LTE eNodeB Demonstrator with Real and Simulated Interfaces

[1]Titus-Constantin BĂLAN, [2]Florin SANDU, [3]Szilard CSEREY, [4]Virgil CAZACU [13]*SIEMENS Program and System Engineering Bd Mihai Kogalniceanu nr.21 bl.C6 RO-500090 Brasov,*
[2]"Transilvania"University of Brasov Bd Eroilor nr. 29A RO-500036, Brasov, [4]Bit Defender str. Preciziei, nr.24, RO-062204, Bucharest, [1]titus.balan@siemens.com, [2]sandu@unitbv.ro,[3]szilard.cserey@siemens.com
[4]vcazacu@bitdefender.com

***Abstract* — The difference between a simulator and an emulator resides in the feature of emulation to replace the functions of real equipment in operability conditions. When it comes to networking, the main distinction between simulation and emulation can be confined to the option of connectivity with real interfaces that makes possible intercommunication between virtualized and real network elements. The paper describes a method of integrating real interfaces to communicate with simulation modules of OMNeT++ environment. The case study presented is oriented towards Long Term Evolution eNodeB functionality emulation.**

***Index Terms* — Communication system software, Discrete event simulation, Internetworking, Long Term Evolution, Network interfaces, Mobile Data Communications, Simulation software**

I. INTRODUCTION

With simulators, one of the drawbacks is that, even if real protocols are replicated, real messages cannot be passed into the simulated environment. Interconnecting real and emulated elements is not a new idea. Emulation dedicated tools started with complex equipment including a special hardware platform. But implementing everything at software level seems simpler and more cost-effective. Software solutions can have performance limitations, but this type of emulators will be used mostly for research and development purposes, with an important educational value, where functionality is evaluated with respect to performance.

In the case of networking, implementing everything in programming languages like C, at network kernel level, can lead to greater performance as compared to using discrete event simulation messages. But one of the great benefits of a simulator is the graphical programming option that allows building a network in a visual way. In the case of OMNeT++ the graphical environment is called NED. OMNeT++ needs to be extended to interact with real world, in order to support a network mixture of both real and simulated connected elements, a concept similar to the "hardware-in-the-loop" approach in LabVIEW.

Most of the testing equipment is focused on pushing traffic in different scenarios to the real equipment. But the dimensions and complexity of new network architectures makes end-to-end testing almost impossible only with real equipment, even in the case of important telecom vendors, so emulation of elements is an important step in development and testing.

Long Term Evolution architectures represent an element of novelty, and testing access to real equipment that has not been released in commercial versions is limited for common users. For learning and research purposes the only solution is the emulated equipment.

This paper will describe a socket-based method for connecting real interfaces to simulation environment, which can be applied to different protocols. For example, for the LTE (3GPP Long Term Evolution) eNodeB (evolved Node B), a real S1_U interface is integrated. Our goal is to devise an eNodeB emulator, using the same method of emulation for other interfaces.

II. INTEGRATING REAL INTERFACES IN OMNET++

The simulation engine used is OMNeT++, a well-know open source discrete event simulator, that has reached a maturity state in its version 4.0, and which also includes an integrated Eclipse development environment.

OMNeT++ is an object-oriented modular network simulator, which can be used for: traffic modelling of telecommunication networks, protocol modelling and other network-related simulations.

Communication between simulation models and real-time applications is problematic, especially from the point of view of event synchronization, but also from a message coordination perspective. For this reason, an adaptation layer is introduced.

The PC that runs the event discrete simulator software will act as a listening server for a number of applications that also run on the real equipment that we try to emulate: TCP and UDP sockets are the most used interfaces, but also raw sockets can be used. The emulator should be prepared any

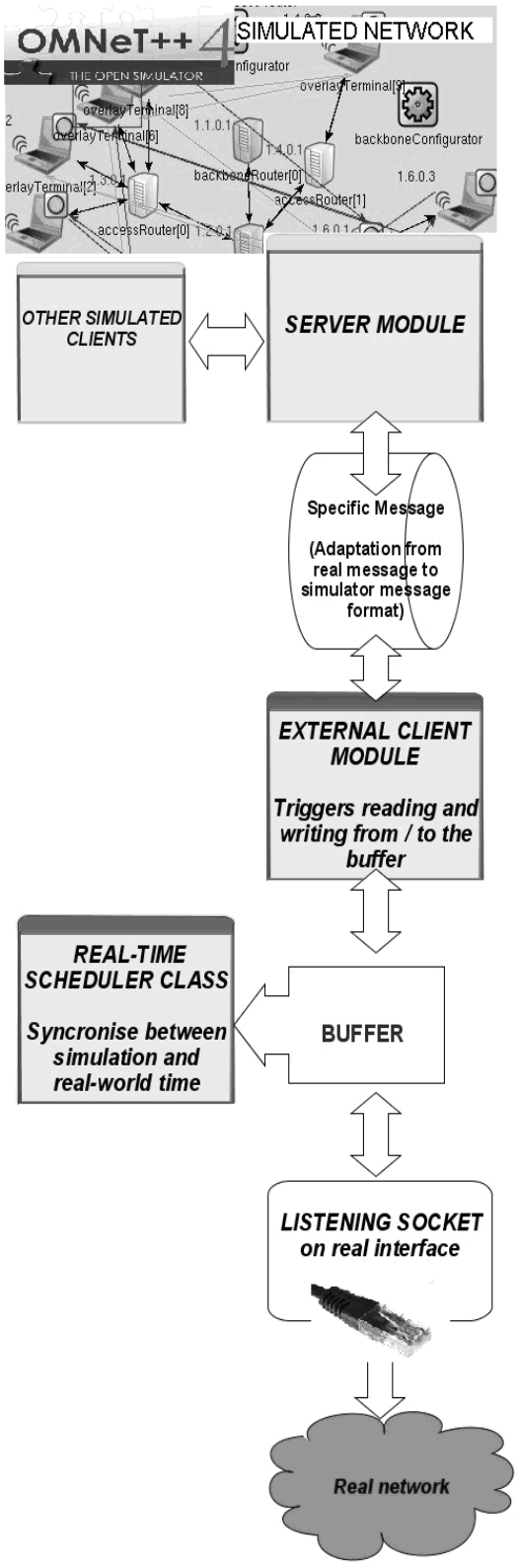

Fig. 1 Integration of a real interface into the OMNNeT++ simulator. All events are scheduled, all data is buffered, and reading from/ writing to the buffer is triggered by the external client OMNeT++ model

time to accept messages, the same way as a real element.

There are two time domains that need to be coordinated: the simulation time of OMNeT++ with the real time used by the equipment that generates the traffic on the real interface.

For that purpose, a scheduler class was implemented, based on the model of cSocketRTScheduler that OMNET++ offers. This class has the role of receiving real messages and buffering them in a byte array variable. Also, the time of receiving packets is registered.

For reading from and writing to the buffer, an external interface class is implemented (extS1UClient). At simulator level, the external interface is visible as a separate module, which acts as client for another module that is the server (S1UServer is incorporated in the eNodeB class). In order to start the real listening socket, the external client has to run an initialization method that also triggers the listening state.

For reading incoming packages from the buffer, the external interface class is using self-destined packages. This is a common method used by OMNET++ as trigger for events execution. The incoming packets are replayed, one by one, in a serialized way into the simulator, in close coordination with the scheduler class for time synchronization. Before each event execution there will be some small delays.

The real messages are mapped into the message structure of the simulator. The OMNET++ INET framework has support for many types of protocols, so the message mapping can be done directly to one of the INET classes.

For sending packages, the process is reversed: the message is converted into a byte array, and passed to the buffer.

A method of the scheduler class is invoked, so outgoing packets are played on the real interface using the socket.

This adaptation layer may prove disadvantageous when the number of real messages sent is greater than the scheduler class' ability to process, like the case of flooding one interface with traffic. The effect will be a break of the socket connection.

In order that the simulation keeps up with real time, the OMNET++ simulation should be run in fast or express mode. For didactic purpose, normal animation time can be used in message analysis. When simulation is running with animation and the external interface is waiting for a reply, timeout that might occur represent quite a big risk, because the simulator time is delayed.

III. LTE eNodeB INTERFACES

As a demo for integrating the simulator with real interfaces, we have chosen to implement the real interfaces of the LTE eNodeB. The Long Term Evolution 3GPP architecture tries to simplify the telecom networks and one of the intentions is to reduce the number of network elements. As a step forward from UMTS (Universal Mobile Telecommunications System) networks, LTE eNodeB combines the functionality of the NodeB and RNC (Radio Network Controller), including more routing capabilities, and the inter eNodeB handover as a solution for signalling overhead limitation in the core network.

So far, the eNodeB real interfaces integration with simulation models is focused on the connectivity to the System Architecture Evolution, the core network of LTE. As User-Plane protocol, the GTPu tunnelling (GPRS Tunnelling Protocol - user) is used on top of UDP, on the allotted port 2152. For Control-plane, S1AP (S1 Application Part) and X2AP are the used protocols, which run on top of SCTP.

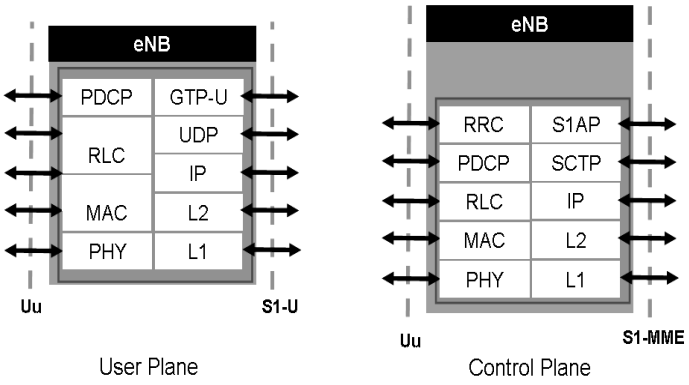

Fig. 2 Protocols used at eNodeB level for User Plane and Control Plane

Two interfaces ensure the communication between eNodeB and the core network:

- S1_MME is the interface towards MME (Mobility Management Entity), so it is the main interface used for Control-plane communication.
- S1_U is the interface towards the S-GW (Serving Gateway) and it is responsible for user plain traffic.

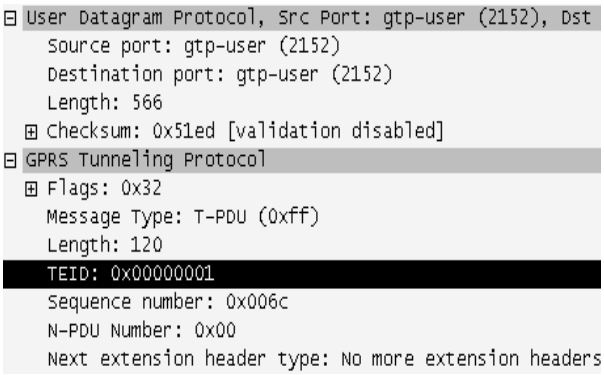

Fig. 3. GTPu header: the TEID field is an important element when the forwarding traffic path has to be chosen.

At eNodeB level, the traffic is tunnelled using GTPu. An important element of the GTP header is the TEID (tunnel endpoint identifier). In downlink traffic, this ID is used by the eNodeB to forward the traffic to the correct radio cell.

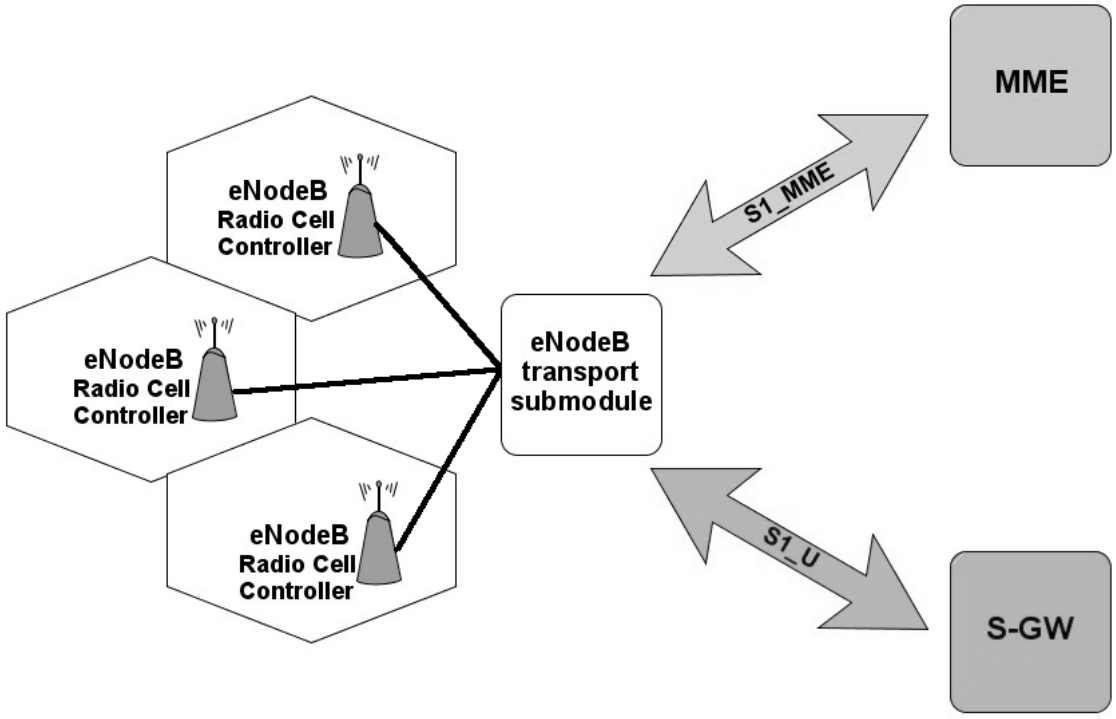

Fig. 4 The eNodeB module consists of several interconnected sub-modules. The Transport sub-module forwards downlink traffic to the proper Radio Cell Controller based on the TEID internal mapping to the Radio Controllers

The eNodeB controls more radio cells (at least 3 cells) at the same time, and the internal structure of eNodeB consists of more elements that communicate also at IP level.

For downlink traffic coming from the S-GW, the traffic forwarding is done to one of the cells according to the TEID.

Steps in downlink traffic path establishment at LTE eNodeB level:

- MME and eNodeB set during the S1AP Initial Context Setup (Radio Access Bearer Setup) a TEID for DL traffic.
- When MME sends Handover Request (S1AP PathSwitchRequest), it also includes a GTP-TEID within the ERAB field, and the eNodeB should reply in the Handover Request Acknowledge that it accepts the proposed forwarding of downlink data for this bearer.

- The eNodeB keeps an internal database for mapping the radio cells that it controls with the TEIDs (MAC addresses of Radio Cell Controlling Elements are mapped to the TEIDs)
- When S-GW sends downlink traffic, that is GTPutunnelled, the eNodeB forwards the traffic to the proper cell based on the mapping between the TEID in the header and the stored MAC addresses (that are usually fixed addresses in the eNodeB internal network).

IV. CONNECTING S1_U REAL INTERFACE WITH ENODEB SIMULATION MODEL

In our implementation, we started from a signalling discrete event simulator for LTE-SAE networks and integrated a real S1_U interface that is connected to the existing eNodeB module described. For this implementation, an UDP socket SOCKET_DGRAM was opened on the allotted GTPu port, 2152.

The real-time scheduler class was created for synchronization. An external S1_U (extS1UClient) interface client was described. When starting the simulation, the initiation procedure of this external S1_U client sets the socket in listening state. S1UPkt.msg class was created for the message adaptation.

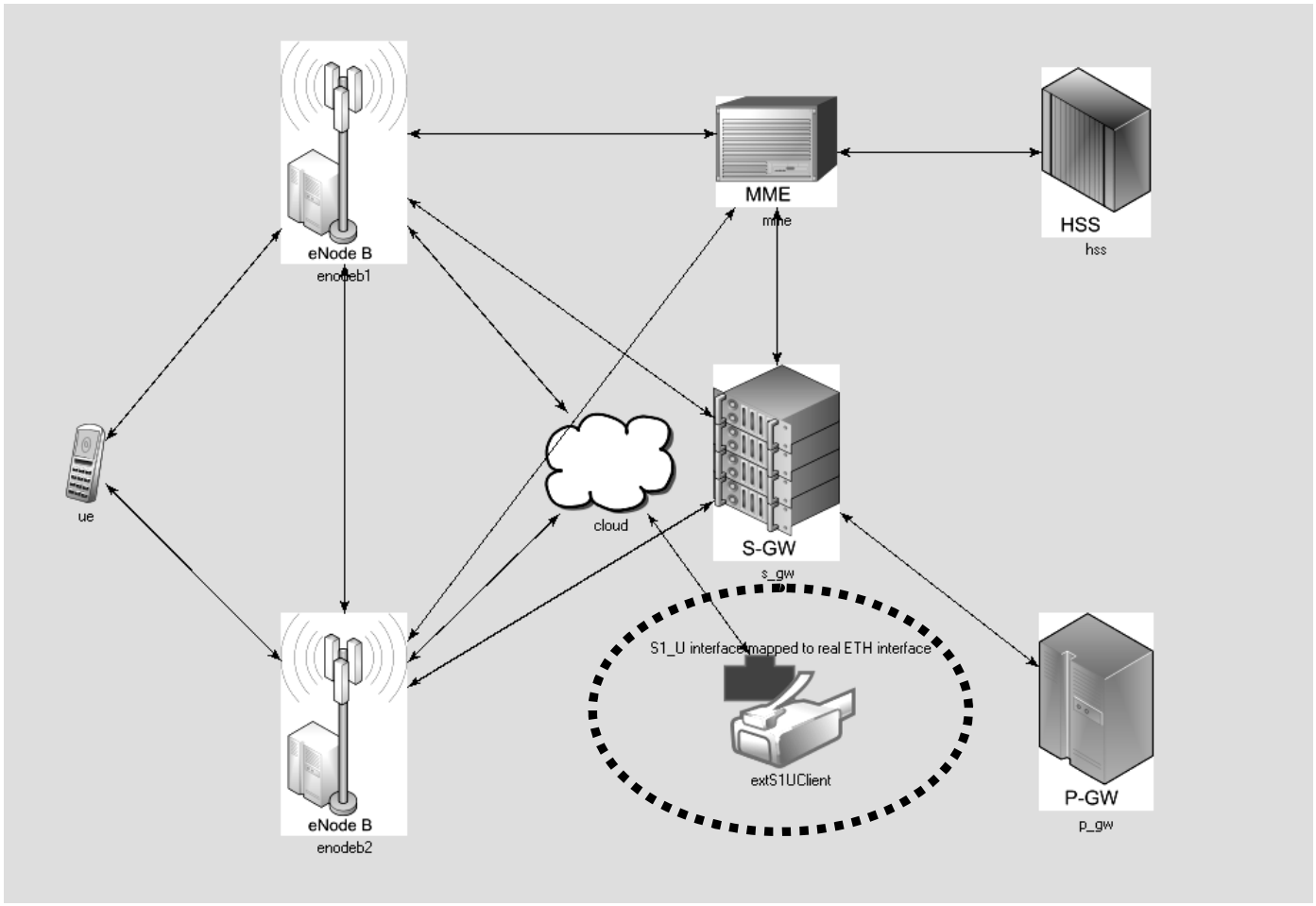

Fig. 5 To the LTE signalling simulation network, an external S1_U Interface was integrated to the eNodeB simulated model. The OMNET interconnection model is called extS1Client

In Fig. 5, this connection module is shown as an Ethernet RJ45 interface, because it makes the connection to the real Ethernet interface on the simulation host PC. The TEID is extracted form the received buffered message, because this parameter determines the forwarding traffic path for the eNodeB.

The TEID with MAC address mapping was statically set for our test, just for a limited number of examples. OMNET++ also offers the possibility to integrate a MySQL Database.

The eNodeB simulation model is capable of communication with other simulation models and at the same

time acts as a server to the real interface client. Real packets simulation messages can be forwarded to other virtual modules.

For testing the setup on the real interface, a packet player software was used, Colasoft Packet Builder, playing a GTPu message from a real GTPu eNodeB trace. The packets were sent from another PC connected via Ethernet to the PC running the simulation.

The simulator runs under MS Windows in MINGW environment. We had difficulties in sending generated packets from the host PC (from Colasoft Packet Builder) to the simulator running on the same host. These packages are considered to be intended for localhost, because we try to send them to the listening socket on the same machine, and Windows restricts this functionality. Installing a Microsoft Loopback adapter did not solve the problem, so running simulation in Unix is recommended or sending the real packages from another host, connected to the host where the simulator is running.

CONCLUSIONS AND FUTURE WORK

Integrating real interfaces with discrete event simulators offers the possibility for the virtualized network to take part in real test scenarios. The visual programming is one of the benefits of using simulators with graphical interface, because each element is visible as an object, which is easily integrated with other elements.

For emulating real equipment, the equipment behaviour should be replicated at simulator host level, meaning the simulator should act as a server, and listen to exterior events. The same servers, which we replicate in the emulator, also run on the real emulated equipment. Reaction to external stimuli is a new function of the simulator, which listens to incoming messages or generates packages based on an algorithm. The problem of time synchronization and message adaptation can be solved by introducing an adaptation layer.

As future work, we are planning to integrate other that are received from a real interface and converted into interfaces of the eNodeB as well, and this implies that other servers should run in our simulated environment, and other sockets should be in listening state, depending on the emulated protocols. SCTP traffic integration is the next step, as this protocol is the base for most of the signalling with LTE core network. For the message adaptation part, the best solution would be the correlation with the OMNET++ INET framework, which offers support for many protocols simulation. Improvement of emulation model can be done by comparing the testing results with the real equipment tests, and making a fine tune of the virtual model.

REFERENCES

[1] A.Varga: "OMNeT++ Discrete Event Simulation System Version 3.2 User Manual", 2005, www.omnetpp.org

[2] Florin Sandu, Szilárd Cserey, Titus Constantin Balan, Mihai Romanca: "Simulation-based UMTS e-learning software, PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments", July 2008

[3] Christoph P. Mayer, Thomas Gamer, "Integrating real world applicationsinto OMNeT++", Institute of Telematics, Universität Karlsruhe (TH), Technischer Bericht, Nr. TM-2008-2, Feb 2008

[4] Balan Titus-Constantin, Sandu Florin," Fourth-Generation Wireless Networks: Applications and Innovations", Pages: 405-423 pp, IGI Global, 2010

[5] Harri Holma, Antti Toskala : "LTE for UMTS - OFDMA and SC-FDMA Based Radio Access", Wiley, April 2009

[6] 3rd Generation Partnership Project: "3GPP TS 36.413 Evolved Universal Terrestrial Radio Access Network"; (E-UTRAN); S1 Application Protocol (S1AP), V9.1.0 (2009-12)

[7] I. Baumgart, B. Heep, and S. Krause. OverSim: "A Flexible Overlay Network Simulation Framework. Proceedings of 10th IEEE Global Internet Symposium". pages 79–84, May 2007.

[8] http://www.omnetpp.org