SDR Virtualization in Future Mobile Networks: Enabling Multi-Programmable Air-Interfaces

Maicon Kist*, Juergen Rochol*, Luiz A. DaSilva‡, Cristiano Bonato Both†
*Federal University of Rio Grande do Sul, Brazil, ‡Trinity College Dublin, Ireland,
†Federal University of Health Sciences of Porto Alegre, Brazil
{mkist, juergen}@inf.ufrgs.br*, dasilval@tcd.ie‡, cbboth@ufcspa.edu.br†

***Abstract*—The fifth generation of mobile networks is envisioned to provide connectivity services to a multitude of devices with vastly different requirements. Current mobile systems rely on inflexible hardware-based RF front-end that provide a "one-size-fits-all" air-interface. Instead, future mobile networks should be flexible, providing different air-interfaces for particular users and applications. In this paper, we present HyDRA, a software-defined-radio virtualization layer that enables the execution of multiple programmable air-interfaces on top of one RF front-end. Our solution multiplexes digitized IQ signal samples of multiple virtual radios into a single stream. We have implemented HyDRA and experimentally evaluate its performance in a scenario that considers a base station executing LTE and NB-IoT VRs. Results obtained show that HyDRA is able to efficiently multiplex these two technologies, while the computational analysis shows that HyDRA is not CPU-intensive and can run in standard, commodity computers. We also show that HyDRA is a promising framework to enable RRH slicing, multi-radio access networks, and flexible multi-tenant networks.**

I. Introduction

The fifth generation (5G) of mobile networks is envisioned to provide connectivity services to a multitude of devices with vastly different requirements, ranging from mobile subscribers with high-bandwidth services (*e.g.,* video-streaming) to IoT devices with bursty and low-bandwidth services (*e.g.,* utilities metering, and assisted living monitoring). To efficiently support services with such distinct requirements, providing a single "one-size-fits-all" air-interface is not desirable. Instead, future mobile networks should be flexible, providing different air-interfaces for particular users and applications [1] [2] [3]. For example, the 3GPP is considering base stations capable of offering connectivity to both LTE mobile subscribers and Narrow-Band Internet-of-Things (NB-IoT) devices. We argue that radio virtualization is a promising solution to achieve such flexibility, enabling multiple Virtual Radios (VRs) to coexist on top of one base station [4] [5] [6].

With radio virtualization, multiple VRs can coexist and share the same RF front-end, while providing some isolation of computational and communication resources. The basic premise is the adoption of a physical radio device, such as a Software Defined Radio (SDR) front-end, and a virtualization manager, *i.e.,* a hypervisor. The hypervisor must abstract a physical radio device into VR devices and schedule the resources available in the physical radio between the VRs. Although radio virtualization is currently being considered in state-of-the-art architectures, an implementation is still lacking in the literature. Such implementation is essential to correctly analyze its benefits and performance impact, *e.g.,* latency and throughput of a VR due to hypervisor overheads, and isolation between VRs.

In this paper, we explore the design of a hypervisor for SDR platforms, named HYpervisor for software Defined RAdio (HyDRA), and demonstrate its architecture and interfaces. Our hypervisor receives raw IQ samples from VRs coexisting in the same RF front-end and uses software baseband processing to multiplex these samples into a single radio signal. HyDRA is implemented purely in software and is fully extensible to receive raw IQ samples from any air-interface. The current version of HyDRA abstracts an SDR RF front-end, essential to guarantee isolation and multiplex multiple VRs. We also show that HyDRA is an enabler for RRH slicing, multi-radio access networks, and flexible multi-tenant networks, while at the same reducing the network CAPEX.

The remainder of this article is organized as follows. We first present the state-of-the-art in radio virtualization. After, we discuss the requirements that a hypervisor for radio virtualization must satisfy. Next, we introduce the architecture of HyDRA and its application interface. In the sequence, we evaluate HyDRA in the context of a use-case aligned with the 3GPP vision for 5G networks. Finally, we close this article with concluding remarks.

II. Background and Motivation

The first radio virtualization framework can be traced back to the VR Framework proposed by Sachs and Baucke [7]. In this framework, an abstraction to a common wireless resource, such as a spectrum or Physical Radio Block (PRB) in LTE, can be allocated to different VRs. A Virtualization Manager is responsible for managing "which resource will be allocated to which mobile operator" and guaranteeing its isolation, *i.e.,* a VR has access only to the wireless resources allocated to it. Although this embryo contains the initial ideas of introducing virtualization into the mobile networks area, it lacks a practical implementation of the mechanisms that it proposes, such as the radio resource allocation and isolation.

Since the work of Sachs and Baucke, most approaches on wireless virtualization have focused on 3GPP LTE systems. Zaki *et al.* [8] introduce a hypervisor responsible for two tasks: (*i*) abstracting the physical eNodeB into a number of virtual eNodeBs, and (*ii*) scheduling PRBs to each of the virtual eNodeBs. The scheduling of PRBs is performed according to the

TABLE I
COMPARISON OF RADIO VIRTUALIZATION FRAMEWORKS

Authors	Virtual resource type	Isolation	Technology	Integration	Implementation and Evaluation	Contributions
Sachs and Baucke [7]	PRB	PRB	LTE	NA	NA	Introduced the concept of VR
Zaki *et al.* [8]	PRB	PRB	LTE	Requires modifications on the LTE Base Station	Software, simulation	Designed a hypervisor for eNodeB virtualization
Tan *et al.* [9]	Spectrum	Frequency	WiFi and Zigbee	VRs interface with function calls	Software/hardware, experimental	Hypervisor for spectrum virtualization
Kist *et al.* [this paper]	RF front-end	Frequency	Any	Emulates a RF front-end	Software, experimental	Hypervisor for SDR front-end virtualization

virtual mobile operator requirements, such as user data rates, channel conditions, traffic load, or a combination of them. The approach presented by Zaki can only be integrated in LTE-based networks, due to its virtualization being dependent on PRBs.

Tan *et al.* propose the Spectrum Virtualization Layer (SVL) framework [9], with focus on abstracting how radio devices access the spectrum in dynamic spectrum access networks. SVL abstracts the radio spectrum and presents it as virtual spectrum to traditional radios. When the spectrum allocation changes under the dynamic spectrum access regime, SVL performs real-time reshaping of baseband signals so that the baseband of traditional radios maps to the current physical spectrum band allocation, which can be of different width, or in non-contiguous bands.

Table I compares the aforementioned frameworks for radio virtualization considering six aspects:

- **Virtual resource type**: Refers to the resource type that is abstracted and used by VRs. The framework of Sachs and Baucke, and Zaki *et al.* allocate PRBs to virtual radios. Tan *et al.* allocate spectrum directly, which is arguably more flexible.
- **Isolation**: Refers to how VRs are isolated among themselves. The isolation in Sachs and Baucke, and Zaki is performed by ensuring that only one VR can access a particular PRB. Similarly, Tan *et al.* allocate each frequency to only one VR.
- **Technology**: Refers to the air-interface that VRs can implement. In the work of Sachs and Baucke, and Zaki, the VR must implement LTE (or LTE-based) air-interfaces due to their virtual resource type being PRB. In contrast, the framework of Tan *et al.* allows VRs to implement any air-interface.
- **Integration**: Refers to the modifications that current physical systems require in order to adopt the virtualization framework (note that this is different from virtualizing the air-interface). The frameworks of Sachs and Baucke, and Zaki require removing some of the physical layer functions from current eNodeB systems. The framework of Tan *et al.* requires the physical layer of VRs to generate digitized IQ samples representing the signal to be transmitted and then use the interface provided by the framework to transform these samples into radio signals.
- **Implementation and evaluation**: Refers to the implementation, *i.e.,* software or hardware, and evaluation methodology, *e.g.,* simulation or experimental, of the proposal. Zaki *et al.* implement their virtualization framework as a software module and perform simulations, whereas the framework of Tan *et al.* comprises both a proprietary software and hardware components and is evaluated in an experimental environment.
- **Contribution**: Sachs and Baucke introduce the concept of radio virtualization and provide a detailed framework showing the essential functions required to realize it. Zaki *et al.* design a virtualization framework for an LTE-based system that can be integrated in real LTE networks. Finally, Tan *et al.* design a framework for spectrum virtualization in dynamic spectrum access networks.

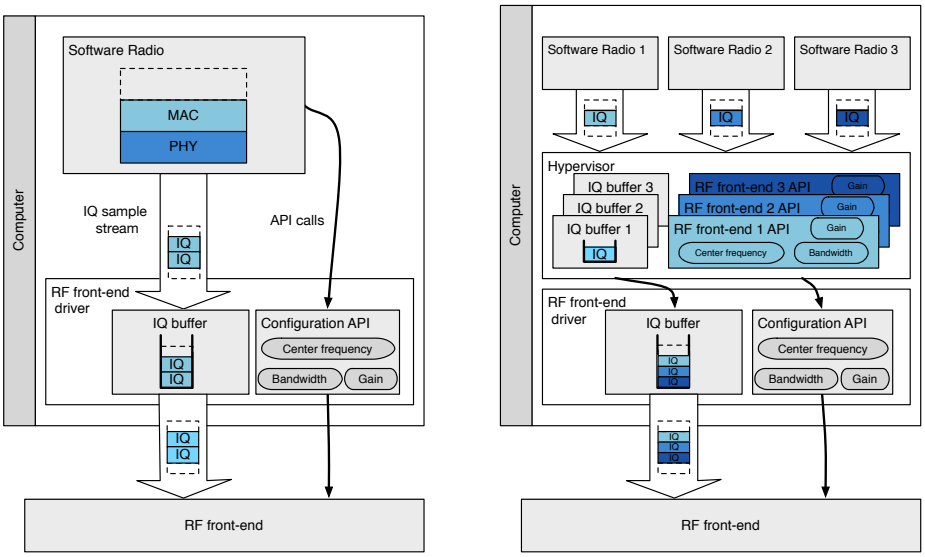
From the frameworks presented, we can note that radio virtualization can occur at different levels of abstraction. For example, Sachs and Baucke, and Zaki perform radio virtualization at the PRB level, *i.e.,* their framework abstracts, slices, isolates, and shares PRB from a physical eNodeB; in this case, VRs are software instances of an eNodeB that can access only the PRBs allocated to them. The framework of Tan *et al.* goes a step further, abstracting the radio spectrum through interfaces provided by a hypervisor-like software.

HyDRA is designed to perform radio virtualization of the RF-front end. This level of virtualization enables our solution to support air interfaces implementing any technology. HyDRA is developed as a software module and is evaluated experimentally; different from Tan *et al.*, HyDRA runs on any computer with GNURadio. It is publicly available online at GitHub (https://github.com/maiconkist/gr-hydra). The complete source code is accompanied with several examples with configurable parameters such as the number of VRs, air-interface, central frequency, and bandwidth of each radio, and GUI elements such as spectrum waterfall and plotters to see data transmitted and received. In the next section, we describe our vision of radio virtualization.

III. RADIO VIRTUALIZATION FOR SDR

Radio virtualization is "the process of abstracting a physical radio and slicing it into VRs holding certain corresponding functionalities and isolating each other" [10]. In other words, it is the process of abstracting, slicing, isolating, and sharing the radio hardware between multiple VRs that hold all or part of the functionalities of their physical counterpart.

Fig. 1(a) illustrates a conventional SDR platform. It comprises a software radio performing the baseband signal processing, the RF front-end driver, and the RF front-end. The software radio implements the signal processing operations related to transforming a sequence of bits, *e.g.,* user data,



(a) Standard SDR platform. Only one software radio can access the RF front-end.

(b) Virtualized SDR platform. In this case the RF front-end is abstracted, sliced, and shared among multiple software radios.

Fig. 1. Illustration of a standard SDR platform (a) and a virtualized SDR platform (b).

into digitized IQ samples that represent the radio signal that must be transmitted (external applications usually perform the data acquisition). The software radio can be executed on top of general purpose processors, taking advantage of highly-optimized signal processing libraries. The digital IQ samples generated are transferred to the RF front-end driver.

The RF front-end driver is usually implemented by the hardware vendor and provides an Application Programming Interface (API) that enables the software radio to recover and send digitized IQ samples from the RF front-end and to configure, among other parameters, the RF front-end center frequency, bandwidth, and gain. The RF front-end is responsible for translating the digitized IQ samples into/from radio signals that are transmitted/received by the antenna.

Based on the aforementioned characteristics, we can view virtualization of an SDR as the process of abstracting and slicing an RF front-end into a number of virtual RF front-ends. Virtual RF front-ends must have independent center frequency, bandwidth, and transmission/reception gain. Such abstraction requires the design and implementation of a SDR hypervisor, in line with other virtualization frameworks. Figure 1(b) illustrates where the hypervisor is located.

The main requirements for a SDR hypervisor are [10]:

- **Coexistence:** the hypervisor must ensure that multiple VRs can coexist on the same physical RF front-end. Moreover, since VRs can implement different air-interfaces, their requirements can be vastly different. The SDR hypervisor must support multiple simultaneous VRs, implementing air-interfaces with different processing constraints, bandwidths, channel access schemes, etc.
- **Isolation:** the hypervisor must ensure that any configuration or misconfiguration will not be able to affect and interfere with other coexisting VRs. Since several VRs can coexist on the same physical RF front-end, isolation is the fundamental requirement that guarantees fault tolerance, security, and privacy [10].
- **Programmability:** VRs should have a level of programmability similar to standard SDRs. This means that a virtual RF front-end must hold the same set of functionalities of its physical counterpart, *e.g.,* configuration of center frequency, bandwidth, and transmission/reception gain, while fulfilling the previous two requirements.

In the next section, we present HyDRA, a hypervisor for SDRs that complies with the requirements outlined above.

IV. HyDRA Architecture and Configuration API

The architecture of HyDRA is shown in Fig. 2. HyDRA abstracts the RF front-end by adding a layer of indirection between VRs. VRs operate as if they were interfacing directly with a standard SDR RF front-end by sending/receiving digitized IQ samples. HyDRA ensures isolation while allowing VRs to configure the central frequency, bandwidth, and sampling rate on-the-fly. HyDRA makes use of a spectrum map to keep track of these configurations.

The spectrum map is extremely flexible in HyDRA. For example, VRs can request any central frequency or bandwidth, as long as bands of operation of VRs do not overlap. Usually, these configurations are set during the bootstrap of HyDRA, but our architecture allows for computational inexpensive on-the-fly changes, as they only trigger an update in the IQ mapping.

The core and challenging part in designing HyDRA was to multiplex the incoming raw IQ samples of each VR into a single signal that is transmitted by the RF front-end. Our multiplexing is based on Fast Fourier Transform (FFT)/Inverse FFT (IFFT) operations. First, the incoming digitized IQ samples are transformed from time to frequency domain in an FFT with N_i points (N_i is a function of the bandwidth of the VR i and the

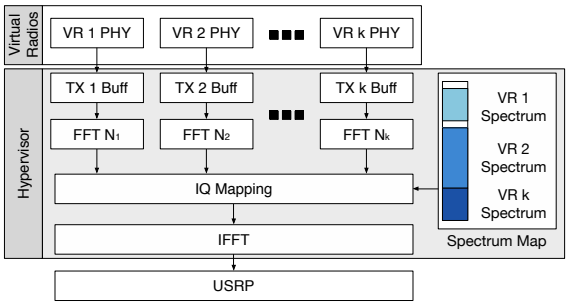


Fig. 2. Main architectural blocks of HyDRA.

sampling rate of the RF front-end). After that, the resulting N_i frequency components are mapped into the buffer of an IFFT of size M according to the spectrum map. After the mapping, we perform the IFFT to combine the frequency components of all VRs into the resulting multiplexed time domain signal, which can be transmitted by the RF front-end.

The multiplexing process based on FFT/IFFT operations is a perfect fit for our virtualization objectives. First, FFT/IFFT operations are low-level baseband processing operations agnostic of access technologies; this enables HyDRA to multiplex several VRs, each implementing a different access technology. Second, modifications in the original signal caused by the multiplexing are not distinguishable from well-known wireless disturbances, *e.g.,* path-loss, frequency shift, and phase distortion; this allows receiving devices to recover the data transmitted using conventional physical layer equalization mechanisms. Finally, the FFT/IFFT are computationally efficient operations, highly optimized for modern processors through Single Instruction Multiple Data (SIMD); this allows HyDRA to multiplex several VRs simultaneously while using only a fraction of the resources of modern processors.

A. HyDRA Configuration API

HyDRA defines a set of configuration APIs to manage VRs and to configure the virtual RF front-end associated to them. These APIs are summarized in Table II. The interface *create_virtual_radio* can be used to create a new slice on the physical RF front-end and abstract the slice onto a virtual RF front-end, which is accessed in the same way that a standard SDR RF front-end would be. The *bandwidth* parameter determines the spectrum bandwidth of the virtual RF front-end, and *center_frequency* specifies its center frequency.

We highlight that before creating the virtual RF front-end, HyDRA verifies whether the bandwidth and the center frequency requested are valid, checking that the requested channels do not overlap with another VR and whether the physical RF front-end supports the required configurations. If accepted, HyDRA returns an object representing a standard RF front-end, which is used by the VR.

The remaining APIs provide the functionalities that a VR expects from a standard SDR RF front-end. The *get_bandwidth*, *get_center_frequency* and *get_gain* interfaces communicate with the spectrum map and return the current bandwidth, center frequency or gain of the virtual RF front-end. Similarly, their *set* versions allow a VR to change its bandwidth, center frequency and gain, provided that the spectrum map accepts the requested configurations.

TABLE II
MAIN HYDRA APIS

Return type	API name	Parameters
rf_front_end*	create_virtual_radio	(int bandwidth, float center_frequency)
void	set_bandwidth	(int bandwidth)
int	get_bandwidth	()
void	set_center_frequency	(float center_frequency)
float	get_center_frequency	()
void	set_gain	(float decibels)
float	get_gain	()
void	send_samples	(IQ_samples *buffer, int size)
int	recv_samples	(IQ_samples *buffer, int buff_size)

VRs call *send_samples* to output a buffer of IQ samples to HyDRA. The parameters *buffer* and *size* specify the pointer and the number of digital samples to send, respectively. Similarly, VRs use *recv_samples* to receive all samples since the last call to this function. HyDRA fills *buffer* until all samples for the VR are transferred or until *buff_size* is reached.

V. MULTIPLEXING OF LTE AND NB-IOT

To evaluate the functionality of HyDRA, we focus on a use case whereby a 5G base station may be called upon to provide connectivity services to both LTE mobile subscribers and NB-IoT devices, as illustrated in Fig. 3. The implementation of this use-case consists of (*i*) one Universal Software Radio Peripheral (USRP) acting as the base station, (*ii*) one USRP acting as the LTE mobile subscriber, and (*iii*) one USRP acting as the NB-IoT healthcare data receiver. At the base station side, a dedicated computer executes HyDRA and the VRs for the LTE transmitter and NB-IoT transmitter. Similarly, we have one dedicated computer and USRP for the LTE and NB-IoT receivers. Table III summarizes the main parameters for HyDRA and the VRs. To analyze HyDRA's ability to multiplex the LTE and NB-IoT air-interfaces, we selected two services aligned with the capabilities of each one: a constant high-bandwidth video streaming in the LTE VR, and a bursty low-bandwidth healthcare sensor data for the NB-IoT VR.

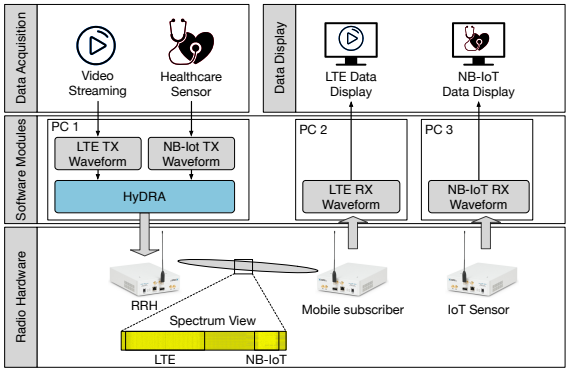


Fig. 3. Experimental scenario used to evaluate HyDRA.

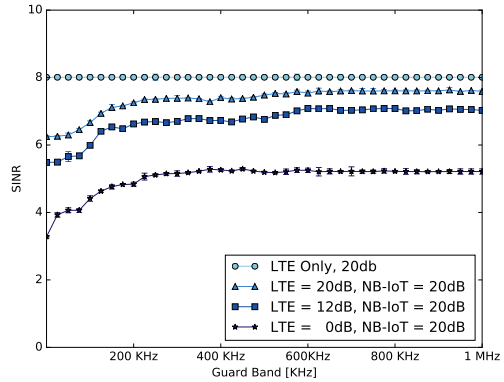
A. Virtual RF front-end isolation

In this analysis, we are interested in evaluating the isolation between the virtual RF front-ends created by HyDRA. As HyDRA multiplexes the virtual RF front-ends in the frequency

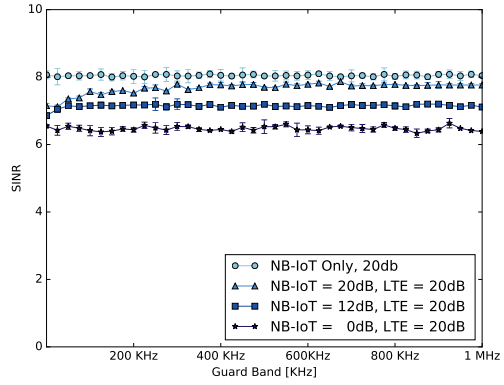
TABLE III
HyDRA, LTE, AND NB-IoT DEFAULT CONFIGURATION

Parameter	Value(s)
HyDRA	**CF**: 950 MHz **BW**: 8 MHz, **FFT**: 4096
LTE VR	**CF**: 947 GHz, **BW**: 1.4 MHz, **FFT**:128, **CP**: 7 symbols (short), **MOD**:QPSK
NB-IoT VR	**CF**: 951MHz, **BW**: 200 KHz, **FFT**:64, **CP**: 7 symbols (short), **MOD**:BPSK

domain, the best way to measure their isolation is considering different distances, *i.e.,* guard-bands, among them. More precisely, we measure the Signal-to-Interference-plus-Noise-Ratio (SINR) at the mobile subscriber and the NB-IoT receiver, for different guard-bands between the virtual RF front-ends being used by the LTE and NB-IoT VRs. In addition to the guard-band, we are also interested in analyzing the impact of gain differences in the multiplexing process, *i.e.,* when one of the RF front-ends utilizes a signal gain higher than the others.



(a) LTE



(b) NB-IoT

Fig. 4. SINR observed at the receiver as a function of the guard-band between the virtual RF front-ends.

The results obtained are shown in Fig. 4(a) for the LTE mobile subscriber and 4(b) for the NB-IoT receiver. In Fig. 4(a), the curve labeled *LTE Only* shows the SINR at the receiver when using a physical RF-front-end without virtualization. We compare against the case where HyDRA is used to simultaneously support LTE and NB-IoT VRs, in three different configurations; in each one we reduce the gain of the LTE virtual front-end, while maintaining the gain of the NB-IoT constant. First, we measured the SINR at the mobile subscriber when both virtual front-ends are set to 0 dB gain, then when the virtual RF front-end assigned to the NB-IoT transmitter uses a gain 6 dB higher than the LTE virtual front-end, and when the NB-IoT uses a gain 20 dB higher. In all curves, we can see that guard-bands larger than 200 KHz present a SINR degradation of only 1 dB. The same reasoning applies for Fig. 4(b), which shows the SINR at the NB-IoT receiver. In this case, the NB-IoT signal presented a minimal reduction in the SINR for all signal gains and guard bands. These results shows that the virtual RF front-ends are isolated, but with some limitations that arise from the physical RF front-end.

B. Virtual RF front-end multiplexing impact

HyDRA uses a sequence of FFT/IFFT operations to multiplex the signals of multiple virtual RF front-ends. The spectral leakage of FFT/IFFT can introduce offset errors in the amplitude or phase of the original signal. To quantify such errors, we evaluated the multiplexing operation of HyDRA considering a different number of bins for the IFFT, *i.e.,* the value of M. For each value of M, both LTE and NB-IoT signals are multiplexed by HyDRA, transmitted over a noisy channel and then immediately demultiplexed. Then, we calculate the Mean Square Error (MSE) between the original and demultiplexed signals.

Table IV shows the MSE values obtained. The row "standalone" presents the MSE when the IQ samples are transmitted over a noisy channel without being multiplexed by HyDRA. In this case, the MSE is due only to the noise. The remaining rows show the values of the MSE after the signals are multiplexed and transmitted by HyDRA.

TABLE IV
MSE OF THE MULTIPLEXING PROCESS CONSIDERING DIFFERENT IFFT SIZES

	LTE	NBIoT
Standalone	2.1 dB	1.1 dB
IFFT 1024	11.2 dB	7.9 dB
IFFT 2048	6.1 dB	4.1 dB
IFFT 4096	3.4 dB	2.2 dB
IFFT 8192	2.4 dB	1.1 dB

The MSE decreases as the number of IFFT bins increases. LTE presents a higher MSE due to its original signal being generated from a more complex OFDM configuration with higher number of carriers and constellation symbols. When HyDRA is used with a large enough number of IFFT bins (8192), the resulting MSE, for both the LTE and NB-IoT cases, is comparable to the MSE measured in the absence of HyDRA. These results show it is possible to eliminate the multiplexing error by using sufficiently large IFFTs.

C. HyDRA Computational Overhead

In this analysis, we consider the computational overhead introduced by HyDRA. We measure the CPU usage of HyDRA for different IFFT sizes and normalize it as a percentage

of one core in a Intel i5-6440HQ processor. The results are shown in Fig. 5. As we can see, the LTE and NB-IoT VRs require 19.68% and 4.71% of the CPU processing power, respectively, independently of the configuration used in HyDRA; this is expected, as the VR processing is independent. HyDRA requires an additional 6% of the CPU processing power for an IFFT of 512 bins, up to 15.30% for 8192 bins. This results indicate that HyDRA is not CPU intensive and can run in standard commodity computers.

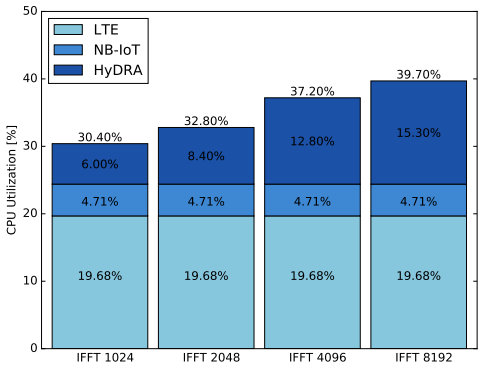


Fig. 5. CPU utilization.

D. Qualitative Benefits

In this subsection, we summarize the main qualitative benefits of the virtualization design adopted in HyDRA.

- **RRH slicing:** HyDRA slices the RRH by partitioning its physical RF front-end into multiple virtual RF front-ends. This is an effective form of RRH sharing, as virtual front-ends are isolated in the spectrum domain. RRH sharing is a major step towards future multi-tenancy networks, as being introduced by 3GPP [11].
- **Multi-radio access networks:** HyDRA simplifies the integration and operation of multi-radio access technologies in one device. Moreover, different from the current approach, where devices can only operate in technologies defined in its design phase, HyDRA enables devices to operate in any technology simple by creating a virtual RF front-end and loading a new VR software module.
- **Flexible multi-tenant access networks:** Current multi-tenant access networks are restricted by the underlying access technology, *i.e.,* all "tenants" share the same LTE base station. In contrast, HyDRA enables different mobile operators to run the technology of their choice on top of the virtual RF front-ends.
- **CAPEX reduction:** HyDRA can run on top of the infrastructure of cloud-radio access networks, while at the same time enabling multiple VRs to share the same RRH, *i.e.,* the RF front-end. As a consequence, mobile operators can deploy new access technologies simply by creating a new VR and multiplexing it through HyDRA.

VI. Concluding Remarks

We demonstrate the feasibility of radio virtualization as a mechanism to provide versatile connectivity services in the next generation of mobile networks. To this end, we designed HyDRA, a hypervisor capable of creating multiple virtual RF front-ends that are multiplexed into a single physical RF front-end. Virtual radios operate on top of each virtual front-end as if they were interfacing directly with a standard SDR RF front-end by sending/receiving digitized IQ samples. HyDRA ensures isolation while allowing each virtual RF front-end to be independently configured, *e.g.,* central frequency, bandwidth, and sampling rate.

We evaluated the virtual RF front-end isolation, multiplexing impact, and computational overhead added by HyDRA in a scenario that considers a base station executing LTE and NB-IoT VRs. The results obtained show that HyDRA can multiplex these two technologies with less than 1 dB degradation in the SINR when using a guard-band larger than 200 KHz and that the signal error introduced is inferior of that of standard noise. Moreover, the computational analysis shows that HyDRA is not CPU-intensive and can run in standard, commodity computers. As benefits, HyDRA enables RRH slicing by efficiently splitting the RF front-end into virtual ones; multi-radio access networks by giving devices the flexibility to select in real-time its air-interfaces; flexible multi-tenant access networks by allowing each tenant to run any access technology; and CAPEX reduction by allowing a single RRH to provide connectivity services to multiple technologies.

Acknowledgements

This work has received funding from the European Union's H2020 for research, development, and demonstration under grant agreement no. 645274 (WiSHFUL).

References

[1] I. F. Akyildiz, P. Wang, and S.-C. Lin, "SoftAir: A Software Defined Networking Architecture for 5G Wireless Systems," *Computer Networks*, vol. 85, no. 7, pp. 1–18, 2015.

[2] P. Rost, I. Berberana, A. Maeder *et al.*, "Benefits and challenges of virtualization in 5G radio access networks," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 75–82, 2015.

[3] C. Sexton *et al.*, "5G: Adaptable Networks Enabled by Versatile Radio Access Technologies," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 688–720, 2017.

[4] T. Taleb *et al.*, "EASE: EPC As a Service to Ease Mobile Core Network Deployment Over Cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, 2015.

[5] J. Liu *et al.*, "CONCERT: A Cloud-Based Architecture For Next-Generation Cellular Systems," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 14–22, 2014.

[6] S. Abdelwahab *et al.*, "Network function virtualization in 5G," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 84–91, 2016.

[7] J. Sachs and S. Baucke, "Virtual radio: a framework for configurable radio networks," *International Conference on Wireless Internet*, pp. 1–7, 2008.

[8] Y. Zaki *et al.*, "LTE wireless virtualization and spectrum management," in *Wireless and Mobile Networking Conference (WMNC)*, 2010, pp. 1–6.

[9] K. Tan, H. Shen, J. Zhang, and Y. Zhang, "Enable flexible spectrum access with spectrum virtualization," *IEEE International Symposium on Dynamic Spectrum Access Networks*, pp. 47–58, Oct 2012.

[10] C. Liang and R. Yu, "Wireless network virtualization: A survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 17, 2015.

[11] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, jul 2016.