Cellular Access Multi-Tenancy through Small-Cell Virtualization and Common RF Front-End Sharing

Jose Mendes
NEC Laboratories Europe
Heidelberg, Germany
jose.mendes@neclab.eu

XianJun Jiao
Ghent University - imec, IDLab,
Department of Information Technology
Ghent, Belgium
Xianjun.jiao@ugent.be

Andres Garcia-Saavedra
NEC Laboratories Europe
Heidelberg, Germany
andres.garcia.saavedra@neclab.eu

Felipe Huici
NEC Laboratories Europe
Heidelberg, Germany
felipe.huici@neclab.eu

Ingrid Moerman
Ghent University - imec, IDLab,
Department of Information Technology
Ghent, Belgium
Ingrid.Moerman@UGent.be

ABSTRACT

Mobile traffic demand is expected to grow as much as eight-fold in the coming next five years, putting strain in current wireless infrastructure. One of the most common means for matching these mounting requirements is through network densification, essentially increasing the density of deployment of operators' base stations. In this paper we take a step in that direction by implementing a virtualized base station consisting of multiple, isolated LTE PHY stacks running concurrently on top of a hypervisor deployed on a cheap, off-the-shelf x86 server and a shared radio head. In particular, we show that it is possible to run multiple virtualized base stations while achieving throughput equal or close to the theoretical maximum.

CCS CONCEPTS

• **Networks** → **Wireless access points, base stations and infrastructure**; **Network experimentation**; *Network measurement*; • **Hardware** → *Signal processing systems*; *Wireless devices*;

KEYWORDS

Small-cell virtualization; fronthaul sharing; radio multi-tenancy

ACM Reference Format:
Jose Mendes, XianJun Jiao, Andres Garcia-Saavedra, Felipe Huici, and Ingrid Moerman. 2017. Cellular Access Multi-Tenancy through Small-Cell Virtualization and Common RF Front-End Sharing. In *Proceedings of WiNTECH'17, Snowbird, UT,USA, October 20, 2017,* 8 pages.
https://doi.org/10.1145/3131473.3131474

WiNTECH'17, October 20, 2017, Snowbird, UT,USA

ACM ISBN 978-1-4503-5147-8/17/10…$15.00
https://doi.org/10.1145/3131473.3131474

1 INTRODUCTION

In the coming years, growth in mobile data traffic, fueled by the continued adoption of mobile devices and their use for downloading video and other content, will continue to expand at a rapid pace, with reports claiming as much as an eight-fold increase over the course of the next five years [7]. In that same time period, 70% of the world's population is forecast to use mobile devices. Along these lines, 5G networks are supposed to cope with 1000 times higher data volume per geographical area, 10-100 times more connected devices and 10-100 times higher typical user data rate, among others [22].

Such towering numbers will put significant strain on existing mobile infrastructure. *Network densification* [5] is a well-recognized means to increase spectrum efficiency in cellular systems, and thus data traffic capacity. The obvious way of densifying Radio Access Networks (RANs) is to deploy more radio access points per unit area. However, deploying such infrastructure represents a significant cost for network operators, rendering this approach less than attractive in practice. It is reported, for instance, that today 50% of radio sites yield less than 10% of operators' revenue [14].

In order to achieve a good degree of densification without compromising cost efficiency, *infrastructure sharing* has become a pivotal strategy guiding the design of next generation mobile networks. It is estimated that network sharing can make up for 20% of operational costs in typical European operators, halving the infrastructure cost of passive RAN components (which make up to 50% of the total network cost) [12].

However, efficiently and safely sharing such radio access points remains challenging. In this paper, we argue that the combination of applying virtualization technologies (e.g., Xen, KVM [2, 13]) to base station software, along with the use of inexpensive radio front-ends (also called radio head) is a key enabler of network densification. Virtualization provides the strong isolation needed to safely run multiple (virtualized) base stations belonging to different operators on shared hardware, thus increasing the density of each of those operators' networks and improving the efficiency of the deployed hardware through statistical multiplexing. Regarding radio front-ends, LTE modems based on Software Defined Radio

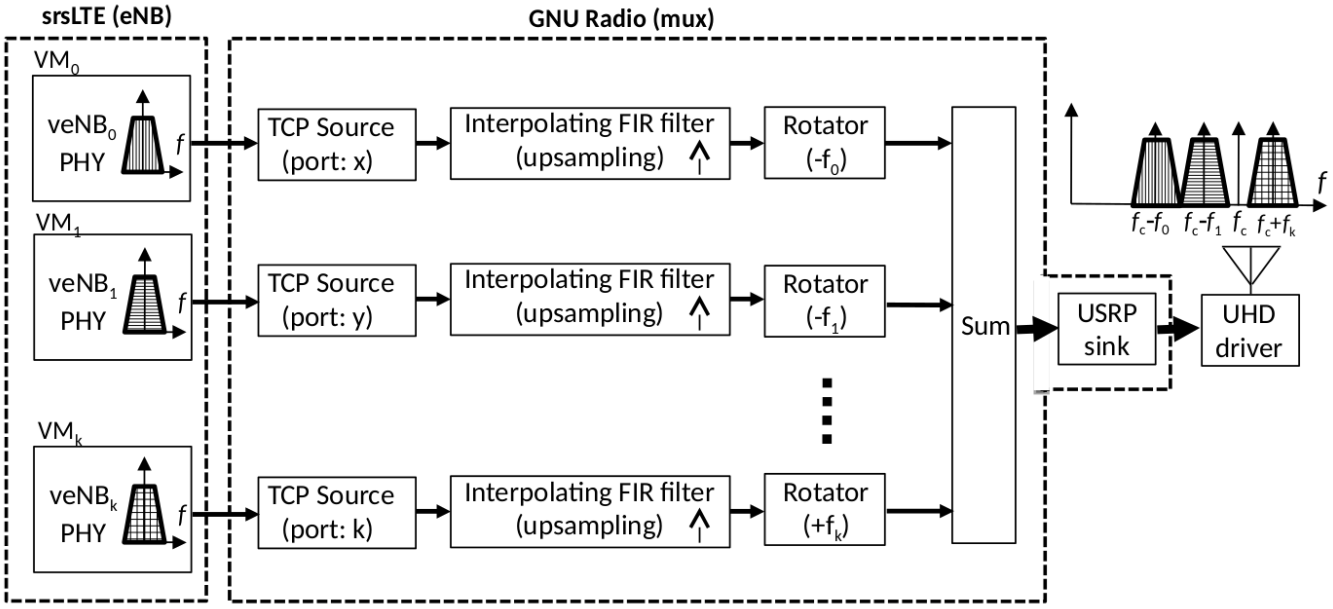


Figure 1: Virtualized base station overall architecture.

(SDR) [10, 16] are gaining momentum as a solution to future dense deployments [20]; a remarkable example is Facebook's OpenCellular project [8].

Towards this vision of network densification and infrastructure sharing, we provide a prototypical implementation of a high performance virtualized platform consisting of an off-the-shelf, inexpensive x86 server running the PHY layer of multiple virtualized LTE base stations (called eNodeBs or eNBs for short) instances along with a common, shared radio head (called SRH hereafter). We focus on the PHY layer since this has been shown to be far the most computationally expensive part of an eNB, sometimes consuming up to 2/3 of the available CPU cyles [4, 19, 25]. In greater detail, in this paper we show that:

- Standard x86 hardware is capable of handling the LTE PHY stacks of multiple (independent) eNBs, properly multiplexing their access to a common radio front-end;
- Inexpensive SDR equipment can satisfy the bandwidth requirements needed by mobile device applications, including content delivery.
- The use of full-fledged virtualization (i.e., as opposed to containers) does not degrade performance.

To the best of our knowledge, this is the first work presenting promising results of multiple, virtualized LTE PHY layer stacks sharing commodity SDR equipment. In particular, our results show that a virtualized eNB can yield throughput at the theoretical maximum rate in some setups, and that up to 4 virtualized eNBs can concurrently run on an inexpensive 4-core x86 server without maxing out its CPU resources.

2 DESIGN AND IMPLEMENTATION

The overall architecture of our virtual eNB environment is depicted in Figure 1. Our system consists of three modules: eNB, mux, and radio front-end.

The first module, represented in the left-most part of Figure 1, consists of a set of virtual eNBs (VeNBs). Each VeNB, in turn, comprises the LTE eNB software itself, a virtualization environment, and a guest operating system running on commodity x86 servers. Regarding software, we use srsLTE [10], a highly modular open-source LTE library which is relatively simple to modify and use. In addition, we choose KVM as our virtualization platform and use Linux guests to house the virtualized base station software.

In the right-most part of the figure, we represent the actual radio front-ends or shared radio heads (SRH). To this aim, we employ an USRP B210, commonly use for software defined radio [10]. This board provices 56 MHz of real-time bandwidth, a programmable Spartan6 FPGA, and fast SuperSpeed USB 3.0 for connectivity with the eNB software. For tests purposes, we employ a set of additional USRP boards and servers deploying the LTE UE software counterpart that allow us to connect to each of the virtual eNBs.

In between, we implement and deploy a mechanism to multiplex signals from the multiple virtual base stations onto the shared radio head (SRH) for transmission (Tx) as well as the ability to split the incoming signal back to the corresponding eNBs, i.e. reception (Rx). To this end, we implement a frequency multiplexing IQ switch that receives IQ samples (digitized radio signals) from the VeNBs and shifts those to different frequency locations in a wider bandwidth. The merged signal, which has higher sampling rate and wider bandwidth than those of the individual VeNBs, is sent to the SRH. To avoid overlapping or interference between the VeNBs, we use a

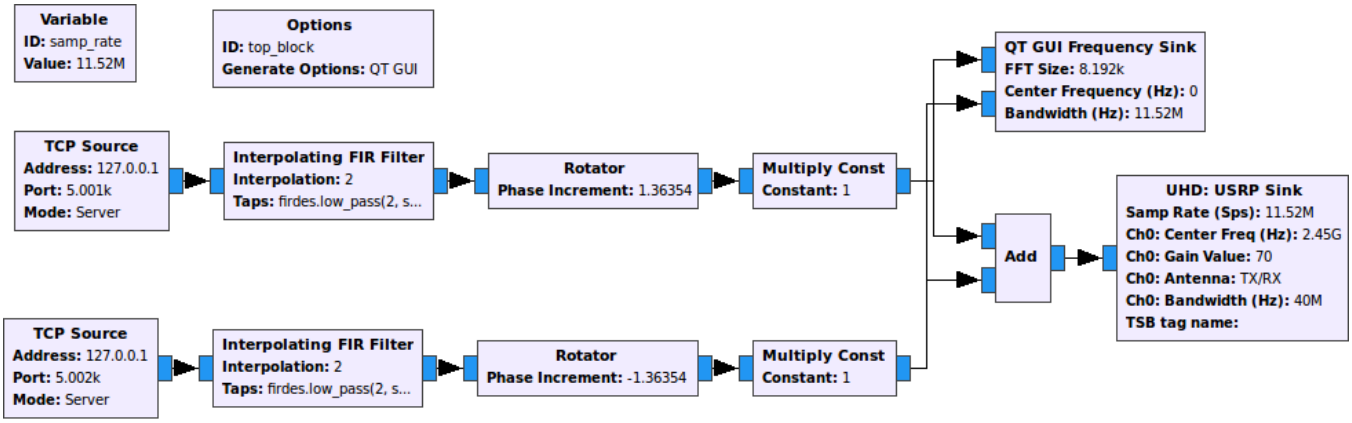


Figure 2: Experimental setup to validate our IQ switch design.

DUC (Digital Up Converter) composed of an upsampling filter and a frequency shifting module.

In order to comply with the Nyquist theorem and achieve efficient computation, we set the sampling rate of the final signal as follows. First, we calculate the least common multiplier from the baseband sampling rate of the different VeNBs (e.g., 30 if three VeNBs have sampling rate 3 MHz, 5 MHz and 10 MHz). Then, we take increasing multiples of this multiplier until the result is higher than the sum of the VeNBs' sampling rate (in the example, the total is $3+5+10=18$, so we would set the final signal's sampling rate to 30 MHz). Finally, it is worth pointing out that the IQ switch/demux is implemented using GNU Radio [9] (see Figure 2).

In summary, the workflow is as follows. Each VeNB runs srsLTE in a Linux VM and output the IQ samples over a TCP socket. From there, the samples arrive at the GNU radio IQ switch where their TCP/IP headers are stripped in the TCP Source block. After that, upsampling and frequency shifting are done in the Interpolating FIR (Finite Impulse Response) Filter and Rotator blocks, respectively, and the signals from all the VeNBs are merged in the Sum block. Next, the signals are sent to the SRH via the USRP Sink block which communicates with the UHD driver and, eventually, with the SRH (a USRP B210 radio in our case) over USB3. Note that due to space constraints we do not show a diagram for the demux (i.e., for receiving IQ samples from the SRH going to the eNBs).

3 VALIDATION AND EVALUATION

In this section we first validate the design approach taken in the design of our IQ switch, and then we provide a thorough performance evaluation of our virtualized multi-VeNB platform.

3.1 IQ Switch Validation

Our first set of experiments is aimed at validating our IQ switch implementation. The experimental setup, illustrated in Figure 2, consists of two sources of IQ samples emulating two VeNBs using a common USRP radio front-end. Each VeNB is configured with 5 MHz of channel width. The parameter of our FIR is decided according to the specific properties of the LTE signals to handle. In case of 5 MHz VeNBs, each IQ flow is comprised of 300 subcarriers with

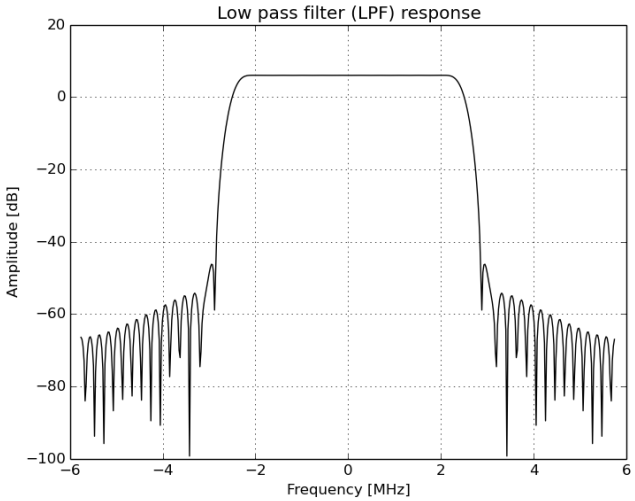


Figure 3: Frequency response of our FIR design.

subcarrier spacing 15 kHz. That means that the effective bandwidth occupied by these subcarriers is 4.5 MHz. In other words, LTE already provides a guard band which allows us to use a more relaxed FIR design. Based on this information, we design a FIR with cutoff frequency equal to 5 MHz and transition width equal to 1 MHz. This causes about 50 dB attenuation between adjacent 4.5 MHz effective LTE bandwidth by using 55 coefficients. The frequency response of the FIR is shown in Figure 3.

In turn, the actual spectrum of our two 5 MHz VeNBs before being sent to the USRP radio front-end is shown in Figure 4. According to the figure, the interference level to adjacent channel is about 60 dB lower than the signal in that channel. This is a very good isolation at the transmitter side, since it causes negligible signal to interference and noise ratio (SINR) degradation. Note that we validate this in our next experiments, where RF hardware non-ideal effects are also involved.

Table 1: SNR threshold to achieve 1% BLER.

LTE MCS index	0	5	10	15	20	25	28
Without adjacent channel interference	4.4 dB	8 dB	9 dB	13.5 dB	17 dB	22 dB	28.6 dB
With adjacent channel interference	8 dB	10 dB	11.3 dB	13.5 dB	17.8 dB	24.7 dB	29.5 dB

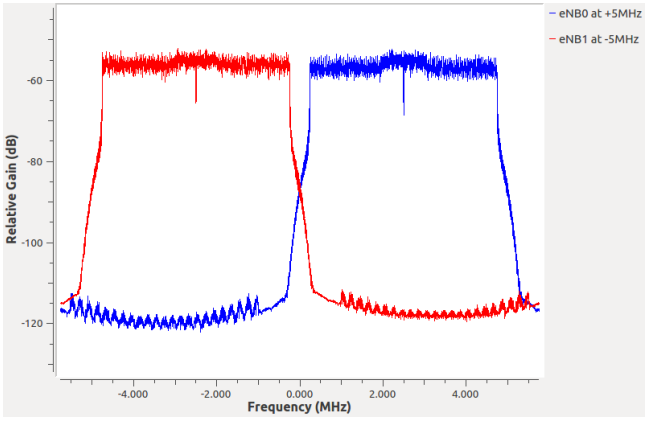


Figure 4: Performance of two IQ generators multiplexed by our IQ switch design.

Finally, we validate the performance of our IQ switch for different LTE modulation and coding schemes (MCSs). Specifically, Table 1 reports experimental data taken in our validation setup that shows the SNR threshold required to achieve 1% Block Error Rate (BLER)—a threshold that indicates a noticeable performance drop. We perform the same experiment for two cases: (*i*) with a single TX chain, i.e. no adjacent channel interference (only additive white Gaussian noise), and (*ii*) both TX chains, i.e. with AWGN noise and adjacent channel interference. The results show that the additional interference caused by the secondary LTE channel is minimal, validating in this way the design approach taken.

3.2 End-to-end Performance Evaluation

In the sequel, we are interested in (*i*) assessing whether current, off-the-shelf x86 hardware is able to concurrently host multiple (software-based) base stations with high throughput, and (*ii*) whether virtualization, which is a requirement to keep isolation among VeNBs, results in significant overhead. We carry out all our experiments on a pair of servers with an Intel Xeon E5-1620 v2 3.7 GHz CPU (4 cores) and 16GB of RAM (Linux 4.4.1, QEMU 2.1.2) connected over USB3 to a USRP B210 acting as a shared radio head

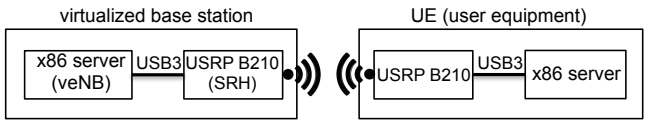


Figure 5: Experimental setup showing the virtualized base station and UE (user equipment) each consisting of an x86 server connected to a USRP B210 via a USB3 interface.

(see Figure 5). To guarantee more deterministic results, we disable hyper-threading, turbo boost, and all power saving features. Further, we limit the amount of cores that an eNB can use to one. That is, for baremetal (no virtualization), we pin the eNB process to a single core and, for each VeNB, we pin the entire QEMU process to a core, including the main QEMU thread, the QEMU I/O threads and the VM's vCPU thread. In terms of wireless channel bandwidth for the eNBs, we consider 5 and 10 MHz, a common configuration in femto and small-cell deployments [18], and use the unlicensed 2.4 GHz band as frequency carrier. In addition, we ensure that our experiments are properly isolated from external interfering transmitters using the same band, e.g. WiFi networks. Finally, we use `iperf` to generate UDP traffic.

3.2.1 Single eNB/VeNB Throughput. We begin the evaluation by measuring Tx/downlink throughput (i.e., from the base station to the UE) when running a single eNB, labelled as baremetal or "BM"), and then assess the overhead from virtualization when using a single VeNB. In both cases, we use a wide range of Modulation and Coding Schemes (MCSs), and carry out experiments for the 5 MHz and 10 MHz bands as previously mentioned. In addition, we downclock the CPU's frequency to determine at which value the base station can no longer match the theoretical maximum throughput.

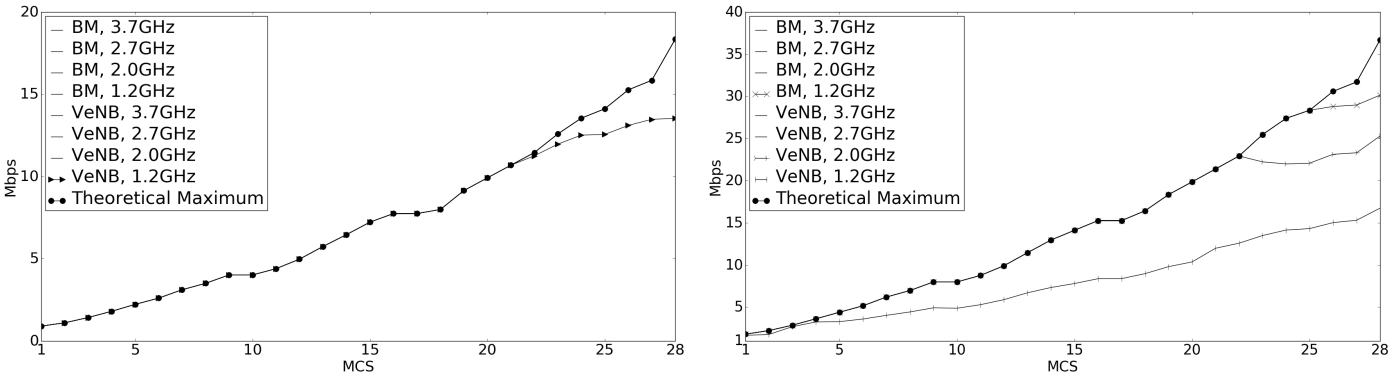
The results are plotted in Figure 6 (for convenience, we plot a line depicting the theoretical maximum throughput, that is, the transport block size (TBS), for each MCS). In the 5 MHz case (Figure 6), all setups, both virtualized (VeNB) and non-virtualized, i.e. baremetal (BM), are able to yield the theoretical maximum throughput for all MCSs (up to a maximum of 18 Mb/s) even when the CPU frequency is scaled down. The only exception is for the VeNB when runnion on a CPU at 1.2 GHz, which experiences a slight drop for MCSs higher than 22.

The 10 MHz case, shown in Figure 6, shows that most setups can still reach the theoretical max of up to 37 Mb/s, except in the case where the CPU is running at 1.2 GHz for both the eNB and the VeNB, and at 2 GH for the VeNB.

Finally, a remarkable observation is the fact that virtualization (VeNB) does not have a noticeable overhead over its baremetal counterpart in both cases (5 and 10 MHz).

3.2.2 Single eNB/VeNB CPU Utilization. Next, we use the `top` tool to evaluate CPU utilization when running a single eNB and a single VeNB. Figure 7a shows the CPU usage of both baremetal and VeNB cases, operating at 5 MHz over different CPU frequencies. Importantly, this result justifies the fact that a CPU frequency of 1.2 GHz could not reach the maximum theoretical throughput in the previous experiment: the CPU is maxed out.[1]

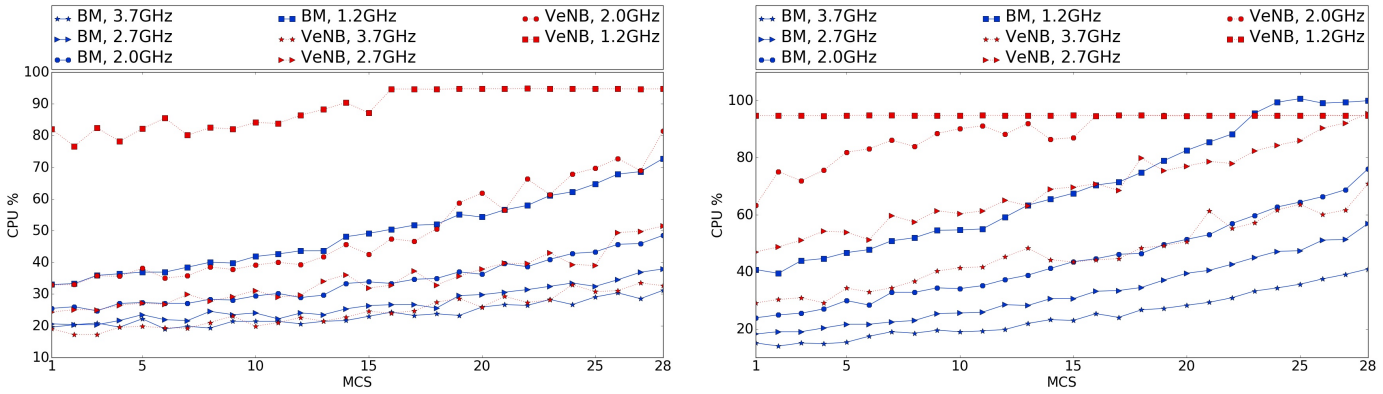
[1]Note that the whole QEMU process (QEMU threads plus vCPU thread) is pinned to a single core; this explains that the vCPU never gets 100% of the core share, as opposed to baremental.



(a) 5 MHz bandwidth

(b) 10 MHzbandwidth

Figure 6: Throughput for a single baremetal eNB and for a single virtualized eNB for the 5 MHz and 10 MHzbands, different CPU frequencies and different MCSs.



(a) 5 MHz bandwidth

(b) 10 MHz bandwidth

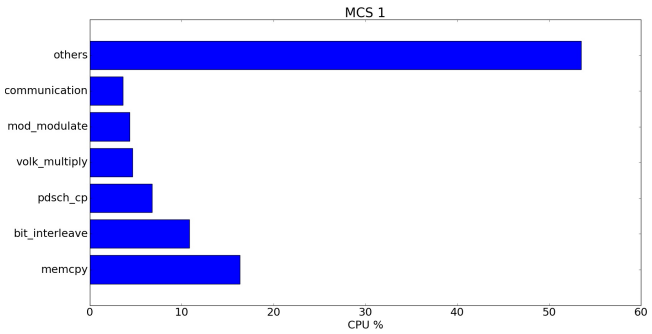
Figure 7: CPU utilization when running a single eNB and a single VeNB on the 5 MHz and 10 MHz bands for different CPU frequencies and MCSs.

With 10 MHz bandwidth (Figure 7b), the experimental results provide the same explanation for the throughput drop shown in Figure 6: a CPU frequency of 1.2 GHz is overly low for the eNB (and subsequently for VeNBs too) to achieve the theoretical maximum since the CPU is fully utilized (as is the case in the VeNB at 2GHz and higher MCS values). Still, for common CPU frequencies, we are more than able to host a single VeNB instance. In Section 3.2.4, we evaluate multiple concurrent VeNBs. Prior to this, we show next an evaluation of the decoding process, typically a more costly procedure.

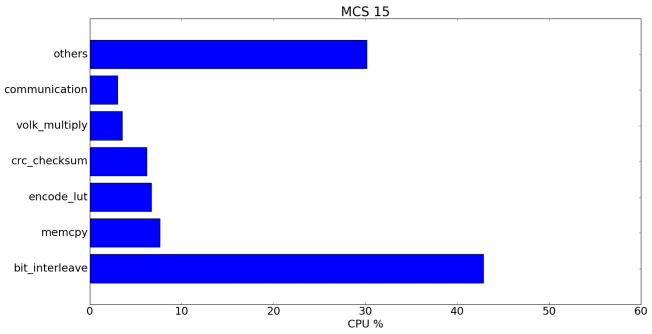
In the following, we assess the CPU consumption of independent components within the eNB software (srsLTE). Our results are summarized in Figure 8 for different MCSs. In the figures, we represent with bars the most representative consumers (functions) of CPU and aggregate the remaining in a meta function labeled as "others". In particular, it is worth highlighting how bit interleaving gains weight as the modulation level grows, consuming up to 60% of the overall CPU usage with the largest MCS compared to a (roughly) 12% consumption with MCS equal to 1.

3.2.3 PHY Receiving. So far we have focused on the eNB downlink scenario (i.e., signal transmission). However, since receiving is one of the most computationally expensive operations of an LTE stack of an eNB (i.e. uplink), we also need to prove that it is feasible for our commodity server to perform this operation, both for the baremetal eNB and the VeNB.

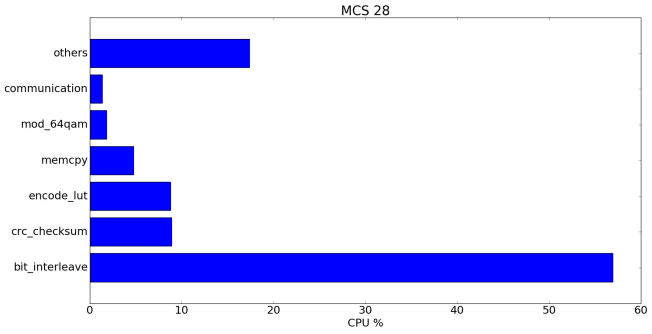
Evaluating the PHY receiving cost by setting up UE to eNB link is non-trivial since it implies the use of L2 and above protocols



(a) MCS 1



(b) MCS 15



(c) MCS 28

Figure 8: CPU profiling of individual components of srsLTE when using 5 MHz.

(MAC, PDCP, etc.) and requires the eNB to provide UL grants to the UE on a separate channel (also a non-trivial process which requires scheduling decisions). This effectively means that we would not only be measuring the PHY receiving capabilities of the eNB but also other signaling and protocol overhead.

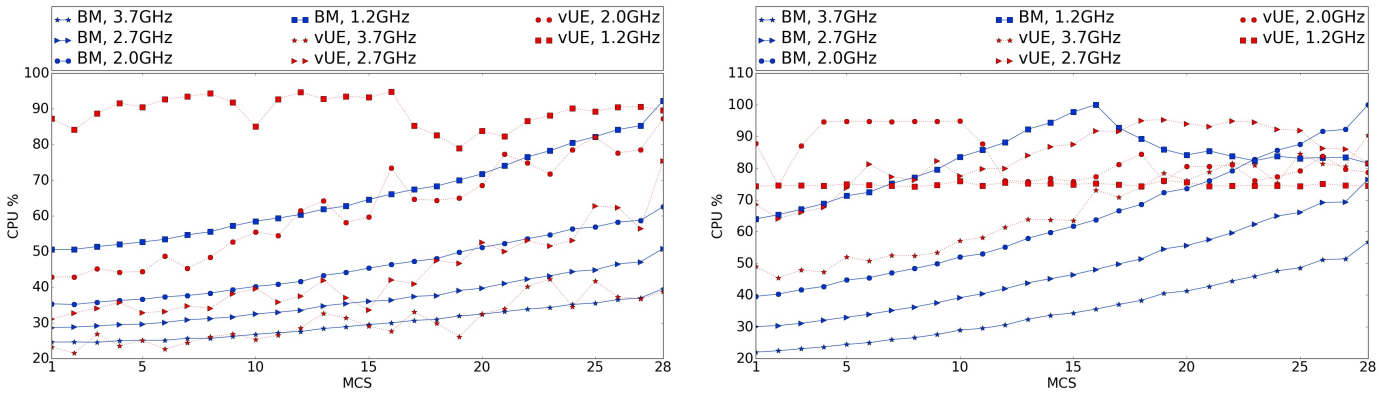
Since our goal in this section is simply to evaluate the computational expense of PHY receiving, we resort to evaluating the PHY receiving capabilities of the UE. This procedure is on the same order of complexity than the process of eNB receiving—in fact, it is roughly the same process with the exception of one less FFT computation—thus allowing to assess the viability of eNB receiving LTE signal on software. Figure 9 depicts our experimental evaluation on both 5 MHz and 10 MHz channels. The case of 5 MHz is similar to the transmission experiment performed earlier with CPU starvation issues when the CPU runs at 1.2 GHz.

In the 10 MHz case, the graph shows that we can correctly process signals at 10 MHz for all CPU frequencies and MCSs when using baremetal (eNB). The exception is at 1.2 GHz, which shows a spike when the MCS index is 15: at this point the CPU is out of cycles; subsequently any higher MCS shows lower CPU utilization because the system drops samples and thus it does not consume cycles for decoding. In the virtualized case, we see a substantial difference in CPU consumption with respect to baremetal simply because, as stated before, the QEMU threads are scheduled on the same core. That is, under load, the vCPU utilization will not reach 100% of the core usage because the remaining threads also need to be executed on the same core (and under load also require CPU time to execute their tasks). Aside from that phenomenon, the behaviour is similar to baremetal decoding: there is a spike in usage from which CPU utilization drops because samples are dropped and not decoded. As opposed to baremetal, the values shows that virtualized PHY receiving is only viable at 3.7 GHz.

3.2.4 Multiple VeNBs. Finally, we measure the CPU utilization and network throughput performance for the whole system: including the IQ switch and concurrent virtualized base stations (vENBs) at both 5 and 10 MHz channel widths. We evaluate up to 4 concurrent veNBs, to match the 4 CPU cores we have in our testbed. The represented IQ switch values correspond to a baseline computational effort to merge N signals at a given sampling rate and is evaluated independently of the concurrent VeNBs.

The results in Figure 10 evidence the feasibility of running multiple eNBs on the same physical infrastructure. Note that, on our 4-core machine we can run up to 4×5 MHz eNBs as well as 2×10 MHz without exhausting our CPU resources. In all, this shows the feasibility of running multiple, virtualized base stations over shared, inexpensive commodity hardware.

We now evaluate the memory requirements of the VeNB software in comparison to its guest OS (Debian) for different number of VeNBs. The experiment consists of a downlink scenario with 1 to 3 VeNBs with different MCSs and bandwidth configuration. A first conclusion raised out of our experiment is that memory consumption is practically independent of the MCS and bandwidth configuration. Due to this, we represent in Figure 11 the memory utilization of a scenario with 5 MHz and MCS equal to 28. It is shown that memory usage grows linearly with the number of virtualized eNBs. This is explained by the fact that each srsLTE instance is independent (running on its own VM) and therefore, no libraries are shared between instances. In addition, the behavior of the Linux dynamic loader is to load all required libraries (e.g. VOLK, libboost) making the amount of memory required by srsLTE external libraries independent on the MCS/bandwidth.



(a) 5 MHz bandwidth

(b) 10 MHz bandwidth

Figure 9: CPU utilization for the PHY receiving process using a 10 MHz bandwidth with different MCSs and CPU frequencies.

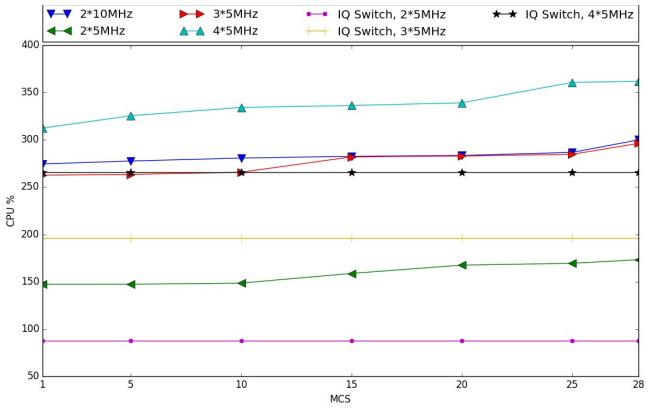


Figure 10: (multi-)CPU utilization for multiple VeNBs running concurrently and different IQ switch configurations.

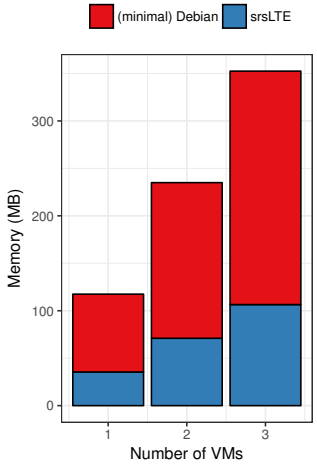


Figure 11: Memory consumption of multiple VeNBs.

4 RELATED WORK

Virtualization of resources in radio access networks is not new, although most of the work focuses on spectrum efficiency. FlexRadio, for instance, focuses on enabling sharing of RF resources through efficient allocation by unifying MIMO, full-duplex and interference alignment techniques [6]. SplitAP [3] "virtualizes" the network by providing air-time guarantees toto clients sharing an access point.

Closer to the topic of this paper, a recent survey [15] mentions the possibility of using a hypervisor to virtualize an LTE base station. The work in [24] also suggests using hypervisors to virtualize eNBs, but focuses instead on algorithms to schedule the air interface [24]. Perhaps the closest work to ours is Virtual Wifi [23], which, as the name suggests, uses KVM to virtualize a WiFi access point as opposed to an LTE base station. The work also only uses a single VM/virtualized access point and reports much higher delay overheads from virtualization (up to 35% more).

A number of research papers have looked into running wireless processing on different kinds of inexpensive hardware. For example, Atomix [1] introduces a modular framework for building applications on wireless infrastructure with high performance by leveraging multi-processor DSPs. Further, Sora [21] provides a high performance software radio using a custom radio control board and implements an 802.11a/b/g WiFi transceiver. Ziria [11] extends this work by presenting a novel programming model that makes it easier to program the Sora platform [11]

Finally, there are a number of modular, software frameworks for running wireless applications on commodity x86 hardware. Perhaps the most widely used one is GNU Radio, which can be used with external hardware to create software-defined radios or without it as simulation [9]. A number of other platforms target LTE, including OpenAirInterface [16], OpenLTE [17] and srsLTE [10]. In this work we settled on the latter since OpenLTE is incomplete and many

features are still under development and OpenAirInterface's code is complex and hard to split each LTE layer processing for rapid, early evaluation and prototyping. We further used GNU radio to implement our mux/demux.

5 CONCLUSION AND FUTURE WORK

In this work we introduced a working proof-of-concept system able to run concurrently multiple, virtualized LTE PHY stacks over shared, inexpensive commodity hardware with network throughput performance equal or close to the theoretical maximum. We have also shown that it is possible to use an IQ switch software module and a single shared radio head to multiplex the signals from the base stations. To the best of our knowledge there is not a large body of work on end-to-end base station virtualization sharing common infrastructure (including the radio front-end).

There are important points to work out as next steps. One important drawback is the fact that our testbed is not comprised yet of a fully functional virtualized LTE base station. As we explained in our paper, we evaluate a PHY-only eNB instead which is the most computationally expensive part, and its performance evaluation shows that the CPU is able to cope with multiple concurrent base stations. In addition, we do not yet know where the major performance bottlenecks in our system are, nor have we compared our system to other base station software such as OpenAirInterface. Note, moreover, that if the price, power consumption or physical size of our solution were too large, operators might be reluctant to deploy it. In future work we are looking at the possibility of instantiating virtual base station instances on the fly, when needed, in order to bring down power usage. We are also looking at using single-board computers (e.g., an Intel NUC) instead of the physically-large x86 server we used in this work. Regarding the IQ switch software module, as an improvement to frequency multiplexing, we would need to investigate time multiplexing and statistical multiplexing approaches in the context of LTE protocols to improve spectrum efficiency.

ACKNOWLEDGMENTS

The projects leading to this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 67156 (Flex5Gware) and No. 732174 (ORCA project).

REFERENCES

[1] Manu Bansal, Aaron Schulman, and Sachin Katti. 2015. Atomix: A Framework for Deploying Signal Processing Applications on Wireless Infrastructure. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. USENIX Association, Oakland, CA, 173–188. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/bansal

[2] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. 2003. Xen and the Art of Virtualization. *SIGOPS Oper. Syst. Rev.* 37, 5 (Oct. 2003), 164–177. https://doi.org/10.1145/1165389.945462

[3] Gautam Bhanage, Dipti Vete, and Ivan Seskar. 2010. SplitAP: Leveraging Wireless Network Virtualization for Flexible Sharing of WLANs. In *Global Telecommunications Conference*. IEEE. https://doi.org/10.1109/GLOCOM.2010.5684328

[4] Sourjya Bhaumik, Shoban Preeth Chandrabose, Manjunath Kashyap Jataprolu, Gautam Kumar, Anand Muralidhar, Paul Polakos, Vikram Srinivasan, and Thomas Woo. 2012. CloudIQ: A Framework for Processing Base Stations in a Data Center. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (Mobicom '12)*. ACM, New York, NY, USA, 125–136. https://doi.org/10.1145/2348543.2348561

[5] Naga Bhushan, Junyi Li, and Durga Malladi. 2014. Network densification: the dominant theme for wireless evolution into 5G. In *IEEE Communications Magazine*. IEEE. https://doi.org/10.1109/MCOM.2014.6736747

[6] Bo Chen, Vivek Yenamandra, and Kannan Srinivasan. 2015. FlexRadio: Fully Flexible Radios and Networks. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. USENIX Association, Oakland, CA, 205–218. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/chen

[7] Cisco. [n. d.]. 10th Annual Cisco Visual Networking Index (VNI) Mobile Forecast Projects 70 Percent of Global Population Will Be Mobile Users. https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1741352. ([n. d.]).

[8] Facebook. [n. d.]. Introducing OpenCellular: An open source wireless access platform. https://code.facebook.com/posts/1754757044806180/introducing-opencellular-an-open-source-wireless-access-platform. ([n. d.]).

[9] GNURadio. [n. d.]. GNURadio, the Free and Open Software Radio Ecosystem. http://gnuradio.org/. ([n. d.]).

[10] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D. Sutton, Pablo Serrano, Cristina Cano, and Doug J. Leith. 2016. srsLTE: An Open-source Platform for LTE Evolution and Experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization (WiNTECH '16)*. ACM, New York, NY, USA, 25–32. https://doi.org/10.1145/2980159.2980163

[11] Mahanth Gowda, Gordon Stewart, Geoffrey Mainland, Bozidar Radunović, Dimitrios Vytiniotis, and Doug Patterson. 2014. Poster: Ziria: Language for Rapid Prototyping of Wireless PHY. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking (MobiCom '14)*. ACM, New York, NY, USA, 359–362. https://doi.org/10.1145/2639108.2642893

[12] GSMA report. 2012. *Mobile Infrastructure Sharing*. Technical Report. http://www.gsma.com/publicpolicy/wp-content/uploads/2012/09/Mobile-Infrastructure-sharing.pdf

[13] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. 2007. KVM: the Linux Virtual Machine Monitor. In *In Proc. 2007 Ottawa Linux Symposium (OLS '07)*.

[14] K. Larsen. 2014. Network Sharing Fundamentals. https://techneconomyblog.com/2014/05/21/the-abc-of-network-sharingthe-fundamentals-part-i/. (May 2014). [Online; accessed 2017-02-27].

[15] Chengchao Liang and F. Richard Yu. 2015. Wireless Network Virtualization: A Survey, Some Research Issues and Challenges. In *IEEE Communications Surveys and Tutorials*. IEEE. https://doi.org/doi:10.1109/comst.2014.2352118

[16] Navid Nikaein, Mahesh K. Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Chri stian Bonnet. 2014. OpenAirInterface: A Flexible Platform for 5G Research. *SIGCOMM Comput. Commun. Rev.* 44, 5 (Oct. 2014), 33–38. https://doi.org/10.1145/2677046.2677053

[17] OpenLTE. [n. d.]. OpenLTE: An open source 3GPP LTE implementation. https://sourceforge.net/p/openlte/wiki/Home/. ([n. d.]).

[18] Jonathan Rodriguez. 2015. *Fundamentals of 5G mobile networks*. John Wiley & Sons.

[19] Peter Rost, Salvatore Talarico, and Matthew C. Valenti. 2015. The ComplexityâĂŞRate Tradeoff of Centralized Radio Access Networks. In *IEEE Transactions on Wireless Communications*. IEEE. https://doi.org/10.1109/TWC.2015.2449321

[20] S. Sun, M. Kadoch, L. Gong, and B. Rong. 2015. Integrating network function virtualization with SDR and SDN for 4G/5G networks. *IEEE Network* 29, 3 (May 2015), 54–59. https://doi.org/10.1109/MNET.2015.7113226

[21] Kun Tan, He Liu, Jiansong Zhang, Yongguang Zhang, Ji Fang, and Geoffrey M. Voelker. 2011. Sora: High-performance Software Radio Using General-purpose Multi-core Processors. *Commun. ACM* 54, 1 (Jan. 2011), 99–107. https://doi.org/10.1145/1866739.1866760

[22] The 5G Infrastructure Public Private Partnership. [n. d.]. KPIs. https://5g-ppp.eu/kpis. ([n. d.]).

[23] Lei Xia, Sanjay Kumar, Xue Yang, Praveen Gopalakrishnan, York Liu, Sebastian Schoenberg, and Xingang Guo. 2011. Virtual WiFi: Bring Virtualization from Wired to Wireless. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '11)*. ACM, New York, NY, USA, 181–192. https://doi.org/10.1145/1952682.1952706

[24] Carmelita Goerg Yasir Zaki, Liang Zhao and Andreas Timm-Giel. 2010. LTE Wireless Virtualization and Spectrum Management. In *IFIP WMNC*.

[25] Chun Yeow Yeoh, Mohammad Harris Mokhtar, and Abdul Aziz Abdul Rahman. 2016. Performance study of LTE experimental testbed using OpenAirInterface. In *International Conference on Advanced Communication Technology (ICACT)*. IEEE. https://doi.org/10.1109/ICACT.2016.7423494