS-LTE and 802.16e/m networks with the additional possibility for creating mesh network topologies. They do not provide any form of compliance, however, with these standards. The emulation environment comes in both real-time and non-real-time flavors based on RTAI/Linux open-source developments. Novel ideas for physical layer (PHY) abstraction are also reviewed.

## 1. INTRODUCTION

Performance evaluation of protocols and applications for wireless networks is typically done through the use of regular simulators, small testbeds, or costly protocol testers. In the first approach, even if relatively large networks are simulated, abstractions of some crucial parts of the network stack are made in order to make the simulation feasible. This abstraction can hide important issues that unfortunately may only be revealed when the software is implemented and deployed on a large scale. The use of testbeds, on the other hand, is not only expensive but also the measurements produced on them are hard to predict and reproduce. The best environment to evaluate applications and protocols for wireless networks would one where one could use a real network stack (e.g. provided by Linux or BSD), which is easy to configure, with nodes deployed predictable way and that results could be reproduced to verify the true difference between two different solutions. The emulation suite described here intends exactly to fulfill these requirements. It is part of the OpenAirInterface development platform [1].

The main difference between the methodology used here with respect to existing open-source simulation tools is firstly that it is built with a real-time framework in mind, using the open-source real-time extension to Linux, RTAI [2], and secondly that it is part of a validation chain for a real protocol implementation. With respect to more generic simulation tools such as ns2 [3], the study of Cavin et al. [4] shows that even the simulation of simple protocols, using different simulators, may yield significantly divergent results. Indeed, as the simulators implement different models for the MAC and physical layers, the result of the simulators say as much about the simulated protocol, as it does about the particular lower level implementation of the simulator. Offline discrete simulators, such as ns-2, OPNET and Glomosim are highly flexible and scalable, but event based simulators hardly will be able to represent real-time applications without deviations . Even though it is possible inject real time data in the NS simulation and it is planned to ns3 to have emulation support for traffic [5], these features are limited to the network layer only. Our approach goes a step beyond that, the proposal is to use the real stack to perform more realistic and reliable simulations. This also reduces development time since the code can be ported to the final real-time implementation with very little redesign.

A similar approach to ours, in the sense of real-time behavior, is based on an extension of ns-2 and is described in [8]. Both virtualization of nodes using the user-mode linux (UML [9]) framework and distributed simulation of large networks over Ethernet is provided along with the ability to run real Linux applications on top of the simulated network. This differs from our approach in two ways; firstly that hard real-time framing constraints are not respected (since a real-time operating system is not used), and secondly that the medium-access layer (MAC) protocols are modeled in addition to the physical-layer (PHY). While for 802.11 networks this remains a reasonable approach, due to the simplicity of the MAC and PHY specifications, for the evolving 802.16e/m [6] and UMTS-LTE [7] air interface specifications this is much less the case. Our approach relies on a full real-time implementation of the MAC protocol and accurate modeling of the PHY.

The paper is organized as follows: section 2 provides an overview of the OpenAirInterface emulation architecture, section 3 provides a description of how we make use of PHY abstraction techniques and some concrete examples for emerging air interface technologies. Finally in we present some conclusions and directions of our ongoing work.

## 2. OPENAIRINTERFACE EMULATION ARCHITECTURE

The OpenAirInterface emulation environment can be configured for real-time PC-based targets and user-space non-real-time PC based targets. Both allow for virtualization of network nodes within physical machines and distributed

*This research is partially supported by Eurecom's industrial partners: BMW, Cisco Systems, France Telecom, Hitachi Europe, SFR, Sharp, ST Microelectronics, Swisscom, Thales. The research work leading to this paper has also been partially supported by the European Commission FP6 project CHORIST and by the French RNRT project APOGEE.

deployment on wired Ethernet networks. Virtualization is done within the same operating system instance (i.e. we do not need to make use of virtualization tools such as UML, although in some cases the use of such OS virtualization tools can help for the development of layer 3 protocols) and the Linux IP protocol stack is shared among nodes in the same physical machine. Nodes in the network communicate via direct-memory transfer when they are part of the same physical machine and via multicast IP over Ethernet when they are in different machines. The communication between nodes allows for the exchange of transport data at the PHY and MAC interface, the so-called *transport channels* in ETSI UMTS and LTE terminology, see [7]. Nodes filter MAC-layer PDUs on reception based on radio measurements which are locally simulated, in the sense that channels that are not destined for a particular receiving node are dropped. The presence of a particular channel is potentially used, however, in the calculation of interference in the PHY abstraction entity discussed in section 3.

The real-time version of the emulator is designed to represent the behavior of the wireless access technology in a real network setting while obeying the temporal frame parameters of the air-interface. It makes use of the open-source real-time operating system extension to Linux, RTAI [2] to guarantee hard real-time behavior. With virtualization of the protocol stack, many instances (on the order of 30 on a 2GHz Quad-core Xeon) can reside in the same physical machine. A typical setup for a large-scale emulation would consists of several PCs in a cluster network each housing tens of virtual nodes. At Eurecom, 8 Quad-core Xeon servers are used to this end. The layer 3 networking protocols reside in the standard linux kernel or user-space and are interfaced using a custom networking device driver. In a typical large-scale emulation scenario a combination of real applications and traffic generators (such as mgen [10] or iperf [11]) would be used. This targets large-scale repeatable emulations on a real protocol stack using real applications.

The non real-time version of the emulator runs in normal Linux user-space and kernel-space and maintains true frame times *on average* if the CPU processing power is sufficient and the Ethernet network is fast enough. Execution is in non-real time but with a true Linux networking device so that higher-layer protocols (routing, mobility management, etc.) can be integrated into the emulation if needed. It should be mentioned that although the user-space emulation is non-real-time, in the sense that radio frame timing is not guaranteed, real applications can still be executed on the user-space emulator. The user-space mode is particularly useful in debugging a network deployment prior to large-scale simulations or when used with real OpenAirInterface radio equipment (see [1]). Additionally, it is a necessary prototyping step during testing of upgrades to the protocol stack since standard GNU-Linux debugging tools can be used.

## 2.1 OpenAirInterface Example Protocol Stack

OpenAirInterface[1] provides a complete wireless protocol stack and radio hardware. It is an open-source hardware and software initiative for collaborative innovation in the area of digital radio communications funded primarily from public sources. OpenAirInterface implements the PHY, MAC, RLC(Radio Link Control), RRC(Radio Resource Control) layers as well as providing a IPv4/IPv6/MPLS network device interface under Linux. In addition to the protocol emulation environments described here which do not make use of radio hardware, fully-functional real-time two-way RF hardware (5 MHz channels at 1.9 GHz) is provided and has been made available to partner institutions. The initiative now targets 4th generation wireless systems (UMTS Long-term-evolution (LTE), 802.16e/m) and rapidly-deployable MESH networks using similar, yet simplified, radio interface technologies. The OpenAirInteface example protocol stack is depicted in Figure 1 and comprises C-language implementations of the protocol stack and the PHY abstraction unit each corresponding to a particular node in the network. The development currently targets generic Linux PC-based hardware. Embedded FPGA-based system-on-chip (SoC) targets are in the implementation phase.

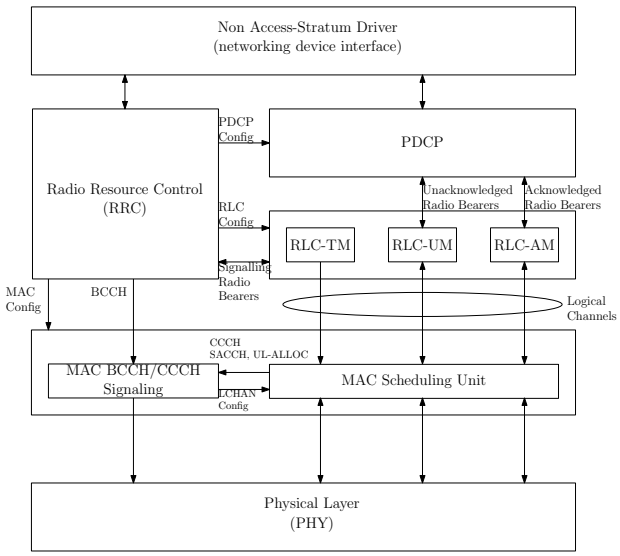


**Figure 1: OpenAirInterface Protocol Stack**

The protocol suite allows for mesh or cellular network topologies. The linux network device provides IP and MPLS (based on Linux-MPLS [12]) interconnection and permits quality-of-service (QoS) classification (packet filtering) of IP and MPLS traffic on layer 2 resources.

# 3. PHY ABSTRACTION

PHY abstraction is a procedure done at the receiver of each node in order to firstly compute the error statistics of the received packets before delivery to the MAC, and secondly to generate measurement information for radio resource management algorithms (scheduling, call-admission, adaptive coding and modulation, power control, etc.). Subband signal strength indicators are computed every transmission frame (sub-frame in LTE terminology) based on the RF topology and pre-defined propagation models. The function of the abstraction module can be system dependent (i.e. based on precomputed probability of error simulations for specific modulation and coding formats) or generic based on semi-analytical formulas. As in the 802.16m methodology described in [13], the output of the radio simulation is random PDU loss indicators for each transport channel block traversing the PHY/MAC interface.

In each radio frame (sub-frame), the PHY_Abstraction

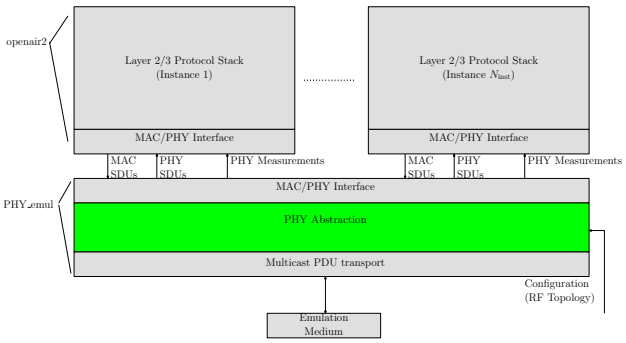


**Figure 2: PHY Abstraction Module**

unit analyzes the set of received SDUs from the emulation medium and determines those which are sources of information and those which represent interference. The target PHY SDUs to be received are those programmed by the MAC as in the case when operating with a true PHY. The interferers, however, are naturally present with the true PHY (i.e. in the signal itself) and thus their impact must be simulated in the abstraction unit. Since a particular node in the network is not aware of all sources of interference *a priori*, this is done by adding a physical resource description to each transport block in the emulation medium which is not present in the real PHY. With this additional information both accurate signal and interference powers can be simulated.

A secondary task of the PHY abstraction unit is to provide the stimulus on the measurement interface for the Layer 2 protocol stack. In system simulations this is required for validation of the Layer 2 mechanisms related to adaptive resource control (modulation and coding adaptation) and wideband resource scheduling which are typically both functions of the MAC-layer scheduling algorithm. Eventually, these measurements also make their way up to Layer 3 protocols which are responsible for connection management and admission control procedures.

The environment of the PHY abstraction unit is shown in Figure 2. It takes input from the emulation medium (Ethernet or direct memory transfer) corresponding to the MAC-layer SDUs from corresponding nodes in the network. These SDUs correspond to transport blocks for different transport channels to be encoded by PHY or just decoded by PHY. It also receives information from an RF topology server regarding slowly-varying propagation parameters (mobility, path loss/shadowing models, multipath intensity profiles, etc.). The other end implements the PHY/MAC interface in order to interconnect with the true layer 2 protocol stack.

# 4. A CONCRETE EXAMPLE OF PHY ABSTRACTION (MIMO/OFDMA)

Consider the following example for PHY abstraction at node $j$ in the network. This targets a wideband multicarrier multiple-input multiple-output system, for example OFDMA or single-carrier FDMA as in UMTS-LTE [7]. Let $\overline{\text{RSSI}}_{i,j}[n]$ be the average received strength in frame (sub-frame) $n$ between node $i$ and node $j$. Its' temporal variation represents mobility of the node or the environment around it. This can be generated locally in each node based on a model for mobility or can be signaled by a topology server dynamically. As in the 802.16m simulation methodology [13] the goal of the PHY abstraction entity is to simulate the block error rate process (BLER) of each transport block of a particular received resource. Evaluation of the BLER boils down to the evaluation of a function of the statistics of the received signal and interference vectors at node $j$. Let $\sqrt{\overline{\text{RSSI}}_{i,j}[n]}\mathbf{H}_{i,j}[n,k]$ be the spatial channel matrix of dimension $M(i) \times M(j)$, with $M(i)$ being the number of antennas for node $i$ in frequency band $k$ for the signal from node $i$ to $j$ in frame (sub-frame) $n$. Also let

$$K_{\text{I},i,j,m}[n,k] = \sigma^2 + \sum_{i' \neq i, i' \neq j} \sum_{m=0}^{M(i')-1} \sqrt{\text{RSSI}_{i',j}}[n]\mathbf{h}_{i',j}[n,m,k]\mathbf{h}_{i',j}[n,m,k]^*, \quad (1)$$

be the second-order statistical description (co-variance matrix) of the received vector, where $\mathbf{h}_{i',j}[n,m,k]$ is the spatial channel column-vector for transmit antenna $m$ in band $k$ corresponding to interferer $i'$. In a system like UMTS-LTE the frequency index would correspond to a *resource block* which is a group of contiguous OFDM sub-carriers representing the smallest entity which can be allocated by layer-2 scheduling procedures.

The random variables $\mathbf{H}_{i,j}[n,k]$ and $\mathbf{h}_{i',j}[n,m,k]$ depend on the space/time/frequency description of the propagation environment, more specifically a second-order description of the power-delay profile (PDP), the Ricean factor which relates the energy of direct path and the reflected paths, the antenna correlation, and the mobility (see [14] for details on these subjects). As is common in the literature and radio channel simulators, these are usually modeled using Gaussian deviates each frame (sub-frame) based on previously generated channels and modifications to the propagation environment.

## 4.1 Characterization of the BLER

Obtaining an accurate description of the BLER as a function of $\sqrt{\overline{\text{RSSI}}_{i,j}[n]}\mathbf{H}_{i,j}[n,k]$ and $K_{\text{I},i,j,m}[n,k]$ is the key issue in PHY abstraction. The description is strongly dependent on the coding and multiple-access techniques. Here we outline a few cases which highlight the key issues in system simulation of emerging air interface technologies.

Consider first the simplest form of transmission where feedback information on return channels is not used (e.g. BCH in LTE[7]). This is typically the case on broadcast channels or basic signaling channels.The first issue is to define a particular receiver structure (bit-interleaving metric, minimum-mean squared-error receiver, etc.) for which wideband signal-to-interference-and-noise ratio (SINR) expressions (similar to those considered in 802.16m[13] can be derived from the above second-order interference description. These are then used for BLER lookup based on tabulated performance of a particular coded-modulation scheme. This consists of running a series of computer simulations of BLER vs. SINR curves for the given average SINR. The difficulty in this approach is that SINR is a vector which possibly characterized by a significant number of degrees of freedom and thus the offline procedure could be very time-consuming. Another alternative would be the use of PHY-agnostic information-theoretic bounds based on information-outage probabilities under the assumption of either Gaussian codebooks or finite QAM constellations (see

for example [16]. In the first case, reasonably simple analytical formulas can be used, in the second tabulated precomputed data must be generated. A third PHY-agnostic possibility would be to use finite block-length error-rate expressions (upper and lower bounds) based on random coding experiments. This is the most challenging approach from a research perspective. For example, under the assumption of Gaussian transmit signals and a particular receiver structure (e.g. MMSE which is optimal for Gaussian statistics) a model for achievable BLER (on each transport block sent by PHY to MAC) could be bounded as

$$P_e\left(\left\{\sqrt{\overline{\mathrm{RSSI}}_{i,j}[n]}\mathbf{H}_{i,j}[n,k]\right\}\{K_{\mathrm{I},i,j,m}[n,k]\}\right) \leq K_{\mathrm{impl},i,j}2^{-N_{\mathrm{TB}}(R_{i,j}(n)-R_{\mathrm{MAC},i,j}(n))} \quad (2)$$

where $K_{\mathrm{impl},i,j}$ is an SINR-dependent implementation degradation factor, $R_{\mathrm{MAC},i,j}(n)$ is the allocated code rate in bits per dimension by the MAC layer scheduler in frame (sub-frame) $n$ and

$$R_{i,j}(n) = \frac{1}{|\mathcal{A}_{i,j}(n)|}\sum_{f\in\mathcal{A}_{i,j}(n)}\log_2\left(1+\mathrm{SINR}_{i,j}[n,f]\right)$$

where $\mathcal{A}_{i,j}(n)$ is the set of frequency resources allocated in frame (sub-frame) $n$ and $\mathrm{SINR}_{i,j}[n,f]$ is the SINR of the chosen receiver structure in frame (sub-frame) $n$ and frequency-band $f$.

With HARQ-based schemes, block errors at a particular time also depend on past values of the signal and interference components. Here additional protocol information from the MAC signaling channel header are required, but well-known semi-analytical models can be used to describe the BLER on a particular transport block as a function of the round index of the HARQ protocol and the current and past SINR values. The can be used for both repetition coding with chase combining (type I HARQ) or Incremental redundancy (type II HARQ). Again this can be done using either tabulated BLER performance curves for the coded-modulation scheme under test or information-theoretic PHY-agnostic formulas. In the latter case, equivalent formulas to those described in the previous section can be applied (see [15] for details).

When precoding is performed based on channel state feedback at the transmitting end, be it linear or non-linear, additional PHY layer information must be transported in the emulation process along with MAC information, namely the linear/non-linear spatial filtering description at the sending nodes. This is required to compute the received SINR at the nodes which now depends, in addition to the channel, on the spatial filtering done at the transmitter. Once this information is incorporated into the SINR characterization, the methods described above can be employed for PHY abstraction.

# 5. CONCLUSIONS

We described the The OpenAirInterface wireless network emulator, a tool with the dual objective of performing protocol and application performance evaluation, in addition to real-time layer 2/3 protocol implementation validation. A brief overview of the current example protocol implementations was also provided. These closely resemble the specifications for evolving UMTS-LTE and 802.16e/m networks with the additional possibility for creating mesh network topologies. In principle, any layer 2 protocol suite could be implemented or adapted to use our methodology. We also briefly described some novel ideas pertaining to physical layer (PHY) abstraction procedures which can be used in wireless network emulation. Our current work focuses on demonstrating the scalability of these techniques in real-time emulation of wireless networks comprising hundreds of nodes distributed on a PC cluster with a focus on efficient and accurate PHY abstraction methods. The target experimentation includes layer 2 scheduling algorithm development, layer 3 mobility and handover protocols as well as advanced radio resource management strategies for 4th generation cellular networks and rapidly-deployable mesh networks for public-safety applications.

# 6. REFERENCES


[1] OpenAirInterface, `www.openairinterface.org`.

[2] Real-Time Application Interface, `www.rtai.org`.

[3] Daniel Mahrenholz and Svilen Ivano, "Real-Time Network Emulation with ns-2," *Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'04)*, Budapest, Hungary, October 2004.

[4] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of MANET simulators," *Second ACM international Workshop on Principles of Mobile Computing*, Toulouse, France, October 30 - 31, 2002.

[5] Thomas R. Henderson, Sumit Roy, Sally Floyd and George F. Riley, "ns3-Project Goals, Workshop on ns-2: the IP network simulator," Pisa, Italy October 10, 2006.

[6] IEEE 802.16 Task Group m, "IEEE 802.16m System Description Document",see `wirelessman.org/tgm`.

[7] P. Lescuyer, T. Lucidarme, "Evolved Packet System-The LTE and SAE Evolution of 3G UMTS," Wiley, 2008.

[8] D. Mahrenholz, S. Ivanov, "Real-Time Network Emulation with ns-2", *Proceedings of The 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications, Budapest Hungary,* October 21-23, 2004.

[9] User-Mode Linux Kernel, `http://user-mode-linux.sourceforge.net/`

[10] The Multi Generator, `http://pf.itd.nrl.navy.mil/mgen/mgen.html`

[11] Iperf, `http://sourceforge.net/projects/iperf`.

[12] MPLS-Linux, `http://mpls-linux.sourceforge.net/`.

[13] Project 802.16m Evaluation Methodology Document (EMD), 802.16 Task Group m, IEEE 2006, see `wirelessman.org/tgm`.

[14] William C. Jakes, Editor (February 1, 1975). Microwave Mobile Communications. New York: John Wiley and Sons Inc.

[15] H. El Gamal, G. Caire, M.O. Damen,"The MIMO ARQ Channel: Diversity Multiplexing Delay Tradeoff," *IEEE Transactions on Information Theory,* vol. 52, no. 8, 3601-3621, 2006.

[16] R. Knopp and P.A. Humblet, "On Coding for Block-Fading Channels", *IEEE Transactions on Information Theory,* vol. 46, no. 1, Jan. 2000, pp. 189-205.