Implementation of an Open-Source UTRAN Testbed

Van Peteghem, Hugues; Schumacher, Laurent

Published in:
2006 Symposium on Communications and Vehicular Technology

DOI:
10.1109/SCVT.2006.334380

Publication date:
2006

Document Version
Early version, also known as pre-print

Implementation of an Open-Source UTRAN Testbed

Van Peteghem H. and Schumacher L.
Pôle "Réseaux et Sécurité"
FUNDP - The University of Namur
Namur, Belgium
{hvp,lsc}@info.fundp.ac.be

***Abstract*— We present here the implementation of our UTRAN testbed. The article focuses on the organisation of the Linux traffic controller to investigate, over a classical Ethernet (wired) link, two typical properties of the air interface of an UMTS network, namely the availability of three different transport channels known as DCH, DSCH and FACH, and the significant error rate of radio communications.**

I. Introduction

Time- and event-driven network simulators implemented in software are valuable tools for researchers to develop, test and diagnose network protocols, architecture, Quality of Service (QoS) management, etc. They can figure out a number of issues to be solved before even thinking about a real deployment. But network simulators also have their limitations. As the network to simulate becomes very large or very complex, they begin to be really difficult to implement and they require more and more computing time.

In this perspective, a testbed can be a complementary tool for researchers. Indeed, testbeds are cheaper than a real deployment while approaching even more the real network behaviour than a simulator would do in certain cases. They are also capable to reveal issues hard to notice using simulators. The testbeds are then seen as a good tradeoff between computer simulations and tests over a real life network.

The rest of this paper is organised as follow. Section II presents our testbed and describes the architecture designed to emulate a part of a Universal Mobile Telecommunication System (UMTS) network known as the UMTS Terrestrial Radio Access Network (UTRAN). Section III delivers our first results about the emulation of the variable Bit Error Rate (BER) communications undergo over an UMTS radio link. Section IV explains our next steps in this research. Finally conclusions are drawn in Section V.

II. Testbed and UMTS Structure

Our work is based on a computer testbed emulating the UTRAN [1]. It consists of 9 PCs respectively standing for the Radio Network Controller (RNC), 4 NodeBs and 4 sets of User Equipments (UEs). We emulate 4 macrocell trisectorial NodeBs, each of them managing a population of UEs over a 27 km^2 world surface.

A. Transport Channels

Traffic whose distribution complies with the characteristics of the 4 traffic classes (Conversational, Interactive, Streaming and Background) [2] is generated from the RNC to the PC emulating a given UE (downlink) using a synthetic traffic generator [3]. Each of these data flows is mapped to one of the three available UMTS transport channels (Table I) following the guidelines given in [4]:

- The Dedicated CHannel (DCH): this is the only dedicated transport channel in the UMTS standard, it carries the service data of a single user at variable bit rate,
- The Dedicated Shared CHannel (DSCH): this transport channel carries signalling and service data of several users at variable bit rate, or
- The Forward Access CHannel (FACH): this last transport channel carries also the service data of several users but at fixed bit rate.

TABLE I
Traffic Classes and Channels Mapping

3GPP Traffic Classes	Authorised Transport Channel(s)
Conversational	DCH
Interactive	DCH, DSCH or FACH
Streaming	DCH
Background	DCH, DSCH or FACH

We consider that each sector into the emulated world may only have one FACH, one DSCH and a number of DCHs depending on the RNC's Spreading Factor (SF) management. Those sole FACH and DSCH however collect traffic from several active users, as further explained in Section II-C.

B. Typical Radio Link BER

The main objective of our testbed is to emulate the lower layers of the UMTS air interface over an Ethernet (wired) link between two PCs respectively emulating a population of UEs and their serving trisectorial NodeB. The emulation of the Radio Link Control (RLC) and Medium Access Control (MAC) layers is achieved using the Linux traffic controller (TC) which copes with packet shaping, scheduling, policing and dropping. More precisely, we use the TC NETwork EMulation component (NETEM) [5] which is a waiting queue able to delay, drop, corrupt, duplicate or even reorder packets.

0-7803-9785-1/06/$20.00

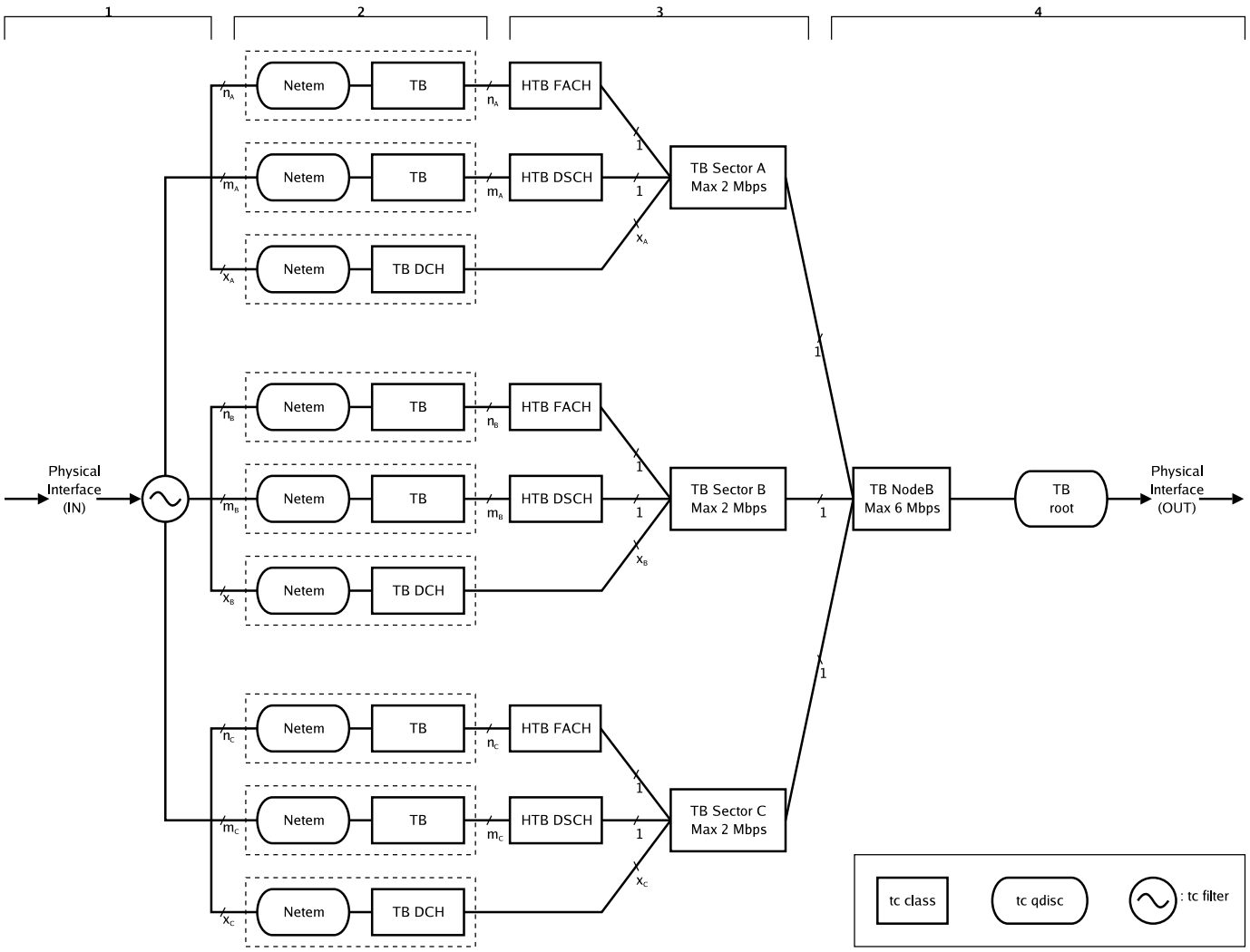

Fig. 1. Air Interface Emulation

The NETEM packet corruption functionality is the most appropriate to approach what packets really undergo over the UMTS radio link. Indeed, it introduces a single bit error at a random position in the packet. Unfortunately, being on Ethernet, this bit error inevitably leads to a CRC error and a packet drop which is slightly more aggressive than the real Automatic Repeat Request (ARQ) protocol implemented into the RLC layer. Anyway, this way of doing does not simply drop the packet, it corrupts it and allows the packet to be transmitted to the UE and then to consume a part of the bandwidth allocated to the flow it belongs to.

The RLC layer is able to transmit RLC Packet Data Unit (PDU) in three different modes: Acknowledge Mode (AM), Unacknowledged Mode (UM) and Transparent Mode (TM). Note that the ARQ mechanism is only available in the AM. As a consequence, the testbed amplifies radio impairments of AM as corrupted packets are dropped by Ethernet and trigger retransmission at higher layers. Each of these modes has a different tolerance level to a PDU loss or reordering. Keeping this in mind and also that RLC PDUs are typically smaller than IP packets, the resulting BER will have to be adapted. Figure 2 presents an example of a resulting BER dependant on the Signal-to-Noise plus Interference Ratio (SNIR) level.

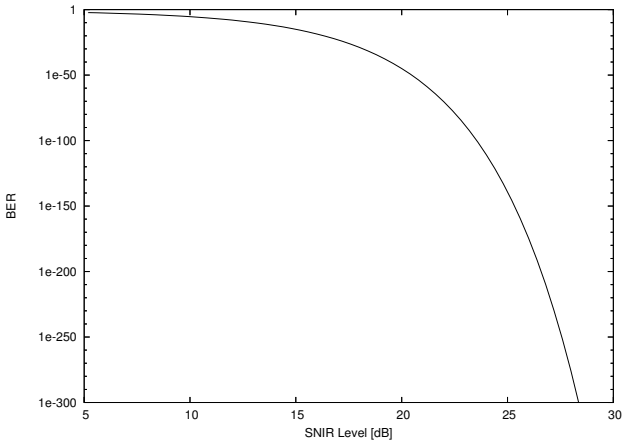

Fig. 2. BER Vs. SNIR with QPSK modulation on a AWGN channel without spreading nor coding.

C. Implementation

Using TC and its Token Bucket (TB) module to mimic the different available transport channels and its NETEM module to emulate the packet loss over the UMTS air interface, we can now build a structure to manage all the packets going from a serving NodeB to one of its UEs. Fig. 1 illustrates the implementation of one of our NodeBs:

1 First of all, a new downlink data flow is filtered based on the destination IPv6 address and the "Traffic Class" field of the IPv6 header. Following these header fields we know exactly in which sector the UE receiving this data flow is located and which transport channel this flow will use over the air interface.

2 A new instance of a NETEM queue connected to a TB is then created. We corrupt some of the packets passing through this queue since this is what happens over radio links. The packet corruption process is guided by the SNIR level of the destination UE. Due to the NodeB's fast power control, the target SNIR setpoint is permanently adjusted and aims at a constant quality, usually defined as a certain target BER, depending on the traffic class [2]. This might be achieved only on the transport channels supporting fast power control, namely the DSCH and the DCH. From the FACH point of view, a poor SNIR level can not be thwart by the power control, meaning a UE with a low SNIR level will get a higher BER without any bound.
It has also to be said that the SF allocated to a flow influences the BER level. As we get a long spreading code, the user's symbols are spread on a larger number of chips over the radio link allowing them to better support the link impairments. So, the longer the spreading code, the lower the BER.
The TB is only responsible of the bandwidth consumption of the flow ensuring that it never exceeds the bandwidth it has received. In this architecture, there is a NETEM-TB instance for each single data flow. That means there are n_i flows sharing the FACH, m_i flows sharing the DSCH and x_i flows each of them occupying a single DCH of the sector i (i=A, B or C).

3 Notice that if the flow is using a common channel (FACH or DSCH), it has to share the channel bandwidth with other flows. All NETEM-TB instances belonging to the same common channel are then supervised by a Hierarchical TB (HTB) which has the ability to borrow some unused tokens from a given TB to give them to another TB waiting for tokens. If the flow is using a DCH, it is directly scheduled at the TB responsible of a sector. This last TB has to equitably manage (x_i+2) entries: the x_i DCHs + 1 FACH + 1 DSCH of the sector i (i=A, B or C). It also has to guarantee that the total bandwidth consumption of the sector never exceeds 2 $Mbps$.

4 Finally, the three TBs managing the three sectors are controlled by the last TB representing the NodeB itself.

III. Results

Using this testbed and the presented TC structure, we have launched an emulation loading each sector with 7 UEs (total of 84 UEs in the emulated world) generating the different kinds of traffic during $5,400$ s and evolving into an urban environment. In the urban area scenario, the UE speed distribution is the following: 30% at 3 km/h, 40% at 30 km/h, 20% at 70 km/h and 10% at 120 km/h. For the moment, the radio link BER is straightly linked to the SNIR level of the UEs (no fast-power control at all). The results of the emulation are summarised in Table II.

On one hand, the Interactive and Background traffics undergo a greater mean packet delay with the NETEM module included into the TC structure. This is obvious since these kinds of traffic are considered as non real-time, which means they are more focused on minimising the error during the transfers than on minimising the mean packet delay. Interactive and Background flows use TCP as transport protocol considering its reliability. With TCP, each corrupted/loss packet has to be retransmitted but it also triggers the congestion control. As a result, following packets will get a lower bandwidth and will then reach their destination with a greater delay. The jitter standard deviation increases also in the same proportions.

TABLE II
Traffic Classes Delay

Mean Packet Delay	without NETEM	with NETEM
Conversational [ms]	42	48
Interactive [ms]	844	1,264
Streaming [ms]	22	29
Background [ms]	2,149	7,295

On the other hand, the Conversational and Streaming traffics are mentioned as real-time traffic meaning they may support some packet losses while the focus is mainly set on delivery time. Conversational and Streaming flows use UDP as transport protocol. This leads to an equivalent mean packet delay since no retransmission is supported using UDP, but the number of lost packets raises from 0.02% without the NETEM module (only due to handovers) to 3.09% with it (handovers + corrupted packets). As for the mean packet delay, the jitter standard deviation of this kind of traffic tends to be stable.

IV. Future Works

It has been proven that the typical bit errors over a radio link are not totally decorrelated and tend to appear in bursts. One of the analytical models trying to describe this property is the Gilbert-Elliot model [6], [7]. It uses a two-state (error and error-free) time-homogeneous discrete time Markov chain to describe the channel variations. This model is a popular one for wireless bit errors since it is complex enough to capture burstiness and simple enough to be treated analytically.

More over, a recent research has noticed that an enhanced version of the Gilbert-Elliot model approaches even more the UMTS radio channel [8]. The error state in this new model is the same than the one in the simple two-state Markov model. However, the error-free state of the new model has been modified. In that state the gaplength is now taken from a Weibull distributed random number which correctly fits the measurements over real networks either in a static scenario and in the case of car movements.

The next step of our work will then be to modify the NETEM module to respect the specifications of this enhanced Gilbert-Elliot model.

V. Conclusion

We have shown that, with a minimal investment, it is possible to establish a testbed mimicking a real UMTS network. This paper has presented how to emulate the UMTS air interface over a wired network, especially the different transport channels and the high BER. Such a testbed is able to evaluate the quality of user experience of services delivered over a cellular network. This way of doing reproduces the real network in a different way than simulators do. It is really important since such a new approach may reveal unsuspected problems and gives another point of view to known issues leading sometimes to easier solutions.

References

[1] H. Van Peteghem and L. Schumacher, "Description of an IPv6 Linux-Based UTRAN Testbed," in *Proceedings of the* 2^{nd} *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom)*, Barcelona (Spain), Mar. 2006.

[2] Technical Specification Group Radio Access Network, "TS 23.107 V6.4.0, Quality of Service (QoS) concept and architecture," 3^{rd} Generation Partnership Project (3GPP), Tech. Rep., Mar. 2006.

[3] "Traffic generator tool," http://www.postel.org/tg/tg.htm, last visited: 03 November 2006.

[4] H. Holma and A. Toskala, *WCDMA for UMTS*, 3rd ed. Wiley interscience, Oct. 2004.

[5] S. Hemminger, "Network Emulation with NetEm," in *Proceedings of the* 6^{th} *Australia's National Linux Conference (LCA)*, Canberra (Australia), Apr. 2005.

[6] E. N. Gilbert, "Capacity of a Burst-Noise Channel," *Bell Systems Technical Journal*, vol. 39, pp. 1253–1265, Sept. 1960.

[7] E. O. Elliot, "Estimates of Error Rates for Codes on Burst-Noise Channels," *Bell Systems Technical Journal*, vol. 42, pp. 1977–1997, Sept. 1963.

[8] W. Karner, "A UMTS DL DCH BLER Model Based on Measurements in Live Networks," COST 273, Tech. Rep., Jan. 2005.