Mendes, J., Jiao, X., García-Saavedra, A., Huici, F. y Moerman, I. (2019). Cellular access multi-tenancy through small-cell virtualization and common RF front-end sharing. *Computer Communications,* 133, January, pp. 59-66.

DOI: 10.1016/j.comcom.2018.10.010

Computer Communications 133 (2019) 59–66

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Cellular access multi-tenancy through small-cell virtualization and common RF front-end sharing

Jose Mendes [a,*], XianJun Jiao [b], Andres Garcia-Saavedra [a], Felipe Huici [a], Ingrid Moerman [b]

[a] NEC Laboratories Europe, Germany
[b] Ghent University - imec, IDLab, Belgium

ABSTRACT

Mobile traffic demand is expected to grow as much as eight-fold in the coming next five years, putting strain in current wireless infrastructures. Meanwhile the diversity of traffic and standards may explode as well. 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 many small cells and handling timing critical traffic at the edge. In this paper we take a step in that direction by implementing a virtualized small cell 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. In contrast to C-RAN (Cloud/Centralized Radio Access Network), our virtualized small cell base station has full stack at the edge so that a low latency high throughput front-haul, which is necessary in C-RAN architecture, is not needed. This approach brings all the flexibility and configurability (from network management point of view) that a software based implementation provides while the transparent architecture enables the possibility of multiple standards sharing the same radio infrastructure.

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 [1]. 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 [2].

Such towering numbers will put significant strain on existing mobile infrastructure. *Network densification* [3] is a well-recognized mean 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 [4].

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) [5].

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 [6,7]) 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 (SDR) [8,9] are gaining momentum as a solution to future dense deployments [10]; a remarkable example is Facebook's OpenCellular project [11].

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 [12–14]. The sharing of Radio Head is achieved by manipulating IQ samples which implies transparent multitenancy. So, it would be possible for the LTE base station to be multiplexed over different technologies/standards. This manuscript is an extended

* Corresponding author.
E-mail addresses: jmml.mendes@gmail.com (J. Mendes), xianjun.jiao@ugent.be (X. Jiao).

https://doi.org/10.1016/j.comcom.2018.10.010
Received 12 January 2018; Received in revised form 11 July 2018; Accepted 25 October 2018
Available online 3 November 2018

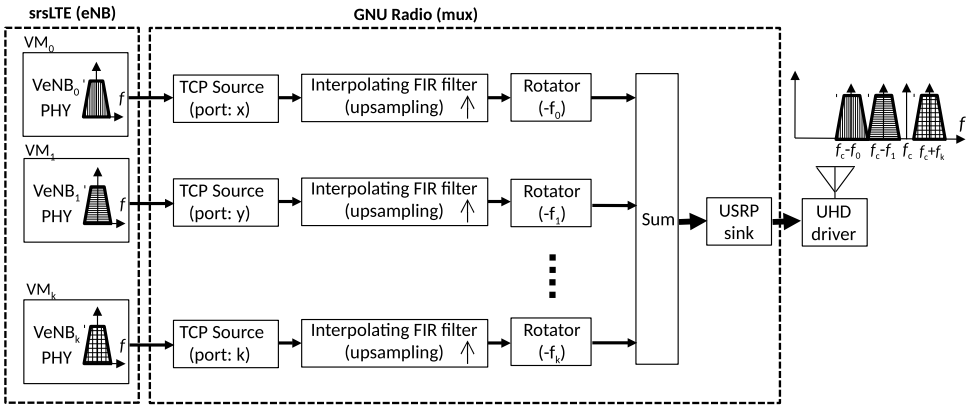


Fig. 1. Virtualized base station overall architecture.

Table 1
Sampling rate and bandwidth of different systems.

	Baseband sampling rate	Channel bandwidth
Wi-Fi	20/40/80/160 Msps	20/40/80/ 160 MHz
Bluetooth Low Energy (4.0)	1 Msps	2 MHz
GSM	270.883 Ksps	200 KHz
IS-95	1.2288 Msps	1.25 MHz
CDMA2000	3*1.2288 Msps	5 MHz
WCDMA	3.84 Msps	5 MHz
LTE standard	1.92/3.84/7.68/15.36/23.04/30.72 Msps	1.4/3/5/10/15/ 20 MHz
srsLTE	1.92/3.84/5.76/11.52/15.36/23.04 Msps	1.4/3/5/10/15/ 20 MHz

version of our preliminary work [15]. 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.
- The multiplexing/de-multiplexing of IQ samples, which traditionally is done in hardware (FPGA/ASIC), can be done in software and fulfill the performance requirements.

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 certain setups (virtual eNBs with 5 MHz bandwidth), 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 Fig. 1. Our system consists of three modules: eNB, mux, and radio front-end.

The first module, represented in the left-most part of Fig. 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 [8], 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. To this aim, we employ an USRP B210, commonly used for software defined radio [8]. This board provides 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 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 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 [16] (see Fig. 2).

The main challenges of IQ switch implementation are high computational intensity and flexible control of each signal path. High computational intensity comes from the fact that merged signal has the highest sampling rate in the whole system, which means IQ switch module needs to process multiple times number of samples compared with each signal path. Flexible control is required in real world deployment considering each operator/base station may need different bandwidth and center frequency according to government or operator's frequency plan. In the domain of digital signal processing, a wide design space has been explored to converge to our current IQ switch solution. Polyphase and FFT based channelizer is explored at first, but they only achieve a small aggregated bandwidth (also seen in [17]). The reason is that polyphase and FFT based channelizer's high computation efficiency relies on special conditions: baseband sampling rate and channel spacing/bandwidth are equal or has integer times relationship which is true for system like Wi-Fi and Bluetooth Low Energy but not true for most of mobile communication system (LTE in our case). Table 1 shows

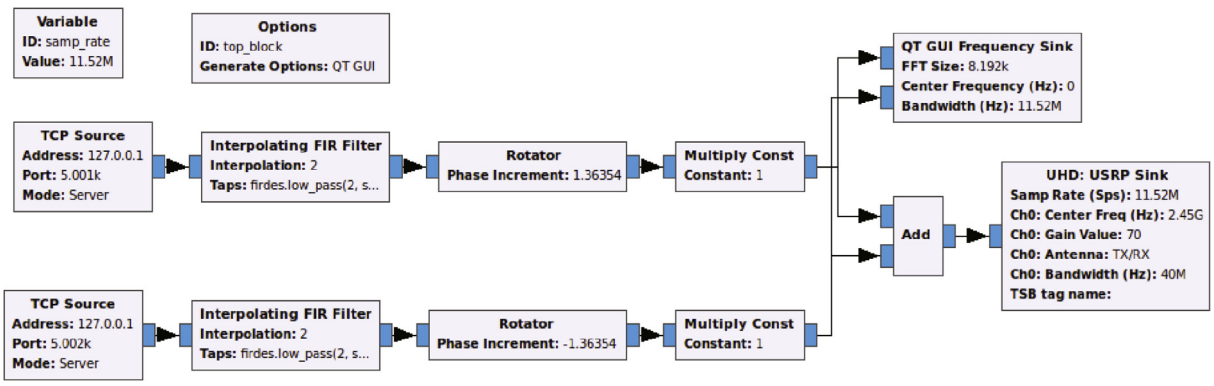


Fig. 2. Experimental setup to validate our IQ switch design.

Table 2
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

this phenomenon. Complicated resampler (in time domain or frequency domain involving different size of FFT) has to be used together with fractional relationship between sampling rate and channel bandwidth, and this extra complication finally results in only a small aggregated bandwidth according to our profiling. Polyphase Channelizer also has limitation on central frequency of each channel, which leads to less flexibility regarding fine tuning central frequency of each base station. After exploring many design options, such as FFT Filter and Polyphase Channelizer in GNU Radio, our final design (see Fig. 2), accelerated by VOLK (Vector Optimized Library of Kernels) [18], achieves good computational efficiency (much bigger aggregated bandwidth compared to [17]) and flexible control of each signal path: both bandwidth (via coefficients of Interpolating FIR Filter) and central frequency (via phase increment of Rotator).

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).

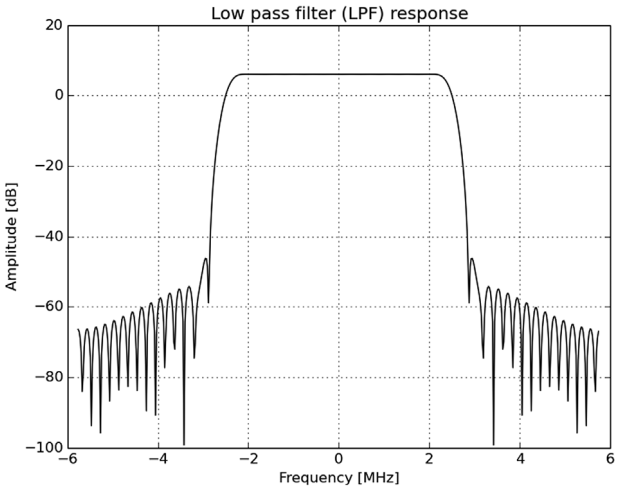


Fig. 3. Frequency response of our FIR design.

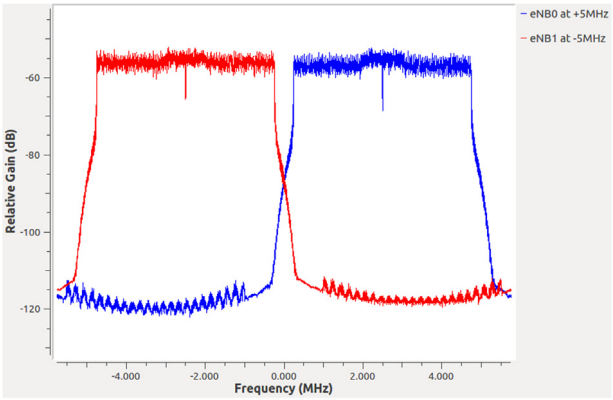


Fig. 4. Performance of two IQ generators multiplexed by our IQ switch design.

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 Fig. 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 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 Fig. 3.

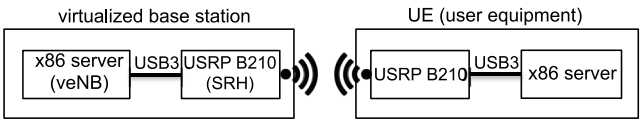


Fig. 5. Experimental setup showing the virtualized base station and UE (user equipment) each consisting of an x86 server connected to a USRP B210 via a USB3 interface.

In turn, the actual spectrum of our two 5 MHz VeNBs before being sent to the USRP radio front-end is shown in Fig. 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.

Finally, we validate the performance of our IQ switch for different LTE modulation and coding schemes (MCSs). Specifically, Table 2 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. Although the adjacent channel interference is guaranteed small by filter design, the colorized interference (higher at one edge of channel than the other edge of channel) could bring out extra effects out of white noise, being also necessary to evaluate sensitivity degradation under this situation. The results, meanwhile, show that the sensitivity degradation is minimal. It is important to note that this evaluation is carried out on end-to-end test via real USRP RF hardware, 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 (see Fig. 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 (i.e., non-virtualized), 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 [19], 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) by connecting the USRPs directly using SMA cables. To guarantee further accuracy of the results, each represented value is obtained by doing an average over 25 individual runs. That is, besides multiple measurements for the same value (e.g. throughput at 5 MHz, MCS 28), between sequential measures the software components being evaluated are restarted.

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, labeled 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

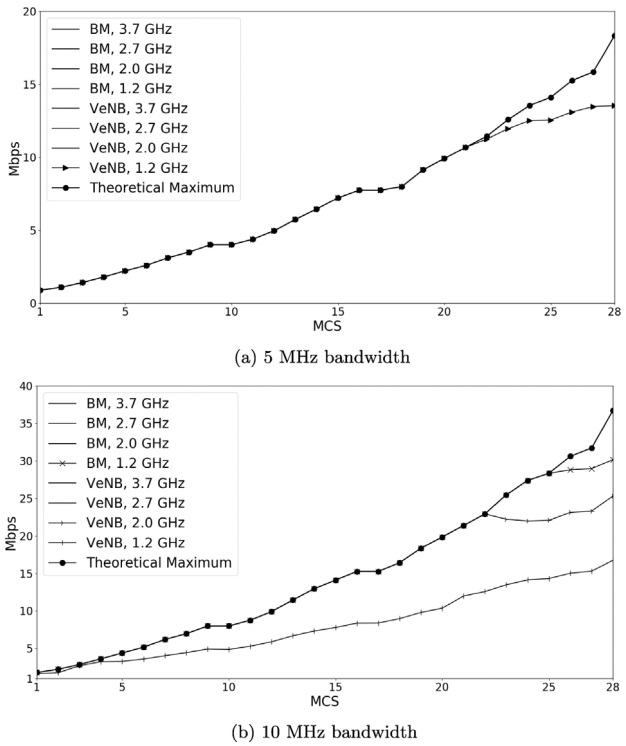


Fig. 6. Throughput for a single baremetal eNB and for a single virtualized eNB for the 5 MHz and 10 MHz channels, different CPU frequencies and different MCSs.

the 5 MHz and 10 MHz channels 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 Fig. 6 (for convenience, we plot a line depicting the theoretical maximum throughput for each MCS). In the 5 MHz case (Fig. 6), all setups, both virtualized (VeNB) and non-virtualized (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.

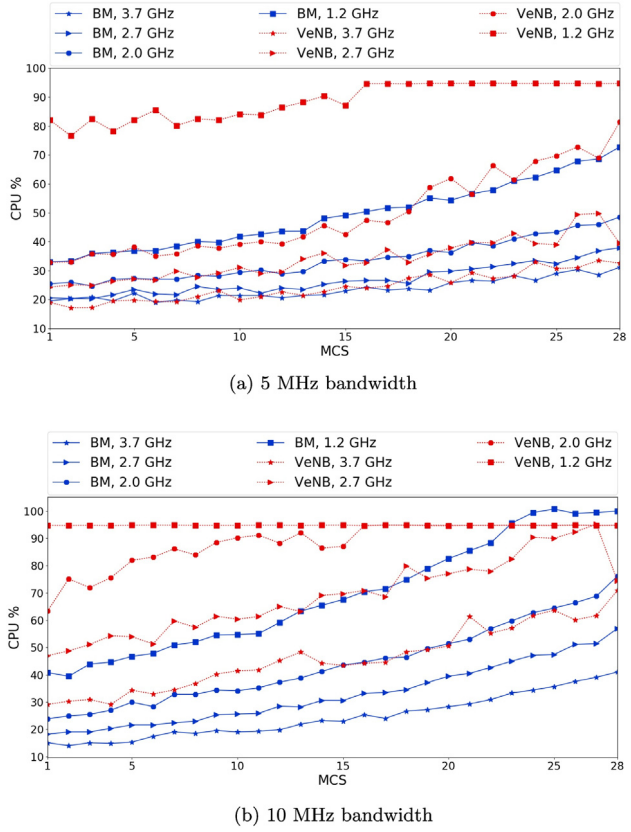
The 10 MHz case, shown in Fig. 6, shows that most setups can still reach the theoretical max of up to 37 Mb/s, except in the case where the CPU is running at 1.2 GHz for both the eNB and the VeNB, and at 2 GH for the VeNB.

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. 7 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.

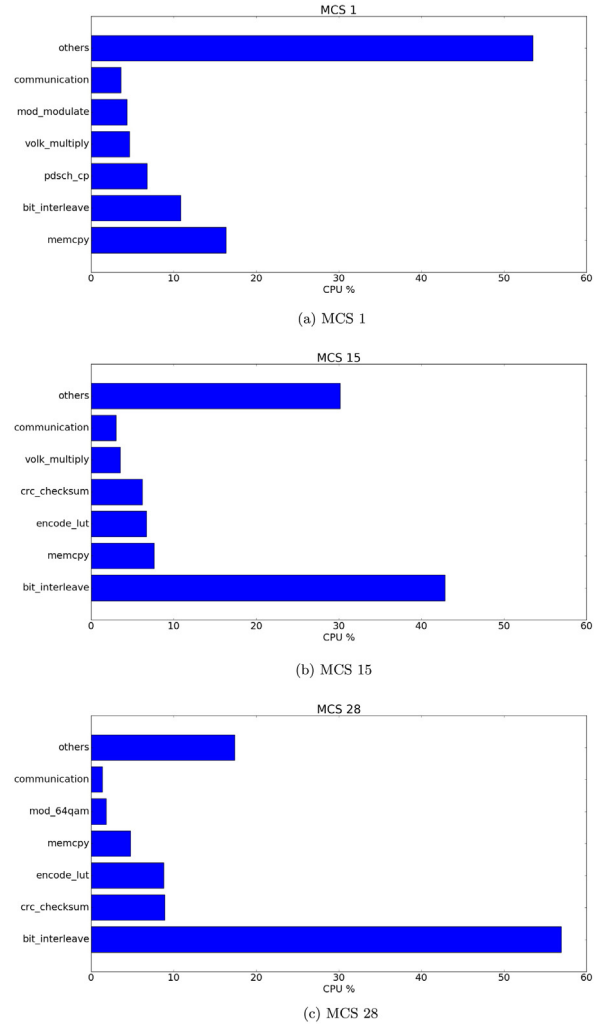
With 10 MHz bandwidth (7), the experimental results provide the same explanation for the throughput drop shown in Fig. 6: a CPU frequency of 1.2 GHz is overly low for the eNB (and subsequently for VeNBs too) to achieve the theoretical maximum since the CPU is fully utilized (as is the case in the VeNB at 2 GHz 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.



(a) 5 MHz bandwidth

(b) 10 MHz bandwidth

Fig. 7. Single-core CPU utilization when running a single eNB and a single VeNB on the 5 MHz and 10 MHz channels for different CPU frequencies and MCSs.



(a) MCS 1

(b) MCS 15

(c) MCS 28

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

Worth noticing, on both figures, the VeNB saturates the CPU at visibly less than 100% while in baremetal, the utilization nears 100%. This is due to the fact that the represented VeNB lines reflect CPU utilization of the QEMU vCPU thread which shares the core with the remaining QEMU threads. Since those also need CPU time, the VeNB vCPU utilization cannot ever reach 100% of the physical CPU usage. Measuring only the vCPU thread has the intention of showing the actual CPU utilization of the VeNB instead of evaluating QEMU and its additional threads.

In the following, we assess the CPU consumption of independent components within the eNB software (srsLTE). Our results are summarized in Fig. 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.

Another important point to observe is the cost of communication, that is, the time spent transferring samples between a VeNB and the IQ switch and thus representing an overhead of using virtualization instead of a baremetal deployment. It is observable that the cost of communication even at MCS 1 (where the relative amount of time spent on communication is higher) does not exceed around 3.6% and, at MCS 28 where the CPU is under heavier usage, the relative cost drops to around 1.4%. This indicates that communication (as an overhead of virtualization) is not a bottleneck and, in fact, is a minor contributor to CPU load increase when compared with other operations.

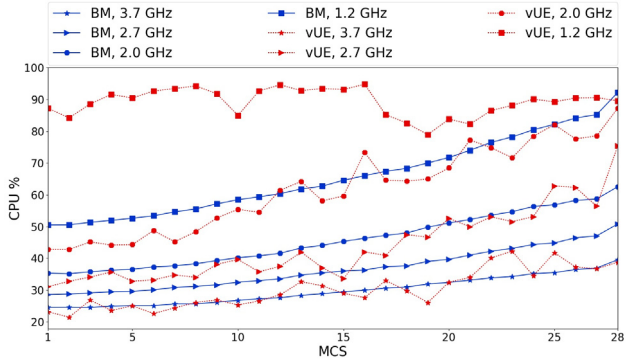
3.2.3. PHY receiving

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.

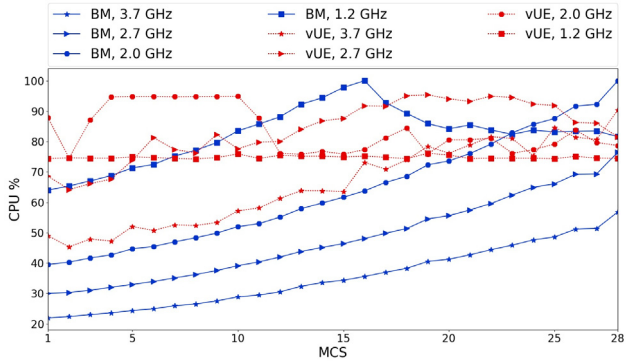
Evaluating the PHY receiving cost by setting up UE to eNB link is non-trivial since it implies the use of L2 and above protocols (MAC, PDCP, etc.) and requires the eNB to provide UL grants to the UE on a separate channel (also a non-trivial process which requires scheduling decisions). This effectively 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. Fig. 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



(a) 5 MHz bandwidth



(b) 10 MHz bandwidth

Fig. 9. Single-core CPU utilization for the PHY receiving process using a 10 MHz bandwidth with different MCSs and CPU frequencies.

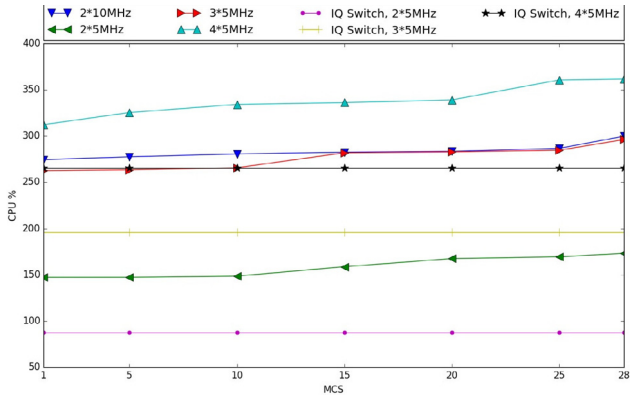


Fig. 10. Multi-core CPU utilization for multiple VeNBs running concurrently and different IQ switch configurations.

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 behavior 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.

Another thing to note is that the lines for the virtualized case at 1.2 and 2.0 GHz are only included for completeness: with the CPU completely starved, there are so many samples dropped that the lines

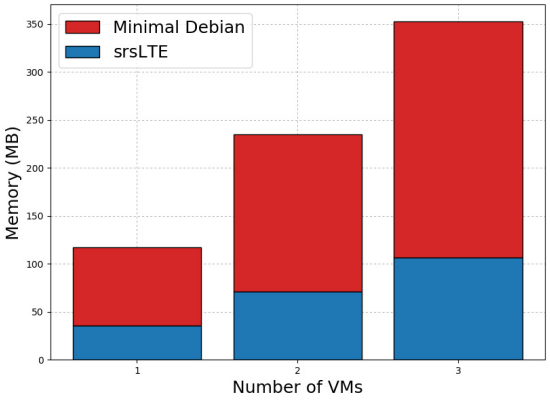


Fig. 11. Memory consumption of multiple VeNBs.

are not representative of a genuine decoding effort, rather additional computations.

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 Fig. 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 configurations. A first conclusion drawn out of our experiment is that memory consumption is practically independent of the MCS and bandwidth configuration. Due to this, we represent in Fig. 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.

4. Use case: Operator infrastructure sharing

The previous evaluation section proves the concept of the architecture successfully. Based on the successful evaluation, we foresee an obvious use case: Operator/technology independent infrastructure.

For that, and considering the results of our evaluation, a generalized infrastructure architecture is proposed in Fig. 12. The architecture is designed to be operator and technology agnostic. It can also be used for one operator to host multiple virtual operators (MVNOs). The benefit of technology independency is derived from the fact that multi-stream data manipulation is performed at IQ sample level, which is independent from specific standard/technology. In this architecture, the IQ sample multiplexing/de-multiplexing is called IQ switch, because it will connect multiple Shared Radio Heads (SRH) and virtual machines (VM). Likewise, multiple radio heads are connected to the server, and are shared among virtual machines via IQ switch.

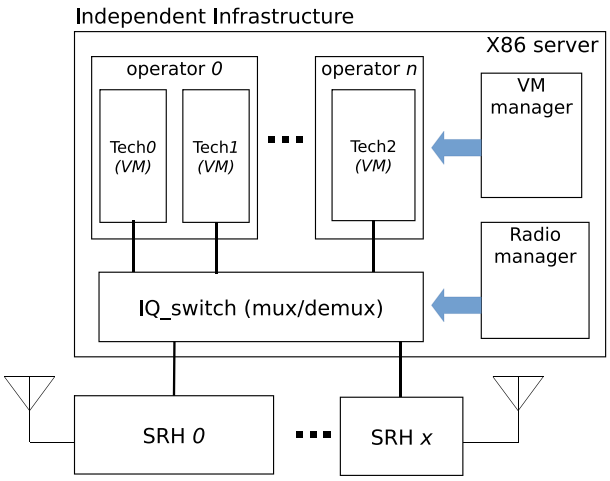


Fig. 12. Generalized architecture adopting infrastructure and spectrum management.

To fit the dynamic/diverse services/operators requirements of this scenario, two management modules are needed: virtual machine manager and radio manager. Through the virtual machine manager, different operators could deploy their own base station software via virtual machine images. Different base stations could use the same protocol stacks or different technologies, allowing for example, the coexistence of LTE with UMTS-only basestations (or any other technology). The radio manager configures the IQ switch according to each virtual machine's requirement: bandwidth, central frequency, number of antennas, latency, etc. Performing in a similar manner to a network switch, the IQ switch is also supposed to forward IQ samples among different ports dynamically under the control of radio manager, which could fit into the Software Defined Network (SDN) concept. Cooperating with virtual machine manager, a highly dynamically and configurable infrastructure would be made possible.

With this type of infrastructure deployed and managed (which could be done by an independent third party rather than one of the operators using the infrastructure), operators can deploy or retract their own base station (using any desired protocol stack) easily. More importantly, the configuration of the operator's network can be adjusted dynamically according to the operator's plan or customer situation changing (i.e. adjusting resources available dynamically). All the operations are performed in software domain – virtual machine image deploying, but operator still has control of full stack – from physical layer to network layer. This approach gives operator a full base station (from the functionality point of view) as if there was a real physical base station available but without the need to operate and maintain its own hardware and dedicated site.

5. 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 [20]. SplitAP [21] "virtualizes" the network by providing air-time guarantees to clients sharing an access point.

C-RAN is a cloud based centralized RAN architecture which is composed by a BBU (Baseband Unit) pool located at a datacenter and multiple RRUs (Remote Radio Head) located in coverage area (cell). Virtualization is natural in the cloud, however, a very low latency and high throughput front-haul is needed between BBU and RRU to meet critical service/technology requirements. C-RAN is usually deployed by a single operator to get the benefits of one virtual "super base station": inter-cell coordination; dynamic resource (computation power, spectrum) allocation among multiple cells. On the contrary, our virtualized small cell architecture does not require a high quality front-haul because the Radio Head is bundled together with digital unit of base station in the small cell area. Also different from one "super base station" for a single operator, our small cell architecture allows multiple operators to deploy their own full stack base station at the edge in the same shared infrastructure.

Closer to the topic of this paper, a recent survey [22] mentions the possibility of using a hypervisor to virtualize an LTE base station. The work in [23] also suggests using hypervisors to virtualize eNBs, but focuses instead on algorithms to schedule the air interface [23]. Perhaps the closest work to ours is Virtual Wifi [24], 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 [25] introduces a modular framework for building applications on wireless infrastructure with high performance by leveraging multi-processor DSPs. Further, Sora [26] provides a high performance software radio using a custom radio control board and implements an 802.11a/b/g WiFi transceiver. Ziria [27] extends this work by presenting a novel programming model that makes it easier to program the Sora platform [27].

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 [16]. A number of other platforms target LTE, including OpenAirInterface [9], OpenLTE [28] and srsLTE [8]. 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.

6. 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 in our future work. 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 full-fledged 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 to improve spectrum efficiency.

To extend its performance supporting more radio heads and base stations for generalized infrastructure, multi-threading parallel processing architecture needs to be investigated to utilize modern many-core CPU platform.

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), no. 732174 (ORCA project) and no. 761536 (5G-Transformer).

References

[1] Cisco. 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.
[2] The 5G Infrastructure Public Private Partnership. KPIs. https://5g-ppp.eu/kpis.
[3] N. Bhushan, J. Li, D. Malladi, Network densification: the dominant theme for wireless evolution into 5G, in: IEEE Communications Magazine, IEEE, 2014.
[4] K. Larsen, Network Sharing Fundamentals. https://techneconomyblog.com/2014/05/21/the-abc-of-network-sharingthe-fundamentals-part-i/, May 2014. [Online; accessed 2017-02-27].
[5] GSMA report. Mobile Infrastructure Sharing. Tech. rep., September 2012.
[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, SIGOPS Oper. Syst. Rev. 37 (5) (2003) 164–177.
[7] A. Kivity, Y. Kamay, D. Laor, U. Lublin, A. Liguori, KVM: the linux virtual machine monitor, in: In Proc. 2007 Ottawa Linux Symposium, in: OLS '07, 2007.
[8] I. Gomez-Miguelez, A. Garcia-Saavedra, P.D. Sutton, P. Serrano, C. Cano, D.J. Leith, 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, in: WiNTECH '16, ACM, New York, NY, USA, 2016, pp. 25–32.
[9] N. Nikaein, M.K. Marina, S. Manickam, A. Dawson, R. Knopp, C.s. Bonnet, Openairinterface: A flexible platform for 5G research, SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 33–38.
[10] S. Sun, M. Kadoch, L. Gong, B. Rong, Integrating network function virtualization with sdr and sdn for 4g/5g networks, IEEE Network 29 (3) (2015) 54–59.
[11] Facebook. Introducing opencellular: An open source wireless access platform. https://code.facebook.com/posts/1754757044806180/introducing-opencellular-an-open-source-wireless-access-platform.
[12] C.Y. Yeoh, M.H. Mokhtar, A.A.A. Rahman, Performance study of lte experimental testbed using openairinterface, in: International Conference on Advanced Communication Technology (ICACT), IEEE, 2016.
[13] P. Rost, S. Talarico, M.C. Valenti, The complexityrate tradeoff of centralized radio access networks, in: IEEE Transactions on Wireless Communications, IEEE, 2015.
[14] S. Bhaumik, S.P. Chandrabose, M.K. Jataprolu, G. Kumar, A. Muralidhar, P. Polakos, V. Srinivasan, T. Woo, Cloudiq: A framework for processing base stations in a data center, in: Proceedings of the 18th Annual International Conference on Mobile Computing and Networking, in: Mobicom '12, ACM, New York, NY, USA, 2012, pp. 125–136.
[15] J. Mendes, X. Jiao, A. Garcia-Saavedra, F. Huici, I. Moerman, Cellular access multitenancy through small cell virtualization and common rf front-end sharing, in: Proceedings of the 11th Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization, in: WiNTECH '17, ACM, New York, NY, USA, 2017, pp. 35–42.
[16] GNURadio. GNURadio, the Free and Open Software Radio Ecosystem. http://gnuradio.org/.
[17] M. Kist, J. Rochol, L.A. DaSilva, C.B. Both, Hydra: A hypervisor for software defined radios to enable radio virtualization in mobile networks, in: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017, pp. 960–961.
[18] VOLK. VOLK: Vector-Optimized Library of Kernels. http://libvolk.org/.
[19] J. Rodriguez, Fundamentals of 5G Mobile Networks, John Wiley & Sons, 2015.
[20] B. Chen, V. Yenamandra, K. Srinivasan, Flexradio: Fully flexible radios and networks, in: 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15), USENIX Association, Oakland, CA, 2015, pp. 205–218.
[21] G. Bhanage, D. Vete, I. Seskar, Splitap: Leveraging wireless network virtualization for flexible sharing of wlans, in: Global Telecommunications Conference, IEEE, 2010.
[22] C. Liang, F.R. Yu, Wireless network virtualization: A survey, some research issues and challenges, in: IEEE Communications Surveys and Tutorials, IEEE, 2015.
[23] Yasir Zaki, C.G. Liang Zhao, A. Timm-Giel, Lte wireless virtualization and spectrum management, in: IFIP WMNC, 2010.
[24] L. Xia, S. Kumar, X. Yang, P. Gopalakrishnan, Y. Liu, S. Schoenberg, X. Guo, Virtual wifi: Bring virtualization from wired to wireless, in: Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, in: VEE '11, ACM, New York, NY, USA, 2011, pp. 181–192.
[25] M. Bansal, A. Schulman, S. Katti, 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, 2015, pp. 173–188.
[26] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, G.M. Voelker, Sora: High-performance software radio using general-purpose multi-core processors, Commun. ACM 54 (1) (2011) 99–107.
[27] M. Gowda, G. Stewart, G. Mainland, B. Radunović, D. Vytiniotis, D. Patterson, Poster: Ziria: Language for rapid prototyping of wireless phy, in: Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, in: MobiCom '14, ACM, New York, NY, USA, 2014, pp. 359–362.
[28] OpenLTE. OpenLTE: An open source 3GPP LTE implementation. https://sourceforge.net/p/openlte/wiki/Home/.