RESPECT: A Real-time Emulator for Service Performance Evaluation of Cellular neTworks

Oumer M. Teyeb, Malek Boussif, Troels B. Sørensen
Department of Communication Technology,
Aalborg University,
Niels Jernes Vej 12
9220 Aalborg East, Denmark
e-mail: {oumer, mb, tbs}@kom.aau.dk

Jeroen Wigard, Preben E. Mogensen
Nokia Networks, Aalborg R&D
Niels Jernes Vej 10
9220 Aalborg East, Denmark
e-mail: {jeroen.wigard,
preben.mogensen}@nokia.com

Abstract— **The evaluation and optimization of packet service performance in wireless networks is a complex process, considering the number of heterogeneous entities and protocols that are involved. An emulation platform comes at hand for such performance investigations as it provides a means to see the performance from an end-2-end (E2E), user-perceived Quality Of Service (QoS) point of view. In this paper, the design and implementation of RESPECT, an easily configurable network emulator is described. RESPECT was originally geared towards Universal Mobile Communications System (UMTS) networks, but thanks to its modular and scalable design, it is being extended for generic heterogeneous networks. Using RESPECT, QoS studies can be carried out to study the behavior of different services in different network conditions, identify generalized service dependent performance metrics for already existing services and predict network environment requirements for future services.**

I. Introduction

Evaluation of the performance of different services running over wireless networks could be done in several ways: *mathematical analysis, simulation, live testing, emulation* or a combination. Analytical modeling is the most logical step to get some insight on the service performance aspects. However, it turns out to be complex even for a network as simple as a tandem networks [1] and is usually only feasible when considering an isolated protocol from the full network protocol stack (such as the one done for TCP in [2]). Due to this, simplified analytical models are mainly used as the starting point for performing simulation studies.

Simulation offers a simple solution where lots of scenarios can be investigated in a relatively short amount of time. However, widely used simulators such as Ns-2 [3], usually utilize abstract implementations for the whole network protocol stack, which could differ significantly from real world implementations (for example TCP versions used in NS-2 vs. Linux TCP). On top of that, simulation tools usually ignore the effects of operating systems, other concurrently running process and memory overheads. This could turn out to be the bottleneck rather than the network protocols themselves (see for example [4]).

Live tests could be performed either on isolated testbeds or even on a real network such as the Internet. Though live testing avoids the problem of using abstract network protocols and entities, the fact that it is almost impossible to change the implementations of the different network protocols or parameter settings just for studying the performance of services limits the scope of this method. Performing live tests can turn out to be very expensive. Results are not reproducible because of the conditions of the live network could vary considerably from one test run to another, which makes it difficult to conduct parameter optimization tests.

Emulation uses a combination of simulated and real network protocols and components. The part of the network that is being evaluated will be simulated while the other parts will be used as in real systems. The advantage of emulation over simulation is that real E2E QoS studies could be done with the aid of end users, and the advantage over live testing is that it is cheaper, experiments could be replicated (to some extent), and protocols for future networks could be tested.

In this paper, we present an emulation platform called RESPECT (Real-time Emulator for Service Performance Evaluation in Cellular neTworks), which provides a link-level, real-time emulation of UMTS networks. The remainder of the paper is structured as follows. Section II gives a brief overview of other related work in the area of network emulation, followed by a high level overview of RESPECT in section III. Section IV describes the choices that were made during the design and implementation of the emulator. Section V gives a glance at some of the results that have been generated so far using the emulator. Section VI gives a description of some of the ongoing enhancements we are making on RESPECT. Finally, section VII gives conclusions.

II. Related Work

Network emulation has been an active research area, especially in the realm of wired networks. One of the earliest network emulators is the Long Link Emulator (LLE) [5]. LLE uses specialized hardware to introduce a constant delay and bit error rates to a given link. DummyNet [6] is an application transparent emulator that is able to introduce queue and bandwidth limitations; constant delays, and errors over a link. NetBed [7] is a configurable Internet emulator, which uses DummyNet. NetBed uses a cluster of hundreds of PCs

0-7803-9152-7/05/$20.00 © 2005 IEEE

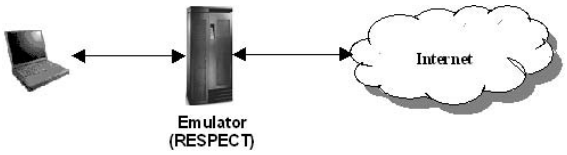

Fig. 1. RESPECT usage scenario

with specialized hardware to create what is like the emulation equivalent of ns-2 simulations.

Generally two approaches have been used to emulate wireless networks. The first approach is to use trace based emulation where the goal is to reproduce real measurements. Such is the approach in [8], where measurements are taken for different scenarios in a WLAN environment, the measurements are then mapped into a simple model, and wired network traffic is modulated with the model. The other approach is less strict, in the sense that, apart from real measurements, the emulation can be based also on more generic models derived from simulation data of dynamic network simulators, or other modeling information. The WINE emulation platform [9] uses this approach to emulate the effect of mobility on application performance. The ARROWS [10] and WINEGLASS [11] emulation platforms are some recently developed system level emulators for the investigation of Radio Resource Management (RRM) algorithms in UMTS that use this approach. RESPECT also uses the second approach.

What makes RESPECT unique is that it uses a very open framework where developers can easily implement protocol enhancements or even new protocols. While most of the emulators described above are very simplistic supporting only fixed delay and bandwidth limitations (such as Dummynet [6])or designed with only one system in mind (such as ARROWS [10]), RESPECT tries to provide a framework that can easily be extended to different kinds of networks. Using the framework, an initial version that is geared towards UMTS networks is implemented. RESPECT is completely based on free software tools, and is able to run on a stand alone PC running Linux, with no additional hardware/software requirements. In contrast to other emulators designed for the Linux platform, RESPECT is implemented completely in *user-space* domain, i.e. it does not require any recompilation of the Linux Kernel.

III. Overview of RESPECT

A simplified diagram of the emulator's usage scenario is shown in Fig. 1. The emulator runs on a PC that is also configured to act as a proxy server[1] and users are connected to the Internet through this machine. Users specify their emulation setup using configuration files, which includes information such as the data rate, the error rate, and delays of different network entities. The settings can be either static or dynamic,

[1]The emulator could also be installed on a normal PC running Linux instead of the proxy server. The use of proxy server allows users to access the emulator remotely and also avoids possible real-time timing problems that may arise if the user is also running a lot of programs on the PC

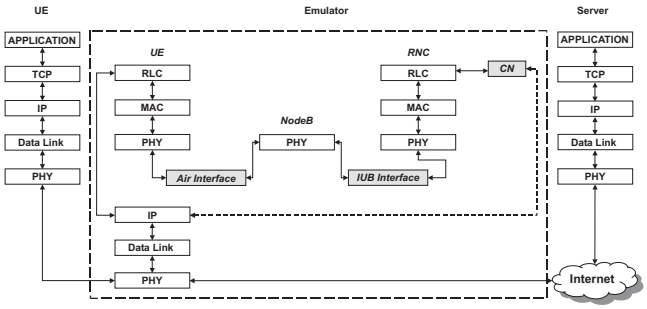

Fig. 2. Emulated UMTS Protocol Stack

describing the different states the network can be in and a stochastic model of the state transitions. Several different parameters can also be specified that affect the workings of the core protocols of the emulated network.

A user starts some network-based application such as FTP file download session or streaming, which is accessed via the proxy server where the emulator is also running. The emulator intercepts the packets that are flowing into and out of the proxy, processing them according the specified set up for the session. Thus, from a higher layer application point of view, the application data will seem to be traversing a UMTS network, instead of just a wired Internet connection. The overall effect of the different entities within a network manifest themselves through delay or packet loss, and these artifacts, both at link and IP level, are recorded into log files for later analysis. In such a way, both user level (the perceived QoS) and statistical (using the log files) analysis of performance is possible with the aid of RESPECT.

The functionality of the emulator, as seen from a protocol point of view, is depicted in fig. 2. The dotted box represents the parts of the protocol stack in the emulator machine that are relevant to us. In a nutshell, RESPECT inserts the additional protocol layers that comprise a UMTS network between the IP and TCP layers of a normal TCP/IP protocol stack. In a real UMTS network these layers are distributed in different network entities, as shown by the names in italics, namely the User Equipment (UE), Base Station (NodeB), Radio Network Controller (RNC), and Core Network (CN). In RESPECT, they are all lumped together into one entity, and doing so will not have different characteristics from that of the real case, as long as the processing delay of the different network entities are taken into account. The Uplink(UL) IP packets that are leaving the UE, when they arrive at the emulator are diverted into the Radio Link Layer (RLC) representing the UE, and they pass all the way through the simulated UMTS protocols, and then released to the destination server via the Internet. On the other hand, the Downlink(DL) packets, when they arrive from the server they are directed to the CN (the dashed path), which will simulate the delay experienced in a UMTS core network, and then passed through the RNC's RLC layer, all the way through the simulated UMTS protocols and released to the UE.

IV. Design and Implementation

RESPECT is composed of four core modules, each module containing a set of interrelated classes. The four different modules are called *Common, Framework, Sniffer* and *Emulator*. The Common, Framework and Sniffer modules are generic modules that can be used to extend the emulator to represent other networks. The Emulator module is a collection of classes, which are mostly inherited from the Framework and Common modules, that specifically model the UMTS protocols. The following sub-sections give a brief description of these modules. The reader is referred to [12] for detailed description of these modules.

A. Common Module

The common module provides base classes that are used to define interfaces for the different parts of the system. This module contains five main classes, namely *Data_Handler*, *Data_Source_Sink*, *Thread*, *Configuration_Reader* and *Logger*. *DataSourceSink* provides an interface for entities that act as sources or sinks of data. The *DataHandler* class, on the other hand, provides interface for those entities that are responsible for handling data. The *DataHandler* and *DataSourceSink* are associated 1-to-1, i.e. every source/sink should have a handler that handles its data, and every handler should have an associated source/sink where it gets its data and where it passes the data after it has processed it. The *Configuration Reader* is a simple parser that handles parameter files.

Thread is an abstract class that provides a wrapper for creating and configuring the priority levels and the scheduling policies of threads using C++ classes. The *logger* class is used for logging data, such as results from emulation runs and debugging information about the emulator operation. If extensive logging to the hard disk is to be done at the same time as the emulation process, timing requirements will not be met, as writing to the hard disk is very slow and takes a considerable amount of CPU time compared to emulation event periods. The logger mitigates this drawback of logging using remote logging accompanied by priority scheduling and voluntary processor yielding.

B. Sniffer Module

The *Sniffer* module provides the functionality of intercepting incoming/outgoing packets from/into the network on a real-time basis and supplies the data to the emulator module, which handles the packets and gives a decision on what is to be done. The interception and re-injection of packets is done using the Netfilter/Iptables packet handling framework [13]. Using this framework, it is possible to drop, accept or queue packets based on criteria such as IP addresses, port number or packet sizes. During its setup, the Sniffer initializes the Netfilter packet filtering rules for queuing the concerned packets. When a packet arrives at the IP layer in the emulator, the Sniffer raises an event to the associated data handler, which is the emulator, letting it know that new data has been received. This event includes information such as the size, identification and the direction of flow of the packet, i.e. whether it is incoming or outgoing. The packet will be kept inside the queue till the Sniffer receives a verdict to discard or release the packet from the emulator. The Sniffer also logs the header information of every incoming/outgoing packet in a *pcap* file format so that the connection could be analyzed further by using powerful tools such as *tcptrace* [14].

C. Framework Module

The core protocol components of the emulator were designed based on CASINO (Component Architecture for Simulating Network Objects) [15]. CASINO is an object-oriented protocol development platform that was targeted for developing network communication protocol stacks. A scaled-down version of the CASINO framework, with some modifications to fit the needs of RESPECT, was reimplemented, based mainly on the design specifications of CASINO [16]. The *Framework* module provides generic classes that implement basic functionalities of protocols, packets, events, and timer handler that control the events.

D. Emulator Module

This is the module that actually performs the UMTS emulation process. The main class in the emulator module, known as *Emulator*, initializes the different protocols and network entities and interconnects them for proper packet flow. To get the emulation process started, the initial timers that would keep on firing every Transmission Time Interval (TTI) are started. The Emulator accepts information about arriving packets from the Sniffer, propagates it down its protocol stack and also notifies the Sniffer on behalf of its components when packets are dealt with, i.e. when it has passed through the whole protocol stack and ready to be released back to the network or when a drop decision is made due to some errors in the air interface or buffer overflows.

Apart from the emulator class, the other main classes in this module are the RLC, Medium Access Control (MAC), Physical (PHY) and Air Interface. The RLC is implemented following the 3GPP RLC specifications [17], with full support of the retransmission facilities described in the specification. The MAC layer is the one that determines the rate of information flow and the granularity of the emulation process. Every TTI, the MAC sends information to the RLC, telling it the amount of data that could be accommodated in one TTI, the length of subsequent TTIs and the PDU sizes that the RLC should use next time it segments packets. The PHY layer is a simple class that passes packets coming from the MAC to the air interface and vice versa. The air interface provides a memory-less channel that has some given FER (frame erasure rate) in the UL and DL directions and associated delays in both directions.

V. Sample Results

So far RESPECT has been mainly used to investigate the impact of several RLC and TCP parameters on the performance of FTP and HTTP services. Some results comparing the performance of 2.5G systems such as GPRS (General Packet

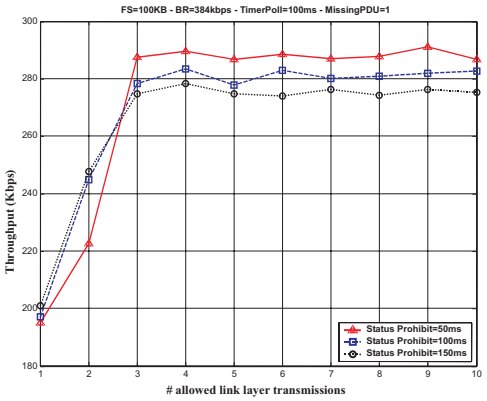

Fig. 3. Throughput of FTP download sessions

Radio Service) and EDGE (Enhanced Data rates for GSM Evolution) with UMTS have also been generated. Following is a brief description of some of these results.

The impact of the different RLC retransmission mechanisms on FTP performance are investigated in [18]. The main finding from the investigation is that the main parameter that affects the performance is the number of allowed link layer transmissions. This parameter specifies how many times the link layer will try to transmit an erroneous RLC segment, including the first transmission, before giving up. Fig. 3 and fig. 4 show the average throughput and goodput, respectively, of a 100KB FTP file download, for a 384kbps connection with a constant FER of 10%. From the figures, it can be seen that a maximum link layer transmission of three or more leads to optimal results. Detailed explanation of these and other related results can be found in [18].

In [19], the impact of link layer delivery sequence on FTP performance is investigated. The impact of using TCP extensions such as TCP Selective Acknowledgments /Forward Acknowledgment (SACK/FACK) is also discussed. It is found out that out-of-sequence delivery deteriorates the performance of FTP at high bit rates. Not only that, the performance

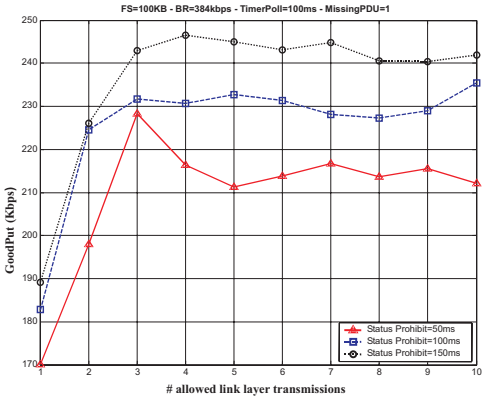

Fig. 4. Goodput of FTP download sessions

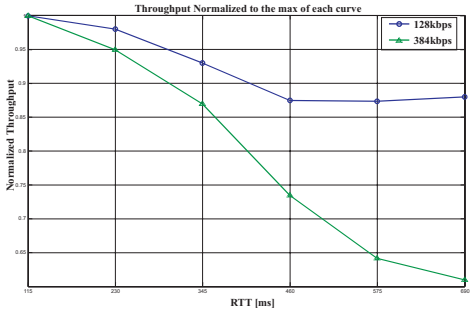

Fig. 5. Effect of RTT on FTP download sessions

becomes even worse if TCP FACK/SACK are used as they employ aggressive retransmission strategies that lead to redundant link and transport layer retransmissions.

The impact of RTT on FTP performance is shown in fig. 5. A 100KB file download sessions are tested for different RTT values, and for different bit rates. As shown in the figure, the RTT greatly affects the throughput and the effects are more pronounced for the high bit rate connection.

Finally, fig. 6 shows the comparison of the throughput for HTTP web browsing for different technologies. The only thing that is modified in the emulator for these tests is some of the parameters, specially that concern processing delays of different components and the available bandwidth, as GPRS, EDGE and UMTS, use almost the same link layer mechanisms. The test results represent the average throughput while browsing the www.nokia.com web page under the different settings. For the UMTS case three different bit rates of 64, 128 and 384kbps are used in the DL, while the UL bandwidth was fixed at 64kbps. For the GPRS case the bandwidth available is 21.4kbps and for EDGE it is set to 64kbps, in both directions. As shown in the figure, UMTS performs much better than the other technologies, but still the bandwidth utilization (percentage of available bandwidth used) becomes worse due to the inherent nature of HTTP and TCP protocols. These low bandwidth utilization for HTTP services agrees with recent

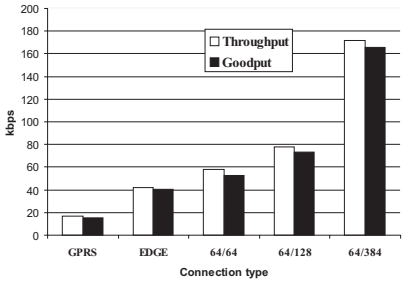

Fig. 6. Comparison of the performance of web browsing throughput and goodput of different technologies

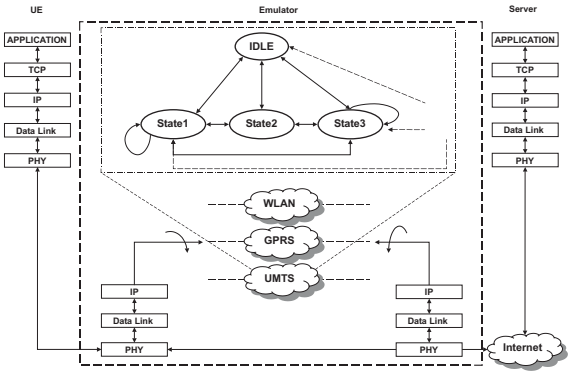

Fig. 7. Extending RESPECT for heterogeneous networks.

live network measurement results described in [20].

VI. Ongoing modifications

Several wireless technologies exist, each one with its own advantages and limitations. For example, GPRS/UMTS provide high mobility but with limited bandwidth while WLAN provides higher bandwidth with restricted mobility. Considering this, the current trend in wireless networks is not one network type replacing another but rather the interoperability between different networks. Due to this fact, we are extending RESPECT to support a heterogeneous network comprising UMTS, WLAN, and GPRS networks.

Fig. 7 shows the structure of this new extension. The detailed UMTS protocol stack implementation that was shown in fig. 2 is removed, and instead, the network behavior is described in a probabilistic model. A network simulator is being developed that will generate the parameters for the model. In the simulator, an overlayed layout consisting of the three different networks is initialized in a wrap around configuration. Depending on the scenario to be investigated, some of the cells from each network are activated, and the load in each activated cell is distributed accordingly. A user is released into the network, following a certain mobility pattern (which at the moment is limited to a constant speed and constant direction), and it is assumed the user always has something to transmit. Depending on the instantaneous load, the instantaneous channel conditions and the link adaptation mechanisms that are employed in each network, the user experiences bit rate fluctuations. Also, when the user reaches an area of common coverage by two or more networks, a handover maybe initiated if the new network is able to provide a better connection. The bit rate fluctuations and the handover decisions are captured using higher order Markov chains. These models are then fed into RESPECT.

VII. Conclusions and future Work

There is an increasing interest in the proper provision of QoS in cellular/wireless networks, as it would lead to the satisfaction of both the end user (in terms of service guarantees) and network operators (with efficient utilization of resources and hence increased revenues). Usually, QoS is measured by some basic parameters such as delay, throughput, and delay jitter. However, QoS is a subjective issue and it is very difficult to find a set of parameters that satisfy every user. So, as the main impact of QoS provision is on the end user, a detailed usability study of QoS provision is a necessity. It is with this main aim in mind that RESPECT was developed. RESPECT have used so far performance investigation of FTP and HTTP services in UMTS and other related networks. Preparations are being made to undertake usability tests where subjective QoS will be compared with the objective QoS metrics gathered so far. The emulator is being extended to support a heterogeneous network consisting of UMTS, GPRS, and WLAN. Arrangements are also being made to make remote access to RESPECT possible for the research community.

References

[1] L. Kleinrock. *"Communication nets; stochastic message flow and delay"*. Mc-Graw-Hill, 1964.

[2] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. "Modeling TCP Throughput: A Simple Model and its Empirical Validation". In *ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, 1998.

[3] The network simulator-ns-2. http://www.isi.edu/nsnam/ns/.

[4] D. D. Clark, V. Jacobson, J. Romkey, and H. Salwen. "An Analysis of TCP Processing Overhead". *IEEE Communications Magazine*, pages 94–101, 2002.

[5] W. Milliken. "The Long-Link Emulator System Description". BBN Systems and Technolgies, White Paper, September 1993.

[6] L. Rizzo. "Dummynet: A Simple Approach to the Evaluation of Network Protocols". *ACM Computer Communication Review*, 27:31–41, 1997.

[7] B. White. "An Integrated Experiment Environment for Distributed Systems and Networks". *USENIX Symposium on Operating Systems Design and Implementation*, pages 255–270, December 2002.

[8] B. Noble, M. Satyanarayanan, G. T. Nguyen, and R. H. Katz. "Trace-Based Mobile Network Emulation". *SIGCOMM*, pages 51–61, 1997.

[9] T. Perennou, E. Conchon, and L. Dairaine. "Two-stage Wireless Network Emulation". *Western Simulation MultiConference*, 2004.

[10] X. Revs, A. Umbert, R. Ferrs, A. Gelonch, and F. Casadevall. "Real-Time Demonstrator for QoS testing in 3G/4G mobile networks with IP Multimedia Applications". *The 5th European Wireless Conference*, February 2004.

[11] O. Sallent, J. Perez-Romero, F. Casadevall, and R. Augsti. "An Emulator Framework for a New Radio Resource Management for QoS Guaranteed Services in W-CDMA Systems". *IEEE J. Select. Areas Commun.*, 19(10):1893–1904, October 2001.

[12] O. M. Teyeb. "Design and Implementation of a UMTS Emulation Platform". Technical report, Aalborg University, 2004. http://cpk.auc.dk/FACE/documents/deliverable_2.2.pdf.

[13] The Netfilter/Iptalbes Project. http://www.netfilter.org/.

[14] The TCPtrace Project. http://www.tcptrace.org/.

[15] A. Battou, B. Khan, D. C. Lee, S. Marsh, S. Mountcastle, and D. Talmage. "CASiNO: component architecture for simulating network objects". *Software:Practices and Experience*, 2002.

[16] CASINO documentation. http://www.nrl.navy.mil/ccs/project/public/casino.

[17] 3GGP TS 25.322 v5.4.0. *RLC Protocol Specification*, March 2003.

[18] O. M. Teyeb, M. Boussif, T. Sørensen, J. Wigard, and P. E. Mogensen. "Emulation Based Performance Investigation of FTP File Downloads over UMTS Dedicated Channels". *Lecture Notes in Computer Science*, 3420:388–396, April 2005.

[19] O. M. Teyeb, M. Boussif, T. Sørensen, J. Wigard, and P. E. Mogensen. "The Impact of RLC Delivery Sequence on FTP Performance in UMTS". *The Eigth International Symposium on Wireless Personal Multimedia Communications (WPMC05)*, September 2005. accepted for publication.

[20] C. Gomez, M. Catalan, D. Viamonte, J. Paradells, and A. Calveras. "Internet Traffic Analysis and Optimization over a Precommercial Live UMTS Network". In *Vehicular Technology Conference, Spring*, May 2005.