Received October 18, 2019, accepted November 6, 2019, date of publication November 11, 2019, date of current version November 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952619

Data Plane Framework for Software-Defined Radio Access Network Based on ETSI-Standard Mobile Device Architecture

HEUNGSEOP AHN[1], SEUNGWON CHOI[1], (Member, IEEE), MARKUS MUECK[2], (Member, IEEE), AND VLADIMIR IVANOV[3], (Member, IEEE)
[1]Department of Electronics and Computer Engineering, Hanyang University, Seoul 04763, South Korea
[2]Intel Germany GmbH, 85622 Munich, Germany
[3]Saint Petersburg State University of Aerospace Instrumentation (SUAI), 190000 Saint Petersburg, Russia

Corresponding author: Seungwon Choi (choi@dsplab.hanyang.ac.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion (IITP) through Korean Government (Ministry of Science and Information & Communication Technology) under Grant 2017-0-00723, and in part by the Development of Software-defined Service-oriented Integrated base station platform using Reconfigurable Radio System Technology.

ABSTRACT This paper addresses how to achieve efficient programmability and software portability in the data plane of a software-defined radio access network (SDRAN). We assume a cloud RAN environment that builds on multi-vendor hardware components. Recent literature on SDRAN data plane indicates that software portability remains an issue in terms of efficient execution of software, even if the software is abstracted from the underlying hardware. In addition, software interfaces typically vary across different hardware components in the SDRAN data plane, leading to platform-dependent software management. Generalizing the European Telecommunications Standards Institute approach for a mobile device architecture, this paper presents a novel SDRAN data plane framework, providing efficient hardware platform-independent programmability and software portability. First, to resolve the software portability issue, the proposed data plane framework employs a specific (radio) virtual machine as well as a radio library; the heterogeneous hardware platforms are abstracted, enabling the joint optimization of the radio application code and hardware platform. Second, to achieve platform-independent software management, the proposed data plane framework adopts a double-layered structure enabling users to exploit high-level software management for the SDRAN data plane. Third, the feasibility of the proposed data plane framework is verified through a proof-of-concept (PoC) system with the proposed double-layered structure. Based on this PoC system, we show that users can efficiently perform software management. According to the numerical results obtained from the PoC system, the proposed double-layered structure introduces negligible additional footprint in terms of computational resources, memory requirements, and latency.

INDEX TERMS Software-defined RAN, C-RAN, ETSI-standard, data plane programmability, software portability.

I. INTRODUCTION

In fifth-generation (5G) mobile networks, there is an urgent need to operate in accordance with the requirements set forth by various services, which include e-health, self-driving vehicles, and internet of things [1]. To cope with these new requirements, studies on softwarized network solutions have been actively performed and seek to control the network configuration with software [2]. Recent research on softwarized networks has been performed on both the radio access network (RAN) side and the core network (CN) side. The software-defined radio access network (SDRAN) and cloud-RAN (C-RAN) belong to the former, while software-defined core networks and network function virtualization (NFV) belong to the latter. Using a softwarized network, network operators can optimize the network according to various 5G requirements while also significantly reducing network capital expenditures and operating expenditures through efficient use of the network infrastructure [3].

This paper addresses the problems of data plane programmability and software portability in SDRAN, which is a typical solution for softwarising the RAN [4]. Recently,

The associate editor coordinating the review of this manuscript and approving it for publication was Kashif Sharif.

OpenRadio [5] and PRAN [6] have been suggested to provide programmability to the SDRAN data plane. However, previous works, including OpenRadio and PRAN, suffer from the software portability problem because they are platform-specific. The software portability issue is serious, especially in the data plane processing of layer 1 (L1) and layer 2 (L2) because L1/L2 data plane processing is generally performed on hardware platforms consisting of not only a general-purpose processor (GPP) but also various types of special-purpose processors, such as digital signal processors (DSPs), field programmable gate arrays (FPGAs), graphic processing units (GPUs), and application-specific integrated circuits (ASICs), due to the heavy computational loads and real-time processing constraints required for modem functionalities [5], [7]–[9]. In an SDRAN, as each of the hardware platforms for the data plane processing of L1/L2 may be provided by different vendors, the software portability issue becomes even more complicated. Consequently, with previous works involving this issue, software reconfiguration would be limited only to specific hardware platforms.

A C-RAN, a typical application of NFV in a RAN, is often employed in an SDRAN for the efficient utilization of computational resources and collaborative processing among baseband units (BBUs) in the infrastructure layer [8]. It is noteworthy, however, that unless the software portability issue is resolved, the software reconfiguration of a C-RAN would also be limited only to the specific platform because the heterogeneous BBUs in a given C-RAN may possess differing hardware platforms with various levels of programmability and computing power. It is interesting to observe that the above-described situation regarding the software portability issue in a C-RAN is antithetic to the case of an NFV-based CN, which does not incur the software portability problem. Without the requirement of a real-time processing constraint or heavy computational load, an NFV-based CN generally operates on an industry standard server, which makes it possible for the NFV-based CN to be reconfigured through open-source virtual network function (VNF) applications [9]–[11].

This paper presents a novel data plane framework for an SDRAN. The basis of the proposed framework is taken from the reconfigurable mobile device architecture, which has been standardized by the European Telecommunications Standards Institute Technical Committee Reconfigurable Radio Systems (ETSI TC-RRS) [12]–[14]. The ETSI-standard mobile device architecture is only applicable for a mobile device consisting of a single hardware platform. Thus, to develop the novel SDRAN data plane framework, we first generalize the ETSI-standard mobile device architecture in such a way that it can be applicable for generic radio equipment consisting of multiple hardware platforms. It is also essential to prove that this generic framework can be instantiated for the specific needs of the SDRAN data plane. The novelty of the proposed approach can be summarized as follows:

- Generalization of the ETSI-standard mobile device architecture for any radio equipment consisting of multiple hardware platforms in Section III
- Instantiation of the generalized framework, specifically for the SDRAN data plane in Section IV
- Demonstrating the suitability of the proposed framework for the SDRAN data plane through proof-of-concept (PoC) system implementation in Section V

The key contribution of the proposed framework is to resolve the problem of software portability in the SDRAN data plane such that the proposed framework provides platform-independent programmability. Ultimately, the proposed data plane framework makes it possible for the SDRAN data plane, which employs a C-RAN consisting of various heterogeneous BBUs for real-time processing of the L1/L2 data plane, to be reconfigured through open sources provided by third-party software developers. In this paper, it is assumed that the SDRAN employs the C-RAN, which provides efficient use of computational resources and collaborative processing among BBUs in the infrastructure layer. It is noteworthy that platform-independent software management enables users to exploit a high-level reconfiguration of the SDRAN data plane. In Section V, the feasibility of platform-independent software management provided by the proposed data plane framework is verified through a PoC system with numerical results obtained from various experimental tests.

The proposed data plane framework exploits two key advantages of the ETSI-standard mobile device architecture. First, a radio virtual machine (RVM) has been adopted for the execution of a given platform-independent software with a platform-specific RVM setup for each target hardware platform in the SDRAN infrastructure layer [14], [15]. Second, the double-layered structure for software management has been adopted for high-level abstraction of configuration-related operations in the SDRAN data plane [12], [13]. In addition, the proposed data plane framework is advantageous for accomplishing real-time processing requirements of the softwarized RAN because the proposed framework is based on a mobile device architecture that operates in a real-time domain.

II. OVERVIEW OF SOFTWARIZED RAN SOLUTIONS

This section briefly summarizes the SDRAN and C-RAN as typical softwarized RAN solutions. This section also introduces some previous works related to SDRAN and/or C-RAN to observe inherent problems with conventional approaches.

A. SDRAN

The 5G signal environment is associated with densely deployed small cells and heterogeneous wireless networks. As a result, a need arises for efficient wireless access management in a RAN, including multi-radio access technology (multi-RAT) management, mobility management, and inter-cell interference handling. Under these circumstances, active research on SDRAN has been performed to apply the concept of SDN in wired networks, i.e., the decoupling of the data plane and control plane [3], [4], to wireless networks.

Figure 1(a) shows a conceptual diagram of a conventional SDRAN consisting of application, control, and infrastructure layers. The infrastructure layer, which generally includes multiple base stations, performs the entire L1/L2 data plane processing in the SDRAN according to the control decisions in the control layer. Other than the data plane processing itself, the infrastructure layer further provides two core functionalities required for data plane reconfiguration as follows. First, it reports context information (e.g., mobility, inter-cell interference level, traffic load) and resource utilization status to the control layer; second, it executes the control command related to the data plane reconfiguration transferred from the control layer.

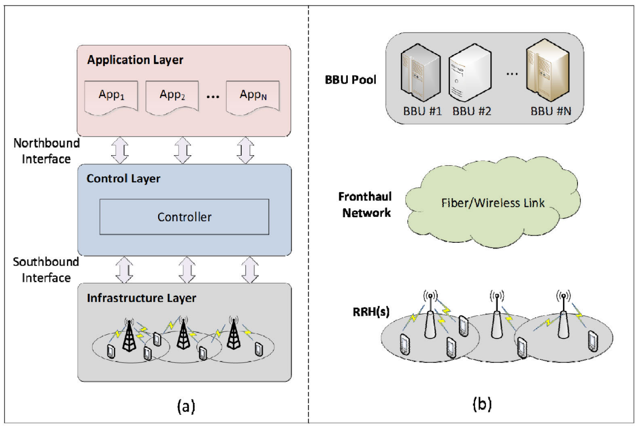


FIGURE 1. Conceptual diagram of SDRAN and C-RAN as typical solutions for enabling the softwarized RAN: (a) SDRAN architecture; (b) C-RAN architecture.

The control layer, located between the application and infrastructure layers, uses northbound and southbound interfaces to interact with the application and infrastructure layers, respectively. Through the southbound interface, the controller in the control layer can access each base station in the infrastructure layer. Through the northbound interface, users are provided with application programming interfaces (APIs) required to control the infrastructure layer. Using the APIs, users can receive context information and resource utilization status for use in a corresponding application.

The application layer supports various applications for optimizing the infrastructure of a given RAN, such as multi-RAT management, mobility management, and inter-cell interference handling.

B. C-RAN

A C-RAN centralizes BBUs to provide cloud computing with efficient use of computational resources. Using programmable hardware platforms in a given BBU pool, a C-RAN ensures remarkable flexibility in terms of network upgrades and network maintenance [8]. Figure 1(b) is a conceptual diagram of a conventional C-RAN, which consists of a BBU pool, fronthaul networks, and remote radio heads (RRHs). The BBU pool consisting of multiple BBUs performs the baseband processing required mainly for modem functionalities. Each BBU is generally composed of various processors, such as GPPs, DSPs, FPGAs, and ASICs, which differ depending on the vendor. The fronthaul determines the interconnection between the BBU pool and the RRHs. The common public radio interface (CPRI) is a typical radio interface protocol used for data transmission between the BBU pool and the RRHs.

C. RELATED WORKS

SoftRAN [16] provides a software-defined centralized control plane of an SDRAN by applying the SDN concept, i.e., decoupling the control and data planes from each other, to the RAN domain for the first time. SoftRAN abstracts multiple base stations located at different geographical regions into a single virtual big-base station to control each from a central locale. However, as data plane programmability has not been considered in SoftRAN, software reconfiguration is only confined to the control plane.

FlexRAN [17], the first developed open-source SDRAN platform, is another concept for SDRAN design. With the programmability provided by FlexRAN, in addition to RAN control applications, control functions in a given controller can be updated as well. Unfortunately, the programmability provided by FlexRAN is focused only on the control plane. Although FlexRAN does not provide data plane programmability, it has been claimed [17] that FlexRAN can be complementarily used with another work that provides data plane programmability, such as the data plane framework proposed in this paper.

Recently, starting from [16], [17], there has been remarkable progress in terms of control plane programmability as presented in [18], [19]. Nevertheless, recent state-of-the-art literature mainly focuses on control plane programmability, rather than data plane programmability, in an SDRAN.

OpenRadio [5] and PRAN [6] are typical works that provide programmability for the SDRAN data plane. OpenRadio provides a modular and declarative programming interface that allows the modularization of L1/L2 functional blocks of a given communication protocol stack to support communication standard evolution. Consequently, OpenRadio can support data plane programmability. However, its programming interface is platform-specific only for a particular DSP, which implies that the software for data plane processing of OpenRadio cannot be ported on other hardware platforms. Meanwhile, PRAN also provides data plane programmability that allows software reconfiguration of a given SDRAN by switching the data paths of L1/L2 processing at the central controller. However, without the multi-target back-end compiler [6], which is applicable to various heterogeneous platforms, data plane programmability of PRAN as well as OpenRadio is limited to specific platforms.

From the above discussions, the software portability issue should be resolved to provide complete programmability that allows software reconfiguration of the SDRAN data plane with no limit on platforms. Consequently, the need arises for efficient compilation and execution of a given radio application (RA) code [12] for each of the various hardware platforms in the BBU pool of a given SDRAN [6]. Of course, when the RA code is given as a platform-specific executable code, there is no software portability issue. The main concern of this paper, however, is to consider more generic cases when the RA codes are distributed as platform-independent codes that can be provided by third-party software developers. More specifically, as will be discussed later in this paper, each RA code can be distributed as an open source once the software portability issue is resolved.

In previous works, such as OpenRadio [5] and PRAN [6] providing programmability for the data plane, the RA is coupled with the BBU hardware platform such that RA management becomes platform-specific as well. RA management includes installation/uninstallation and/or activation/deactivation of RAs, multi-RAT management, and data flow management for each RA. Consequently, when the BBU pool in the infrastructure layer of a given SDRAN consists of heterogeneous hardware platforms, RA management becomes extremely complicated in conventional OpenRadio and PRAN because the network operator should provide platform-specific RA management using their software interfaces, i.e., APIs from the central controller.

As will be described later, this paper proposes a novel data plane framework that resolves the software portability problem and provides a uniform way of managing the RA through the high-level abstraction of data plane configuration-related operations for each hardware platform in a given BBU pool. No other work has ever provided any specific method of resolving the SDRAN data plane programmability and software portability on multi-vendor hardware components.

III. GENERALIZATION OF ETSI-STANDARD MOBILE DEVICE ARCHITECTURE FOR A GENERIC RADIO EQUIPMENT CONSISTING OF MULTIPLE HARDWARE PLATFORMS

ETSI TC-RRS developed a standard architecture and related interfaces for a reconfigurable mobile device whose configuration is determined by a downloaded RA code [12]–[14]. The key objective of the ETSI-standard mobile device architecture is to provide efficient mobile device software reconfiguration with guaranteed software portability between the RA code and various hardware platforms.

Figure 2 illustrates ETSI-standard mobile device architecture components consisting of a communication services layer (CSL), radio control framework (RCF), unified radio application (URA), and radio platform. These four components are interconnected through interfaces as follows:

- Multiradio interface (MURI) for interconnecting CSL and RCF;
- Unified radio application interface (URAI) for interconnecting URA and RCF;

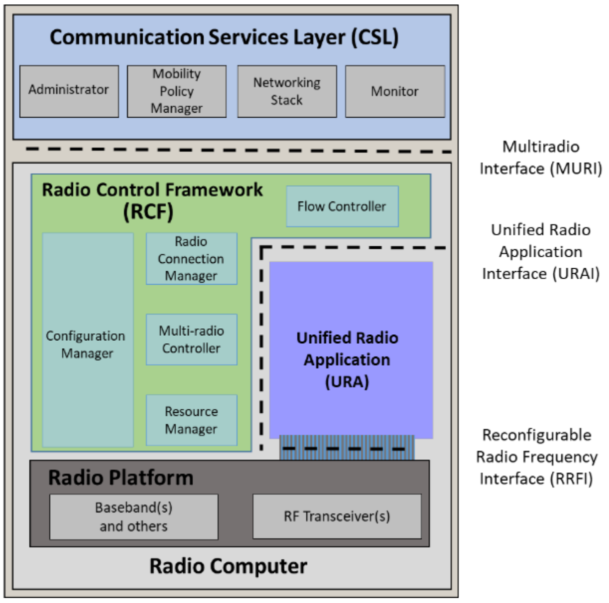


FIGURE 2. ETSI-standard mobile device architecture [12].

- Reconfigurable radio frequency interface (RRFI) for interconnecting URA and RF transceiver in radio platform.

In addition to the aforementioned three interfaces, the radio programming interface (RPI) [14] has been defined for the (third-party) RA provider to implement the RA code with. Using the ETSI-standard mobile device architecture shown in Figure 2, various RA codes can be downloaded onto a reconfigurable platform from a radio app store. As the framework shown in Figure 2 can be viewed as a computer that executes a given RA code using its own hardware platform on its own OS, it is denoted as a radio computer.

In the following three subsections, we explain how the ETSI-standard mobile device architecture shown in Figure 2 can be generalized in such a way that the framework is applicable to any radio equipment consisting of multiple hardware platforms. The generalized framework will then be instantiated specifically for the SDRAN data plane in the next section.

A. EXTENSION OF SINGLE RADIO COMPUTER INTO MULTIPLE RADIO COMPUTERS

Figure 3 shows a generic radio equipment consisting of multiple radio computers, each of which interacts with the CSL through the MURI via the routing entity. The key difference between the mobile device architecture and the generic radio equipment framework shown in Figure 3 is that only a single hardware platform is involved in the former, whereas multiple hardware platforms that may be provided by different vendors with different OSs, are encountered in the latter. Consequently, the generic radio equipment framework generally includes multiple radio computers, each of which might employ its own OS corresponding to its hardware platform.

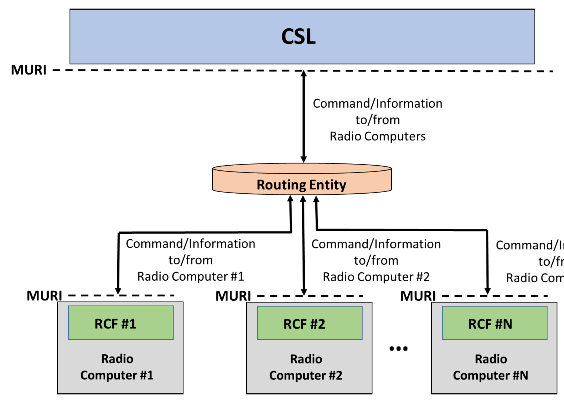

CSL
MURI
Command/Information to/from Radio Computers
Routing Entity
Command/Information to/from Radio Computer #1
Command/Information to/from Radio Computer #2
Command/Information to/from Radio Computer #N
RCF #1
Radio Computer #1
RCF #2
Radio Computer #2
RCF #N
Radio Computer #N

FIGURE 3. Block diagram of generic radio equipment consisting of multiple radio computers.

Because of the multiple radio computer architecture, the CSL should specify the radio computer ID as well as the command itself to be executed by the corresponding RCF for each radio computer. Thus, each CSL command shall be transferred to the corresponding RCF of the target radio computer via a routing entity.

B. MODIFICATION OF DOUBLE-LAYERED STRUCTURE FOR HIGH-LEVEL RA MANAGEMENT

This subsection explains how the double-layered structure consisting of the CSL and each RCF should be modified to achieve a uniform way of managing RAs executed on heterogeneous hardware platforms. We first briefly summarize the ETSI-standard double-layered structure developed for the RA management of mobile devices. Then, we suggest how the double-layered structure should be modified to be employed in generic radio equipment. Figure 4 illustrates the double-layered structure consisting of the CSL and RCF, which are interconnected through the MURI [12], [13].

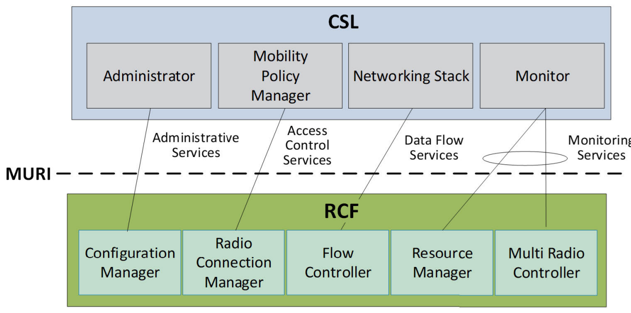


FIGURE 4. Modified double-layered structure for high-level RA management.

The CSL is an abstraction layer providing high-level management for RAs that are executed on a hardware platform. More specifically, a CSL abstracts each of the configuration-related operations performed on a RCF to provide a uniform approach to RA management. The software entities defined in the CSL include the administrator, mobility policy manager, networking stack, and monitor, all of which support multi-RAs. Functionalities of each software entity in the CSL are shown in Table 1(a).

The RCF is a software component that provides the functionalities required for RA management. More specifically, an RCF is a control framework that extends OS capabilities in terms of RA management. It performs software reconfiguration through the execution of CSL commands. In other words, any ordinary real-time OS with the functionalities of the RCF being implemented can be used as an OS of the proposed data plane framework, which is denoted as the Radio OS in this paper [12]. The software entities defined in the RCF include the configuration manager, radio connection manager, flow controller, resource manager, and multi-radio controller, all of which provide the execution environment for software reconfiguration. Functionalities of each software entity in the RCF are shown in Table 1(b).

The MURI, the interface between the CSL and RCF, provides the services summarized in Table 1(c). It is noteworthy that to apply the double-layered structure to generic radio equipment consisting of multiple hardware platforms, monitoring services should be included in addition to the three types of services provided by the original MURI (administrative, access control, and data flow services) as defined for the ETSI-standard mobile device architecture.

In mobile device applications, the RCF autonomously manages the computational and spectral resources according to the L1/L2 processing of the present RA. In generic radio equipment, however, each of the multiple RCFs on the corresponding hardware platform must report the context information and usages of its computational and spectral resources to the CSL monitor such that the CSL can provide central management of the RA for generic radio equipment.

The new monitoring services of the MURI enable efficient management of the context information and computational/spectral resource usages among the various hardware platforms in the generic radio equipment. As monitoring services are added in a MURI, the functionality for the resource manager and multi-radio controller to report the status of the computational and spectral resources to monitor in the CSL should be added correspondingly. Additionally, the monitor should present the status of the computational and spectral resources to the user. Functionalities to be added for multiple hardware platforms are shown in italics in Table 1.

C. GENERATION AND EXECUTION OF RA CODE

This subsection introduces the procedure of generating and executing the RA code using a radio library and an RVM, respectively, for generic radio equipment consisting of multiple radio computers. More specifically, an RA code is composed of functional blocks properly chosen from the radio library, and heterogeneous hardware platforms of the radio equipment are abstracted by the RVM. It is claimed in this subsection that the RVM implementation should be

TABLE 1. (a). Functionalities of each software entity in the CSL. (b). Functionalities of each software entity in the RCF. (c). Services of the MURI.

	Entities	Functionalities
Communication Services Layer (CSL)	Administrator	• requests (un)installation of a RA • requests creation or deletion of a RA instance • provides information about the spectral and computational requirements for each RA and status of each RA
	Mobility Policy Manager	• requests (de)activation of the RA • monitors the radio environments and hardware platform capabilities
	Networking Stack	• sends/receives backhaul/fronthaul network data to/from flow controller
	Monitor	• presents context information such as received signal strength, interference level, etc. • *presents status of computational/spectral resource management*

(a)

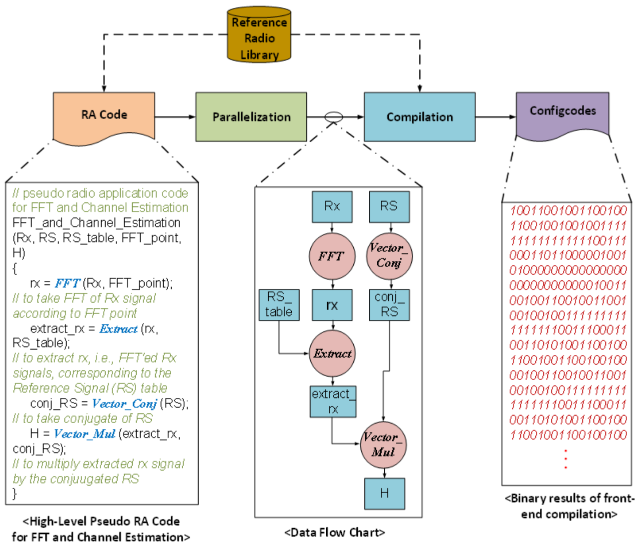
	Entities	Functionalities
Radio Control Framework (RCF)	Configuration Manager	• (un)installs of the RA • creates/deletes instances of the RA • manages the access to the radio parameters of the RA
	Radio Connection Manager	• activates/deactivates the RA according to user request
	Flow Controller	• manages data flows
	Resource Manager	• manages computational resources in order to share them among simultaneously active RAs, and guarantees their real-time execution • *reports computational resource status to monitor*
	Multi Radio Controller	• schedules the requests for spectral resources issued by concurrently executing RAs • detects and manages the interoperability problems among the concurrently executed RAs • *reports spectral resource status to monitor*

(b)

	Services	Services
Multiradio Interface (MURI)	Administrative Services	• used for some configuration applications, i.e., administrator, to (un)install a new RA into the corresponding hardware platform(s), to create/delete an instance of the RA
	Access Control Services	• used by the mobility policy manager to maintain user policies and preferences
		related to the usage of different RATs
	Data Flow Services	• used by the networking stack to send/receive backhaul/fronthaul network data
	Monitoring Services	• *used by the monitor to present context information and status of computational/spectral resource management among hardware platform s*

(c)

provided specifically for each radio computer to apply the RVM concept of the ETSI-standard mobile device to the case of multiple radio computers.



```
// pseudo radio application code
for FFT and Channel Estimation
FFT_and_Channel_Estimation
(Rx, RS, RS_table, FFT_point,
H)
{
    rx = FFT (Rx, FFT_point);
// to take FFT of Rx signal
according to FFT point
    extract_rx = Extract (rx,
RS_table);
// to extract rx, i.e., FFT'ed Rx
signals, corresponding to the
Reference Signal (RS) table
    conj_RS = Vector_Conj (RS);
// to take conjugate of RS
    H = Vector_Mul (extract_rx,
conj_RS);
// to multiply extracted rx signal
by the conjuugated RS
}
```

```
10011001001100100
11001001001001111
11111111111100111
00011011000001001
01000000000000000
00000000000010011
00100110010011001
00100100111111111
11111110011100011
00110101001100100
11001001100100100
00100110010011001
00100100111111111
11111110011100011
00110101001100100
11001001100100100
        ⋮
```

FIGURE 5. Front-end compilation using reference radio library.

1) RADIO LIBRARY

The radio library consists of a set of standard functional blocks that serve as benchmarks for the implementation of RA codes. The radio library is classified into two types: i) the reference radio library for front-end compilation of a given RA code and ii) the native radio library for execution of the front-end compilation result on a target platform. The front-end compilation result of an RA code is denoted as "configcodes" in this paper [14].

- The reference radio library provides a normative description of each corresponding standard functional block. For a third-party software developer to generate an RA code using the reference radio library, the normative description shall include not only the contents of the functional block itself but also additional information, such as the requirement for spectral/computational resources for the functionality of the corresponding standard functional block. The normative description might be provided in a high-level language, e.g., C, C++, or Java. The reference radio library, which is platform-independent, is used for the front-end compilation of a given RA code.
- The native radio library provides a platform-specific description of each corresponding standard functional block. The native radio library can be generated from the reference radio library by mapping the specific characteristics of a target hardware platform (e.g., dedicated hardware accelerators and programmable processors) onto the reference radio library. The native radio library is used for the execution of the front-end compilation result, i.e., configcodes.

2) FRONT-END COMPILATION USING REFERENCE RADIO LIBRARY

Figure 5 shows the procedure for the front-end compilation of an RA code using the reference radio library that generates

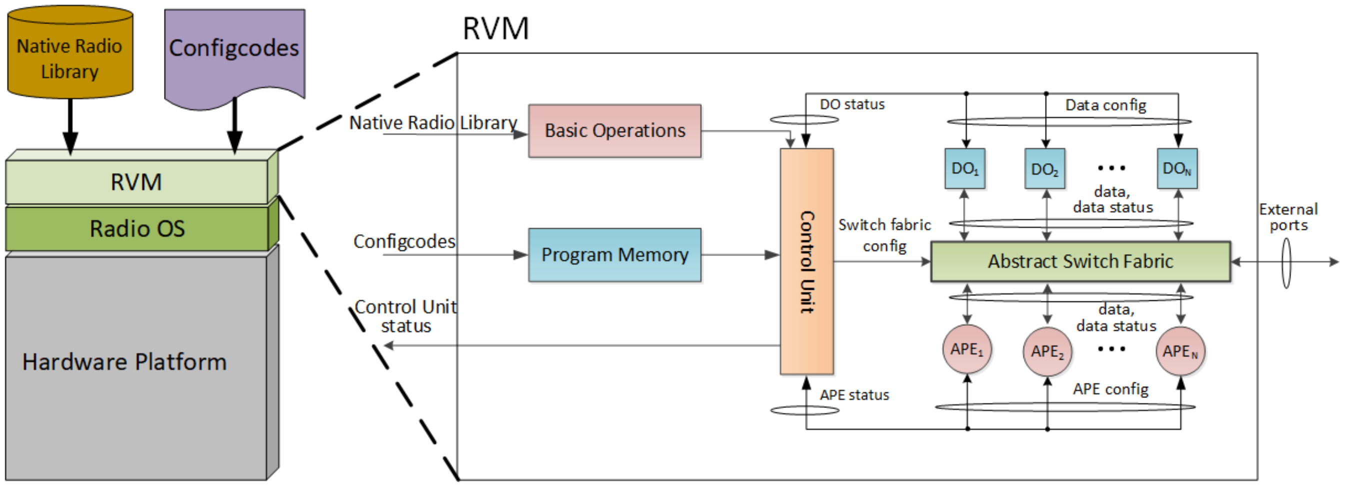


FIGURE 6. Execution of configcodes using radio computer-specific RVM and native radio library.

configcodes. The RA code, developed from the reference radio library, is converted into binary configcodes through the parallelization and compilation procedure. As configcodes are generated through front-end compilations with the platform-independent reference radio library, configcodes are also platform independent.

Figure 5 explicitly shows how a high-level pseudo RA code is front-end-compiled utilizing fast Fourier transform (FFT) and channel estimation. The functional blocks shown as ***FFT***, ***Extract***, ***Vector_Conj***, and ***Vector_Mul*** in pseudo RA code, shown at the lower left-hand side of Figure 5, are provided by the reference radio library. The RA code is described as a data flow chart through the parallelization procedure, as shown in the lower middle portion of Figure 5. Finally, through the compilation procedure, the data flow chart is converted into a binary representation according to a predefined binary format [14].

3) EXECUTION OF CONFIGCODES USING NATIVE RADIO LIBRARY AND RVM

Figure 6 shows the conceptual block diagram describing the execution of configcodes on a target platform that includes RVM implementation as an interpreter or just-in-time compiler. The Radio OS shown in Figure 6 is a real-time OS, on which given configcodes are executed using the resources of a given hardware platform. The RVM is an abstract machine that abstracts heterogeneous platforms to execute all algorithms included in the given configcodes on various types of physical resources. The abstract resources of the RVM include, as shown in Figure 6, an abstract processing element (APE), a data object (DO), and an abstract switch fabric (ASF) for abstracting computational resources, memory resources, and switch resources, respectively. This means that APE and DO correspond to an operator and memory, respectively, while ASF properly interconnects each APE and DO.

The "basic operations" and "program memory" blocks shown in Figure 6 include operators downloaded from the native radio library and store the configcodes, respectively, while the "control unit" initializes and sets up all instructions of the APE, DO, and ASF by decoding the configcodes stored in the "program memory". To execute the platform-independent configcodes on each target platform, an RVM is set up in accordance with the target hardware platform, which is represented by the platform-specific native radio library. Consequently, RVM implementation should be provided specifically for each radio computer when the target radio equipment consists of multiple radio computers.

To efficiently execute the configcodes on a target platform, the RVM implementation shown in Figure 6 is based on a data-driven structure. This provides minimum latency in the execution of the configcodes, of which the binary description represents the data flow chart shown in Figure 5. This indicates that operation of an APE is automatically executed as soon as the data status of the corresponding DO changes from "empty" to "full". Furthermore, all operations parallelized during the front-end compilation are executed concurrently. The DO, ASF, and APE associated with a given operation share the data status and data. In addition, the status of the DO and APE is reported to the "control unit." A combination of RVMs, i.e., the vertical/horizontal scaling of RVMs, can provide an RVM extension [14]. In that case, each APE may be composed of multiple RVMs.

It is important that the RVM implementation is included at each hardware platform only when the computational resources at each hardware platform allow for it. Otherwise, the back-end compilation should be provided separately, e.g., as a cloud service provided by the hardware platform manufacturers. In this case, the back-end compiler generates the executable code for each target platform in a cloud using the configcodes and the native radio library corresponding to the target platform. Users can choose between these two methods of obtaining executable codes in such a way that their expenditures can be fully optimized, thus providing flexibility.

IV. PROPOSED DATA PLANE FRAMEWORK FOR SDRAN

Using the generalized framework shown in the preceding section, this section presents how the generic framework can be instantiated for the specific needs of SDRAN data plane, i.e., software portability and programmability. Figure 7 illustrates the scope of the data plane framework proposed in this section within the RAN architecture. To support software-based centralized network management, the RAN architecture adopting the SDRAN consists of the application, control, and infrastructure layers [3]. Furthermore, to support efficient use of computational resources together with collaborative processing and efficient network upgrade/maintenance, the infrastructure layer of the RAN architecture includes the C-RAN architecture, which consists of the BBU pool [8]. The controller in the SDRAN control layer might include various software components, such as those for RA management, radio resource management, and quality of service (QoS) management [16]. In contrast, the proposed data plane framework concerns only the software component for RA management that provides software reconfiguration management of the SDRAN data plane, as shown in Figure 7. In particular, this means that the proposed data plane framework concerns only data plane processing in the BBU pool and functionalities of RA management in the controller. It is important that the applications contained in the application layer shown in Figure 7 are differentiated from the RA executed on the corresponding BBU in the infrastructure layer.

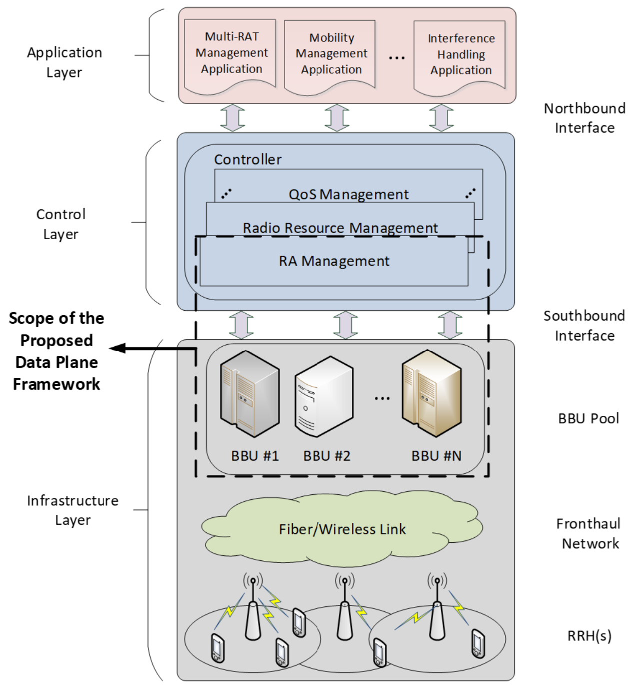


FIGURE 7. Scope of the proposed data plane framework within RAN architecture.

Now, we present the proposed data plane framework that includes the radio library, RVM, and double-layered structure for multiple radio computers explained in the preceding section. Discussions are focused on its capability to resolve the software portability issue and provide high-level abstraction for RA management. Then, we present typical use cases of the proposed data plane framework.

A. PROPOSED DATA PLANE FRAMEWORK

Figure 8 illustrates the proposed data plane framework consisting of two main parts: i) multiple radio computers, each of which performs L1/L2 data plane processing, and ii) CSL, which provides high-level abstraction for RA management.

As shown in Figure 8, each radio computer employs its own Radio OS and RCF to execute a given RA on its hardware platform. Using the RVM and native radio library described in the preceding section, each radio computer can execute a given RA with the target platform-specific RVM setup. Although the RVM is specific to each radio computer, as shown in Figure 8, platform-specific implementation of each RVM does not lead to any additional burden on RVM providers. This is because the difference in RVM implementation at each radio computer simply comes from different machine instructions used in the RVM implementation procedure for each Radio OS. It is envisioned that the RVM is provided by the hardware platform vendor together with the Radio OS.

The CSL, which composes the double-layered structure together with the RCF as described in the preceding section, provides users with a uniform way of managing the RA. As the CSL abstracts the operation of the RCF for each radio computer, it is possible for users to exploit high-level RA management. The CSL is interconnected to each of the RCFs of the corresponding radio computer through the MURI, which is a part of the southbound interface, as shown in Figure 7. Table 1(c) summarizes the services provided by the MURI with the specifications of corresponding software entities of the CSL and RCF involved in each of the MURI services.

As mentioned in the preceding section, the CSL interacts with each of the RCFs via a routing entity, the detailed operations of which are not included in Figure 8. Furthermore, the inter-communication among multiple radio computers is outside the scope of this paper. We believe that the inter-BBU connection approach used in the case of C-RAN can be adopted in multiple radio computers as well [8], [20].

Using the proposed data plane framework shown in Figure 8, we explain below how the problem of software portability can be resolved and how the proposed data plane framework can provide high-level abstraction for RA management.

1) RESOLUTION OF SOFTWARE PORTABILITY PROBLEM

Depending on the desired levels of programmability, flexibility, power consumption, and processing speed, the hardware platform of each BBU generally consists of different types of

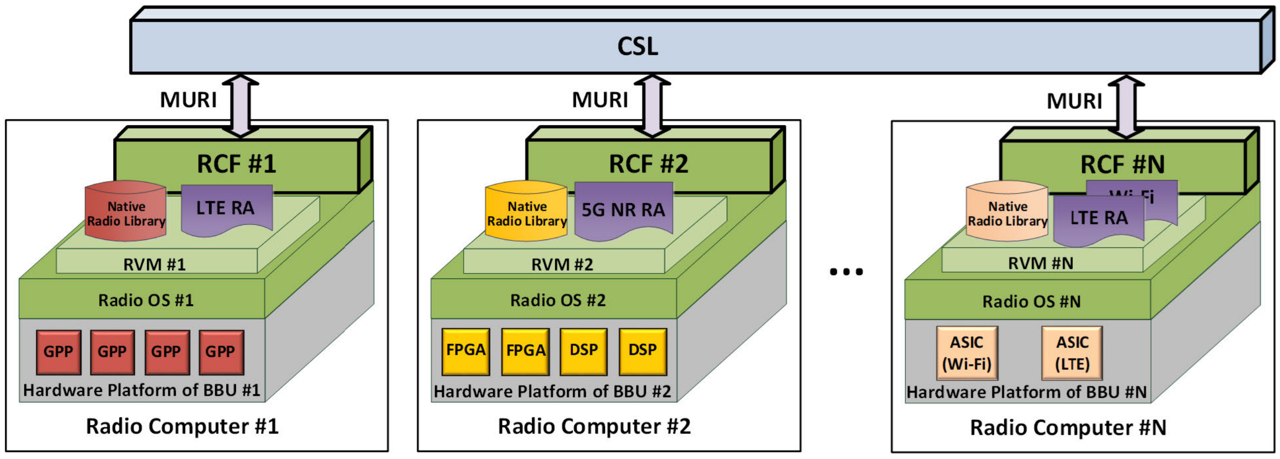


FIGURE 8. Proposed data plane framework.

processors. In Figure 8, for instance, the hardware platform of BBU #1 consists of only GPPs, while that of BBU #N consists mainly of ASICs. Consequently, without a separate back-end compiler for each of the various heterogeneous hardware platforms, the problem of software portability is inevitable.

To resolve software portability problems arising due to heterogeneous hardware platforms, an RVM is introduced between each RA and the corresponding Radio OS running on its BBU hardware platform, as shown in Figure 8. As mentioned in the previous section, each RA shown in Figure 8 is given in the form of configcodes after front-end compilation. As an RVM abstracts physical resources in each hardware platform into virtual resources, the RVM guarantees that the RA code, i.e., platform-independent configcodes, can be ported onto any type of hardware platform. During RVM execution, the native radio library is used to interpret the configcodes in a platform-specific way. Consequently, execution of an RVM provides joint optimization of both software (i.e., a given RA code) and hardware (i.e., the target hardware platform of the designated BBU). The software communication architecture (SCA) [21], a conventional approach suggested for resolving the software portability problem, separates the software and hardware using middleware. As a result, joint optimization of software and hardware cannot be achieved [22]. In addition, borrowing the data-driven structure of RVM implementation, system latency and hardware platform resource usage can be reduced by eliminating the resource allocation and control overhead, respectively. The data-driven structure is a key enabler for the proposed RVM approach to outperform other ETSI-standard virtualization methods based on a virtual machine (VM) or container [23], as well as the conventional SCA-based method.

2) HIGH-LEVEL ABSTRACTION FOR RA MANAGEMENT

Using the northbound interface, third-party software developers can provide various applications related to wireless access, such as multi-RAT management, mobility management, and interference handling. For the implementation of SDRAN applications, it is often necessary for software developers to consider special requirements related to RA management. For example, for implementation of a multi-RAT management application, when a new (group of) RA(s) is activated, another (group of) RA(s) might have to be deactivated. Similarly, for an interference handling application, the transmit power of a certain (group of) RA(s) processed in a particular (group of) BBU(s) might have to be properly reduced or enhanced. In addition to these two simple examples, various situations arise where SDRAN applications should control the RAs processed in various BBUs in a given BBU pool. The problem here is how to execute the command regarding RA management from the application layer on each of the heterogeneous BBU platforms, which may employ different OSs, as shown in Figure 8. Using the proposed data plane framework, the application layer can simply transfer the desired command to the CSL without having to know the hardware characteristics of each BBU because the CSL would send the command to the target RCF(s) prepared as a part of the Radio OS for each BBU. In other words, using the double-layered structure of a CSL and RCFs, the application layer does not have to consider various heterogeneous BBU hardware platforms.

As the RA management required in a given application is executed on the SDRAN server, corresponding commands are transferred from the application layer to the CSL through the northbound interface. Thereafter, the corresponding entity(ies) in the CSL transfer(s) the command(s) to the target RCF(s) through MURI. Finally, the corresponding entity(ies) in the designated RCF(s) execute(s) the command(s) for the corresponding BBU hardware platform to be reconfigured in accordance with the command(s). In particular, this means that the CSL can be viewed as an abstraction layer for operation(s) of the multiple RCFs according to the command(s) transferred from the application layer. Consequently, the proposed data plane framework provides users with a uniform way of managing the RAs through the double-layered structure consisting of a CSL and RCF.

It is noteworthy that each RCF in the proposed framework shown in Figure 8 is prepared as a part of the Radio OS of the corresponding specific BBU hardware platform, while the command regarding the RA management is issued from the application layer without considering each BBU hardware platform.

B. USE CASES

In this subsection, we present two typical use cases for the application of the proposed data plane framework. Each use case is focused on the capabilities of the proposed data plane framework with respect to the resolution of the software portability problem and high-level abstraction for RA management.

1) RECONFIGURATION OF SDRAN DATA PLANE THROUGH OPEN-SOURCE RA DOWNLOAD

As the open-source community such as OPNFV [24] has become very invigorated recently, CN reconfiguration can now be achieved by downloading the open-source VNF application from the OPNFV server and executing it on a target server. The key reason why the NFV open-source community can be so successful is that no software portability issues exist with the VNF applications as they are executed, in general, on industry standard servers [9]–[11], [25].

As the software portability issue in the SDRAN data plane is resolved using the proposed data plane framework that employs the radio library and an RVM, network operators can now reconfigure the SDRAN data plane by downloading the desired RAs distributed as open source. In short, using the proposed SDRAN data plane framework, the SDRAN data plane can be reconfigured with open-source RAs as in the CN reconfiguration achieved with the open-source VNF application.

More specifically, third-party software developers can upload their RA codes, e.g., 5G new radio (NR), on an open-source community server in the form of platform-independent configcodes obtained as a result of front-end compilation. Then, network operators download the platform-independent configcodes of the RA code of the 5G NR onto their BBU hardware platform and execute it on a desired hardware platform. While the BBU hardware platforms can be heterogeneous, the RVM at each BBU provides platform-specific execution of the RA code using the platform-specific native radio library. Although the RVM is equipped within each of the BBU platforms in Figure 8, back-end compilation could be provided outside the RAN in a cloud, as mentioned in the preceding section. The key part of the proposed data plane framework is that network operators can achieve SDRAN data plane reconfiguration with open-source RAs using the RVM and native radio library optimized to each of the BBU hardware platforms.

Borrowing the complete resolution of the software portability problem in the SDRAN data plane, we can exploit the merits of open-source RAs as in the case of NFV-based CN,

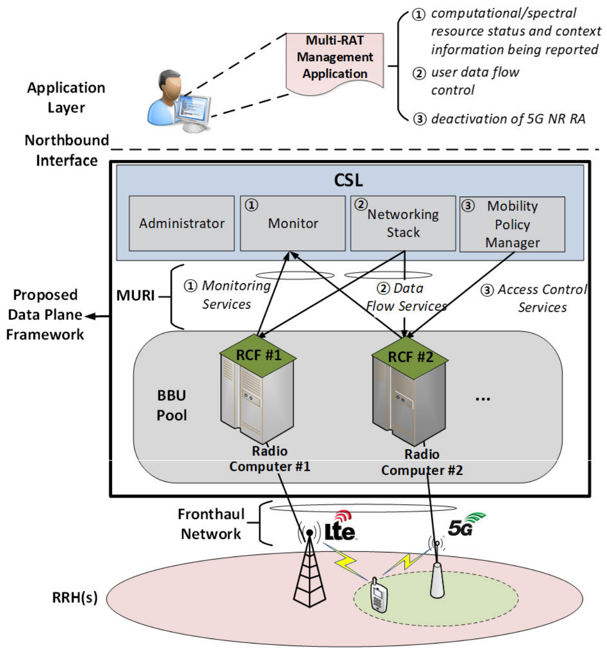


FIGURE 9. Conceptual diagram of short-term multi-RAT management procedure using the proposed data plane framework.

which will ultimately prevent data plane reconfiguration from being limited to a specific (group of) hardware platform(s).

2) HIGH-LEVEL ABSTRACTION FOR MULTI-RAT MANAGEMENT IN HETEROGENEOUS NETWORKS

Current wireless networks can be represented as a heterogeneous group of various RATs, such as GSM, WCDMA, LTE, 5G NR, and Wi-Fi [26]. Thus, network operators need to optimize their networks using multi-RAT management that can be classified into two categories: short-term and long-term management. A typical example of the former is one that switches off a 5G NR small cell overlaid on an LTE macro cell to save the energy of the 5G NR small cell in non-standalone 5G networks [27], [28]. A typical example of the latter is GSM spectrum refarming, with which operators can assign the GSM spectrum to LTE and/or 5G NR for more efficient spectrum utilization. GSM spectrum refarming has been developed to address the recent decline in GSM spectrum usage [29].

As the procedure of long-term multi-RAT management is quite well known [29], this subsection is focused on short-term management. Figure 9 illustrates a conceptual diagram showing the procedure of short-term multi-RAT management using the proposed data plane framework. As shown in Figure 9, assume that Radio Computer #1 executes the LTE RA, providing LTE macro cell coverage to user equipment (UE) as a master node, while Radio Computer #2 executes the 5G NR RA, providing the 5G NR cell coverage to the same UE as a secondary node. It is also assumed that the UE considered in this discussion supports dual connectivity, including both LTE and 5G NR RATs [27]. In short-term multi-RAT management, where the 5G NR small cell is to be switched off according to the reduction of the traffic load, the proposed data plane framework equipped with the double-layered structure provides high-level abstraction for data plane configuration-related operations as follows. It is noteworthy that the short-term multi-RAT management explained in this subsection only concerns data plane reconfiguration and not control plane reconfiguration involving radio resource control (RRC) connections and/or RRC signaling.

A network operator first makes a decision to switch off the 5G NR RA after checking the report on the context information related to the traffic load and computational/spectral resources status received at the CSL monitor from each BBU platform of both Radio Computers #1 and #2. This report is transferred from the corresponding RCF entities of Radio Computers #1 and #2 via the monitoring services in MURI. Then, the UE, which is mainly provided with data traffic by the 5G NR small cell, should be served only with the data traffic from the LTE macro cell. Consequently, the network operator transfers a command for removing data flow of the 5G NR small cell and adding data flow of the LTE macro cell to the flow controller in the RCF of Radio Computer #2 and Radio Computer #1, respectively, from the networking stack in the CSL via flow control services in the MURI. Finally, the network operator can switch off the 5G NR small cell by deactivating the 5G NR RA executed on Radio Computer #2 using the mobility policy manager in the CSL via access control services in the MURI.

As explained above, the multi-RAT application, although it is processed in the application layer, requires reconfiguration of the corresponding BBU hardware platforms in the infrastructure layer as well as reconfiguration of the controller in the control layer. The key aspect provided by the proposed data plane framework is that whenever the application layer needs to control the data plane, the application layer communicates with the corresponding entities within the CSL rather than the individual BBU hardware platforms. Consequently, users can manage data plane processing of RAs at each BBU hardware platform through the CSL, of which the details can be summarized in three steps, as follows. First, using the monitor in the CSL, the computational/spectral resource status and context information can be reported. Second, using the networking stack in the CSL, the data flow of the 5G NR cell can be moved to the LTE macro cell. Third, using the mobility policy manager in the CSL, the 5G small cell can be switched off. In short, with the double-layered structure of the proposed data plane framework, reconfiguration of the SDRAN data plane can be achieved without considering all the different characteristics of each of the heterogeneous hardware platforms.

V. IMPLEMENTATION OF PROOF-OF-CONCEPT SYSTEM

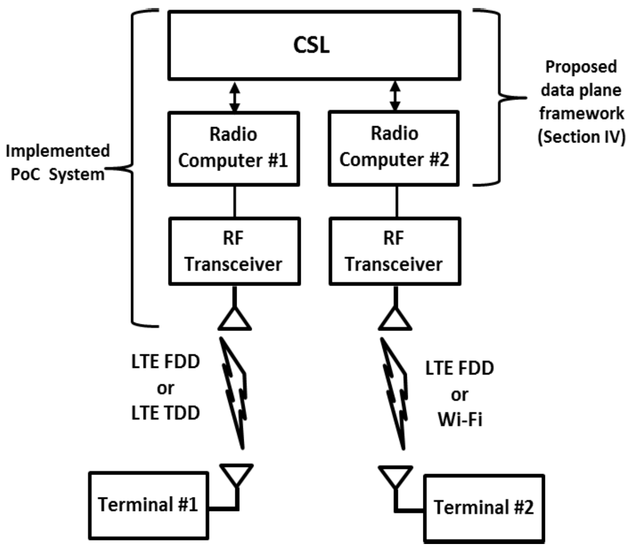
This section implements a PoC system with the proposed data plane framework introduced in Figure 8. Figure 10(a) presents a block diagram of the experimental environment, which includes the implemented PoC system consisting of RF transceivers and antennas as well as the proposed data plane framework. Figure 10(b) presents a photograph of the experimental environment corresponding to the block diagram shown in Figure 10(a). The PoC system includes two radio computers, Radio Computer #1 and Radio Computer #2, which are controlled using the double-layered structure consisting of a CSL and each of the two RCFs, as introduced in Section III. B and Section IV. A. Each of the two radio computers emulates a base station system communicating with Terminal #1 and Terminal #2, respectively, as shown in Figure 10. For the PoC system, we implemented RA codes for three RATs: (1) LTE Frequency Division Duplexing (FDD), (2) LTE Time Division Duplexing (TDD), and (3) Wi-Fi. The RA codes for (1) and (2) support all the physical channels of the 3rd-generation partnership project (3GPP) Release 12, while that for (3) complies with the IEEE 802.11 ac standards. Table 2 shows the detailed specifications of each RA.

TABLE 2. System parameters for each RA.

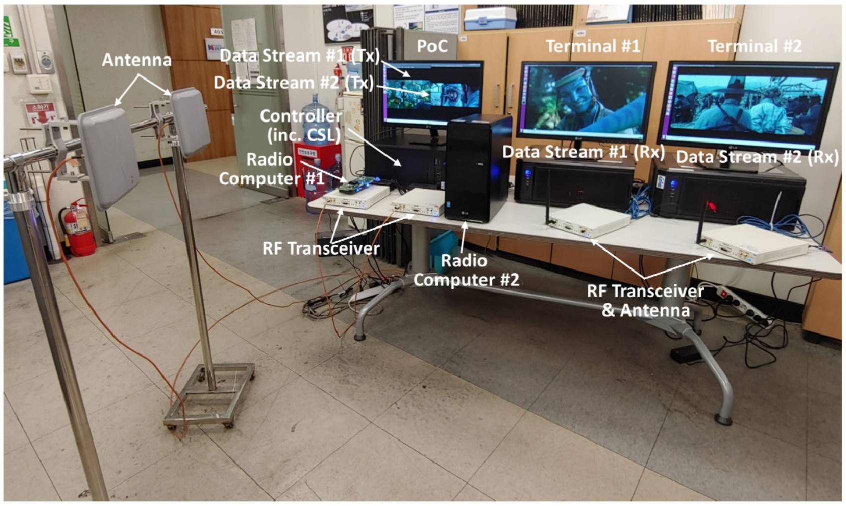
	LTE FDD	LTE TDD	Wi-Fi
Waveform standard	3GPP Release 12	3GPP Release 12	IEEE 802.11 ac
Center frequency	2.12 GHz	2.65 GHz	5.18 GHz
Bandwidth	20 MHz	20 MHz	20 MHz
Modulation	64 QAM	64 QAM	256 QAM
Channel coding	Turbo code (code rate: 0.75)	Turbo code (code rate: 0.75)	Convolutional code (code rate: 5/6)
Peak data rate	55.4976 Mbps	27.8768 Mbps	55.4 Mbps

In the experimental environment shown in Figure 10, Radio Computer #1 sends a dedicated video data stream to Terminal #1 using LTE FDD or LTE TDD, whereas Radio Computer #2 uses LTE FDD or Wi-Fi to transmit another dedicated video data stream to Terminal #2. Both Terminals #1 and #2 employ a CPU/GPU-based hardware platform that supports all of the abovementioned RAs, i.e., LTE FDD, LTE TDD, and Wi-Fi [30].

The key to the implementation in Figure 10 is that the PoC system employs a double-layered structure consisting of a CSL and RCF to perform RA management, as introduced in Figure 4. In this section, we demonstrate the merit of the double-layered structure, which provides high-level RA management. Based on the numerical results of the implemented system, we verify that RA management can be achieved in a uniform way using the double-layered structure with a negligible overhead. The performance of the proposed system is compared to that of an ordinary single-layered structure. In more detail, RA management in the single-layered structure is controlled by a controller, and the corresponding control command is directly sent to each radio computer without using the RCF. As explained later in this section, the hardware platform for Radio Computer #1 consists solely of a DSP,



(a)



(b)

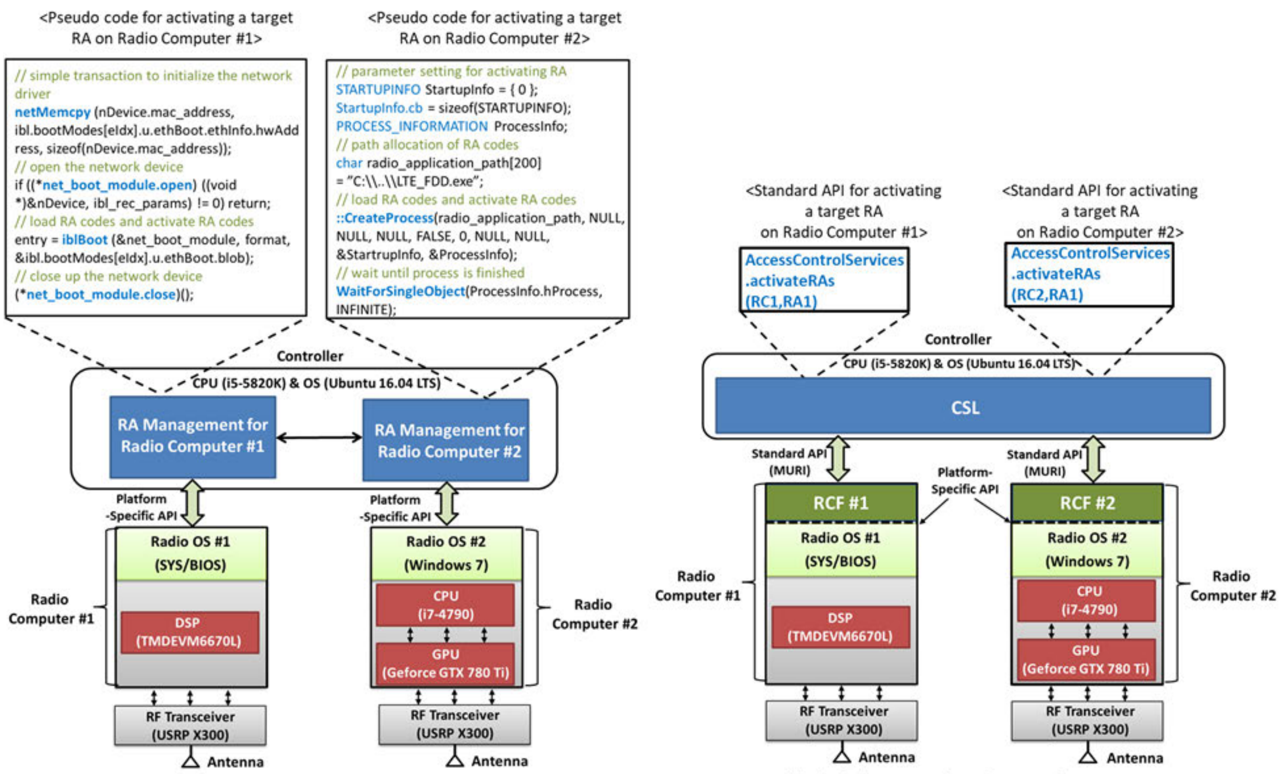
FIGURE 10. (a) Block diagram of the experimental environment. (b) Photograph of the experimental environment.

while that of Radio Computer #2 consists of a CPU and GPU, whose compilers are provided by the corresponding vendors. For simplicity but without loss of generality, we used an executable code for each RA code to determine the configurations of Radio Computer #1 and Radio Computer #2, implying that RVM was not implemented in our PoC. Note that the RVM for back-end compilation and the Radio OS shown in Figure 8 are supplied by the hardware platform vendors, as mentioned in the previous section.

A. IMPLEMENTATION

Figure 11 presents the block-diagram of the implemented PoC system that can employ either a single-layered or double-layered control structure, depending on whether the RA management command from the controller is sent directly or indirectly (by way of the corresponding RCF) to the designated radio computer, respectively. As shown in Figure 11, the implemented PoC system consists of the following three parts: (1) a controller for RA management, (2) a radio computer for L1/L2 data plane processing, and (3) an RF component for sending/receiving the RF signals. As the RF component is outside the scope of this paper, our discussion in this subsection focuses on the controller and radio computer, which are included in the proposed data plane framework shown in Figure 8.

Table 3 shows detailed specifications of the controller and two radio computers included in the implemented PoC system. The controller employs an Intel i5-5820K @ 3.3 GHz with 16 GB random access memory (RAM) and Ubuntu 16.04 LTS as its hardware platform and OS, respectively. A 1-gigabit ethernet (GbE) interface was used so that the controller could transfer the RA management command to


<Pseudo code for activating a target RA on Radio Computer #1>
<Pseudo code for activating a target RA on Radio Computer #2>
Controller
CPU (i5-5820K) & OS (Ubuntu 16.04 LTS)
RA Management for Radio Computer #1
RA Management for Radio Computer #2
Platform -Specific API
Platform -Specific API
Radio OS #1 (SYS/BIOS)
Radio OS #2 (Windows 7)
DSP (TMDEVM6670L)
CPU (i7-4790)
GPU (Geforce GTX 780 Ti)
Radio Computer #1
Radio Computer #2
RF Transceiver (USRP X300)
Antenna
<Standard API for activating a target RA on Radio Computer #1>
<Standard API for activating a target RA on Radio Computer #2>
Controller
CPU (i5-5820K) & OS (Ubuntu 16.04 LTS)
CSL
Standard API (MURI)
Platform-Specific API
Standard API (MURI)
RCF #1
RCF #2
Radio OS #1 (SYS/BIOS)
Radio OS #2 (Windows 7)
DSP (TMDEVM6670L)
CPU (i7-4790)
GPU (Geforce GTX 780 Ti)
Radio Computer #1
Radio Computer #2
RF Transceiver (USRP X300)
Antenna

```
// simple transaction to initialize the network
driver
netMemcpy (nDevice.mac_address,
ibl.bootModes[eIdx].u.ethBoot.ethInfo.hwAdd
ress, sizeof(nDevice.mac_address));
// open the network device
if ((*net_boot_module.open) ((void
*)&nDevice, ibl_rec_params) != 0) return;
// load RA codes and activate RA codes
entry = iblBoot (&net_boot_module, format,
&ibl.bootModes[eIdx].u.ethBoot.blob);
// close up the network device
(*net_boot_module.close)();
```

```
// parameter setting for activating RA
STARTUPINFO StartupInfo = { 0 };
StartupInfo.cb = sizeof(STARTUPINFO);
PROCESS_INFORMATION ProcessInfo;
// path allocation of RA codes
char radio_application_path[200]
= "C:\\..\\LTE_FDD.exe";
// load RA codes and activate RA codes
::CreateProcess(radio_application_path, NULL,
NULL, NULL, FALSE, 0, NULL, NULL,
&StartrupInfo, &ProcessInfo);
// wait until process is finished
WaitForSingleObject(ProcessInfo.hProcess,
INFINITE);
```

```
AccessControlServices
.activateRAs
(RC1,RA1)
```

```
AccessControlServices
.activateRAs
(RC2,RA1)
```

(a) Block diagram of implemented PoC system using single-layered structure

(b) Block diagram of implemented PoC system using (the proposed) double-layered structure

FIGURE 11. Block diagram of implemented PoC system.

TABLE 3. Specifications of the PoC system.

	Controller	Radio Computer #1	Radio Computer #2
Hardware Platform	CPU (i5-5820K @ 3.3 GHz)	DSP Evaluation Board (TMDEVM6670L)	① CPU (i7-4790 @ 3.6 GHz), ② GPU (Geforce GTX Titan)
OS	Ubuntu 16.04 LTS	SYS/BIOS version 6.46.5.55	Windows 7
Operating RAs	-	LTE FDD & LTE TDD	LTE FDD & Wi-Fi
Hardware Interface with Controller	-	1GbE	1GbE

the designated target radio computer and receive the corresponding response from the radio computer. Radio Computer #1 uses a DSP evaluation board (TMDEVM6670L) and SYS/BIOS version 6.46.5.55 as its hardware platform and Radio OS, respectively. It supports LTE TDD and LTE FDD RAs, as mentioned earlier. Radio Computer #2 uses both an Intel i7-4790 @ 3.6 GHz with 24 GB RAM and an NVIDIA Geforce GTX Titan as its hardware platform. Exploiting the large number of cores provided in the GPU, all the parallelizable L1/L2 data processing is performed using the GTX Titan GPU, while the sequential data processing and GPU control are performed using the i7-4790 CPU. Windows 7 was employed as the Radio OS of Radio Computer #2 to support LTE FDD and Wi-Fi, as mentioned earlier.

When the RA management command from the controller is transferred directly to the target radio computer as shown in Figure 11(a) (the single-layered structure), the RA management command should be provided with the platform-specific APIs corresponding to the target platform because the command should be executed directly on the target platform. In contrast, in the case of the double-layered structure shown in Figure 11(b), the RCF is prepared on top of the corresponding Radio OS. In this case, as the RCF executes the platform-specific APIs corresponding to the RA management command sent from the CSL, users do not have to consider platform-specific APIs for RA management.

Let us consider the pseudo codes given at the upper side of Figure 11. For simplicity but without loss of generality, we considered an RA management command for activating an RA. For the single-layered structure shown in Figure 11(a), as the DSP is used as the hardware platform in Radio Computer #1, the desired RA should be activated by booting the DSP using the "iblBoot" API from the DSP chip support library. Similarly, for Radio Computer #2, RA activation should be performed using "CreateProcess," a window API, as shown on the right-hand side of the upper part of

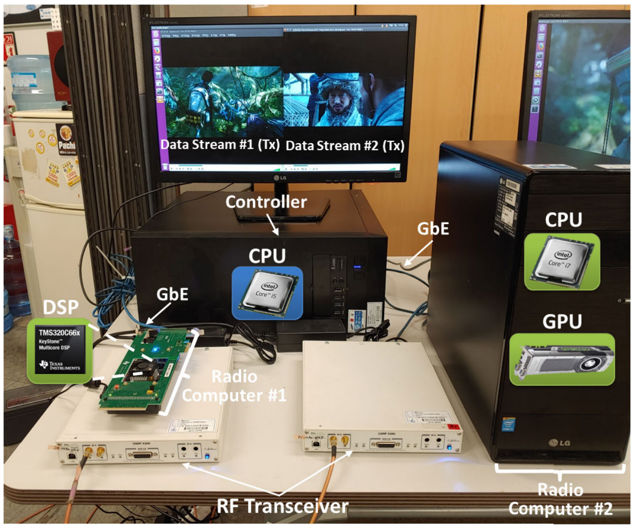


FIGURE 12. Photograph of implemented PoC system.

Figure 11(a). In contrast, for the case of the double-layered structure in Figure 11(b), the platform-specific API does not have to be considered at all. Activation of this RA can be achieved simply by calling a standard API (i.e., MURI introduced in Section III. B.), "AccessControlServices.activateRAs()," from the CSL. Not just the RA activation, but any RA management can be achieved by calling the corresponding MURI from the CSL, as shown in the upper part of Figure 11(b).

In short, by using the double-layered structure, users can achieve RA management in a uniform way with the standard API, i.e., MURI, without having to consider platform-specific APIs.

Figure 12 presents a photograph of the implemented PoC system consisting of RF parts and the proposed data plane framework. Figure 12 exactly corresponds to the block diagram shown in Figure 11(b), which employs the double-layered structure for controlling RA management. As mentioned earlier, the controller implemented with the CPU transfers the RA management command to the RCF of the designated radio computer via the 1GbE interface. For example, upon the arrival of the activation command, the RCF executes the command, allowing the target RA to become active on the platform of the designated radio computer. Then, the data stream is sent from the designated radio computer to the corresponding terminal using the RA activated by this command.

B. NUMERICAL RESULTS

In the previous subsection, it was observed that the proposed double-layered structure provides a high-level abstraction for RA management without having to consider platform-specific commands. To achieve this, however, the RCF should be implemented on the Radio OS of each radio computer, which may cause some additional overhead. Table 4 shows the computational resource utilization and memory footprint

TABLE 4. Comparisons of resource utilization status in single-layered and double-layered structures.

	Radio Computer #1		Radio Computer #2	
	wo RCF	w RCF	wo RCF	w RCF
Computational resource utilization (%)	7.2	9.7	3.8	5.1
Memory footprint (MB)	8.8	10.7	4.6	6.4

overhead according to the employment of the RCF at each radio computer involved in our PoC system. As the RCF for activating the desired RA is overlaid on top of the Radio OS (SYS/BIOS version 6.46.5.55), the required computational resources are increased by approximately 2.5%, and the memory footprint increased by approximately 1.9 MB for Radio Computer #1. Similarly, as the same RCF (i.e., the activation of the desired RA) is added to the Radio OS (Windows 7), the required computational resources and memory footprint increased by approximately 1.3% and 1.8 MB, respectively, for Radio Computer #2. From this analysis, it can be concluded that implementation of the RCF on the corresponding Radio OS does not cause considerable overhead. In fact, the increase in computational resource utilization by 1.3%-2.5% is almost equivalent to adding a low-complexity functional block into the RA. For instance, the scrambler functional block required in both LTE FDD and LTE TDD requires about a 3.2% and 1.7% increase in computational resources for Radio Computer #1 and Radio Computer #2, respectively. Furthermore, as the total memory sizes for Radio Computer #1 and Radio Computer #2 are nearly 512 MB and 24 GB, respectively, the increase in memory footprint due to the addition of the RCF is negligible.

Table 5 shows the time required for the PoC system to generate the RA management command and receive execution acknowledgement from each radio computer in our implemented PoC system. In Table 5, we present the times for (1) activation/deactivation of the RA, (2) creation/deletion of data flows, and (3) report of the computational/spectral resource status and context information. These measurements are required for reconfiguring the hardware platform as shown in the use case, "high-level abstraction for multi-RAT management in heterogeneous networks," introduced in the previous section. When the RCF is overlaid on the Radio OS in both Radio Computer #1 and Radio Computer #2, the round-trip time is increased by 0.019 ms and 0.0144 ms on average, respectively, which is less than a 4.95% increase, as shown in Table 5. Consequently, the latency increase of

TABLE 5. Round-trip time for RA management command to start from CSL and come back to CSL from the RCF of each radio computer.

	Radio Computer #1			Radio Computer #2		
	wo RCF	w RCF	Increase (%)	wo RCF	w RCF	Increase (%)
RA activation	18.93 ms	18.95 ms	0.11	7.487 ms	7.5 ms	0.17
RA deactivation	2.124 ms	2.14 ms	0.75	1.134 ms	1.15 ms	1.41
Data flow creation	0.511 ms	0.529 ms	3.52	0.468 ms	0.481 ms	2.78
Data flow deletion	0.467 ms	0.485 ms	3.85	0.451 ms	0.463 ms	2.66
Computational/spectral resource status, context information report	0.444 ms	0.466 ms	4.95	0.438 ms	0.456 ms	4.11

the RA management command to travel the double-layered structure is negligible.

VI. CONCLUSION

In this paper, we have demonstrated that the proposed framework is suitable for instantiating the specific needs of the SDRAN data plane. The proposed approach provides an explicit solution for programmability and software portability for the SDRAN data plane. Using an RVM as an abstract machine, RA codes can be ported to various kinds of heterogeneous hardware platforms in a BBU pool. To provide joint optimization of both software and hardware when executing the RA code, the RVM uses a platform-specific native radio library. For RA management, the proposed data plane framework employs a double-layered structure consisting of a CSL and RCFs, thus providing a uniform way of managing RAs executed on heterogeneous hardware platforms. Furthermore, the feasibility of the proposed data plane framework is verified through a PoC system with the proposed double-layered structure. Based on the implemented PoC system, users can easily perform RA management with a negligible increase in hardware costs. The PoC system consists of a main controller (CSL) and two radio computers, DSP-based Radio Computer #1 and CPU/GPU-based Radio Computer #2. The additional amounts of computational resources and memory footprint due to the proposed double-layered structure are 1.9% and 1.8 MB, respectively, and the additional latency of RA management commands is only 0.017 ms.

REFERENCES

[1] A. Gupta and E. R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, Jul. 2015.

[2] P. Demestichas, A. Georgakopoulos, D. Karvounas, K. Tsagkaris, V. Stavroulaki, J. Lu, C. Xiong, and J. Yao, "5G on the horizon: Key challenges for the radio-access network," *IEEE Veh. Technol. Mag.*, vol. 8, no. 3, pp. 47–53, Sep. 2013.

[3] T. Chen, M. Matinmikko, X. Chen, X. Zhou, and P. Ahokangas, "Software defined mobile networks: Concept, survey, and research directions," *IEEE Commun. Mag.*, vol. 53, no. 11, pp. 126–133, Nov. 2015.

[4] I. T. Haque and N. Abu-Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2713–2737, 4th Quart., 2016.

[5] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: A programmable wireless dataplane," in *Proc. Workshop Hot Topics (SDN)*, Helsinki, Finland, Aug. 2012, pp. 109–114.

[6] W. Wu, L. E. Li, A. Panda, and S. Shenker, "PRAN: Programmable radio access networks," in *Proc. 13th ACM Workshop Hot Topics Netw. (HotNets)*, New York, NY, USA, Oct. 2014, pp. 1–6.

[7] X. Ge, J. Yang, H. Gharavi, and Y. Sun, "Energy efficiency challenges of 5G small cell networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 184–191, May 2017.

[8] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for mobile networks—A technology overview," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 405–426, 1st Quart., 2014.

[9] Y. Yang, J. Xu, G. Shi, and C.-X. Wang, *5G Wireless Systems: Simulation and Evaluation Techniques*. New York, NY, USA: Springer, 2017.

[10] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[11] V. G. Nguyen, A. Brunstrom, K.-J. Grinnemo, and J. Taheri, "SDN/NFV-based mobile packet core network architectures: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1567–1602, 3rd Quart., 2017.

[12] *Reconfigurable Radio Systems (RRS); Radio Reconfiguration Related Architecture for Mobile Devices*, Standard ETSI EN 303 095, Jun. 2015.

[13] *Reconfigurable Radio Systems (RRS); Mobile Device Information Models and Protocols; Part 1: Multiradio Interface (MURI)*, Standard ETSI EN 303 146-1, Nov. 2015.

[14] *Reconfigurable Radio Systems (RRS); Mobile Device Information Models and Protocols; Part 4: Radio Programming Interface (RPI)*, Standard ETSI EN 303 146-4, Apr. 2017.

[15] V. Ivanov, M. Mueck, S. Choi, H. Ahn, K. Kim, and E. C. Strinati, "Radio virtual machine," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Singapore, Dec. 2017, pp. 1–6.

[16] A. Gudipati, D. Perry, L. Li, and S. Katti, "SoftRAN: Software defined radio access network," in *Proc. SIGCOMM Workshop Hot Topics (SDN)*, Hong Kong, Aug. 2013, pp. 25–30.

[17] X. Foukas, N. Nikaein, M. M. Kassem, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proc. ACM CoNEXT*, Irvine, CA, USA, Dec. 2016, pp. 427–441.

[18] E. Coronado, S. N. Khan, and R. Riggio, "5G-EmPOWER: A software-defined networking platform for 5G radio access networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 715–728, Jun. 2019.

[19] K. Koutlia, R. Ferrús, E. Coronado, R. Riggio, F. Casadevall, A. Umbert, and J. Pérez-Romero, "Design and experimental validation of a software-defined radio access network testbed with slicing support," *Wireless Commun. Mobile Comput.*, vol. 2019, Jun. 2019, Art. no. 2361352. Accessed: Sep. 3, 2019. [Online]. Available: https://www.hindawi.com/journals/wcmc/2019/2361352/abs/

[20] J. Zhang, Y. Ji, J. Zhang, R. Gu, Y. Zhao, S. Liu, K. Xu, M. Song, H. Li, and X. Wang, "Baseband unit cloud interconnection enabled by flexible grid optical networks with software defined elasticity," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 90–98, Sep. 2015.

[21] JTNC. (Aug. 2015). *Software Communications Architecture Specification V4.1*. [Online]. Available: http://www.public.navy.mil/jtnc/SCA/SCAv4_1_Final/SCA_4.1_ScaSpecification.pdf

[22] F. Ambrosini, P. Bender, S. Cadzow, S. Choi, V. Ivanov, M. Pagnozzi, and I. Siaud, "Software radio reconfiguration: A highly efficient and modular software reconfiguration approach for mobile devices," Eur. Telecommun. Standards Inst., Tech. Committee, Reconfigurable Radio Syst., Sophia Antipolis, France, White Paper ETSI 21, Oct. 2017. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp21_RRS_FINAL.pdf

[23] *Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the Application of Different Virtualisation Technologies in the NFV Framework*, Standard ETSI GS NFV-EVE 004, Mar. 2016.

[24] OPNFV. (2017). *Open platform for NFV*. Accessed: Oct. 8, 2019. [Online]. Available: https://www.opnfv.org/

[25] D. Rapone, R. Quasso, S. B. Chundrigar, S. T. Talat, L. Cominardi, A. De la Oliva, P.-H. Kuo, A. Mourad, A. Colazzo, G. Parmeggiani, A. Z. Orive, C. Lu, and C.-Y. Li, "An integrated, virtualized joint edge and fog computing system with multi-RAT convergence," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Valencia, Spain, Jun. 2018, pp. 1–5.

[26] R. Wang, H. Hu, and X. Yang, "Potentials and challenges of C-RAN supporting multi-RATs toward 5G mobile networks," *IEEE Access*, vol. 2, pp. 1187–1195, 2014.
[27] *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; E-UTRA and NR; Multi-connectivity; Stage 2 (Release 15)*, document TS 36.300, 3GPP, Mar. 2019.
[28] M. Feng, S. Mao, and T. Jiang, "Base station ON-OFF switching in 5G wireless networks: Approaches and challenges," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 46–54, Aug. 2017.
[29] X. Lin and H. Viswanathan, "Dynamic spectrum refarming with overlay for legacy devices," *IEEE Trans. Wireless Commun.*, vol. 12, no. 10, pp. 5282–5293, Oct. 2013.
[30] Y. Jin, C. Ahn, S. Choi, M. Mueck, V. Ivanov, and T. K. Sarkar, "Design and implementation of ETSI-standard reconfigurable mobile device for heterogeneous network," *IEICE Trans. Commun.*, vol. E99-B, no. 8, pp. 1874–1883, Aug. 2016.

HEUNGSEOP AHN received the B.S. degree from Hanyang University, Seoul, South Korea, in 2013, where he is currently pursuing the Ph.D. degree with the Department of Electronics and Computer Engineering. His current researches focus on reconfigurable systems, RAN architecture, virtualization, LTE-A, mmWave communications, cellular network planning, and various MIMO.

SEUNGWON CHOI received the M.S. degree in computer engineering and the Ph.D. degree in electrical engineering from Syracuse University, Syracuse, NY, USA, in 1985 and 1988, respectively. From 1988 to 1989, he was with the Department of Electrical and Computer Engineering, Syracuse University, as an Assistant Professor. He joined Hanyang University, Seoul, South Korea, in 1992, as an Assistant Professor, where he is currently a Professor with the School of Electrical and Computer Engineering. His research interests include software reconfiguration, digital communications with a recent focus on the implementation of various kinds of MIMO systems for mobile communication systems.

MARKUS MUECK received the Diploma degree in electrical engineering from the University of Stuttgart, Germany, and École Nationale Supériere des Télécommunications (ENST), Paris, France, in 1999, and the Ph.D. degree from ENST, in 2006. He is currently the Senior Standardization Manager with Intel Germany GmbH, and also an Adjunct Professor with the University of Technology, Sydney, Australia. He is a member of the Board with ETSI, the Chairman of ETSI Reconfigurable Radio Systems Technical Body, and the IEEE Special Interest Group on Cognitive Radio in 5G.

VLADIMIR IVANOV received the Diploma degree in mathematics from Novosibirsk State University, Russia, in 1979, and the Ph.D. degree in computer science from Military Communication Academy, Saint Petersburg, Russia, in 1989. In 2003, he joined Intel Corporation. He contributed to the IEEE P1900.4 WG (protocols for cognitive radio), where he acted as the Chair of the System Architecture Subgroup. From 2014 to 2016, he was with LG Electronics, as a Principal Research Engineer. In 2016, he joined Saint Petersburg State University of Aerospace Instrumentation, Russia, as a Professor. His research interests include automated electronic design (ESL and HLS), the mathematical foundations of electronic design, concurrent computations, and reconfigurable radio.