ns-3-based Real-time Emulation of LTE Testbed using LabVIEW Platform for Software Defined Networking (SDN) in CROWD Project

Rohit Gupta and Bjoern Bachmann
National Instruments
Dresden, Germany
rohit.gupta@ni.com,
bjoern.bachmann@ni.com

Russell Ford and Sundeep Rangan
New York University
russell.ford@nyu.edu,
srangan@nyu.edu

Nikhil Kundargi, Amal Ekbal and Karamvir Rathi
National Instruments
Austin, TX, USA
nikhil.kundargi@ni.com,
amal.ekbal@ni.com,
karamvir.rathi@ni.com

Maria Isabel Sanchez
IMDEA Networks Institute
University Carlos III
Madrid, Spain
mariaisabel.sanchez
@imdea.org

Antonio de la Oliva
University Carlos III
Madrid, Spain
aoliva@it.uc3m.es

Arianna Morelli
INTECS, Pisa, Italy
arianna.morelli@intecs.it

ABSTRACT

In this paper, we present initial results on how the ns-3 LTE LENA stack is used to build a LTE testbed in an indoor lab network. We have extended the ns-3 MAC/PHY layer architecture to interface with a LabVIEW implementation of the LTE Physical layer and also extended ns-3 core modules to enable real-time performance. We present how this testbed can be used for prototyping a novel Software Defined Networking (SDN) scheme for interference management within dense heterogeneous deployments of cellular wireless networks. We provide an example case study for a Distributed Mobility Management (DMM) implementation using this testbed, where we demonstrate mobility of a emulated User Equipment device between LTE and WiFi networks. We plan to use this testbed for validation and demonstration of various SDN-based algorithms proposed within the framework of the EU FP7 CROWD (Connectivity management for eneRgy Optimised Wireless Dense networks) Project. We believe the proposed testbed is especially valuable for studying the cross-layer performance of cellular PHY/MAC algorithms in a realistic environment and shows how ns-3 can be used as a unified prototyping and simulation framework for wireless networks.

Categories and Subject Descriptors

I.6.5 [**Computing Methodologies**]: Simulation and Modelling —*Model Development*

General Terms

Algorithms, Design, Performance, Verification

Keywords

LabVIEW, Simulation, Hardware emulation, Software-Defined Radio, Software-Defined Networking, LTE, ns-3

1. INTRODUCTION

There has been growing mobile traffic demand due the explosion of diverse applications and their rising data requirements. As a result, there is significant interest from the wireless industry for denser heterogeneous deployments to improve coverage and data rates for mobile users. However, lack of coordination between neighboring cells can result in severe inter-cell interference, and coordinated transmission of base stations and users to control this interference is one the key challenges in the management of future wireless networks.

This scenario calls for deployment of agile network controllers to improve spectral and energy efficiency. In the FP7 CROWD project [1], we have adopted Software Defined Networking (SDN) as the paradigm for management of dense heterogeneous networks. Our approach is based on a two-level hierarchy of CROWD Local Controllers (CLC) and CROWD Regional Controllers (CRC) that are responsible for tuning the network characteristics on a local or regional scale, respectively. The algorithms proposed within the project span all layers of the protocol stack [2] and demonstration and validation of said algorithms requires an open testbed which allows for easy implementation of novel and flexible network control functions.

It would be quite challenging to incorporate this SDN framework into existing testbeds (see [3, 4]) due to the tightly-optimized integration and complexity of the LTE stack. Moreover, initial algorithm investigation, especially for PHY/MAC cross-layer design is usually done through system-level simulators, such as ns-3 [5]. Hence, we chose to extend the ns-3 LENA stack [6, 7] to enable real-time emulation and define a custom API for interfacing with an external FPGA-based LabVIEW PHY layer. The interested reader is referred to the overall system and physical layer architecture of the CROWD testbed presented in [8, 9, 10].

The main contribution of this paper is to expand on our previous work [8, 9, 10] and describe the changes needed within the MAC/PHY interface in the ns-3 LENA stack and core simulator to enable real-time emulation with the LabVIEW PXI platform. Distributed Mobility Management (DMM) is presented as an example use case for this testbed. In this experiment, we plan to operate a real WiFi network beside the emulated LTE network to show mobility between the WiFi and LTE networks using the CROWD SDN scheme.

The paper is organized as follows. Section 2 gives a brief overview of the CROWD architecture. Section 3 describes the extensions that were made to ns-3 PHY/MAC interface and also to ns-3 core sub-systems to enable real-time execution. Section 4 discusses Distributed Mobility Management (DMM) as a potential use case for this testbed within CROWD. Section 5 presents some initial results from ns-3 logs that demonstrate real-time execution and communication between an LTE eNB and UE. Finally, we conclude the paper with some comments on future work in Section 6.

2. CROWD ARCHITECTURE

In this section, we provide an introduction to the CROWD architecture, which is described in more detail in [1]. Figure 1 shows the system overview and interaction between different network components. The main goal of the framework is to unify network management functions using SDN. In this regard, we propose a two-tier hierarchy of SDN controllers, viz. the previously-introduced CLC and CRC. CROWD controllers are technology-agnostic and expose their *northbound interface* for control applications, whereas they interact with network elements using their *southbound interface*. CROWD Regional Controllers are responsible for wide area network management and executes network functions over long time scales. Each CROWD Local Controller runs short term optimizations and fine tunes the network over shorter time scales. The CROWD vision aims to provide a common set of functions as part of the SDN southbound interface which can be used by *control applications*, for example *LTE access selection* [11] and *LTE interference mitigation* [12], to configure network elements in dense deployments.

Figure 2 illustrates how the CROWD architecture is realized on top of our LTE testbed. The figure shows how the CLC collects statistics regarding channel state information (CSI) and traffic conditions from the eNB and also interacts with the eNB RRC layer and MAC scheduler to influence scheduling decisions at the eNB. The control applications interact with the CLC, providing an efficient and unified way to deploy a multitude of network management functions.

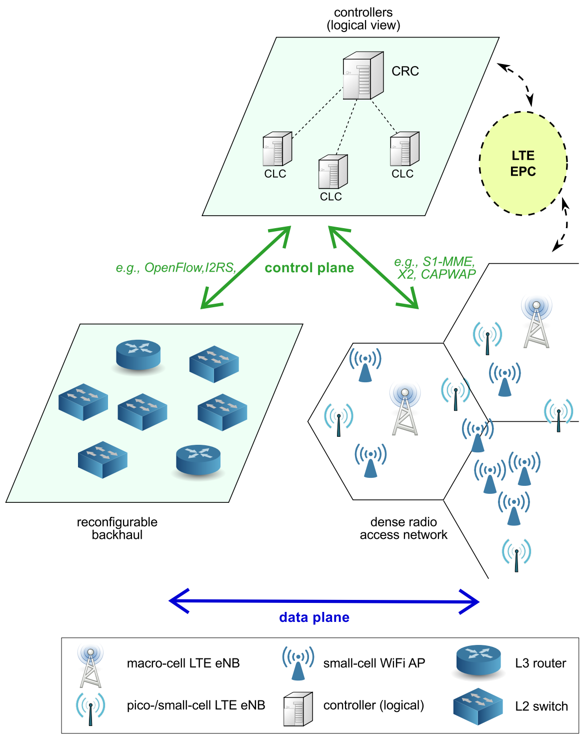

Figure 1: CROWD Architecture

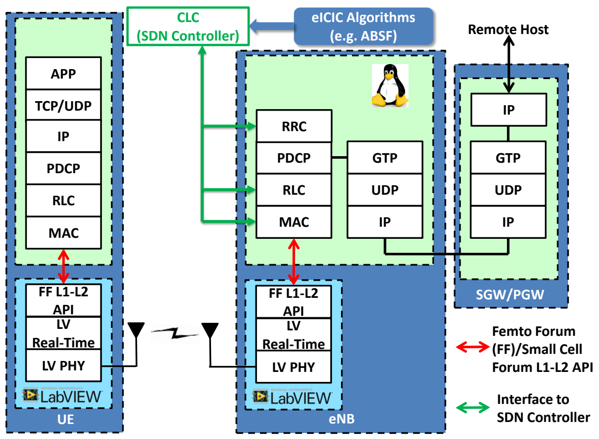

Figure 2: LTE MAC/PHY SDN Architecture for CLC controlling eNBs.

3. MODIFIED NS-3 MAC/PHY ARCHITECTURE

We now describe the modifications to the ns-3 core simulator to allow for real-time execution and to the LENA MAC/PHY code to enable interfacing with the LabVIEW PHY. Ns-3 already provides real-time emulation features that allow simulated TCP/IP networks to interact with real networks. Also the LENA module offers many protocol implementations that are mostly accurate to the LTE/EPC specification, making it one of the most complete open-source and re-configurable implementations of the LTE and core network stack available. The LENA module therefore represents an opportunity for researchers to leverage this source code to emulate LTE devices. However, to make ns-3 function as a wireless device emulator, several major changes to the software architecture are required.

Firstly, the Physical Layer and channel in ns-3 LENA are naturally a system-level simulation. The first component of this project is to completely replace the simulated L1 with a real implementation of the LTE PHY running on FPGA and RF hardware, which can perform the baseband processing in order to transmit and receive the ns-3-generated MAC data Transport Blocks and control signals over-the-air. Clearly this is a fundamental departure from the way LENA functions at L1. However, due to the layered and modular design of the LENA module, we were able to integrate our custom L1-L2 interface without drastic changes to many upper-layer classes, as we will show.

Furthermore, we exploit the built-in real-time core Discrete Event Simulation (DES) scheduler to synchronize LENA functions with wall-clock time. Still, to achieve true real-time performance with the timing constraints of LTE, multi-threading is essential. We shall discuss how we have offloaded certain expensive functions, such as logging and the marshalling and transfer of L1-L2 API messages, to multiple threads in order to increase throughput and reduce latency and jitter. Tight synchronization and low-latency communication between layers is also necessary, so we shall propose how this can be achieved within our UDP API.

3.1 PHY Layer Modifications

Figures 3 and 4 show an overview of the modified version of the LENA eNB stack. The `LteEnbPhy` and `LteUePhy` classes, which model Layer 1 in LENA, are replaced with the `NiLteEnbPhy`, `NiLteUePhy` classes, respectively. The `LteSpectrumPhy` class is bypassed entirely in NI-L1 mode, as `NiLteEnbPhy` and `NiLteUePhy` instances communicate directly with the LabVIEW PHY through the methods provided by the `NiLteL1L2APi` utility class.

For simplifying configuration of a standard LTE/EPC network along with the NI-L1 PHY interface, a modified helper class is provided, which is mostly identical to the LENA `LteHelper` class but omits all code related to the simulated PHY, including the channel model, interference model, antenna model, and TX gain configuration.

Here we only describe the modifications to the downlink, however the general structure and call flow for the uplink is nearly identical (although the control-plane procedures and messages that are generated are, of course, different).

3.1.1 eNB Classes

In the LENA module, the instance of `LteEnbPhy` is responsible for subframe generation from buffered data and control messages produced by the MAC Layer for each eNB. The simulated transmission of a subframe is initiated in *StartSubframe()*; the control period is transmitted at the beginning on the subframe where the data period transmission is delayed by 3 symbol periods. This method, which is repeatedly scheduled every TTI, also triggers the MAC to schedule UEs in DL and UL resources and, in turn, generate the corresponding control and data Transport Blocks for the next subframe.

We should point out that the synchronization between the subframe processing loop in the LabVIEW PHY and the ns-3 MAC layer is not done directly through the PHY triggering the MAC routines to perform scheduling and generate MAC PDUs. The two loops run independently on different hosts (i.e. the Linux laptop and NI PXI controller/FPGA) and their timing sources can be synchronized over Ethernet via Precision Time Prototol (PTP), which is supported by both Linux and the LabVIEW real-time OS. Triggering of the MAC by the PHY though subframe indication messages is currently being tested and will be included in future iterations of the L1/L2 API.

We define a new class, `NiLteEnbPhy`, where we also periodically call *StartSubframe()* to send data for the current subframe. However instead of calling into `LteSpectrumPhy` to transmit the control and data period over the simulated channel, we insert the list of downlink `LteControlMessage` objects and the data `PacketBurst` object for the current subframe into a thread-safe *TX Queue*, which is processed by a separate *eNB Message Handler Thread*. This thread is then signaled via a semaphore to generate and send a corresponding NI-L1 *TX Request* API message to the PHY using the methods provided by `NiLteL1L2Api`.

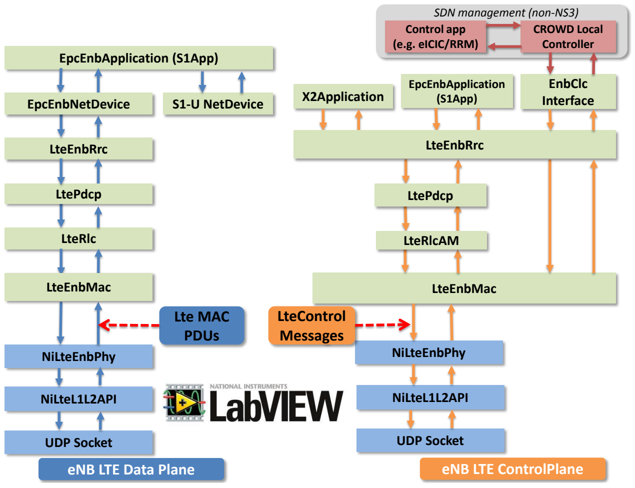

Figure 3: ns-3 LENA LTE eNB Architecture (modified)

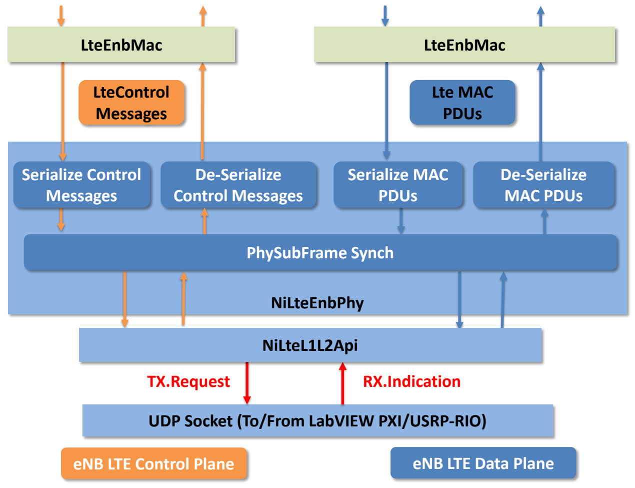

Figure 4: ns-3 LENA LTE eNB NI PHY Module (modified)

`NiLteL1L2Api` is a static utility class that has methods to automatically serialize and de-serialize LTE Control/Data message objects using the Boost serialization library. We elected to use Boost for the convenience of not needing to define our own serialization format and routines, however we later intend for all API messages to derive from the ns-3 `Header` base class, for which we will define a more compact encapsulation. The TX Request is built from the serialized `LteControlMessage` objects and data payload and is then sent over UDP to the LabVIEW PHY (using the Boost socket library non-blocking send), which has a corresponding thread that receives and parses TX Requests (among other API messages). The interested reader is referred to [9] for the detailed description of API and how it interfaces with the LabVIEW PHY.

3.1.2 UE Classes

When the UE successfully receives and decodes downlink TB data for a given subframe, the LabVIEW PHY will generate a *RX Indication* API message, which is sent over UDP to the Linux host running the UE ns-3 process. A *UE Message Handler Thread* instance listens on the UDP socket and receives, deserializes and parses out the RX Indication. The decapsulated downlink `LteControlMessage` and `PacketBurst` objects are enqueued in the control message list for processing by the UE MAC.

Similar to the eNB, the `LteUePhy` class is replaced by `NiLteUePhy`, which also has no relationship to `LteSpectrumPhy`. Another key difference between the original LENA class and the NI-L1 version is that the *SubframeIndication()* method is not initiated by the eNB PHY in *StartSubframe()*. Rather, the method runs (in a separate thread from the message handler thread) independently in a loop and waits on a condition variable, which is set by the message handler on reception of a RX Indication.

3.1.3 NI-L1 Common Control Channel

In our simplified design, only the DCI (the *DL_DCI* message type in ns-3 LTE module) contains parameters necessary for directly controlling the PHY. Therefore this control information must be transmitted as a separate physical channel, i.e. the PDCCH, which can be immediately received at the PHY and used for decoding the data channel. In our design, all other control messaging is only relevant to the MAC and higher layers. The operation of the MAC layer in ns-3 does, however, require control functionality normally provided by the other physical channels (i.e. the PBCH and PHICH), which have not yet been fully implemented in our LabVIEW PHY. As a temporary solution, we have devised a simplified control channel, referred to as the *NI-L1 Common Control Channel* (NI-CCH), which encompasses all control messaging beside the DCI. All *LteControlMessage* objects specific to the downlink are directly serialized, after which they are encoded (always as rate 1/4th QPSK) and regarded as simply another PDSCH Transport Block. The NI-CCH TB is mapped to the first N_{RB}^{NICCH} Resource Blocks (determined by the length of the serialized, coded data), which are decoded by all users and passed up to the UE MAC. While this may be viewed as a stopgap solution, for the purposes of over-the-air experimentation with a real-time ns-3 LTE module, it effectively serves its purpose of transmitting broadcasted messages, such as the Master Information Block, in a much simpler manner. The notable shortcomings of this method are its inefficient use of resources and it being an obvious deviation from the LTE standard. In future iterations, we intend to provide a more accurate LTE frame structure, with all of the basic physical channels implemented.

3.2 Integration with CROWD Local Controller (CLC)

One of the main goals of CROWD project is to show the application of Software-Defined Networking (SDN) concepts for dense LTE deployments. Hence, we have created extensions to the MAC scheduler to integrate with SDN based controllers, for example an OpenDayLight Controller [13] . However, the interface to the ns-3 LTE MAC scheduler is designed to be generic and support other SDN controllers as well. The main function of SDN controller is to collect network performance statistics and dispatch scheduling instructions from SDN applications running on top of the controller. Figure 2 shows the high-level overview of the CLC interface with MAC scheduler, whereas in Figure 5, we show our custom CROWD version of the Round Robin MAC Scheduler class in ns-3 LENA stack. We have essentially created a non-blocking UDP interface that transmits essential CQI and traffic statistics to the CLC controller, while at the same time it also receives MAC scheduling messages from CLC. Such messages include the Almost-Blank Sub-Frame (ABSF) pattern that instructs a particular eNB to mute its data-period transmission for certain TTIs for interference coordination. Here we list some key messages exchanged between eNB MAC scheduler and CLC:

i **DlCqiReportInd**: Sent from eNB MAC scheduler to the CLC to provide DL CQI reports.

ii **DlTrafficStatusInd**: Sent from eNB MAC scheduler to CLC to provide traffic status reports.

iii **SetMcsReq**: Sent from CLC to eNB MAC scheduler to set the MCS of scheduled UEs. UEs are identified using a Radio Network Temporary Identifier (RNTI).

iv **SubframeIndication**: Sent from eNB to CLC every TTI to enable synchronization between CLC and MAC scheduler.

v **SetAbsfPatReq**: Sent from CLC to eNB MAC scheduler to set the ABSF pattern. The ABSF pattern is determined by the ABSF application that interfaces with CLC through the CROWD northbound API.

The implementation of the CROWD architecture within ns-3 LENA predates the recent introduction of various frequency reuse classes, which interfaces with the MAC scheduler in a very similar way. In future work, we will investigate revising our approach to Radio Resource Management, presented here, to be more aligned with the direction of LENA.

3.3 Limitations of ns-3 for Real-time Operation

The ns-3 real-time scheduler has effectively all the hooks necessary for synchronizing execution with real-time clock. However, the object class and other data structures, for example (Packet and Packet Tag objects) are not thread safe. We modified ns-3 reference counting class `SimpleRefCount` to be thread safe by changing private data structure *m_count*

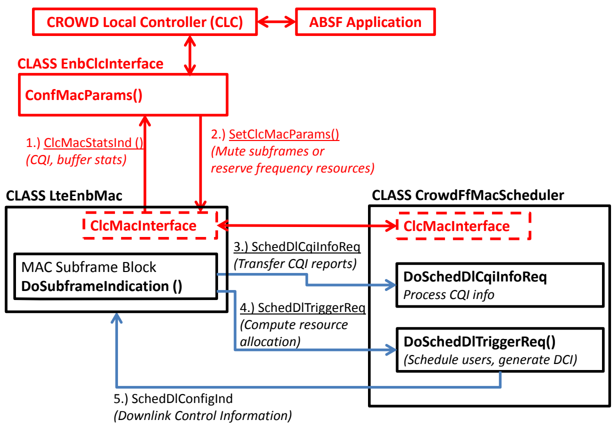

Figure 5: SDN-Based CLC Interface with ns-3 LENA LTE MAC Scheduler

to use the boost::atomic type. However, the packet tags are still not thread safe, which prohibits us to pass `Packet` objects with tags across different threads for more efficient serialization/de-serialization of LTE Control/MAC PDUs. We also set pthread priorities and use the real-time Linux kernel to prioritize ns-3 process and its threads for responsive execution.

3.4 Limitations of ns-3 for Real-time Logging

Ns-3 provides a rich logging framework, which enables module-based logging that can be easily activated and deactivated based on the needs of the simulation. However, the logging infrastructure outputs the log either to file or standard output, which can interfere with real-time execution. Hence, we perform this file output in a separate, low-priority thread, with logging data stored in shared memory between the main high-priority thread and standard pthread synchronization used for thread safety.

3.5 Limitations of ns-3 LENA LTE Stack for Interfacing with Hardware

We found LENA module to be well mostly suited for partitioning of the code so each eNodeB and UE node could be isolated to run in a separate process, with the exception of the *UeManager*. Currently, we have found some workaround for this issue, but separating some of the code in this way requires more pervasive architecture changes throughout LENA, which we plan to carry out with the help of LENA developers.

4. USE CASE: SDN-BASED DMM FOR MOBILITY MANAGEMENT

In this section, we present a SDN-based Distributed Mobility Management (DMM) implementation, which demonstrates CROWD Controller integration in the ns-3 LTE stack. The DMM scheme enables mobility between IEEE 802.11 and LTE networks by means of support at the host as well as network-based mobility management performed by the SDN controller. Media Independent Handover (MIH) or IEEE 802.11u is used for discovery of suitable access point by the client.

The scenario for the inter-technology mobility experiments conducted within CROWD project is shown in Figure 6. The figure represents the switching elements part of the SDN deployment in a single-technology network, which in the scope of CROWD is called a *district*. In our case, we have deployed a WiFi district that is controlled by a local controller (CLC) with several WiFi Access Points (APs) and OpenFlow-enabled switches. The local controllers in the different districts are coordinated by the regional controller (CRC). In every district, there is at least one DMM gateway (DMM-GW) that provides Internet access and manages the different prefixes assigned by the controllers to the mobile nodes that attach to the district. The concept of DMM is to bring the mobility management entities, the DMM-GWs in our case, closer to the edge of the access network. The continuity when a handover occurs is provided by establishing a tunnel between the DMM-GW in the original home and the target visited networks. This is the case when the handover takes place between two districts managed by our SDN-DMM solution. When the mobile node roams to a different district entirely, however, the host takes control of its own mobility management.

In our host-based implementation, the mobile node attaches to a new network (moving from a CROWD district) and, once it has gained connectivity in that new network, the MN notifies the CRC coordinating the district of its previous point of attachment. Then, the CRC notifies the DMM-GW in that district of the new location of the mobile node. In addition, the CRC replies to the MN with the data for connecting to the DMM-GW, as well. The MN and the DMM-GW can, in turn, establish a bidirectional tunnel to encapsulate the mobile node's traffic and ensure continuity for ongoing sessions.

We use as the target network the LTE emulated environment provided by the ns-3 modified LENA LTE and LabVIEW framework already presented. The mobile node and the remote host interact with ns-3 through a virtual tap interface, with the UE, eNodeB and core network running in the real-time ns-3 implementation on Linux laptops, whereas the LTE PHY layer runs over the LabVIEW/PXI platform.

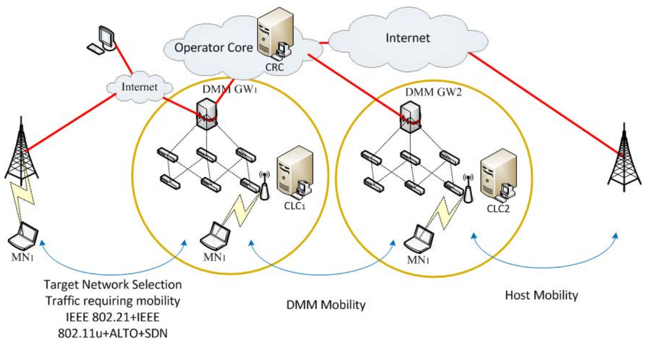

Figure 6: Scenario for inter-technology (WiFi/LTE) Distributed Mobility Management

5. RESULTS

We now show some preliminary results of our efforts to integrate ns-3 with our LabVIEW PHY layer. We run two instances of ns-3, one acting as an eNB and the other as a

UE. In the current setup, the downlink PHY runs Over-The-Air (OTA) within LabVIEW and uplink PHY messages are carried over an Ethernet link, directly connecting the UE PXI system to the eNB host. The two instances of ns-3 for the eNB and UE run on separate hosts and communicate via the MAC/PHY API and UDP, which is supported in both ns-3 and the LabVIEW PHY. The simulation was carried out for a period of 700 seconds. Currently, we are testing the LTE scenario for RLC Unacknowledged Mode without Hybrid ARQ, however we plan to extend our work to support Acknowledged Mode and HARQ feedback. Fig 7 shows the jitter performance of eNB PHY frame counter that runs every 1 ms. Ideally, the curve in Fig 7 should show little deviation from 1 ms, as the subframe generation event scheduled every 1 ms by the core DES scheduler. Any deviation from the ideal execution time is jitter, which we wish to minimize. We observe an average deviation of about 200 microseconds with occasional 300 microsecond peaks. We show similar real-time performance results for the UE in Figure 8. Note that the time representation in y-axis is that of wall-clock time of PC and not that of *simulation time*. The deviations in event execution are due to the fact that ns-3 is not designed from ground up for real-time execution. However, we are investigating a more multi-threaded architecture and further optimizations (to reduce memory allocation and copying of packet buffers, for instance) that we hope will improve real-time performance.

In the current simulation of ns-3, we generated the UDP constant bit-rate source that sends UDP packets from a remote host through the emulated Evolved Packet Core network to UE at 1400 bytes/millisecond. Figure 9 shows the jitter results of the eNB application traffic generation which happens every 1 ms. We can see from Figure 10 that UE sink application receives packets on the average of 1 ms, as expected. We also show below example log from ns-3 real-time logging system of the LTE ATTACH procedure. In the logs below, *Sim Time* refers to ns-3 simulation time, whereas *Wall Clock Time* refers to the real-time-clock of the computer, since the beginning of the simulation.

#grep -i conn /tmp/eNB.log

Sim Time (ms)= 16985 Wall Clock Time (ms) = 20221.8 : IMSI 1 RNTI 1 UeManager INITIAL_RANDOM_ACCESS –> CONNECTION_SETUP

Sim Time (ms)= 17006 Wall Clock Time (ms) = 20242.8 : IMSI 1 RNTI 1 UeManager CONNECTION_SETUP –> CONNECTED_NORMALLY

Sim Time (ms)= 17006 Wall Clock Time (ms) = 20242.8 : IMSI 1 RNTI 1 UeManager CONNECTED_NORMALLY –> CONNECTION_RECONFIGURATION

Sim Time (ms)= 17028 Wall Clock Time (ms) = 20264.9 : IMSI 1 RNTI 1 UeManager CONNECTION_RECONFIGURATION –> CONNECTED_NORMALLY

#grep -i conn /tmp/UE.log

Sim Time (ms)= 31 Wall Clock Time (ms) = 20207.4 : IMSI 1 RNTI 1 UeRrc IDLE_RANDOM_ACCESS –> IDLE_CONNECTING

Sim Time (ms)= 52 Wall Clock Time (ms) = 20228.7 : IMSI 1 RNTI 1 UeRrc IDLE_CONNECTING –> CONNECTED_NORMALLY

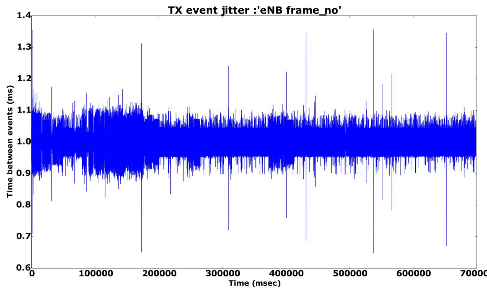

Figure 7: Real-time Performance of eNB PHY Frame Counter

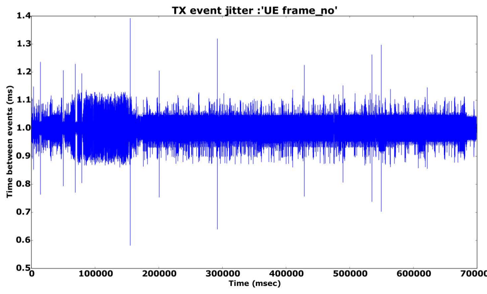

Figure 8: Real-time Performance of UE PHY Frame Counter

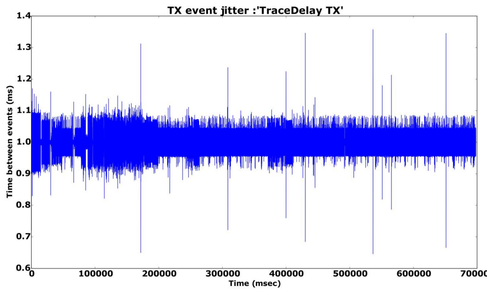

Figure 9: eNB source traffic generation statistics

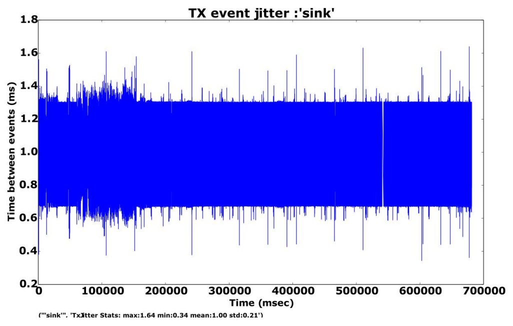

Figure 10: UE sink traffic consumption statistics

6. CONCLUSIONS

In this paper, we propose extensions to ns-3 to integrate a emulated LTE MAC and upper-layer stack from the LENA module with a LabVIEW PXI/FPGA-based LTE Physical layer. This platform provides a flexible Software Defined Radio testbed for prototyping the CROWD SDN system. We introduced several architectural changes, such as multi-threading, non-blocking logging and separation of eNB and UE data structures, that were key to real-time cellular network emulation in ns-3. We also presented Distributed Mobility Management (DMM) to showcase mobility between LTE and WiFi networks using the testbed. We suggest that, since ns-3 is already a widely used simulation tool, adopting it for prototyping, as well, can significantly shorten the time from system-level simulation to prototyping. We plan to continue this work by investigating more efficient ns-3 execution, especially within the LENA stack through a parallel architecture, and finally to contribute this code back the open source community. Also we would like to explore the integration of other ns-3 wireless protocols implementations, such as 802.11 and Zigbee with LabVIEW-based PXI and USRP devices. Another interesting use case for ns-3 as part of a real-world stack would be to reduce the memory footprint and streamline the code execution to allow it to run on System on Chip (SOC) devices such as the embedded USRP E-310 [14] (based on the Xilinx Zinq) platform. We believe this paper presents a strong case that using ns-3 for prototyping as well as simulation can accelerate and advance wireless research.

7. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318115 (Connectivity management for eneRgy Optimised Wireless Dense networks - CROWD).

8. REFERENCES

[1] H. Ali-Ahmad, C. Cicconetti, A. de le Oliva, V. Mancuso, M. R. Sama, P. Seite, and S. Shanmugalingam., "SDN-based Network Architecture for Extremely Dense Wireless Networks," *IEEE Software Defined Networks for Future Networks and Services (IEEE SDN4FNS)*, Nov. 2013.

[2] "EU FP7 CROWD Project." [Online] `http://www.ict-crowd.eu/publications.html`.

[3] "Amarisoft Corp.." [Online] `http://www.amarisoft.com/`.

[4] "OpenAirInterface." [Online] `http://www.openairinterface.org/`.

[5] "ns-3 Open Source Simulator." [Online] `http://www.nsmam.org`.

[6] "Overview of NS3 based LTE LENA Simulator." [Online] `http://networks.cttc.es/mobile-networks/software-tools/lena/`.

[7] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero, "An open source product-oriented lte network simulator based on ns-3," in *Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pp. 293–298, ACM, 2011.

[8] "LabVIEW Based Platform for Prototyping Dense LTE Networks in CROWD Project." [Online] `http://www.ni.com/white-paper/14297/en/`.

[9] R. Gupta, B. Bachmann, A. Kruppe, R. Ford, S. Rangan, N. Kundargi, A. Ekbal, K. Rathi, A. Asadi, V. Mancuso, and A. Morelli, "LabVIEW based Software-Defined Physical/MAC Layer Architecture for prototyping dense LTE Networks," in *SDR WInnComm*, 2015.

[10] R. Gupta, T. Vogel, N. Kundargi, A. Ekbal, A. Morelli, V. Mancuso, V. Sciancalepore, R. Ford, and S. Rangan, "LabVIEW based Platform for prototyping dense LTE Networks in CROWD Project," in *EuCNC*, 2014.

[11] 3GPP, "3GPP TS 24.312; Access Network Discovery and Selection Function (ANDSF) Management Object (MO)," tech. rep.

[12] A. Daeinabi, K. Sandrasegaran, and X. Zhu, "Survey of intercell interference mitigation techniques in LTE downlink networks," in *Telecommunication Networks and Applications Conference (ATNAC), 2012 Australasian*, pp. 1–6, Nov. 2012.

[13] "OpenDaylight Project." [Online] `http://www.opendaylight.org/`.

[14] "USRP Embedded Series Platform." [Online] `http://www.ettus.com/product/category/USRP-Embedded-Series`.