Wireless Personal Communications
https://doi.org/10.1007/s11277-020-07245-x

Enabling Virtual Radio Functions on Software Defined Radio for Future Wireless Networks

Wei Liu[1] · Joao F. Santos[2] · Jonathan van de Belt[3] · Xianjun Jiao[1] · Ingrid Moerman[1] · Johann Marquez-Barja[5] · Luiz DaSilva[2] · Sofie Pollin[4]

Abstract
Today's wired networks have become highly flexible, thanks to the fact that an increasing number of functionalities are realized by software rather than dedicated hardware. This trend is still in its early stages for wireless networks, but it has the potential to improve the network's flexibility and resource utilization regarding both the abundant computational resources and the scarce radio spectrum resources. In this work we provide an overview of the enabling technologies for network reconfiguration, such as Network Function Virtualization, Software Defined Networking, and Software Defined Radio. We review frequently used terminology such as softwarization, virtualization, and orchestration, and how these concepts apply to wireless networks. We introduce the concept of Virtual Radio Function, and illustrate how softwarized/virtualized radio functions can be placed and initialized at runtime, allowing radio access technologies and spectrum allocation schemes to be formed dynamically. Finally we focus on embedded Software-Defined Radio as an end device, and illustrate how to realize the placement, initialization and configuration of virtual radio functions on such kind of devices.

Keywords Software-Defined Radio · Softwarization · Virtualization · Virtual Radio Function

✉ Wei Liu
wei.liu@ugent.be

[1] Ghent University - imec IDLab iGent Tower Department of Information Technology, Technologiepark-Zwijnaarde 126, 9052 Ghent, Belgium

[2] CONNECT - Trinity College Dublin, Dublin, Ireland

[3] Centre for Intelligent Power - Eaton, Dublin, Ireland

[4] Department of Electrical Engineering, University of Leuven, 3001 Leuven, Belgium

[5] Department of Electronics ICT - FTI, University of Antwerp - imec IDLab, Groenenborgerlaan 171, 2020 Antwerp, Belgium

Published online: 07 April 2020

1 Introduction

Ever since the advent of computer networks, engineers have continued to improve the network: not only is today's network able to sustain significantly higher throughput, it is also more flexible, in the sense that the network connection can be reprogrammed and controlled by software, the primary motivation for Software Defined Network (SDNs). At the same time, general purpose processors and commodity hardware have become more powerful: many network functionalities, e.g., firewalls, routers and load balancers, that in the past required dedicated hardware and firmware, can be now implemented entirely in software. The advantage is that the functionalities running on commodity hardware can be quickly deployed and scaled, and typically result in lower capital expenditures. This approach is also called Virtualization of Network Functionalities.

SDN and Network Function Virtualisation (NFV) technologies are the results of the softwarization and virtualization trends in wired networks. Their benefits include: (1) more diversity of technologies and services, (2) increased resource utilization efficiency, (3) easier management of infrastructure, and (4) on average faster development and deployment cycle.

In recent years, a similar trend is happening in wireless networks. For example, the Cloud-Radio Access Network (RAN) (C-RAN) paradigm is gaining traction amongst mobile operators. This paradigm proposes the decoupling between the radio hardware and the radio functionality, where traditional base stations become a combination of a Remote Radio Head (RRH) and a Baseband Unit (BBU). The BBUs run in the cloud and are connected to the RRHs through high speed links. This architecture leads to the possibility to have multiple distributed base stations managed in a centralized way, thus effectively decreasing the operation cost, among other benefits. C-RAN presents one approach for splitting the functions of Radio Access Technology (RATs) between dedicated hardware and software instances. A more general view of the functional split is given in Fig. 1. The functional split may happen anywhere in the communication stack. The more functionalities are realized by software, the deeper the softwarization. The choice of where to place the functional split is affected by many factors. It is typically subject to the trade-off between flexibility and reaction speed. When the communication stack is fully softwarized down to the physical layer, one has direct access to samples from the radio front end.

There are a variety of radio platforms that support complete programmability until physical layer over the radio functionality: these are often referred to as Software-defined

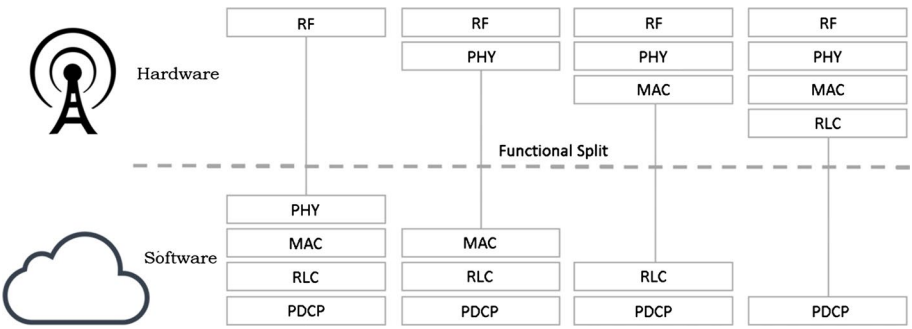

Fig. 1 Functional split of radio communication stack, adapted from [4]

Radio (SDRs). Their capabilities have been improving over the years, supporting increasingly wider RF bandwidth and higher processing power. For instance, about a decade ago the popular USRP reconfigurable radio series could only provide sampling rates up to 25 Msps, and all processing functionalities were implemented in the host computer. Today new generations of the USRP X series can support up to 160 Msps sampling rate, equipped with a 10 times more powerful Field-Programmable Gate Array (FPGAs). In the near future, an SDR with a mmWave front-end may take the bandwidth towards the order of several GHz [9], and the baseband processing will likely be supported by multiple FPGA. The abundance of RF bandwidth and computational power in a single SDR device calls for a new paradigm of usage, shifting the processing loads towards the end SDR devices, with the possibility of sharing the device by multiple RATs via virtualization.

Wireless networks can benefit from softwarization in the same way as wired networks, e.g., with increased flexibility and reduced operational cost. Besides, softwarization of wireless networks can also happen at the lower layers (PHY and MAC), rather than being restricted to upper network layers as is typical in wired networks. Due to this difference, the softwarization of radio communication has unique potential, enabling the creation of RATs according to real-time requirements, e.g., coverage, capacity, reliability, or latency. This extra level of flexibility of softwarization in wireless networks is empowered by the abundant computational resources, which can enable virtualization for optimized resource management. In addition to computational resources, a radio also requires spectrum resources, which are scarce and whose current use and management is not sufficiently flexible. The softwarization and virtualization in the wireless domain should therefore: (i) unleash the potential of constructing flexible PHY and MAC for radio communication in end devices, (ii) enable sharing and, hence, more efficient management of both the abundant computational and the scarce radio spectrum resources.

In this paper, we provide an overview of the relevant enablers of softwarization and virtualization in the wireless domain, introduce the concept *radio function virtualization* and exploit its usage on SDR devices. The remainder of this work is organized as follows: (i) we introduce the key definitions such as softwarization, virtualization, and how they are used in deriving radio functionalities; (ii) we review the state of the art on how these concepts are used in both wired and wireless networks today; (iii) we introduce the realization of softwarization and virtualization on embedded resources of an SDR; (iv) finally we make some concluding remarks.

2 Definitions

In this section, we propose a common understanding of softwarization, virtualization and orchestration, as these domain-specific terms are sometimes used in ambiguous or conflicting ways by experts with different backgrounds. Then, we introduce the concept of radio functions, and how their softwarization and/or virtualization combined with orchestration can benefit wireless networks.

2.1 Softwarization

The processing of communication functionality—be it on a server, switch, radio, etc.—can be implemented through a wide range of equipment, such as (i) through dedicated hardware which is built to execute specific functions, (ii) through domain-specific

programmable-logic chips, or (iii) through software running on general-purpose computers. In general, there is a trade-off between speed of execution and versatility, and so it is important to use processing equipment appropriate to an application's context.

Networks are increasingly required to be reconfigurable, and as such processing functionality which was previously done on dedicated hardware is now being done in software and programmable logic. The advantage of using software is that functions can be quickly reconfigured, added or dropped, and updated in a flexible and dynamic manner. This movement towards using software over hardware to perform the processing of network functions is termed *softwarization*, a movement which can also be seen in other domains such as automotive, manufacturing, transport, and in consumer products.

It may be useful to re-state the distinction between hardware and software, as summerized in Fig. 2. We define software as processing occurring only in the abstract domain (which is the realm of ideas, concepts, logic and mathematics), whereas hardware is defined as processing occurring between the abstract domain and the physical domain (the world of physical objects). The translation between the physical and the abstract domains occurs through a process known as representation. More details about representation can be found in [18].

This definition of software also applies to functionalities running on programmable hardware, whose compilation and execution process can be directly related to standard software execution process, as seen in Fig. 3. For instance, a C program is compiled into assembly, then into machine code, and executed on a processor. The processor itself is hardware, whereas the C program, the assembly program, and the binary machine code are all software representing the same functionality at different levels. Similarly a program written in VHDL language is synthesized and translated into netlist, and compiled into a bitstream for configuring logic gates, fabrics and memory resources an FPGA chip. The FPGA chip is the hardware, whereas the VHDL program, the netlist and bitstream are representations of functionality in software.

To ease the software development process, a programmable framework can be employed to provide general functionality for a particular application domain, which can then be selectively used to create specific implementations. Key features of programmable frameworks are that they are modular, flexible, extensible, and reusable.

2.2 Virtualization

Virtualization is increasingly being employed as a means of managing software resources in a manner appropriate to the application using the resources. We define *virtualization* as a resource mapping in software by a hypervisor, which maps virtual resources, which

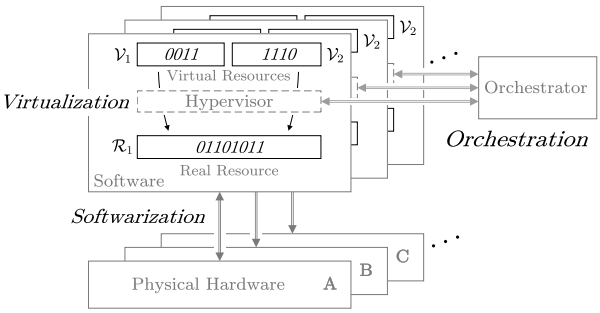

Fig. 2 Softwarization is the use of software rather than hardware to perform some functionality, while virtualization is the tailoring of software resource sizes to specific contexts. Orchestration is the placement of functionality and management of resources for particular purposes

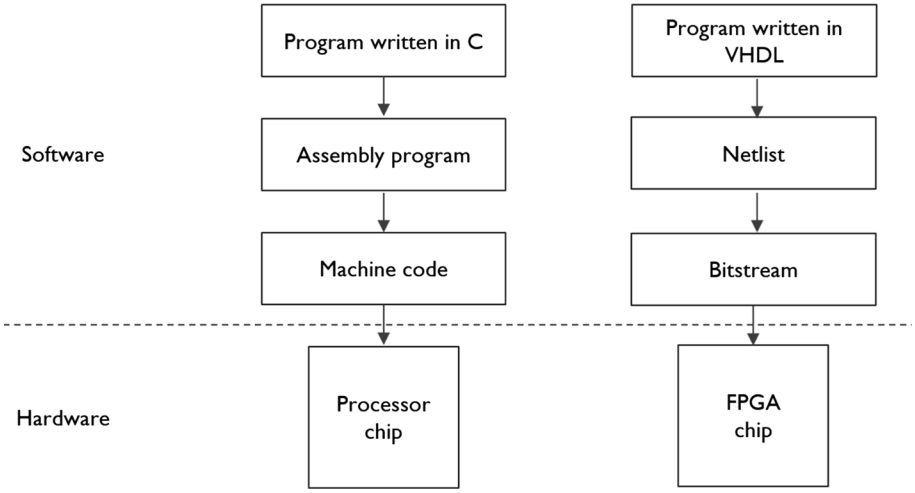

Fig. 3 Extending the concept of softwarization to functionality running on programmable hardware

can be flexibly sized depending on context, to real resources that are fixed in size. Virtual resources are independent of each other and can be used by different users in whatever manner desired. For example, virtualization allows underutilized hardware resources to be used by multiple users independently, or alternatively, can combine multiple hardware resources as if they were one large resource.

It is important to note however, that softwarization and virtualization are distinct and separate processes, but that virtualization depends on softwarization, as it cannot be done in hardware. Figure 2 illustrates the softwarization process, and the subsequent virtualization of software resources. In this case the real hardware resource, $\mathcal{R}_1$, is an 8-bit memory location, to which the hypervisor maps two independent 4-bit virtual software resources, $\mathcal{V}_1$, and $\mathcal{V}_2$.

Hypervisors interface directly with the software resources, but understand the limitations of the real resources, which result from the underlying hardware. Therefore, hypervisors are often specific to one type of hardware to accomplish virtualization effectively. We later introduce how hypervisors can be implemented to virtualize radio spectrum resources.

2.3 Orchestration

So far we have described the softwarization and virtualization processes as they occur on a single device. However, in a network there can be many devices which are cooperating to perform some desired functionality. A device in a network is referred to as a node, whereas the connection between nodes is referred to as a link. *Orchestration* is the management and control of network resources, i.e. node and link resources, and the placement of functionality in a network. Orchestration knows the purpose and specific requirements of a particular functionality, and manages network resources accordingly. Sometimes, orchestrators are known as controllers, but we prefer to use the term orchestrator for consistency between network domains.

Referring again to Fig. 2, an orchestrator is used to place functionality on different nodes A, B, C, etc. and interfaces directly with the hypervisor of each node to manage

resources appropriately. Orchestrators must know the *type* of resources they are managing (such as computing nodes in this example) to place functionality correctly, and typically different orchestrators are used for different types of resources.

Orchestration can be done with or without virtualization, but the use of virtualization greatly increases the effectiveness of orchestration, as the orchestrator can tailor virtual resources to match functionality.

2.4 Radio Function

Given the previous definitions, this section explains what we consider as a radio function, and how the different concepts explained previously are applied.

Radios typically consist of the following building blocks as depicted in Figure 4: (i) an analog Radio Frequency (RF) front end including conversion between analog and digital domains, (ii) processing functions at In phase/Quadrature (I/Q) sample level, (iii) processing functions at symbol level, and (iv) processing functions at bit level.

The front end is the hardware that transmits or receives RF signals, converts the radio frequency to and from an intermediate frequency, and either performs Analogue-to-Digital Conversion (ADC) or Digital-to-Analogue Conversion (DAC), depending on transmission or reception. At I/Q level, digitized I/Q samples are processed, including searching for preambles, estimating and removing a carrier frequency offset, and equalization of the signal. Alternatively, I/Q level processing can also generate outcomes to indicate the energy level of the targeted radio spectrum. At the symbol level, modulation or demodulation occurs, in addition to processes such as sub-carrier mapping, and insertion of pilot signals. Bit level processing refers to any further operations after symbols are mapped to bits, such coding and decoding, forward error correction, and encryption.

A processing function at any of the above levels is referred to as a *radio function*, and can be implemented in either hardware or software. Conventionally, radio functions have been realized on dedicated hardware (i.e., commercial radio chipsets), and were optimised for one particular radio standard such as Long-Term Evolution (LTE). However, although dedicated hardware provides efficient implementation, it takes many iterations to mature the hardware design, and inevitably prolonging the development cycle. Also, due to the difficulty to change hardware, many choices need to be made at design time, resulting in relatively limited runtime reconfigurability. Softwarization of radio functionalities thus resolves both issues by (i) offering fast prototyping approaches and (ii) shifting hardware design time choices to software runtime choices.

The softwarized radio functions can be executed in many different ways. A first method of execution is on general purpose processors, such as servers in the cloud or computers with direct connection to radio devices, commonly referred to as a "Host PC".

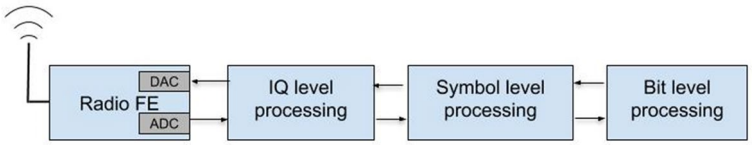

Fig. 4 Typically, radios are comprised of an RF front end, and several processing blocks at I/Q sample level, symbol level, and bit level. Increasingly, these blocks are being executed in software, rather than in hardware

Alternatively, radio functions can be processed on micro-controllers or embedded processors on board; finally, it is also common to implement radio functions partially or completely in programmable hardware such as FPGA, using specific hardware description language.

When radio functionalities are implemented in some form of software as described above, virtualization can be used to ease the management of resources, through the use of an orchestrator. A hypervisor should be present to enable the virtualization of radio functions, taking into account the underlying resources, in terms of the radio spectrum, the computational resources on a device (e.g., the size of an FPGA), and the capability of the RF front end (e.g., the supported RF frequency range and sampling rate).

2.5 Summary of Definitions

- *Abstract Domain:* ideas, concepts, logic, mathematics.
- *Physical Domain:* the world of physical objects.
- *Hardware:* the objects that process abstract operations (maths, logic, etc.) in the physical domain.
- *Software:* the processing of abstract operations in the abstract domain. Dependent on underlying hardware.
- *Programmable Framework:* a software environment that provides modular, extensible, and reconfigurable functionality for a particular application domain.
- *Softwarization:* moving functionality from hardware to software.
- *Virtualization:* a mapping that tailors the size of software resources to context.
- *Hypervisor:* the entity responsible for virtualization.
- *Orchestration:* the placement of functionality and the management and control of resources.
- *Orchestrator:* the entity that performs orchestration.
- *Radio function:* the processing functions at IQ sample, symbol or bit levels to realize wireless communication.

3 An Overview of Related Work on Network Reconfigurability

Having clarified the concepts of softwarization, virtualization, and orchestration, in this section we examine how these concepts are used in networks today. As illustrated in Fig. 5, networks can be divided into three domains: the nodes (or servers), the wired network (switches), and the wireless network (radios). We refer to increasing reconfigurability in nodes as NFV, in wired networks as SDN, and in wireless networks SDR, since these are the most commonly used terms in the literature. We consider radio today with reconfigurability but not fully softwarized down to the physical layer as an intermediate step of softwarization of wireless network; this type of radio is referred to as Reconfigurable Radio System (RRS). Although most commercial wireless devices today fall into this category, in this discussion we only focus on the fully softwarized wireless network based on SDR. We now discuss each of these domains with respect to the following aspects: the hardware, the programmable frameworks (enablers for softwarization), hypervisors (enablers for virtualization), and orchestrator.

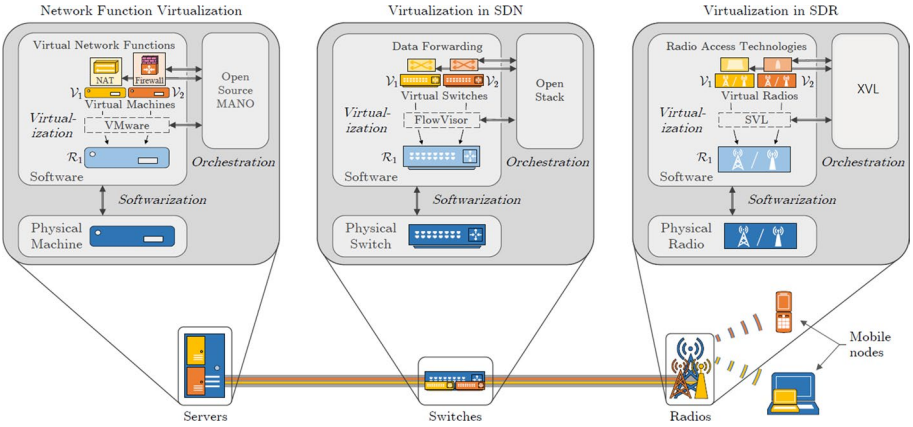

Fig. 5 Illustration of softwarization, virtualization and orchestration in end-to-end networking. The network can be considered as consisting of three domains, namely: nodes, wired links, and wireless links. We refer to the use of reconfigurability in these three domains as NFV, SDN, and SDR respectively

3.1 Network Function Virtualization

In NFV, softwarization occurs through the use of off-the-shelf components (e.g., x86 or ARM processors) rather than proprietary hardware, with the objective of reducing costs and increasing reconfigurability. Typical network functionalities running on a node include Network Address Translation (NAT) and firewalls. The virtualization of these functions can be done through many different hypervisors, such as VMware, KVM, Xen, etc., which provide customizable Virtual Machine (VMs), or containerization (a more limited form of virtualization), such as provided by Docker. Modern containers or virtual machines can be controlled and programmed by a set of common APIs, such as libvirt [3] and LXD [16].

There are also a large number of orchestrators that can be used, such as Open Source MANO, OPEN-O, etc. The orchestrator typically manages the virtual infrastructure through hypervisors, such as VMware as depicted in Figure 5, and at the same time configures the individual functions, such as the address space of NAT or rules used by a firewall, as depicted in Figure 5.

3.2 Software Defined Networking

Softwarization in SDN takes place through the use software-defined switches and routers such as switches that support the OpenFlow standard, which are reconfigurable and reprogrammable. In addition to softwarization, virtualization can be used to create virtual networks through the use of hypervisors such as FlowVisor, CellVisor, or Network Hypervisor [2]. Orchestrators include OpenStack (Neutron), NOX, ONOS, Beacon, etc.

In the example of Fig. 5, the hypervisor's role is fullfilled by FlowVisor, and the orchestrator is implemented in ONOS. ONOS, together with OpenDaylight, RYU are amongst the best known programmable frameworks in the SDN field. The orchestrator implemented in certain programmable frameworks usually has the capability to configure a rich set of hardware, including various switches and network cards, using southbound protocols (e.g, Openfow or netconf combined with the YANG model). From the device's point of view, it must expose a proper northbound interface towards the

orchestrator, through the use of softwarization. Similar to the case of NFV, the orchestrator also has the authority to configure both the virtual switches and the FlowVisor.

3.3 Software Defined Radio

SDR can rely on programmable frameworks running on general purpose processors, such as GNU Radio and Labview, to define functionalities which comprise the radio access technology, while the RF front ends are devices such as Zynq SDR, BladeRF, and USRP [6, 8, 15]. Some programmable frameworks allow users to configure FPGAs or other embedded resources close to the radio front end. For instance, GNU Radio can be combined with RFNoC to distribute radio functionalities between the on-board FPGA and the host PC. Alternatively, SDR applications can also be realized by standalone software without a programmable framework, such as srsLTE [7] and OpenAir-Interface [14]. These are all means to softwarize radio functions.

Virtualization of radio functions is a rather new topic. For functionalities that are running on a host PC or cloud servers, in principle all techniques from the NFV field can be applied. For instance, one can run srsLTE or a GNU Radio flow graph inside a virtual machine. However, this does not mean that radio functions can be placed, initialized and configured as easily as a regular virtual network functions. There are unique challenges to the execution of real-time radio functions: for example, samples must be provided in time, sometimes even precisely time-stamped. These challenges are typically not well handled by hypervisors designed for general NFV purposes. For functionalities that run on embedded systems, we are not yet aware of any hypervisor for embedded resources on an SDR. Therefore we present relevant solutions in Sect. 4.

A few hypervisors have been developed specifically for wireless networks, such as Spectrum Virtualization Layer (SVL), MySVL, and HyDRA [10, 17]. Such hypervisors generally allow samples coming from multiple radio networks to be combined and transmitted by a single radio front end, the reverse process happening in the receiving path. A hypervisor may split one signal into multiple parts in the frequency domain, which can be transmitted by one or multiple radio front ends. The advantage of this kind of virtualization is not only related to sharing the radio front end, but also better utilization of the radio spectrum: By combining or splitting signals using such a hypervisor, spectrum allocation becomes more flexible, hence the name spectrum virtualization layer. When splitting a signal into two separate parts in the frequency domain, filters need to be carefully designed, and various synchronization issues need to be resolved in case the two parts are transmitted by different radio devices. These kinds of challenges are tackled by HyDRA [10].

Orchestration of SDR is also at its initial stage. One example of such an orchestrator is SDN-R, an OpenDaylight controller extended by the SDN community to wireless networks. XVL [17] is a service oriented controller above a radio hypervisor: it allocates resources to create virtual radio front ends for an existing RAT communication stack, but it does not have influence on the RAT itself.

To date, we are not aware of any orchestrator and hypervisor that can achieve the placement and configuration at the level of radio functions, meaning that no solution can fully exploit the flexibility of SDR devices in an end-to-end communication process.

4 Virtual Radio Function and Its Realization on SDR

Inspired by the NFV paradigm and the decomposition of network functions, we assess how the operation of a RAT can be split into functions, and the placement of these functions on SDR platform. In addition, we introduce a way to realize spectrum virtualization layer with a combination of filter banks and mixers. We focus on embedded SDR with FPGA on board.

4.1 Virtual Radio Functions

As shown in Sect. 2.4, the radio functionality can be split into the bit, symbol and IQ level processing functions. Although many different RATs are in use today, they often share similar building blocks. For instance, all Orthogonal Frequency Division Multiplexing (OFDM) based RATs use Fast Fourier Transform (FFT), and almost all packet detection mechanisms rely on certain types of correlation. As such, we can model previously monolithic RATs into chains of radio functions with different configurations [1], akin to the NFV paradigm where services are defined as chains of Virtual Network Function (VNFs).

Such functional decomposition of RATs enables a clear separation of the functional blocks necessary for realizing a given RAT. The implementation of radio functions has been conventionally done in hardware, which requires a distinct and specialized radio device per RAT, with very limited configurability exposed by the driver API. With the support of SDRs, radio functions can now be configured to form different RATs during runtime. Hence, one radio device may switch between multiple RATs at different times, or even act as multiple homogeneous or heterogeneous types of radio interfaces simultaneously if there are sufficient computational resources and RF front end capabilities.

The use of softwarization allows radio functions to be instantiated and placed at various levels (i.e., cloud, host, micro-controller, FPGA). From cloud to FPGA, each of the listed options has its own advantages: in general on the cloud end, there is more flexibility and ease of configuration, whereas at the FPGA side there is faster reaction speed. Such trade-offs are crucial factors in the design and operation of radio functionality. The functional split of RATs is already being explored in the context of C-RAN, where mobile operators leverage the trade-offs of the placement of radio functions for realizing RATs [13]. The mobile operator may place radio functions in the cloud, for achieving better resource utilization and interference control, or in the edge, for achieving latency requirements and reducing fronthaul traffic. The placement of radio functions only affects the performance of the RATs and their computational resource utilization, while the functionality of the RATs is agnostic to where their radio functions are placed [13]. In that regard, radio functions are actually virtual, i.e., they are Virtual Radio Function (VRFs).

We believe that RATs can be realized using a composition of VRFs, analogous to the realization of services using VNFs [11]. Part of the VRFs may be realized using physical devices, e.g., in a split-PHY approach, where part of the lower-level VRFs are realized at the radio itself, but cannot be moved, akin to the placement of physical network functions in the ETSI NVF MANO architecture. Figure 6 illustrates the parallel between a RAN realized through VRFs and a Core Network (CN) realized through VNFs.

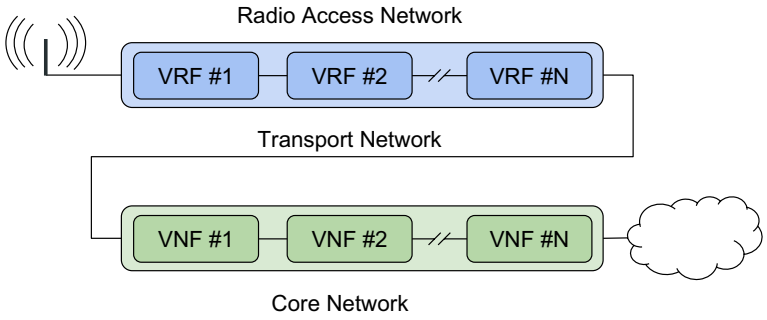

Fig. 6 Example of an E2E network where the CN's functionality is realized through VNFs and the RAT is realized through VRFs

4.2 The Realization of VRF on Embedded SDR

In this section, we discuss the realization of VRF on embedded SDR. Mainstream SDR devices today are composed of programmable logic (i.e., FPGA) and embedded processors. Some SDRs have hard processors (e.g., the ZYNQ SDR has a dual-core ARM processor), whereas others have soft processors made out of the FPGA fabrics (e.g., the USRP X300 series uses Kintex FPGA with a soft processor ZPU). We leverage existing tools to partially reconfigure the FPGA on SDR devices, in order to achieve the placement, initialization and configuration of VRFs. The mechanism is illustrated in Fig. 7.

A hardware module on an FPGA is termed as an Intellectual Property (IP) core. In general, the SDR requires at least one IP core to interact directly with the radio front end. This core is highly hardware-dependent, hence there is not much added value in making it runtime configurable/replaceable. This type of IP core suits best the static part of FPGA design. After this stage, the IP cores can act as processing engines for tasks at IQ/symbol/bit levels: configuring or replacing some of the IP cores can form different RATs or make improvements to the existing RAT. In this case, an IP core is actually a VRF. Some levels of runtime configuration may be achieved at firmware level, such as adapting filter coefficients or the length of the cyclic prefix, while some situations require a total replacement of the IP core, such as changing the FFT size. If only a firmware level update is required, the IP core acting as a VRF can still be placed in the static part of an FPGA design; if an IP core needs to be replaced in runtime, it must be placed

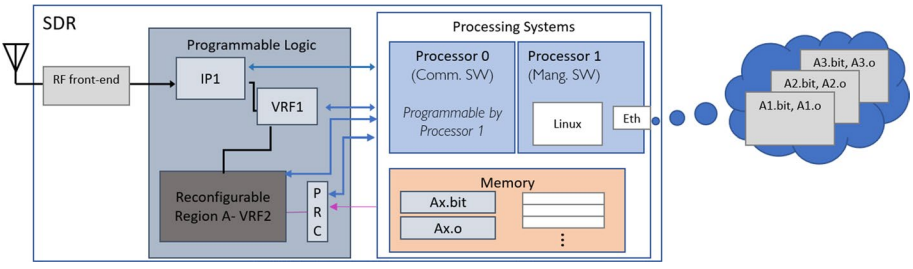

Fig. 7 Partial reconfiguration of FPGA applied for VRF realization on embedded SDR, adapted from [12]

within a Reconfigurable Region (RR) of the FPGA. These two scenarios are represented by *VRF1* and *VRF2* respectively in the example of Fig. 7.

The Partial Reconfiguration Controller (PRC) is an IP core offered by Xilinx that can load a partial bitstream (noted as *Ax.bit* in Fig. 7) from a memory region to the target RR, upon a trigger received from the processing system. The processing system is formed by two processors and some memory resources, which can be either soft or hard processors. One processor (referred to as Processor 0) interacts directly with VRFs and plays a role in the RAT communication functionality, whereas the other processor (referred to as Processor 1) manages the partial reconfiguration and the software running on Processor 0. The tasks of Processor 1 include: (i) fetching partial bitstream and firmware object files from remote storage, (ii) storing the files in predefined memory regions, (iii) triggering the PRC to load the partial bitstream into the RR, (iv) triggering Processor 0 to use the newly available VRF via the updated firmware. The firmware update can be realized by dynamically linking the object file, compiled from functions with a predefined signature. Given that these tasks can be more easily handled by existing tools in the Operational System (OS), Processor 1 runs embedded Linux with dedicated software developed for management purposes.

The network used for fetching the configuration files can be a backbone connection using Ethernet, or the SDR radio interface itself. In both cases, the configuration files need to be correctly obtained, verified and stored on board before the configuration takes place. This is to prevent using incompatible configuration files. In case a backbone network connection is present, the successful rate of the configuration is less critical. However, a restoration mechanism should always be present as a fall back, in case the radio function does not operate as expected. The restoration is triggered by Processor 1 upon a time out condition.

4.3 Spectrum Virtualization Layer on Embedded SDR

In Sect. 3 we observe that several hypervisors exist for sharing the underlying radio front end. Signals coming from multiple communication stacks can be allocated in such a way that the bandwidth supported by the radio front end is optimally used. However, all these solutions are running on host computer. When the RAT formed by chained VRFs is operating on embedded SDR, the hypervisor for the spectrum virtualization must also be realized on the embedded resources.

This section introduces a way to realize the Spectrum Virtualisation Layer (SVL) on embedded SDR by combining a series of mixers and filter banks. The architecture of the receiving path is shown in Fig. 8. IQ samples coming from the radio front end are streamed in parallel to an array of mixers. The purpose of these mixers is to perform frequency shifting, so that target signals at different center frequencies can be moved to baseband. After this stage, a gain module is present: it performs bit-shifting on the coming samples. The output samples of this stage have reduced width, which is helpful to save FPGA resources. The next stage is the filter bank with a certain decimation ratio; the output of this stage is multiple streams of baseband IQ samples that are ready to be further processed by radio functions in a given RAT. This combination of multiple frequency shifting mixers and filter banks is called Direct Down Converter (DDC) filter banks. The transmission path is simply the reverse process, except that the mixers performs Direct Up Conversion (DUC), and the filters perform interpolation rather than decimation.

In the example shown, IQ samples are provided by the radio front end at 40 Msps, which is sufficient to cover 2 WiFi channels and 8 Zigbee channels, and by configuring

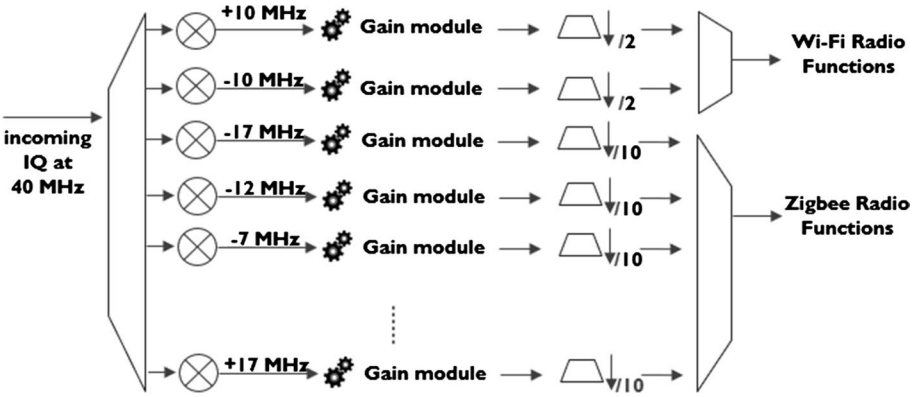

Fig. 8 The architecture of DDC filter banks used for radio front end virtualization on embedded SDR

the frequency shifting in the mixers accordingly, the output of the filter bank is 8 streams of baseband 2 MHz IQ samples (decimated by a factor of 10), and 2 streams of baseband 20 MHz IQ samples (decimated by a factor of 2). These streams of samples are ready to be further processed by VRF chains. Although in principle one may use multiple chains of radio functions to process the samples, in [5] a single preamble detector operating at high speed is used to detect packets on all 10 channels.

By using a series of mixers and filters on an embedded SDR, one single radio front end can be sliced among multiple homogeneous or heterogeneous radio access technologies. The frequency shift of the mixers can be configured in runtime by registers, meaning the center frequency of each virtual radio can be configured. However the total number of channels (depending on the number of filters and mixers) and the bandwidth of each channel (depending on the decimation/interpolation ratio of the filters) are not runtime configurable. As a remedy, we treat the DDC/DUC filter bank as a special VRF, and the entire architecture can be runtime replaced by partial reconfiguration of FPGA. In this way, the number of virtual radios and the individual bandwidth can also be runtime configured.

5 Conclusions

We have observed the softwarization and virtualization trend for both wired and wireless networks, enabled by the abundance of computational resources. Wireless networks, however, present additional complexities: (i) they rely critically on the use of costly radio spectrum, a resource that is physically constrained and hence subject to very different cost and scaling laws, and (ii) the last mile access network can be served by many heterogeneous technologies, depending on the application requirements (e.g., wide coverage, high throughput, or low latency). Because of this, there is a need to enable more flexible allocation of spectrum and dynamic usage of radio access technologies. In this paper, we define softwarization and virtualization for network configuration, introduce the concept of chaining virtual radio functions at runtime for dynamically constructing radio access technologies, and describe the realization of virtual radio functions on embedded SDR devices via partial FPGA configuration. In addition, we apply DDC/DUC filter banks and mixers for flexible allocation of spectrum to individual virtual radios on a wide-band SDR device.

The DDC/DUC filter bank is regarded as a special virtual radio function that can be runtime replaced when needed.

Acknowledgements This publication has emanated from research conducted with the financial support from the European Horizon 2020 Program under the Grant Agreement No. 732174 (ORCA project). It was also partially supported by Science Foundation Ireland Grant No. 13/RC/2077 (CONNECT).

References

1. Bansal, M., Mehlman, J., Katti, S., & Levis, P. (2012). Openradio: A programmable wireless dataplane. In *Workshop on hot topics in software defined networks* (pp. 109–114). ACM.
2. Blenk, A., Basta, A., Reisslein, M., & Kellerer, W. (2016). Survey on network virtualization hypervisors for software defined networking. *IEEE Communications Surveys & Tutorials*, *18*(1), 655–685. https://doi.org/10.1109/COMST.2015.2489183.
3. Bolte, M., Sievers, M., Birkenheuer, G., Niehörster, O., & Brinkmann, A. (2010). Non-intrusive virtualization management using libvirt. In: Design, automation & test in Europe conference & exhibition (DATE) (pp. 574–579). IEEE.
4. Brown, G. (2017). Cloud RAN & the next-generation mobile network architecture. White Paper.
5. de Figueiredo, F. A., Jiao, X., Liu, W., & Moerman, I. (2018). Radio hardware virtualization for software-defined wireless networks. *Wireless Personal Communications*, *100*(1), 113–126.
6. Ettus. https://www.ettus.com/.
7. Gomez-Miguelez, I., Garcia-Saavedra, A., Sutton, P., Serrano, P., Cano, C., & Leith, D. (2016). srsLTE: An open-source platform for LTE evolution and experimentation. In *International workshop on wireless network testbeds, experimental evaluation, and characterization* (pp. 25–32). ACM.
8. Harikrishnan, B., Raghul, R., Shibu, R., & Nair, K. R. (2014). All programmable SOC based standalone SDR platform for researchers and academia. In *International conference on computational systems and communications (ICCSC)* (pp. 384–386). IEEE.
9. Introduction to the NI mmwave transceiver system hardware-national instruments. http://www.ni.com/product-documentation/53095/en/.
10. Kist, M., Rochol, J., DaSilva, L. A., & Both, C. B. (2017). Hydra: A hypervisor for software defined radios to enable radio virtualization in mobile networks. In *Conference on computer communications workshops (INFOCOM workshops)* (pp. 960–961). IEEE.
11. Kist, M., Wickboldt, J., Granville, L., Rochol, J., DaSilva, L., & Both, C. (2019). Flexible fine-grained baseband processing with Network Functions Virtualization: Benefits and impacts. *Computer Networks*, *151*, 158–165.
12. Liu, W., Jiao, X., & Moerman, I. (2018). Live reprogramming of SDR at FPGA and processor level (2018). https://www.orca-project.eu/live-reprogramming-of-sdr-at-fpga-and-processor-level/.
13. Maeder, A., Lalam, M., De Domenico, A., Pateromichelakis, E., Wubben, D., Bartelt, J., Fritzsche, R., & Rost, P. (2014). Towards a flexible functional split for cloud-RAN networks. In *European conference on networks and communications (EuCNC)* (pp. 1–5).
14. Nikaein, N., Marina, M. K., Manickam, S., Dawson, A., Knopp, R., & Bonnet, C. (2014). OpenAirInterface: A flexible platform for 5G research. *ACM SIGCOMM Computer Communication Review*, *44*(5), 33–38.
15. Nuand. https://www.nuand.com.
16. Rosen, R. (2014). Linux containers and the future cloud. *Linux Journal*, *240*(4), 86–95.
17. Santos, J. F., Kist, M., van de Belt, J., Rochol, J., & DaSilva, L. A. (2019). Towards enabling RAN as a service: the extensible virtualisation layer. In *IEEE International Conference on Communications (ICC)* (pp. 1–6). IEEE.

18. van de Belt, J., Ahmadi, H., & Doyle, L. (2017). Defining and surveying wireless link virtualization and wireless network virtualization. *IEEE Communications Surveys & Tutorials*, *19*(3), 1603–1627.

Wei Liu received her master's degree in Electronical Engineering in 2010, and obtained her Ph.D. degree from Ghent University in 2016. During her doctoral education, she participated in multiple research projects related to next generation wireless technologies, she became familiar with several software-defined radio platforms, and gained experiences in testbed operations. She is currently a Post-Doctoral Researcher with IDLab, Ghent University, leading H2020 ORCA project. Her research is in the field of cognitive radio and Software-Defined Radio, focusing on spectrum analysis, interference prevention, reconfigurable and flexible radio architectures.

Joao F. Santos is pursuing a Ph.D. on wireless networks at the CONNECT Telecommunications Research Centre, headquartered at Trinity College Dublin. He holds a B.Sc. in Telecommunications Engineering from Universidade Federal Fluminense. He worked at Rede Nacional de Ensino e Pesquisa (RNP) as the lead developer of the FIBRE testbed federation's Clearinghouse. His research interests include network slicing, radio virtualisation, and end-to-end network orchestration.

Jonathan van de Belt received a Ph.D. in Electronic Engineering from Trinity College Dublin (TCD) in 2018, and is now working at the Centre for Intelligent Power at Eaton. Previously, he was a research fellow at CONNECT, TCD, working with Professor Luiz DaSilva on the ORCA project. Before joining Professor DaSilva's group, Jonathan investigated industrial cyber-physical systems at Intel Labs Europe, and briefly worked at Xilinx Research Laboratory, Ireland. Jonathan received the B.A.I. degree in engineering and the B.A. degree in mathematics from TCD in 2012. His research interests include machine learning, software-defined radio, digital signal processing, wireless network virtualization, and embedded systems.

Xianjun Jiao received his bachelor degree in Electrical Engineering from Nankai university in 2001 and Ph.D. degree on communications and information system from Peking University in 2006. After his studies, he worked in industrial research departments and product teams in the domain of wireless technology, such as Radio System Lab of Nokia Research Center, devices department of Microsoft and Wireless Software Engineering department of Apple. In 2016, he joined IDLab, a core research group of imec with research activities embedded in Ghent University and University of Antwerp. He is working as senior researcher at imec on flexible real-time Software Defined Radio (SDR) platform. His main interests are SDR, signal processing and parallel/heterogeneous computation in wireless communications. On his research track, 30+ international patents/papers have been granted/published.

Ingrid Moerman received her degree in Electrical Engineering (1987) and the Ph.D. degree (1992) from the Ghent University, where she became a part-time professor in 2000. She is a staff member at IDLab, a core research group of imec with research activities embedded in Ghent University and University of Antwerp. Ingrid Moerman is Program Manager of the 'Deterministic Wireless Networks' track at imec. Ingrid Moerman is also coordinating the research activities on mobile and wireless networking at Ghent University, where she is leading a research team of more than 30 members. Her main research interests include: collaborative and cooperative networks, intelligent cognitive radio networks, real-time software defined radio, flexible hardware/software architectures for radio/network control and management, Internet of Things, Next generation wireless networks (5G/6G/ …), and experimentally-supported research. Ingrid Moerman has a long-standing experience in running and coordinating national and EU research funded projects. At the European level, Ingrid Moerman is in particular very active in FP7/H2020 programs, where she has coordinated and is coordinating several projects (CREW, WiSHFUL, eWINE, ORCA). Ingrid Moerman was leading team SCATTER, consisting of researchers from IMEC-IDLab and Rutgers University, in the DARPA Spectrum Collaboration Challenge (SC2). The SCATTER team has been awarded with two prizes of 750,000 USD each in Phase 1 and Phase 2 of the DARPA SC2 competition, and was one of the 10 finalist at the DARPA SC2 championship event organized at Mobile World Congress in LA (October 2019). Ingrid Moerman is author or co-author of more than 750 publications in international journals or conference proceedings.

Johann Marquez-Barja currently is an Associate Professor at University of Antwerpen and imec. He is the Head of the Wireless Cluster at imec IDLab UAntwerp. He was and is involved in several European research projects, being Principal and Co-Principal Investigator for many projects. He is an ACM member, and a Senior member of the IEEE Communications Society as well as the IEEE Education Society, where he participates in the Standards Committee. His main research interests are: 5G advanced architectures including edge computing; flexible and programmable future end-to-end networks; IoT communications and applications. He is also interested in vehicular communications, mobility, and smart cities deployments. Prof. Marquez-Barja is co-leading the Citylab Smart City testbed, part of the City of Things programme, located in Antwerpen, Belgium. Prof. Marquez-Barja has been given several keynotes and invited talks in different major events, as well as received 25 awards in his career so far, and co-authored more than 100 articles. He is also serving as Editor and Guest editor for different International Journals, as well as participating in several Technical Programme and Organizing Committees for several worldwide conferences/congresses.

Luiz DaSilva holds the chair of Telecommunications at Trinity College Dublin, where he is the Director of CONNECT, the Science Foundation Ireland Research Centre for Future Networks and Communications. Prior to joining Trinity College, Prof DaSilva was a tenured professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech. His research focuses on distributed and adaptive resource management in wireless networks, and in particular radio resource sharing and the application of game theory to wireless networks. Prof. DaSilva is a principal investigator on research projects funded by the Science Foundation Ireland and the European Commission. Prof DaSilva is a Fellow of Trinity College Dublin, and a Fellow of the IEEE, for contributions to cognitive networks and to resource management in wireless networks

Sofie Pollin obtained her Ph.D. degree at KU Leuven with honors in 2006. From 2006–2008 she continued her research on wireless communication, energy-efficient networks, cross-layer design, coexistence and cognitive radio at UC Berkeley. In November 2008 she returned to imec to become a principal scientist in the green radio team. Currently, she is associate professor at the electrical engineering department at KU Leuven. Her research centers around Networked Systems that require networks that are ever more dense, heterogeneous, battery powered and spectrum constrained. Prof. Pollin is BAEF and Marie Curie fellow, and IEEE senior member.