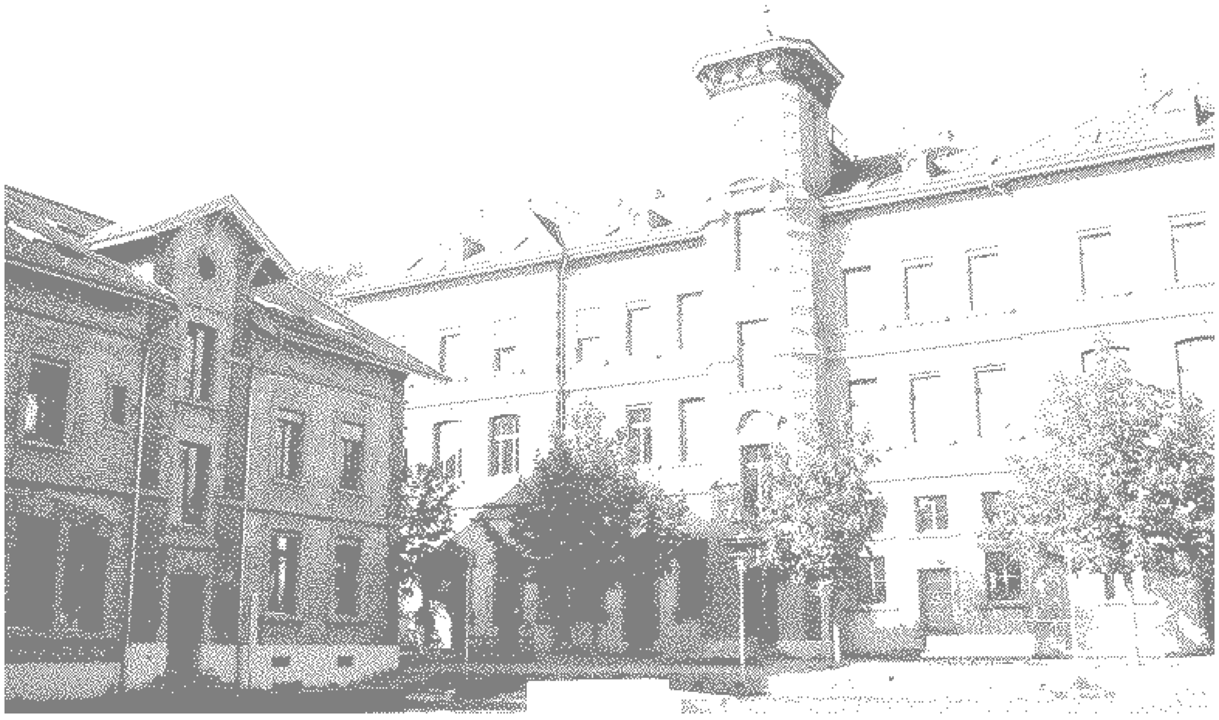
Critical Issues of Centralized and Cloudified LTE-FDD Radio Access Networks

I. Alyafawi, E. Schiller, T. Braun, D. Dimitrova, A. Gomes, N. Nikaein

Technical Report IAM-14-002, October 22, 2014

Institut für Informatik und angewandte Mathematik, www.iam.unibe.ch



Critical Issues of Centralized and Cloudified LTE-FDD Radio Access Networks

Islam Alyafawi, Eryk Schiller, Torsten Braun, Desislava Dimitrova, Andre Gomes, Navid Nikaein

Technical Report IAM-14-002, October 22, 2014

CR Categories and Subject Descriptors:
C.2.1 [Computer-Communication Networks]: Network Architecture and Design; C.4 [Performance of Systems]: Measurement techniques;

General Terms:
Design, Measurement, Performance, Verification

Additional Key Words:
Cloud Computing, Real-Time Cloud Computing, Radio Access Network, Cloudification, Long Term Evolution

Institut für Informatik und angewandte Mathematik, Universität Bern

Abstract

Cloudification of the Centralized-Radio Access Network (C-RAN), in which signal processing runs on general purpose processors inside virtual machines, has lately received significant attention. Due to short deadlines in the LTE Frequency Division Duplex access method, processing time fluctuations introduced by the virtualization process have a deep impact on C-RAN performance. This report evaluates bottlenecks of the OpenAir-Interface (OAI is an open-source software-based implementation of LTE) cloud performance, provides feasibility studies on C-RAN execution, and introduces a cloud architecture that significantly reduces the encountered execution problems. In typical cloud environments, the OAI processing time deadlines cannot be guaranteed. Our proposed cloud architecture shows good characteristics for the OAI cloud execution. As an example, in our setup more than 99.5% processed LTE subframes reach reasonable processing deadlines close to performance of a dedicated machine.

Contents

1 Introduction

Today, we are experiencing a significant increase in end-user data consumption due to an ever-growing number of Internet-capable mobile devices equipped with 3G and 4G. To quickly satisfy increasing mobile traffic demands, operators are forced to seek new cost-effective solutions allowing for upgrades and scaling of the radio network. Current costs of building and operating a new infrastructure able to supply required data rates are superior to the revenue growth rate [1]. The main reason for the high upgrade/maintenance cost is the architecture of the mobile telecommunications network, in which the integrated Base Transceiver Station (BTS) consists of hardware and software components that build a complete Radio Access Network (RAN). A big problem emerges, because typical mobile telephony providers operate on a large scale. Therefore, they have to provide a high BTS density over large geographical areas by installing and maintaining a huge number of expensive integrated BTSs. The problem will grow in the future with the trend of smaller and smaller cell sizes, e.g., picocells.

A new cost-effective RAN solution has to satisfy a multitude of requirements. First, it has to allow for fast upgrades and scaling to satisfy the demand for quickly increasing and highly variable mobile traffic. Second, high capacity and network coverage have to be provided at reduced power consumption to provide a competitive mobile broadband service. Finally, mobile operators need to upgrade their network frequently and operate with multiple air-interfaces in a heterogeneous manner to meet ever-increasing amounts of mobile data traffic.

Centralized-RAN (CRAN) [2][3] could be a solution to reduce costs and power consumption by sharing resources and exploiting load patterns of a certain geographical area at a given time (spatio-temporal load patterns). This solution allows reacting to changes in user data traffic and mobility patterns. It can also allow for increasing spectral efficiency (data rates) by coordinated and joint signal processing. The current RAN architecture is not energy efficient, because only 15–20% of all sites are loaded more than 50% of the total capacity [4]. One of the major benefits of a C-RAN could be a perfect match between computational resources and hence power consumption with spatio-temporal traffic statistics over fairly large geographic areas. Moreover, since signal processing is centralized, it allows for more sophisticated joint spatio-temporal processing of radio signals, which could drastically increase spectral-efficiency. This approach is considered by European projects such as Mobile Cloud Net-

working (MCN) [5], Sail [6] and iJoin [7]. This work has been carried out as part of the MCN vision towards extending the cloud computing paradigm to radio communication networks.
Next steps to lower the costs of running a RAN rely on the successful adoption of virtualization and cloud computing technology allowing for a) migration from expensive specific hardware to general-purpose IT platforms, b) load balancing and c) rapid deployment and service provisioning. Virtualization and cloud computing come at the cost of higher software complexity. The cloudification of the RAN is not new and the benefit of such an approach has demonstrated 71% of power savings in comparison to the existing system [8].
Recent efforts [9] have shown the feasibility of software implementation of LTE RAN functions over General Purpose Processors (GPPs), rather than the traditional implementation over Digital Signal Processors (DSPs), Field-Programmable Gate Arrays (FPGAs) or Application-Specific Integrated Circuits (ASICs). Several software implementations of the LTE evolved Node B (eNB)[1] already exist, e.g., a) Intel solutions based on a hybrid GPP-accelerator architecture aiming at a balance between a flexible IT platform, high computing performance and good energy efficiency [10], b) Amarisoft LTE solution featuring a fully-functional pure-software LTE eNB [11] and c) OpenAirInterface (OAI), developed by EURECOM, which is an open-source Software Defined Radio (SDR) implementation of LTE including both the RAN and the Evolved Packet Core (EPC) [12].
In this paper we evaluate the performance of OAI using GPPs and cloud environments to execute the LTE FDD physical layer (PHY). Our results are based on running the Open Air Interface (OAI) LTE-Release-8 on a cloud platform. This approach provides a precise method for the estimation of required processing resources to adequately handle traffic load by identifying processing bottlenecks. Our main contribution is a description of bottlenecks in cloud environments and a suggestion of a new execution model of the OAI-based LTE PHY layer.
This paper has the following organization. In Section 2, we introduce the LTE FDD PHY layer and calculate its maximal signal processing delay. Section 3 describes the architecture, benefits and challenges of centralized-RAN. The Cloud-based LTE RAN (Cloud-RAN) and its implementations are described in Section 4. Our evaluation of Cloud-RAN with various setups is provided in Section 5. Finally, we conclude and elaborate future works in Section 6.

[1] Currently in LTE, the BTS is referred to as eNB.

2 Processing budget in LTE FDD

This paper considers LTE Frequency Division Duplex (FDD), which requires signal processing with short delays at the subframe level on the PHY layer. The most critical processing deadline is imposed by the Hybrid Automatic Repeat Request protocol (HARQ) on the MAC layer. HARQ, which is a retransmission protocol between eNB and User Equipment (UE), states that the reception status of every received subframe has to be reported back to the transmitter. In LTE FDD-based networks, the HARQ Round Trip Time (RTT) equals 8 ms. The transmission time of a LTE time unit (subframe) T_{subframe} is equal to 1 ms. Each packet received at subframe k has to be acknowledged through an Ack or Nack at subframe k+4, which in turn has to be decoded at the transmitter before assembling subframe k+8. This is due to the fact that acknowledgements steer the retransmission mechanism (i.e., the transmitter has to decide on retransmitting the previously sent information or transmitting a new chunk of data). The total delay budget is therefore considered as 3 ms at the eNB.

From the functional perspective, every eNB consists of a signal processing Base-Band Unit (BBU) and a Remote Radio Head (RRH) [1], which is responsible for transmitting and receiving; they both combined provide the Radio Access Network (RAN).

In this paper, we put a particular focus on the LTE FDD, which consist of the following layers: i) LTE physical layer (PHY) with symbol-level processing, ii) Medium Access Control (MAC) layer, which supports wide-band multiuser scheduling and HARQ. The LTE PHY layer uses an asymmetric access scheme consisting of Orthogonal Frequency-Division Multiple Access (OFDMA) on the downlink and Single-Carrier Frequency-Division Multiple Access (SC-FDMA) on the uplink. The effective data rate over the air interface (goodput) is steered by the Modulation and Coding Scheme (MCS). According to the 3GPP standards, the MCS index varies between 0 and 27, and carries out information about the modulation such as QPSK, 16QAM and 64QAM as well as the coding rate. Modulation decides on the number of bits per symbol, while the code rate defines the amount of redundant information inserted into the data stream [1, 13]. In the LTE PHY layer, the smallest chunk of data transmitted by the LTE eNB is called Physical Resource Block (PRB). The LTE technology uses radio channels of 1.4 MHz, 3 MHz, 5 MHz, 10 MHz, 15 MHz, and 20 MHz, which can allocate 6, 15, 25, 50, 75, and 100 PRBs respectively. Consequently, the MCS index number of PRBs and radio signal bandwidth have an impact

on the generated workload.
From the functional perspective, every eNB consists of a signal processing Base-Band Unit (BBU) and a Remote Radio Head (RRH) [1], which is responsible for transmitting and receiving; they both combined provide the Radio Access Network (RAN).
In this paper, we put a particular focus on the LTE FDD, which consist of the following layers: i) LTE physical layer (PHY) with symbol-level processing, ii) Medium Access Control (MAC) layer, which supports wide-band multiuser scheduling and HARQ. The LTE PHY layer uses an asymmetric access scheme consisting of Orthogonal Frequency-Division Multiple Access (OFDMA) on the downlink and Single-Carrier Frequency-Division Multiple Access (SC-FDMA) on the uplink. The effective data rate over the air interface (goodput) is steered by the Modulation and Coding Scheme (MCS). According to the 3GPP standards, the MCS index varies between 0 and 27, and carries out information about the modulation such as QPSK, 16QAM and 64QAM as well as the coding rate. Modulation decides on the number of bits per symbol, while the code rate defines the amount of redundant information inserted into the data stream [1, 13]. In the LTE PHY layer, the smallest chunk of data transmitted by the LTE eNB is called Physical Resource Block (PRB). The LTE technology uses radio channels of 1.4 MHz, 3 MHz, 5 MHz, 10 MHz, 15 MHz, and 20 MHz, which can allocate 6, 15, 25, 50, 75, and 100 PRBs respectively. Consequently, the MCS index number of PRBs and radio signal bandwidth have an impact on the generated workload.

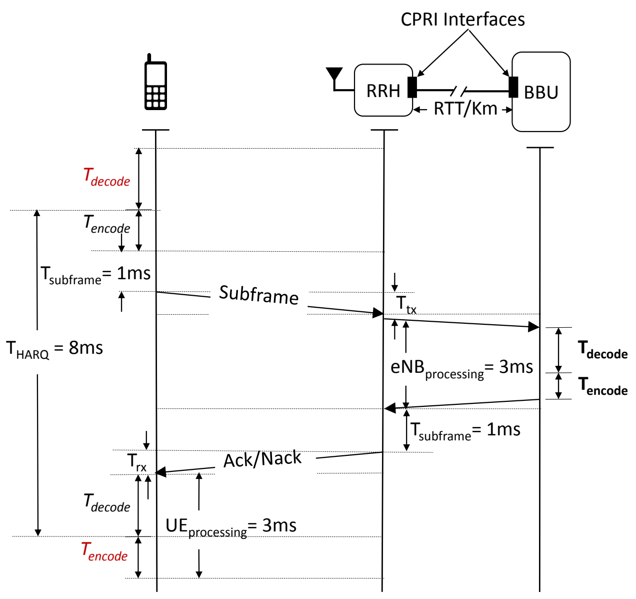


Figure 1: HARQ process timing requirement.

3 Centralized RAN in the LTE Network

In networks with C-RAN, BBU is no longer maintained alongside with a BTS at a remote location. The BBU and RRH are decoupled; the RRH remains at the previous location of the BTS, while the BBU migrates to a centralized processing pool (c.f., Fig. 1). The role of the centralized processing pool is to host a large number of BBUs [9]. Every BBU has to be connected to its RRH through a point-to-point high-speed interface, such as Common Public Radio Interface (CPRI). A very fast link of low delay is necessary as the BBU processes the most computationally heavy physical (PHY) layer of the LTE standards. In Section 2, we calculated the maximum processing delay in LTE FDD. When the BBU and RRH are separated, the processing time at the BBU is reduced and the delay calculation has to consider propagation delays and interface latencies between RRH and BBU. According to *Chanclou et al.* [14], the RTT between RRH and BBU equipped with a CPRI link cannot exceed 700 μs for LTE and 400 μs for LTE-Advanced. Hence, the length of a BBU-RRH link should not exceed 15 km to avoid too high round-trip-delays, while the speed of light in fiber is approximately 200 m/μs. Consequently, this leaves the BBU PHY layer only with around 2.3–2.6 ms for signal processing at a centralized processing pool.

Let us give an example, once the BBU received a subframe (1 ms duration) from the RRH, the BBU has to decode the subframe as well as assemble and return another subframe back to the RRH within a hard deadline ≤ 3 ms depending on the distance between RRH and BBU.

3.1 RAN on a PC

As illustrated in Fig. 2, we focus on the PHY layer, which is the signal processing bottleneck as mentioned in Sec. 2). To run a BBU on a signal processing pool equipped with typical GPP-based computers, we have to operate software-based equivalents of the functionality provided by dedicated hardware previously. The processing architecture consists of the Operating System (OS) equipped with drivers to the CPRI interface and the BBU functionalities implemented as software applications. In the following, we further describe the required properties of the OS and applications deployed.
A general purpose OS requires a kernel, which uses scheduling algorithms to provide processing time to applications. In theory, the kernel is able to instantaneously suspend every user-level task, but in practice, some parts of the kernel code are not preemptible and introduce unpredictable delays. Due to the fact that in C-RAN, BBU is an application, it has to be provided with processing time within a short interrupt-response delay of 100 μs [15] and be able to uninterruptedly process the task within a given processing time window. Such a processing scheme cannot be secured by a typical OS, and therefore a Real Time (RT) OS such as RTLinux is required at the BBU to provide appropriate timing and to avoid processing time fluctuations in our target scenarios. Also, the OAI process, which executes signal processing on the RTLinux operating system, has to be prioritized.

3.2 OpenAirInterface

As mentioned in Sec. 1, there are different implementations of a software-based BBU. In this paper, we consider the OpenAirInterface (OAI) deployed on RTLinux. The OAI emulation platform is open-source software that implements the LTE 3GPP Release-8 standards [16]. OAI provides a complete wireless protocol stack containing the PHY, MAC, Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP) and Radio Resource Control (RRC) layers as well as Non-Access-Stratum (NAS) drivers for IPv4/IPv6 interconnection with other network services [17]. Regarding

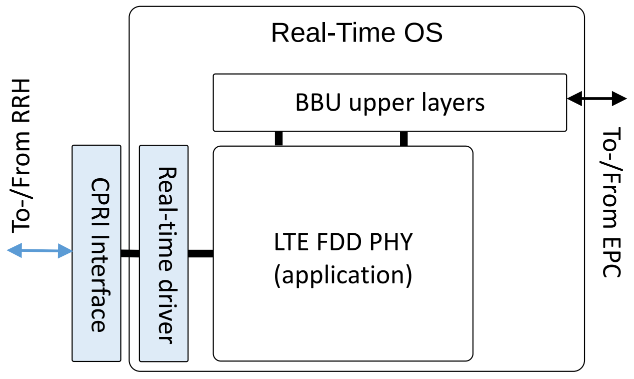


Figure 2: RAN architecture on PC.

the PHY layer, OAI implements the signal processing chain for OFDMA (Downlink) and SC-FDMA (Uplink) as described in Sec. 2. OAI also uses optimized C code for Intel architectures (using MMX/SSE3/SSE4 instruction sets for signal processing) for efficient numerical operations. Fig. 3 illustrates the intended signal processing in OAI [17]. Let us consider that at time instance (1), a mobile device started issuing subframe $N-1$. Once the OAI modem received the complete subframe at (2), a decoding thread at (a) starts processing it. Decoding has to finish within 2 ms at (b), because an encoder thread starting at (b) requires input from the decoder (e.g., to encode an Ack or Nack message). The following subframe has to be scheduled for transmission at (3), hence the encoder is left with only 1 ms for the assembling procedure. Notice, that this description does not contain link and propagation delays that further shorten the processing time by a few hundred microseconds. Therefore, the LTE-OAI has at most 2 ms for decoding and 1 ms for encoding a subframe.

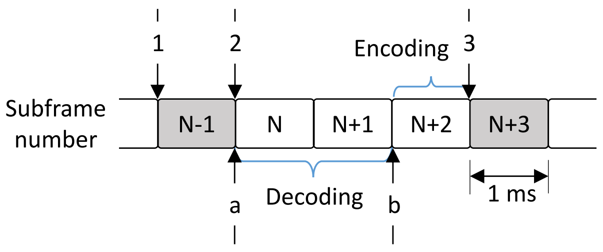


Figure 3: Processing orders in OAI.

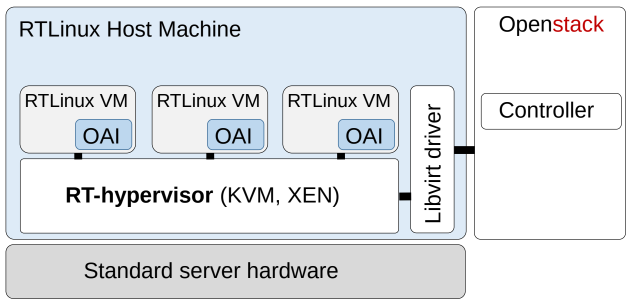


Figure 4: OpenStack management architecture.

4 Cloud-based LTE RAN

Cloudification of C-RAN (software-based BBU), to which we refer now as Cloud-RAN, is an interesting concept for both cloud providers and mobile telephony operators. There are many cloud computing paradigms such as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) [18]. In the following, however, we concentrate on IaaS, which provides users with Virtual Machines (VMs) having processing capabilities, storage, network and other optional services to support various user applications. Running multiple VMs is accomplished through virtualization, which is a process that allows a single physical machine to simultaneously act as a few logical entities by using a software layer called “hypervisor”. Cloud-RAN might be executed on a public cloud platform, in which multiple VMs share computational and storage resources. In such environments, processing time deadlines cannot be guaranteed, while typical practices of cloud providers such as over-subscription of resources (e.g., processors) amplify this trend even further. Therefore, a new model of organizing a publicly available data center has to be worked out. Note that currently, data-centers do not provide virtualization with real time support.

Our starting point is OpenStack, a well known cloud management system that orchestrates various components such as compute, storage and networking to control and manage the execution of VMs on the physical server pool at a data-center. The task of OpenStack is to configure VMs on physical host machines. As illustrated in Fig. 4, our modifications of the typical execution stack include the installation of the RTLinux kernel on the host machine. On the host, we prioritize the KVM hypervisor, which is one of the most popular hypervisors of type 2 in the OpenStack community (we refer to it as RT-hypervisor). The real-time prioritization of the KVM process is provided through the `chrt` Linux command, e.g., `chrt -p --rr 1 {pid}`. Real-time computing on the guest also requires the installation of the RTLinux kernel and the real-time prioritization of the OAI application.

5 Evaluation

In this section, we present our evaluation of the architecture described in Sec. 4. We are interested in processing delays of the OAI application for downlink and uplink processing on VMs and physical (not virtualized) machines (c.f., Sec. 3, Fig. 2). This procedure is important for understanding whether Cloud-RAN based on OpenStack, KVM, and Linux can provide satisfactory processing deadlines and could be used as an execution platform for RAN.

5.1 Hardware Setup

The experimental setup of our testbeds is as follows. A dedicated Intel i5 GPP with configurable CPU frequencies between 1600 and 3300 MHz is used. The memory resources are always fixed to 2 GB RAM (on the physical and virtual testbeds). All the machines (hosts and guests) operate the Ubuntu Linux 12.04 distribution; the kernel version deployed is 3.12.

5.2 Experiments

We are using OAI as a benchmark for profiling the processing time of the LTE PHY layer given different load scenarios configured through PRBs, MCS indices, and radio signal bandwidth. More specifically, we are using two benchmarking tools called “dlsim” and “ulsim” emulating the Physical Downlink Shared Channel (PDSCH) and the Physical Uplink Shared Channel (PUSCH) respectively. Both tools are designed to emulate the behavior of the eNB and UE PHY layers over a simulated wireless medium. The execution time of each signal processing module of downlink and uplink is calculated using timestamps at the beginning and at the end of computing. OAI uses the *RDTSC* instruction implemented on all x86 and x64 processors as of the Pentium to get very precise timestamps. *RDTSC* counts the number of CPU clocks since a reset. Therefore, the execution time is proportional to the value returned by the following algorithm:

```
start = rdtsc();
compute();
stop = rdtsc();
diff = stop - start;
return diff;
```

To get T_{Process} in seconds, we have to divide `diff` by the CPU frequency.

For statistical analysis, we first gather a large number of `diff` samples (i.e., 10000) to calculate the median, first quantile, third quantile, minimum, and maximum processing time for all subframes in uplink and downlink at the BBU side.

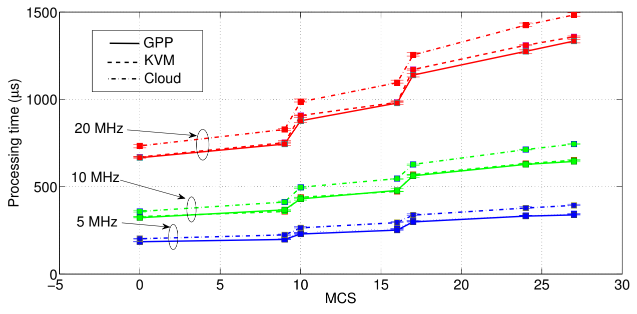
5.3 Results and Analysis

5.3.1 Processing Time

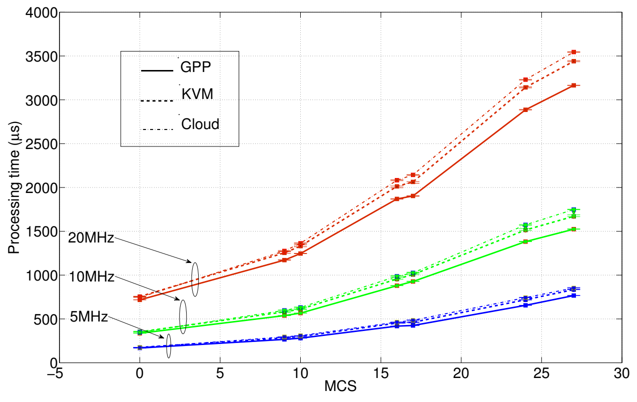
In this section, we study the decoding/encoding processing time by using the receiver/transmitter part of OAI ulsim/dlsim with fixed CPU frequency equal to 2.4 GHz, as illustrated in Figs. 5(a), 5(b). In each figure, we plot the overall processing time for various modulation and coding schemes (MCS: 0, 9, 10, 16, 17, 24, and 27), radio signal bandwidth (5, 10 and 20 MHz), and machine environments: dedicated GPP, KVM, and cloud (ZHAW Openstack-based private cloud [19]). In our figures, the 1st and 3rd quantiles are denoted with short horizontal lines, which in most of the cases lie very close to each other; medians are depicted with filled squares. From our figures, we can draw the following conclusions. The decoding and encoding processing time for LTE subframes grow with the increase of MCS given 25, 50, 100 PRB and 5, 10, 20 MHz bandwidth. On average, the decoding time is twice as long as the encoding time. Hence, the sequential organization of OAI including 2 ms for decoding and 1 ms for encoding is sound. Given that all the considered machines have the same RAM size and CPU frequency, we see that the median values for the cloud VM show lower performance (more processing time) than KVM VMs and GPP machines; this is probably due to resource sharing and slightly different CPUs deployed in the cloud environment. Notice that in the case of OAI, signal processing is executed on a per subframe basis. The encoding process starts 2 ms after the decoding process. Therefore, the decoding process should not compute for more than 2 ms, while in such a case, the encoder has to assemble a Nack message (even if the message was properly received). This increases retransmission rates and degrades the overall goodput.

5.3.2 Required CPU frequency

It is important from the system design point of view to understand the minimum required CPU frequency that fully supports a given data rate. Fig. 6 illustrates the processing time of an eNB (decoding SC-FDMA subframe



(a) Downlink



(b) Uplink

Figure 5: eNB processing time for transmitting packets.

and encoding OFDMA subframe) given different CPU frequencies (1.6, 2.0, 2.4, 2.8, and 3.3 GHz). Note that we consider the worst case scenario defined by the LTE standards, which stated that UE transmits a PUSCH subframe with MCS equals 27 (UE category 5) and the eNB transmits the Ack/Nack using a PDSCH channel. In order to perform experiments with different CPU frequencies, we used Linux *cpupower* tool to limit the available CPU clock [20]. In Fig. 6, we observe the reciprocal behavior of the processing time against CPU frequency. We therefore fit a model of the processing time T$_{\mathrm{subframe}}$ valid for any Intel based processor, which is expressed by the following formula:

$$T_{\mathrm{subframe}}(x)\;[\mathrm{us}] = \alpha/x, \tag{1}$$

where $\alpha = 11740 \pm 26$ for uplink MCS = 27 or $\alpha = 8092 \pm 34$ for uplink MCS = 16 (for currently existing UE of category 4) and x is CPU frequency measured in GHz (note that the downlink MCS is always equal to 27). This

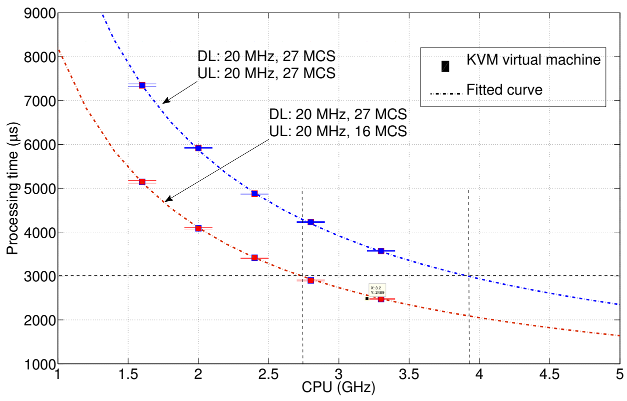


Figure 6: Processing time for transmitting (OFDMA) and receiving (SC-FDMA) packets at eNB given full PRB allocation at 20 MHz.

formula allows us to estimate that cloud operators require VMs with at least 4 GHz CPUs to support the LTE-FDD PHY layer with maximum load.

5.3.3 Processing Time Distribution

In Fig. 7, we depict the Complementary Cumulative Distribution Function (CCDF) of the overall signal processing time for data subframes encoded with MCS 16 over 20 MHz bandwidth (max. for currently available UE of category 4). The left hand side of the figure is a zoom of the interesting region between 2000 and 2400 μs. The CCDF plot for a given value t_x displays the fraction of subframes with execution times grater than t_x. We tested different configuration scenarios. GPP and RTLinux use the configuration setup from Sec. 3, Fig. 2, in which OAI runs on dedicated hardware. In the case of GPP, OAI operates on a typical Linux kernel 3.12, while for RTLinux, the kernel is equipped with a real-time scheduler and OAI is prioritized as real-time. We see that the execution time is stable for both GPP and RTLinux. Notice that we do not study interrupt response delays as our channel is fully simulated and the machine load remains at a low level. In KVM, KVM-RT-OAI, and RT KVM-RT OAI, execution is performed on VMs of a similar setup. The main difference among them is that RT-KVM RT-OAI has both the KVM hypervisor and the OAI process prioritized as real-time, KVM-RT OAI only prioritizes the OAI interface, while the KVM scenario does not use prioritization at all. We also perform experiments on a private cloud serving a small number of VMs and a heavy loaded public cloud. Currently, in the considered cloud environments, we

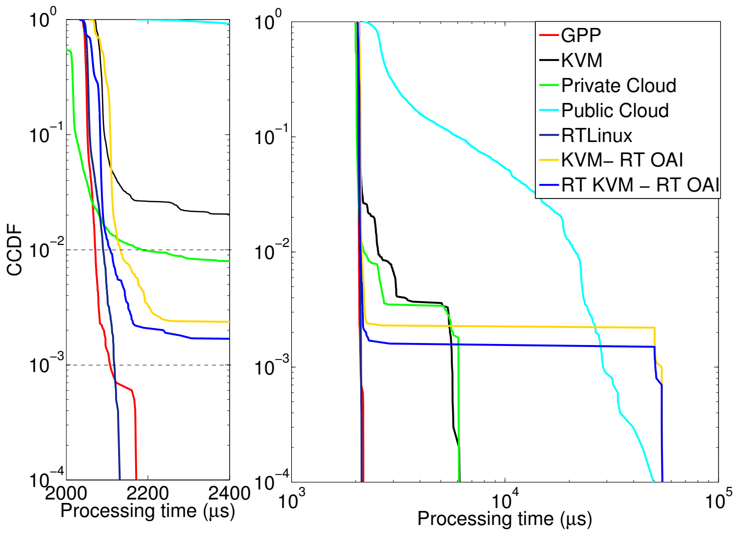


Figure 7: Processing time distribution for received packets (at eNB) in the Cloud environment with BW=20 MHz, MCS=16.

are unable to study real-time prioritization. From our observations, the processing time of virtualized environments is mostly skewed to longer runs due to high variations in the cloud and KVM environments in comparison to a dedicated GPP. For the RT KVM-RT OAI setup, however, the execution time remains close to result obtained for the physical machine in 99.5% of subframes, which is not the case for other scenarios, e.g., KVM heavily diverges from the physical machine behavior for more than 2% of subframes processed. Please note the huge variations in processing time on a public cloud. It leads to unpredictable behaviors, while some subframes surpass the 2 ms deadline for low MCS indices. Through our real-time KVM approach, we were able to reduce the number of fluctuations in processing time, but we were unable to get rid of them completely.

6 Conclusions

In this paper, we have studied and analyzed several important aspects of the LTE radio access network cloudification. We have evaluated OAI in different environments such as dedicated GPP, KVM, RT-KVM, and KVM-based Clouds. Our findings are many-fold. First, we have benchmarked the encoding and decoding workload of the LTE FDD PHY layer subframes on GPP and cloud environments by using the OAI. Second, the reciprocal behavior of the OAI execution time on the Intel CPU family was provided, therefore the processing power required for any type of a physical subframe processing can be estimated. Finally, the bottlenecks of the OAI cloud execution were identified and new models of cloud execution were suggested. We have shown that our configuration setup greatly improves the execution time of OAI signal processing allowing 99.5% of subframes to process data within a reasonable deadline. In the future, we plan to explore implementational details of KVM and further improve deadline handling.

References

[1] E. Dahlman, S. Parkvall, and J. Skld, *4G LTE/LTE-Advanced for Mobile Broadband*. Academic Press, UK, 2011.

[2] NGMN, "Suggestions on Potential Solutions to C-RAN by NGMN Alliance," tech. rep., The Next Generation Mobile Networks (NGMN) Alliance, Jan. 2013.

[3] B. Haberland, F. Derakhshan, H. Grob-Lipski, R. Klotsche, W. Rehm, P. Schefczik, and M. Soellner, "Radio Base Stations in the Cloud," *Bell Labs Technical Journal*, vol. 18, no. 1, pp. 129–152, 2013.

[4] X. Costa-Prez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio Access Network Virtualization for Future Mobile Carrier Networks," *IEEE Communications Magazine*, 2013.

[5] "Mobile Cloud Networking project (FP7-ICT-318109)," *[Online]. Available: http://www.mobile-cloud-networking.eu.*, 2014.

[6] "Scalable and Adaptive Internet Solutions (SAIL). EU FP7 Official Website," *[Online]. Available:http://www.sail-project.eu.*, 2012.

[7] "iJOIN: an FP7 STREP project co-funded by the European Commission under the ICT theme," *[Online]. Available: http://www.ict-ijoin.eu/.*

[8] C. M. R. Institute, "C-RAN White Paper: The Road Towards Green RAN," *[Online]. Available: http://labs.chinamobile.com/cran*, 2013.

[9] D. Wbben, P. Rost, J. Bartelt, M. Lalam, V. Savin, M. Gorgoglione, A. Dekorsy, and G. Fettweis, "Benefits and Impact of Cloud Computing on 5G Signal Processing," *Special Issue "The 5G Revolution" of the IEEE Signal Processing Magazine*, 2014.

[10] R. Schooler, "Transforming Networks with NFV & SDN," *Intel Architecture Group*, 2013.

[11] "Amari LTE 100, Software LTE base station on PC," *[Online]. Available: http://www.amarisoft.com/.*

[12] EURECOM, "Open Air Interface." `http://www.openairinterface.org/`, Oct. 2014.

[13] G. Berardinelli, L. R. de Temino, S. Frattasi, M. Rahman, and P. Mogensen, "OFDMA vs. SC-FDMA: Performance Comparison in Local Area IMT-A Scenarios," *IEEE Wireless Communications*, 2008.

[14] P. Chanclou, A. Pizziant, and et al., "Optical Fiber Solution for Mobile Fronthaul to Achieve Cloud Radio Access Network," *Future Network and Mobile Summit (FutureNetworkSummit)*, 2013.

[15] A. S. Ugal, "Hard Real Time Linux using Xenomai on Intel Multi-Core Processors," *Intel Corporation*, 2009.

[16] "Graduate School and Research Center in Communication Systems," *[Online]. Available: http://www.eurecom.fr/en*.

[17] N. Nikaein, R. Knopp, F. Kaltenberger, L. Gauthier, C. Bonnet, D. Nussbaum, and R. Ghaddab, "OpenAirInterface 4G: an open LTE network in a PC," *International Conference on Mobile Computing and Networking*, 2014.

[18] Oracle, "Oracle Cloud, Enterprise-Grade Cloud Solutions: SaaS, PaaS, and IaaS," *Available Online: [https://cloud.oracle.com/home]*, 2014.

[19] "ZHAW Openstack Testbed:," *[Online]. Available: http://blog.zhaw.ch/icclab/*.

[20] V. Pallipadi and A. Starikovskiy, "The ondemand governor: past, present and future," *Proceedings of Linux Symposium*, 2006.