Landscape of IoT security

Schiller, Eryk; Aidoo, Andy; Fuhrer, Jara; Stahl, Jonathan; Ziörjen, Michael; Stiller, Burkhard

DOI: https://doi.org/10.1016/j.cosrev.2022.100467

Toward a Live BBU Container Migration in Wireless Networks

Eryk Schiller, *Member, IEEE*, Jesutofunmi Ajayi, *Student Member, IEEE*, Silas Weber, Torsten Braun, *Senior Member, IEEE*, Burkhard Stiller, *Member, IEEE*

Cloud Radio Access Networks (Cloud-RANs) have recently emerged as a promising architecture to meet the increasing demands and expectations of future wireless networks. Such an architecture can enable dynamic and flexible network operations to address significant challenges, such as higher mobile traffic volumes and increasing network operation costs. However, the implementation of compute-intensive signal processing Network Functions (NFs) on the General Purpose Processors (General Purpose Processors) that are typically found in data centers could lead to performance complications, such as in the case of overloaded servers. There is therefore a need for methods that ensure the availability and continuity of critical wireless network functionality in such circumstances.

Motivated by the goal of providing highly available and fault-tolerant functionality in Cloud-RAN-based networks, this paper proposes the design, specification, and implementation of live migration of containerized Baseband Units (BBUs) in two wireless network settings, namely Long Range Wide Area Network (LoRaWAN) and Long Term Evolution (LTE) networks. Driven by the requirements and critical challenges of live migration, the approach shows that in the case of LoRaWAN networks, the migration of BBUs is currently possible with relatively low downtimes to support network continuity. The analysis and comparison of the performance of functional splits and cell configurations in both networks were performed in terms of fronthaul throughput requirements. The results obtained from such an analysis can be used by both service providers and network operators in the deployment and optimization of Cloud-RANs services, in order to ensure network reliability and continuity in cloud environments.

***Index Terms*—Cloud-RAN, Live Migration, LoRaWAN, LTE, NFV**

I. INTRODUCTION

CURRENTLY, significant efforts focus on new access methods targeting the Internet-of-Things (IoT). New standards have emerged, with the most prominent examples being Long Range (LoRa) or cellular technologies, such as Long Term Evolution (LTE) Cat. M (LTE-M) or LTE Cat. N (*i.e.,* Narrow Band (NB)-IoT). While LoRa defines a new access method, LTE-M and NB-IoT defined new features of LTE Advanced Pro in Release 13 of the 3rd Generation Partnership Project (3GPP) specifications. With these enhancements evolved Node Bs (eNodeBs) may activate LTE-M or NB-IoT after a software upgrade, depending on the base software release and configuration in use.

The introduction of new access methods into a wireless network ecosystem typically requires significant investments in terms of Capital Expenditure (CAPEX) and Operating Expenditure (OPEX) which do not guarantee profitable returns in a short time. However, with the current network virtualization trend and the goal of managing network infrastructure, platforms, and applications through the Everything-as-a-Service (XaaS) paradigm, significant cost reductions can be obtained since no upfront investments are required, and resources can be traded on-demand, leading to zero-CAPEX for Infrastructure Providers (InfraPs). Furthermore, the risk of ill-estimated CAPEX versus revenue estimations is minimized since resources, and thus infrastructure, platforms, and applications scale up continuously as needed. When the XaaS operational model is applied to the telecommunications (telco) industry, significant benefits of the XaaS model can counter the effect of the ever-decreasing Average Revenue Per User (ARPU) in telco ecosystems. As such, cloud computing and (network) virtualization have stood out as two essential technologies that can be used to create new opportunities to meet current and future goals of Mobile Network Operators (MNOs). Cloud computing enables ubiquitous and on-demand access to a shared pool of scalable computational resources (*i.e.,* processing, networking, and storage). Furthermore, virtualization techniques, such as Network Function Virtualization (NFV) and Software Defined Networking (SDN), use a network abstraction to virtualize network functions and to improve network programmability and intelligence.

Centralized Radio Access Networks (C-RANs) have emerged as a novel mobile network architecture to be deployed in the network providing features, such as network slicing, statistical multiplexing, energy efficiency, and higher network capacity. C-RAN systems replace traditional Base Stations (BSs), where distributed (passive) radio elements, such as the Remote Radio Unit (RRU), are connected to a centralized baseband processing pool at a central location, typically at data centers, through Radio Aggregation Units (RAUs) [19]. Such an architecture supports the up/down-scaling, as well as the migration of virtual Baseband Units (vBBUs) across virtual and physical infrastructure resources to meet varying levels of network and processing demands.

Thus, this paper evaluates the feasibility of live migrating containerized Baseband Units (BBUs) in wireless networks, while specifically targeting LoRa and LTE networks. Different from other works, which focus on the live migration of generic

Submitted 15 October 2021; This paper was partially supported by (a) the University of Zürich UZH, Switzerland, (b) the University of Berne, and (c) the European Union H2020 Research and Innovation Program under grant agreement No. 830927, namely the H2020 Concordia Project.

Eryk Schiller (schiller@ifi.uzh.ch) and Burkhard Stiller (stiller@ifi.uzh.ch) are with the Communication Systems Group CSG, Department of Informatics IfI, Universität Zürich UZH, Zürich, Switzerland

Silas Weber is with the Faculty of Law IUS, Universität Zürich UZH, Zürich, Switzerland (silas.weber@uzh.ch)

Jesutofunmi Ajayi (jesutofunmi.ajayi@inf.unibe.ch) and Torsten Braun (torsten.braun@inf.unibe.ch) are with the Communication and Distributed Systems Group CDS, Institut für Informatik INF, Universität Bern UniBE, Bern, Switzerland.

applications and services in simulated or emulated network environments, this paper approaches the problem of live BBU migration in a systems-oriented manner. This approach enables us to adequately study the practical and technical challenges of performing a live migration in complex and real-time network environments. In this way, this paper evaluates and studies the impact of migrating the BBU on the performance and availability of the network in real-time, something that cannot be adequately investigated in non-real-time scenarios. By providing containerized versions of LoRa (BBU, RRU and Network Server (NS)) and LTE (BBU, RRU) C-RAN networks, this approach is now able to design, implement, and study the impact of live migrations in these widely-used and popular networks. Based on the performed evaluations, dedicated challenges of the considered live migration approach applied to real test-bed setups of the different networks are derived, which further leads to the suggestion of suitable methods that can facilitate a live migration of BBUs in wireless networks, in the future.

The remainder of this paper is organized as follows. Section II lays the background for which related work is detailed, leading to the critical assumptions determined. While the design of the live container migration process is described in Section III, the cloud-based architecture of wireless networks evaluated is specified in Section IV. Driven by the experimental setup defined in Section V, key evaluation results are presented in Section VI. Finally, Section VII summarizes the work and suggests areas for further improvements.

II. Background and Related Work

To allow for a basis for exploiting technical features and functionality of existing networking technologies, those are discussed first. Second, the narrowing down of these approaches to investigate the live migration aspect is performed by summarizing related work specifically, before the main assumptions for the approach undertaken here are derived.

A. Long Range Wide Area Network

LoRa is a spread spectrum modulation technique used in Low Power Wide Area Networks (LP-WAN), allowing for long-range communications in low power IoT applications. It is also used to provide support for devices installed in remote areas[1]. It has become the primary technology in IoT networks, and it provides a Long Range Wide Area Network (LoRaWAN)[2]. The Medium Access Protocol (MAC) protocol is placed on top of the LoRa Physical (PHY) layer within the LoRaWAN open standard. Because IoT sensors show limited battery life, capacity, and range, LoRaWAN is built to optimize LP-WAN, while taking those limits into account.

1) The Things Network

The Things Network (TTN), also a LoRa Alliance member, provides a worldwide LoRaWAN network for the TTN community. Anyone with a LoRa Gateway (GW) can register GWs with TTN, thereby extending the network coverage. At the time of writing, TTN lists 47,713 GWs and is present in more than 80 countries [48]. A LoRaWAN network (*cf.* Fig. 1) shows a star-of-stars topology. GWs act as a relay between end devices and the central network server. GWs use the Internet Protocol (IP)-based communication to connect to the network server by turning LoRa Radio Frequency (RF) signals into IP packets and vice versa [23]. End devices are not associated with any particular GW. Messages are delivered without any regard for the destination on the MAC layer and can be spotted by many GWs simultaneously. A message received by a GW is forwarded to the NS, which filters redundant message instances, performs security checks, and forwards messages to an appropriate application server [24]. The NS is also in charge of Downlink (DL) message scheduling.

2) Long Range Wide Area Network

LoRaWAN is an ALOHA-based protocol allowing any end device to initiate a transmission at any time, supporting 3 end device categories:

- All LoRaWAN devices support the default *Class A* communication. Two DL reception windows, established on an end device right after an Uplink (UL) transmission has been established, are used to receive a response, which enables two-way server-to-end device communication. In Class A communications, DL transmissions origination from the server must wait for the UL transmission from a targeted end device and cannot be launched unilaterally by the network server, while the power consumption of Class A devices is the lowest.
- An end device supporting *Class B* communications (*i.e.,* two-way communication with a deterministic DL delay) open additional receive windows at predetermined periods. This is accomplished by a beacon-based time synchronization in which GWs instruct an end device to open a reception window at a specific moment.
- In the *Class C* communication paradigm (*i.e.,* low latency and two-way communications), devices always open reception windows except when transmitting themselves. If the device is not transmitting, the NS can initiate a DL transmission at any time. Class C devices have the highest power requirements and remain plugged-in rather than being battery-powered.

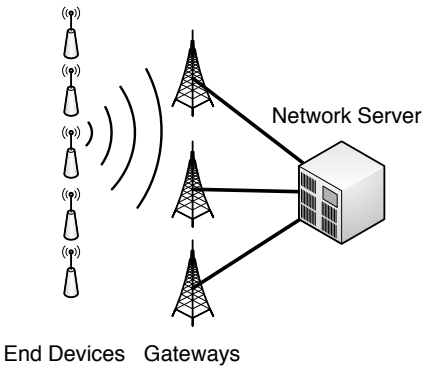


Fig. 1: LoRaWAN Architecture

[1]https://www.semtech.com/lora/what-is-lora
[2]https://lora-alliance.org/resource_hub/what-is-lorawan

3) Long Range Modulation

LoRa is intended for long-range, low-power communication with only a few bytes sent each day. Typically, LoRa shows a range of 2–5 km in urban areas and 15 km in suburban setups [2], however, LoRa communication also demonstrates a successful reception from a low-orbit satellite [47].

In the license-free Industrial Scientific Medical (ISM) band, LoRa signals are modulated as *chirps* using the Chirp Spread Spectrum (CSS) technique [14], [36], where the ISM band in the sub-GHz band is located in the 902-928 MHz and 863-870 MHz Electromagnetic (EM) spectrum range in North America and Europe, respectively, and a regular LoRa channel bandwidth is at 500 kHz and 125 kHz, respectively. The LoRa signal consists of up- and down-chirps. Up-chirps begin with a low frequency and gradually grow in frequency in time, whereas down-chirps (or conjugated chirps) start with a high-frequency signal and progressively reduce the frequency in time. An unmodulated up-chirp (or down-chirp) in Europe, for example, changes its frequency between 868.4375 MHz and 868.5625 MHz in the 125 kHz channel centered around 868.5 MHz. Every UL LoRa frame starts with a preamble, which consists of ten unmodulated up-chirps, preceding the actual payload. The preamble is subsequently followed by two and a quarter unmodulated down-chirps, which signal the conclusion of the preamble and the start of the payload, *i.e.,* Start of the Frame Delimiter (SFD). The payload is composed of symbols (*i.e.,* modulated chirps) that represent the header, the message, and a Cyclic Redundancy Check (CRC) that is used to detect transmission errors [45].

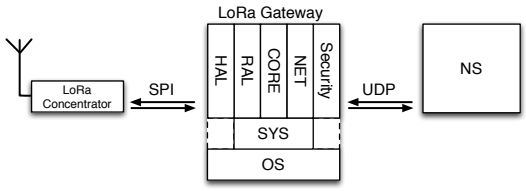


Fig. 2: LoRa GW Architecture

4) Long Range Gateway

The architecture of the LoRa GW [42] (*cf.* Fig. 2) includes the LoRa concentrator, *e.g.,* the SX1276/SX1278 chip, attached to a single board computer through a Serial Peripheral Interface (SPI). The GW is built on top of several abstraction layers. Semtech's Hardware Abstraction Layer (HAL) libraries interact with radio hardware of several designs. The radio functionality is abstracted through the Radio Abstraction Layer (RAL) prepared for different HALs. The System Abstraction Layer (SYS) is provided for different Operating Systems (OSs), such as Linux or FreeRTOS. The Network Abstraction Layer (NET) builds upon a security subsystem and is used for communicating with the NS. The CORE functionality of the packet forwarder uses abstraction layers to communicate with LoRa end devices through the LoRa concentrator and the NS through the NET abstraction layer.

The LoRa concentrator as of Fig. 3, *e.g.,* a SX1276/ SX1278 chip[3], consists of an analog radio chain, a baseband Digital Signal Processing (DSP) unit, and an SPI sub-system. On the UL, the signal passes through the Low Noise Amplifier (LNA), an Analog to Digital (ADC), and a Digital Down Converter (DDC), which provides the baseband In-phase/Quadrature (I/Q) sample stream to the digital baseband LoRa demodulator, demodulating that stream and providing decoded messages toward the SPI subsystem. On the DL, a LoRa message coming from the SPI subunit is modulated with the digital baseband LoRa modulator, providing a baseband I/Q sample stream toward the Digital Up Converter (DUC). The DUC converts the I/Q stream to the passband. The passband signal is converted to the analog signal through the Digital to Analog (DAC), which is amplified through the Power Amplifier (PA) and sent through the air toward end devices.

[3]https://www.semtech.com/products/wireless-rf/lora-core/sx1276

B. Long Term Evolution

The LTE [44] network is one of the most advanced and widely used mobile telecommunications technologies, since it provides high-speed data and voice capabilities, while considerably outperforming the previous generation cellular networks in the above aspects. The LTE network provides IP connectivity between the User Equipment (UE) and a Packet Data Network (PDN), such as the Internet, without a significant disruption to end users' connectivity during mobility. LTE's prominent network architecture consists of two components:

- Radio Access Network (RAN): It facilitates wireless radio connections between UEs and the BSs and performs certain functions, such as Radio Admission Control (RAC), Radio Resource Management (RRM), Radio Resource Control (RRC), and inter-cell interference coordination, at the edge of the network.
- Evolved Packet Core (EPC): the EPC connects UEs to the Internet or a MNO's network and supports user mobility in the network through handovers. It is also responsible for the establishment of radio bearers, *e.g.,* Signalling Radio Bearers (SRBs), in the LTE network.

The 3GPP 4G RAN-specified LTE-Uu protocol is based on a hybrid PHY layer with Orthogonal Frequency-Division Multiple Access (OFDMA)-based UL (UE–eNodeB) and Single-Carrier-Frequency-Division Multiple Access (SC-FDMA)-based DL (eNodeB–UE).

The 3GPP Release 13 LTE Cat. M1 is an LTE standard to support IoT applications [31] and based on an LTE resource grid shared among UEs in the network, where a UE node receives a portion of frequency delivered on a Transmission Time Interval (TTI)-basis, where one TTI lasts for 1 ms.

From the eNodeB perspective, LTE Cat. M1 provides amendments to the regular LTE-Uu protocol stack, specifically

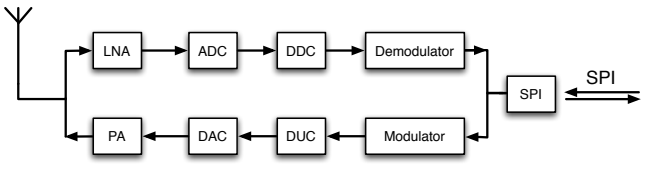


Fig. 3: LoRa Concentrator

at MAC and Radio Link Control (RLC) layers, and uses a subset of PHY functionality of a regular eNodeB. LTE Cat. M supports the UE mobility via a handover operation, UE measurement reporting, RRC connection release, RRC re-establishment, and voice communications.

LTE Cat. N is also known as NB-IoT [18], [33] and defines a new PHY layer for 3GPP Release 13. It uses 200 kHz channels, *i.e.,* one 180 kHz Physical Resource Block (PRB), in the frequency domain with 10 kHz guard bands on each channel edge. LTE Cat. N, unlike LTE Cat. M, does not support mobility (*i.e.,* handovers) or voice. Since the 3GPP Cat. N PHY is entirely new, an eNodeB must support and configure LTE Cat. N within a specific frequency range. On the DL, subcarriers are organized in the same way as in the regular LTE, *i.e.,* 12 subcarriers with 15 kHz spacing with the same TTI organization in time. On the UL, another channel organization was added of 48 subcarriers with 3.75 kHz spacing, where one LTE Cat. N frame consists of five slots of 2 ms each, which is in contrast to regular LTE with one frame composed of 10 TTIs or 20 slots. To support LTE Cat. N, the eNodeB has to implement the new PHY, *e.g.,* in software. However, the new organization does not change much, since it still mainly depends on the OFDMA/SC-FDMA access using different parameters [15].

C. Centralized Radio Access Network

The virtualization of the RAN has emerged as a possible solution for MNOs to improve their services while keeping costs down. This involves decoupling the software that controls the access network from the underlying hardware, which allows for fast upgrades and improved scalability to meet varying network traffic demands, the rapid deployment of new services, and the ability to centralize and pool various resources together. As a result of this abstraction [13], C-RAN implementations consume significantly less power than traditional RAN implementations provided by BSs.

With C-RAN, BBUs are not deployed along with physical BSs at a remote location. Instead, they are decoupled and moved to a centralized processing pool, including other BBUs too. Pooling BBUs together can lead to more sophisticated joint spatio-temporal processing of radio signals, while also improving the spectral efficiency [30].

However, this deployment comes with several challenges. As the result of signal processing performed on General Purpose Processors (GPPs), the C-RAN performance can suffer, at least compared to hardware-based implementations. *E.g.*, if due to a lack of computational resources (*i.e.,* memory or CPU) the GPPs cannot meet signal processing requirements for Ethernet frames received from the front haul network on time, this will have an overall negative impact on the service-level performance of the network. This is further characterized by the fact that in such a scenario, strict timing requirements for Hybrid Automatic Repeat Request (HARQ) would not be met, which leads to more frequent re-transmissions and performance degradation. This is especially critical for services requiring low-latency in the order of a few ms, and/or highly reliable communications, such as Augmented Reality (AR) and Industrial Automation (IA) applications.

The HARQ timing constraint in LTE C-RAN can be considered as one of the most restricting limitations of such deployments, since for every MAC Packet Data Unit (PDU) received it requires an Acknowledgment (ACK) or Negative Acknowledgment (NACK) issued by entities in the network. This NACK is expected to be received in $8\,\mathrm{ms}$, *i.e.,* HARQ Round Trip Time (RTT), which puts stringent requirements on the front haul link to maintain this timing constraint.

The use of virtualization layers in certain C-RAN deployments, such as Virtual Machine (VM), introduces fluctuations in processing time [30]. Such fluctuations can make it challenging to maintain connectivity in C-RANs due to frequently missing real-time requirements caused by the introduction of the virtualization layer. Finally, the high-capacity link required to carry the I/Q sample stream between the RRUs and a centrally placed BBUs located several kilometers away from the network edge (at a datacenter) remains challenging for C-RAN deployments.

The Cloud Radio Access Network (Cloud-RAN) concept is a result of advancements *(i)* to facilitate the development of flexible wireless networks to respond quickly to changes in demand and *(ii)* toward designing service-based network architectures. The concept is very similar to C-RAN, in which compute-intensive signal processing functions are decoupled from specialized hardware and implemented on GPPs.

The cloudification of the RAN has facilitated the opportunity to deploy it as a service in cloud environments. Thus, Radio Access Network as a Service (RANaaS) has emerged as a new cloud computing paradigm, in which an access network can be delivered as a pay-as-you-go service instantiated on top of a cloud infrastructure [30]. This allows MNOs to respond quickly to changes in the load of the network (*e.g.,* increased demand at a particular cell) by deploying new instances of RAN network functions, distributing network and processing load across different cloud instances through load balancing.

D. Functional Splits

To enable the RAN to meet heterogeneous service requirements, 3GPP envisions eight options on how to split the RAN radio stack between the BBU and the RRU. These splits can occur between any of the five principal layers in the radio stack, which includes: RRC-Packet Data Convergence Protocol (PDCP), PDCP-RLC, RLC-MAC, MAC-PHY, and PHY-Radio Front-end, and three internal splits between the MAC, PHY, and RLC layers (*i.e.,* High RLC-Low RLC, High MAC-Low MAC, High PHY-Low PHY), *cf.* Fig. 4.

The current RAN architecture requires baseband signals to be transported between the BBUs located at a Remote Cloud Center (RCC) and the edge located RRUs through, for example, a Radio-over-Fiber (RoF)-based Common Public Radio Interface (CPRI) [51] (*cf.* Fig. 4). However, this places very high bandwidth requirements on the front haul transport network, and such an architecture is unlikely to meet scalability and performance requirements of upcoming 5G use cases, like AR and Virtual Reality (VR) applications. Recent developments have focused on redefining the architecture of the access network by proposing a Next Generation Fronthaul Network

(NGFN), which can split the radio stack between BBUs and RRUs, determining a centralized control of distributed radios. To connect these components, a new front haul interface, the Next Generation Fronthaul Interfaces (NGFI) [6], is proposed and can be used to transport baseband signals using the Radio-over-Ethernet (RoE) protocol or IP interfaces.

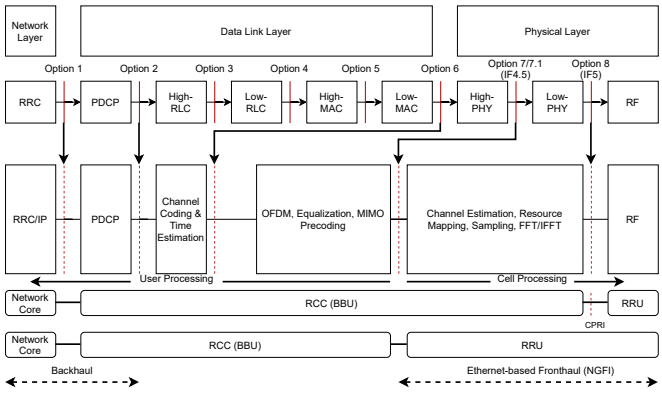


Fig. 4: Functional Split Options

Furthermore, in current C-RANs deployments, I/Q samples are carried from the BBU to the RRU, which places very high bandwidth requirements on the front haul transport network. Such an architecture likely struggles to meet future wireless networks' scalability and performance requirements. The design and implementation of NGFI enable decoupling front haul bandwidth from the number of antennas to ultimately reduce bandwidth and increase transmission efficiency [16]. Through the NGFI architecture, these components can be connected through Ethernet or IP interfaces. More specifically, this allows for point-to-multipoint connections between BBUs and RRUs and for a third component, RAU, to control multiple RRUs that operate in different bands and with different coverage. Functional Splits (FSs) can be established through NGFI, too, which makes C-RAN deployments more flexible and reduces the required front haul capacity [39]. However, the clock-based synchronization between the BBU and RRU is lowered, which impacts the front haul network due to increased data transmission errors [34].

E. Related Work in Detail

C-RAN [19] was first introduced by China Mobile for the optimization of CAPEX and OPEX in future mobile networks. Several approaches exist in the scope of cloudified[4] LoRa, such as [9], [22], [38]. [9] studies different RRU/BBU split configurations and provides a LoRa monitoring application within the cloud environment. It is unclear, whether their cloudified decoder is able to decode LoRa signals. [38] provides a working C-RAN system based on the existing LoRa encoder/decoder implementation [35]. [22] studies front haul compression able to reduce the RRU-BBU communication by a factor of 93.7%. This work mainly extends [38] with the docker.io[5] migration functionality of cloudified BBUs.

[4]The terms "cloudified" and "virtualized" are used interchangeably here.
[5]The terms "docker.io" and "docker" are used interchangeably here.

In 3GPP networks, a C-RAN allows the BBU to be hosted at a central location that is close to the edge or at a data center that is further away. Several cloudified implementations of real-time 3GPP LTE or 5G networks exist, such as Amarisoft OpenAirInterface (OAI) [28], [29].

Several approaches study service migration in various scenarios, including wireless networks. [21] focuses on low-latency strategies for service migration. It discusses connectivity enhancements, migration strategies, and bandwidth slicing as considerations that could be used to support the migration of services with low-latency requirements. However, [21] focuses on service migration in transport networks with a particular focus on schemes for vehicular communications. Furthermore, that evaluation is carried out via simulations using mobility pattern data. [26] addresses the problem of live migrating multiple mobile services across centralized and edge clouds while maintaining high Quality-of-Service (QoS) to ensure that Service Level Agreements (SLAs) are not violated. Their approach is based on a mathematical formulation of the problem, and the proposed model is carried out using simulations, limiting its applicability to real wireless network settings. [25] supports user mobility through live migration of offloaded services in Wide Area Networks (WANs), and proposes an edge computing-based solution leveraging layered storage to reduce the file system synchronization overhead while supporting seamless migrations. [12] discusses requirements and perceived challenges of providing seamless service migration of industrial applications between edge servers. They propose a novel iterative migration scheme that reduces service downtime by excluding the stop-and-copy phase of live migration. [3] studies the dynamic adaptation of Functional Splits between the Central Unit (CU) and the Distributed Unit (DU) in 5G the RAN. It is claimed that such an approach is equivalent to live migration of functions between the CU and the DU. Their proposed approach is based on replicating the MAC and RLC functions of the RAN at both the CU and the DU simultaneously, even when they are not being used, and transferring the state of functions between the old set of MAC and RLC functions and the active set of functions.

F. Assumptions

The majority of these approaches do not study the live migration of containerized C-RAN functions in wireless networks, and they do not address strict timing and processing constraints. While only one approach known so far for live migrating C-RAN functions in wireless networks was proposed in [3], it considers a different functional split scenario from the one studied in this work.

Driven by the background provided and the analysis of related work, two types of NGFI-based splits are considered:

- **NGFI:IF5/CPRI (3GPP Option 8):** The IF5 split is performed at the input, *i.e.,* Receive (RX), and output, *i.e.,* Transmit (TX), of the Orthogonal Frequency-Division Multiplexing (OFDM) symbol generator (*i.e.,* frequency domain signals) and transports resource elements transmitted or received in usable channel bandwidth. RCC and RRU swap baseband I/Q samples across the front

haul link. A low compression is applied to I/Q data being transported across Ethernet front haul. This reduces the front haul link capacity requirement to 28% of the time-domain I/Q split case. Therefore, an RRU with a bandwidth of 20 MHz can be provisioned using a 1 Gbps link between an RRU and RCC [30].
Note that LoRa does not define any splits in its current state. Therefore, this work is inspired by the IF5 split in LoRa, where the RRU generates a raw baseband I/Q sample stream. To this end, Software Defined Radio (SDR) applies DDC/ADC functions, which produce a raw baseband I/Q sample stream as output. This stream does not have to be necessarily provided toward the digital demodulator immediately but can be transported through the front haul interface to the remote cloud LoRa demodulator. However, the problem of IF5 is currently its efficiency, *e.g.,* it requires approximately 1 Gbps to sample 20 MHz channels. LTE defines more efficient splits to protect against high load on front haul interfaces, *e.g.,* IF4.5. However, no splits are currently specified in the LoRa domain.

- **NGFI:IF4.5 (3GPP Option 7.1):** The IF4.5 split is similar to the BBU–RRU interface, in which baseband frequency domain I/Q samples of all PRBs are transported between the BBU and RRU. With this split, the distributed RRUs is responsible for the lower PHY layer processing, namely Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT), which optimizes throughput of the front haul link. As a result, this split can be considered a hybrid centralization solution, since the Cyclic Prefix (CP) removal and transformation of the signal received to the frequency-domain are performed using FFT [20]. Hence, guard-band subcarriers, typically present in OFDM-based systems, like LTE, can be removed at the distributed RRUs and carried over the front haul to the BBU. As the number of guard subcarriers in LTE is 40% [52], the front haul bit rate is lowered using the NGFI IF4.5 (Option 7.1) split and, thus, it is significantly lower than for the NGFI IF5 (Option 8) split.

Thus, the NGFI IF5 and IF4.5 functional splits for LoRa and LTE, respectively, are applied to the setup utilized in this work. Note that the IF4.5 functional split is considered the standard C-RAN-based split in LTE networks and corresponds to the PHY-Radio Front-end split as defined by 3GPP [1]. In the case of LoRa, the *complex* representation of the I/Q sample stream with 64 bits per I/Q sample is used. Thus, in turn, the performance of containerized LTE deployments is evaluated and provides the first practical solution for a containerized migration in such network environments.

III. Live Migration Design

With cloud computing and NFV emerging to enable computing resources to be scaled on demand and packet processing to be moved from hardware middle-boxes (*i.e.,* routers) to software middle-boxes running on commodity hardware, service providers are able to improve the management of computing and network resources in data center environments. The virtualization of Network Functions (NFs) leads to higher energy savings, improved resource efficiency, and better QoS performance. The live migration of NFs can also be used to provide similar benefits, since it can support flexible and reliable provisioning of NFs and services in data centers, too. However, current management and orchestration technologies available for a live migration have been primarily adopted from the IT industry, which limits their suitability in the support of telco-grade performance (*i.e.,* being low-latency, high performance, reliable, and scalable), especially in the scope of next-generation wireless networks, such as upcoming virtualized 5G networks.

The process of live migration has been used extensively in cloud computing environments as it offers a solution for typical data center operational concerns, such as fault management, load balancing, and low-level system maintenance [7]. The process involves transferring running services (or applications) from one physical or virtual resource to another while ensuring minimal disruption to users of the services being transferred. Due to the ease of duplication, MNOs tend to deploy Virtual Network Functions (VNFs) in VMs, which further enables the flexible scheduling and placement of such VNFs in the infrastructure [54]. Unlike VM migration, where the CPU state, memory content, and content storage are considered during the migration process, the migration of containers is primarily concerned with memory content. Memory migration depends on the type of application being migrated and the workload being supported by the underlying operating system [4]. Thus, the more collaborative processes the system handles, the more time the migration process is likely to take, leading to a longer downtime of the service being migrated.

To migrate a containerized BBU between two hosts, while keeping the users' connection active, two possible migration approaches can be applied. In the "stop-and-copy approach", where the running state of the containerized BBU is check pointed at the source node, the BBU on the source is stopped before being migrated and restored on the destination node. The second approach entails performing the migration step in an *iterative* manner by executing several checkpoints of application processes (*i.e.,* BBU functions) before the final migration step is completed. The advantage here is the reduced downtime of the application due to less memory being checkpointed at the beginning of the migration, which is essential to maintain the network's performance.

A. Checkpoint and Restore in Userspace

Docker.io packages software components into self-containing software bundles (*i.e.,* images) and relieves the host operating system from the installation of libraries necessary to support a given application. Such software bundles can be migrated from one host to another without any hassle by copying the image between those machines. To perform (live) migration of docker containers across nodes in a network, the user-level process tool Checkpoint and Restore in Userspace (CRIU) is integrated into the docker system. CRIU supports several types of migrations, including

disk-less Migration, iterative migration, lazy migration, and live migration. In general, the migration procedure can be achieved by check pointing the state of running processes in a container and restoring these processes in user space at the current node or at a different node in the network infrastructure. CRIU stores snapshots of container processes as momentary protocol buffer (protobuf) images in a file system [50]. To access the state of processes during the check pointing phase, CRIU uses the `/proc` file system to collect running threads and child/dependent processes. It uses the `ptrace ()` Linux system call for gathering core parameters that need to be dumped/restored. Upon restoration, CRIU uses the `fork ()` Linux system call to recreate process trees. Other steps during the restore process include mapping necessary code and pages into address spaces needed at the destination node, restarting threads, and rebinding any sockets to resume communications. The CRIU migration process is explained in Fig. 5, where a given process tree (*e.g.,* within a docker container) can be dumped, copied, and restored.

Fig. 5: Migration Framework

B. Container Migration Challenges

A live BBU container migration in a wireless network, especially of containerized BBU in LoRaWAN was rather trivial to perform, compared to doing the same in LTE networks. This is due to the fact that the GW present in LoRaWAN does neither maintain any state (*i.e.,* it is stateless) related to end clients nor does it need any strict synchronization. However, this is not the case for migrating the BBU in LTE networks, which makes it significantly more challenging to perform a live migration in such environments, as detailed in the following:

1) Strict timing and synchronization network requirements exist. In LTE, the Hybrid Automatic Repeat Request (HARQ) is a mechanism used to improve transmission reliability, and it requires the transmitter to retransmit packets in the case of NACKs or acknowledges the reception of packets with an ACK to avoid packet errors [8]. This results in a major impact on the feasibility and performance of the live BBU migration, since the eNodeB is unable to send or receive any ACK/NACK messages to the UEs connected during downtime. Therefore, the period of inactivity at the eNodeB causes multiple packets to be lost and degrades the QoS experienced by users.
2) In a Cloud-RAN deployment, where RAN functional splits exist between the BBU and the RRU, ensuring that packets arriving over-the-air at non-migrated functions (*i.e.,* the Low-PHY and Radio Front-end stacks at the RRU) are later transferred to migrated function(s), is of significant importance to maintain stability and consistency in the network. While [3] showed the feasibility of a similar approach using a custom-made solution for functional splits that occur between layers (*i.e.,* RLC and MAC), their implementation was based on the modular and customizable srsLTE platform. The complexity of the OAI platform, which implements option 7.1 (NGFI IF4.5) function split out of the box, makes such an approach non-trivial, especially for intra-layer functional splits, such as the intra-PHY (*i.e.,* High PHY-Low PHY) split considered in this work.
3) The Cloud-RAN used in this work, OAI-RAN, is a soft real-time system, which means that any excessive burden to the CPU leads to the desynchronization between the eNodeB and the UE [17]. This is also the case during prolonged periods of inactivity of the RAN. Since the migration (checkpoint and restore) of the BBU does require a non-negligible period of downtime, maintaining synchronization between the eNodeB and the UEs during this period is non-trivial.

These observations obtained indicate that any momentary interruptions in the operation of the BBU in an LTE network break the communication between the UE and the Core Network (*i.e.,* Mobility Management Entity (MME) and Serving/Packet Gateway (S/PGW) (Serving Packet Gateway)), which causes the UE to no longer have access to the Internet.

The interruption caused by the procedure of live migrating the BBU in LTE can be seen as an example of a temporary call drop in the network, or more specifically, a Radio Link Failure (RLF). In LTE networks, a RLF typically occurs when the connection between the UE and the eNodeB cannot be established or maintained. If a UE experiences a radio link problem, which would occur due to BBU migration, it typically waits for a specified time until the broken PHY link is either recovered or canceled. If this period passes and the connection is not reestablished with eNodeB, it is released from the eNodeBs list of connections, which requests to the MME the release of the UE-associated logical S1-connection (S1-U/S1AP) as a result of E-UTRAN-generated reasons. However, more studies are needed in this area.

Outside the mentioned network-related observations, it needs to be noted that in order to checkpoint the containerized BBU in LTE, it is required to checkpoint and restore Stream Control Transmission Protocol (SCTP) sockets that are used to communicate with the Evolved Core Network/EPC. However, as at the time of writing, the tool used to perform the migration, CRIU, does not support the check pointing of SCTP sockets, which limits the suitability of the approach to support the migration of the LTE BBU.

IV. Centralized Radio Access Network Architecture Design

The overall architecture considered (*cf.* Fig. 6) envisions a cloud ecosystem composed of compute nodes (*i.e.,* hosts) running virtual BBUs or vBBUs), which in turn support RRUs. Those BBUs are provided as containers and executed among compute nodes, since containers are better adapted for real-time applications [28]. The cloud supports live migration of BBUs through the CRIU engine. Live migration might be used,

for example, to increase energy efficiency of the infrastructure. If services run idle, *e.g.,* at night, it can be beneficial to migrate BBUs to a smaller number of compute nodes, power-off unused infrastructures, and save energy [19], thus, materializing green computing. Live migration shows measurable benefits, since it limits service downtime (*i.e.,* upon migration while swapping containers) and allows for an uninterrupted operation of services being migrated. If these services are efficiently migrated, the user has the impression that the service was not interrupted, even though it was migrated several times.

A. Long Range Centralized Radio Access Network

A "regular" LoRa GW consists of distinct subcomponents (*cf.* Fig. 2). LoRa C-RAN executes the PHY layer processing in a cloudified setup. Thus, signal processing, implemented in software, runs within a cloud environment using general-purpose server infrastructures. This contrasts in comparison to a GW directly executing signal processing in hardware using custom-designed Radio Frequency Integrated Circuit (RFIC) modules. In this cloudified case, the RRU has to remain in a distant location. However, the RRU design is significantly reduced, while the LoRa PHY, which performs demodulation and decoding, is moved to the cloud. Therefore, any specialized pieces of hardware, such as the SX1302 transceivers[6] found on regular LoRa GWs, are not required on the LoRa in the C-RAN setup.

1) Long Range Centralized RAN Architecture

A high level architecture is shown in Fig. 7, including all network functions in the system (*i.e.,* RRU, BBU, and NS). The RRU might be a small-sized computer equipped with a radio chain composed of an antenna, an amplifier, and DAC as well as ADC converters. Similar functionality might be provided by an SDR board attached to the computer using a Universal Serial Bus (USB) port. The SDR board can be used to communicate with LoRa end devices (*e.g.,* Raspberry Pi or Arduino with LoRa compliant chips) in the LoRaWAN through analog radio signals.

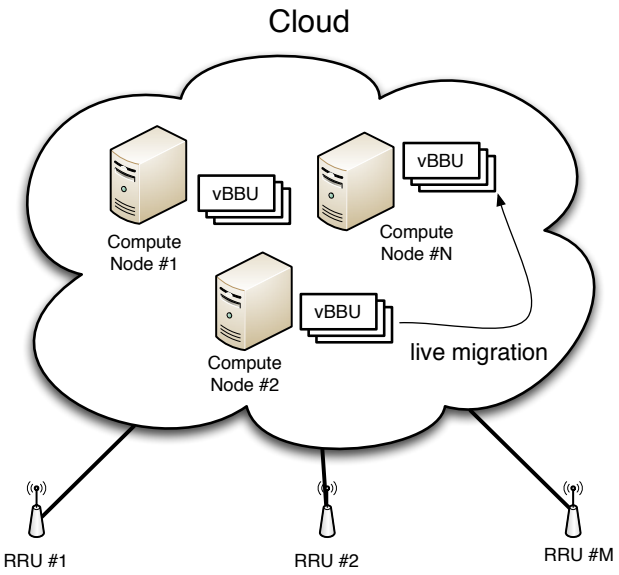


Fig. 6: Cloudified RAN Network

[6]https://www.semtech.com/products/wireless-rf/lora-gateways/sx1302

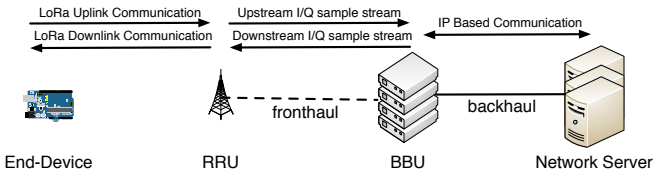


Fig. 7: Cloudified LoRa Network [38]

In essence, on the upstream, the RRU converts analog LoRa radio signals transmitted by end devices over the air into the baseband I/Q sample stream sent over the front haul interface. On the downstream, in turn, the RRU converts the received I/Q baseband sample stream into analog signals sent over the air interface towards end devices with regular LoRa chips.

The overall communication on the front haul interface may be characterized by the following:

- Upstream: the RRU receives LoRa radio signals, converts them into a baseband I/Q sample stream using an ADC/DDC (through SDR), and forwards resulting samples over the Internet (*e.g.,* using an Ethernet connection) to the cloud signal processing unit, the BBU.
- Downstream: the BBU provides LoRa signals to the RRU, which are encoded in the form of a baseband I/Q sample stream, converted to an analog signal by the RRU via the DUC/DAC module (through SDR), and propagated over an antenna.

The RRU fully replaces a LoRa GW allowing for the communication in the LoRaWAN under the assumption that *(i)* the upstream baseband I/Q sample stream is sent over the front haul and appropriately processed at the later stage as well as *(ii)* a necessary downstream baseband I/Q sample stream (*e.g.,* coding DL LoRa signals) arrives at the RRU on time through the front haul interface. Typically, the front haul interface, *i.e.,* the RRU-BBU link meets much higher demands in terms of throughput, latency, and jitter in comparison to the backhaul interfaces between the GW and NS. This is related to radio signals handled as the I/Q sample stream of high bandwidth requirements. This work implements the IF5 functional split as defined by [1] (*i.e.,* working with the baseband I/Q sample stream).

The BBU executes a LoRa modem implemented in software and provisioned as a VNF over an RRC. The task of the LoRa modem is to demodulate and decode in real-time LoRa signals provided within the baseband I/Q sample stream received from the RRU over the front haul on the upstream.

In essence, on the upstream, the LoRa modem accepts the real-time baseband I/Q sample stream on input and output byte-streams containing distinct LoRa packets originating at LoRa end devices. The BBU provides every message decoded toward the NS on the upstream through the backhaul interface for further processing (*e.g.,* using a Datagram (DGRAM) message).

The NS processes messages received over the backhaul interface. If an end device requests a response, the NS originates a downstream transmission. The downstream message is sent over the backhaul toward the corresponding BBU (*e.g.,* using a DGRAM message). The BBU encodes a response message (*i.e.,* DGRAM message) and sends the corresponding downstream I/Q sample stream to the RRU on the front haul interface. Coding and modulation is provided by the LoRa modem, which accepts a byte-stream on input and outputs the corresponding baseband I/Q sample stream on the downstream.

The RRU sends the sample stream to a DUC/DAC (*i.e.,* SDR), which converts it into an analog signal that can be amplified and broadcast into the air via an antenna. The signal may reach an end device using a standard LoRa chip, which would decode the message and send it to higher layers in the communication protocol stack.

2) Containerization and Orchestration

The docker-based containerization used in our setup provides for a uniform resource abstraction layer for all virtual entities, *i.e.,* RRU, BBU, and NS, described in the following subsections. Docker containers are more light-weight than full VMs, and are better adapted for signal processing VNFs [28]. The docker platform provides the orchestration tool *docker-compose*, which stores the configuration of the entire container bundle using Yet Another Markup Language (YAML). As a result, a single orchestration action can start multiple containers implementing a specific composed software function, *i.e.,* cloudified LoRa, or LoRa as a Service (LaaS). Furthermore, docker-compose lets the user scale services according to the momentary load by spawning new containers when required. Therefore, docker is the foundation for the LoRa C-RAN specified and implemented in this work. Docker.io can also equally well deploy functions among open cloud-computing platforms, such as OpenStack[7], which can improve the usability of the system.

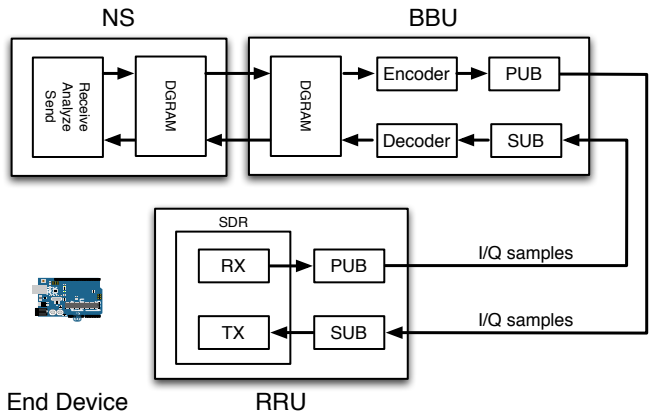


Fig. 8: C-RAN Socket Communications Between Components

3) Remote Radio Unit

The RRU (*cf.* Fig. 8) is a pure GNU Radio-based component [11]. Physically, it consists out of the computing node equipped with a Lime SDR[8] attached through USB and antennas installed on appropriate input and output ports of the SDR device. Logically, the architecture uses the Publish/Subscribe (PUB/SUB) model. RRU uses two Radio Front-end components, *i.e., SDR RX* and *SDR TX*, which correspond to the physical RX and TX of the SDR device.

The RX component of LimeSDR provides an incoming signal as a stream of 64-bit wide I/Q samples and publishes them in a Zero Message Queue (ZMQ)[9]-based Publish (PUB) socket. The first task of the RRU is, therefore, to publish I/Q samples toward registered subscribers, *i.e.,* BBUs. The second task is related to the processing of downstream I/Q samples originating at the BBU. The I/Q sample stream is received through a Subscribe (SUB) socket from the BBU and forwarded toward the TX block of the LimeSDR, which emits corresponding LoRa waveforms through the air. It is noteworthy that the RRU does not decode/encode LoRa packets on its own, but *passively* forwards the sample stream between a remote BBU and the local RF component.

4) Baseband Units

Physically, a BBU (*cf.* Fig. 8) is a cloud-native computing resource, while logically it is a GNU Radio [11] environment running a LoRa [35]/LTE [28] decoder/encoder. It is worth noting that the BBU does not contain any physical Radio Front-end components, such as SDR RX or TX radio chains. The BBU receives the I/Q sample stream from the ZMQ SUB socket and passes it further towards the local decoder. The decoder, in turn, processes LoRa signals and forwards messages decoded to the User Datagram Protocol (UDP)/DGRAM message socket leading toward the NS. Moreover, the BBU shall use the UDP socket to receive a byte-stream from the NS, provide the received byte-stream towards an extended version of the encoder component [35], and send the corresponding I/Q samples (*i.e.,* received from the encoder) towards the PUB socket, publishing I/Q samples on the downstream toward the subscribed RRUs.

Note that the downstream chain is currently only partially implemented because the downstream packet is built by the NS and delivered directly to the RRU, bypassing the BBU.

5) Network Server

An NS (*cf.* Fig. 8) receives and logs data packets received from the BBU through an upstream UDP socket. Furthermore, the NS requests a downstream transmission by contacting the BBU through its downstream UDP socket. In the current implementation, the NS bypasses the BBU and directly provides the RRU with the corresponding I/Q sample stream on the downstream.

6) Communication Diagram

All communication between the components is implemented through sockets. The ZMQ messaging library is deployed here since it offers reliable communication schemes using the Request-Reply or PUB/SUB paradigms. GNU Radio offers ZMQ blocks by default, while Transmission Control Protocol (TCP) source/sink blocks for socket-based communication are still available but deprecated. For the RRU-BBU com-

[7]https://www.openstack.org/

[8]https://limemicro.com/products/boards/limesdr

[9]https://zeromq.org/

munication, the PUB/SUB paradigm is applied unless CRIU restoration has to be supported, *cf.* below.

7) Notes on the Implementation

a) Zero Message Queue: All PUB/SUB blocks, as well as the accompanying data flow, are shown in Fig. 8. The SDR RX chain generates an I/Q sample stream, which is published by the RRU through the PUB socket. While BBU's SUB socket subscribes to the RRU PUB socket, BBU receives I/Q samples from the RRU. Please note that because a TCP connection is used, I/Q samples from the RRU RX radio chain arrive intact at the BBU decoder [35]. The BBU exports decoded LoRa messages towards the NS, which is implemented with the help of a dockerized Python script through a UDP communication. The NS receives decoded LoRa messages provided over UDP and, if desired, feeds I/Q samples of the answer down to the BBU through a DGRAM socket. The BBU encodes the message, which is then propagated towards the PUB communication channel of the BBU. The RRU SUB socket subscribes to the BBU PUB socket, which closes the communication loop. Finally, the message is emitted through the air through the TX radio chain of the RRU.

The advantage of ZMQ is that sockets can ignore *time out* or *disconnect* actions. Thus, subscribing sockets can be started before publishing sockets without interference. The subscribing socket can wait for the publishing socket to be instantiated. In this architecture, this means, in particular, that all docker containers for the RRU and BBU can be started in any order. Furthermore, additional BBU instances can be added at run-time, while the PUB/SUB paradigm allows for new subscribers and publishers to join at any moment.

b) GNURadio TCP and UDP source and sink blocks: Due to the fact that BBU using the ZMQ communication was not restoring properly upon evaluation (*cf.* Table III), the ZMQ communication between RRU and BBU was replaced with the *GrNet*[10] UDP implementation. GrNet sends I/Q sample stream using the UDP socket communication and can replace the ZMQ communication. The UDP message size can be set arbitrarily. However, the message size has to be configured as a multiple of the I/Q sample size. When the *complex* type is used, multiples of 64 bits have to be configured.

B. Long Term Evolution Centralized Radio Access Network

The LTE C-RAN architecture is composed of three main components: BBU, RRU and UEs. The centrally located BBU has the responsibility of allocating network resources with various conflicting objectives (resource, throughput, and energy) and constraints (power, throughput, CPU, and memory) [10]. At the same time, the remotely placed radio units are primarily used for the transmission and reception of signals to and from UEs, respectively. In LTE C-RAN, different layers of the radio protocol stack can be executed at the centralized BBU pool (*i.e.,* RCC) or closer to the edge, at the radio units (*i.e.,* RRU). This division of the protocol stack execution and the exact functional split deployed can be determined by a host depending on factors, such as the available processing resources at each location or the performance requirements for the users being supported, *i.e.,* more protocol stack layers could be offloaded to the radio unit to enable low-latency communications for users at the edge. Similar to the LoRa C-RAN architecture, the signal processing functions at the BBU can be placed and executed in a cloud environment and run on GPPs.

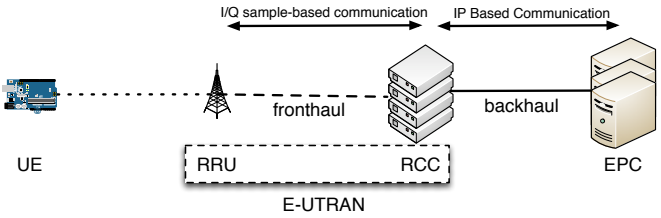


Fig. 9: Cloudified LTE Network

1) Long Term Evolution Centralized Radio Access Network Architecture

The components of the LTE C-RAN architecture can be seen in Fig. 9. The displayed RCC hosts support several RAN-related functions, including Packet Processing Function (PPF), Baseband Processing Function (BPF), *etc.* The RRU is used to handle functions that require special radio hardware such as modulation/demodulation, filtering, signal amplification, Analog/Digital (A/D) conversion. In the Cloud-RAN setup, the typical transceiver and receiver functions of a base station (*i.e.,* RRU) are provided by the Universal Software Radio Peripheral (USRP) B210 SDR board attached to the RRU via a USB 3.0 cable.

Specifically, since the IF4.5 split (Section II-D) is applied, the RRU is configured to perform lower-layer PHY layer processes, such as FFT and IFFT. The EPC contains the NFs, such as MME, Home Subscriber Server (HSS), and S/PGW.

2) Containerization and Orchestration

The uniform resource abstraction layer is the key to the provisioning of network functions. All virtual entities established in the LoRa and LTE C-RAN setup, *i.e.,* RRU, BBU, and NS, may benefit from the universal resource abstraction by utilizing the docker.io containerization. The lightweight nature of docker containers in comparison to full VMs makes them a better-adapted technology for signal processing VNFs [28]. The orchestration of docker containers is similar to VMs. Docker is equipped with an orchestration tool called *docker-compose*, which uses YAML to store the configuration of the system. The system is organized in a so-called bundle of services of the micro-service architecture. As a consequence, multiple containers provisioning various VNFs bundled together to provide a complex use-case (*e.g.,* an LTE C-RAN ecosystem) might be started with a single orchestration action. Docker-compose enables the vertical and horizontal scaling of services according to the momentary network load, by increasing the computational capacity of a given container, or spawning new containers when required. Therefore, the LoRa and LTE C-RAN systems specified and implemented in this work use docker as their foundation. Docker also natively supports cloud management platforms like OpenStack[11] to

[10]https://github.com/ghostop14/gr-grnet

[11]https://www.openstack.org/

deploy its software functions, which improves the system usability and speeds up the deployment phase.

V. Experimental Setup

The experimental setup enables the analyses of the architectures considered and their major characteristics.

A. Long Range Evaluation

The configuration, *cf.* Fig. 10, consists of two standard laptops running Ubuntu 21.04[12] and meeting the requirements of supported applications (*cf.* Fig. 8). The processor is a 64-bit, 4-core x86 processor with a clock rate of 2.70 GHz. Laptops come with an 8GB DDR3 RAM module of a 1,867 MT/s transfer rate and a Samsung 860 EVO Solid State Drive with a 250 GB capacity. Both laptops are equipped with Base-T Ethernet cards supporting a full-duplex 1 Gbps connectivity. The fronthaul interface is provided by connecting the Ethernet cards in those laptops with a category six twisted-pair cable.

A LimeSDR board, with a 2 dBi Sub-Miniature version A (SMA) antenna for the 868 MHz frequency band, is connected with the Compute Node #1 (*i.e.,* RRU) through a USB 3.0 connector. Compute Node #2 is a standard computing node with no extra Radio Front-end components. Both computers are equipped with the Linux kernel 5.10, docker.io version 20.10.8, and CRIU 3.14. Docker.io is configured such that it supports *experimental* features, *i.e.,* checkpointing/restoring containers with CRIU[13]. CRIU is configured with the *tcp-established* option allowing checkpointing process trees, having open TCP connections[14]. GNU Radio modules available on the host and guest docker containers are in version 3.8.2. The end device is an Arduino Mega board with the Semtech SX1276 radio transceiver provided as a Dragino LoRa shield. While the system is still in its early development stages, the focus is currently put on the communication between a single end device and a cloudified LoRa eco-system. It is worth noting that several contributions in this field enable minimal configurations as well, only encompassing a transmitter and the receiver [35], [41], [46].

B. Long Term Evolution Evaluation

1) Hardware and Software Components

To instantiate the LTE-based Cloud-RAN, a real-time implementation of an LTE network provided by the OAI platform was deployed[15]. OAI supports protocol amendments for IoT applications including LTE Cat. M (supported by the *master* OAI branch) and LTE Cat. N (supported in the *develop-nb-iot* branch) [15]. OAI implements 3GPP technology on general purpose hardware and contains the full protocol stack of the LTE 3GPP standard[16], which includes the RAN, EPC, and the UE. The RAN is implemented through an eNodeB application and instantiated as a VNF inside a container. This implementation enables testing multiple network configurations and monitoring network performance and the UEs attached to the network in real-time. The containerized eNodeB is located on a host with an Intel i7-4790 CPU @ 3.60 GHz and offers 32 GB of RAM (*cf.* Fig. 11). To ensure that the eNodeB application is able to meet various processing and scheduling deadlines required by the network (*i.e.,* HARQ deadlines), the eNodeB sees a real-time prioritization on the host, and it applies a deadline scheduling policy for the application process. Furthermore, the eNodeB host runs a 64-bit Ubuntu 16.04 operating system with a low-latency kernel (4.15.0).

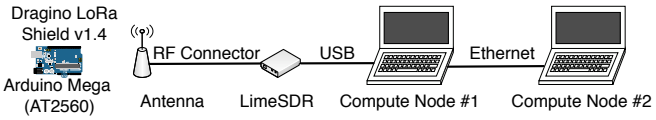


Fig. 10: LoRa Setup

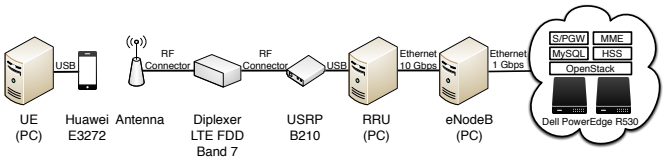


Fig. 11: LTE Setup

The receiver and transmitter functions of a typical BS are realized using a USRP B210 SDR[17] that is connected via a USB 3.0 interface to a host machine running an application that carries out the Low-PHY and Radio Front-end functions of the RAN stack (*i.e.,* RRU operations). The RRU host specifications are: Intel CPU Core i7-3770 @ 3.40 GHz, with 16 GB RAM, running Ubuntu 16.04 and a low-latency kernel (4.4.0). Finally, the UE, which is provided as a Cat. 4 LTE Huawei E3272 device (*cf.* Table I), is run on an Intel CPU i5-2400 @ 3.1 GHz, also running Ubuntu 16.04 and a generic kernel. Following the Cloud-RAN architecture, the eNodeB hosting machine is connected to the RRU host containing via a 10 Gbps Ethernet (10 GbE) cable. Finally, the Core Network/EPC components (MME, HSS, S/PGW) are instantiated as VNFs in an OpenStack Mitaka cloud environment.

2) Cloudification

To cloudify the RAN deployment and enable its delivery as a service, containerization is applied to establish the access network. An eNodeB application is deployed within a docker container delivering the application as a micro-service to take advantage of benefits such as maintainability, flexibility, scalability, and reduced complexity. While in previous work [30] Linux Containers (LXC) were used to deploy the network, docker.io is chosen here as the containerization environment since it offers the option to use CRIU for migration. Compared to other virtualization platforms, such as VMs, docker containers are a more appropriate environment for hosting the eNodeB application, as they do not require a virtualization layer. They also provide other advantages such as running the application directly on the kernel, using less memory

[12]http://releases.ubuntu.com/21.04/
[13]https://criu.org/docker
[14]https://criu.org/TCP_connection
[15]https://gitlab.eurecom.fr/oai/openairinterface5g
[16]Based on LTE Release 8.6; implements a subset of Release 10
[17]https://www.ettus.com/all-products/ub210-kit/

(compared to the VMs), and making runtime executions more efficient [27]. Therefore, there is little performance degradation by deploying the application in such an environment. Finally, docker containers are built on modern kernel features (*i.e.,* cgroups, namespaces, and chroot). These features are essential for ensuring that the host scheduler can meet real-time deadlines [30] needed for the correct operation of the eNodeB application.

TABLE I: LTE Network Parameters

Parameter	**Values**
LTE Release	3GPP Release 8.6
LTE Band	FDD band 7
LTE Frequency	2.5 (UL)/2.6 GHz (DL)
Transmission Mode (TM)	1
Antenna Mode	SISO
System bandwidth	Variable (5, 10, and 20 MHz)
Modulation UL	QPSK, 16-QAM
Modulation DL	QPSK, 16-QAM, 64-QAM
Fronthaul Capacity	10 Gbps Ethernet
CP	Normal
Ethernet MTU Size	1,500 B
UE	Cat. 4 LTE, Huawei E3272

To facilitate two-way communication between the eNodeB, providing the access network and the EPC, a bridge is created by using Open Virtual Switch (OVS) [32]. This supports traffic management between the two network entities (eNodeB, EPC) and a forwards user-received traffic from the eNodeB through an S1-U tunnel to the S/PGW at the core, which uses a Tunnel Endpoint Identifier (TEID) to recognize traffic on a per-user level appropriately, *cf.* [40]. To enable unfiltered communication and access between the containerized eNodeB application and the OVS bridge, the container is bound to the bridge using an SDN-based tool, Pipework[18]. The tool enables us to connect containers in complex scenarios and is a viable alternative to docker bridge networking. From the experiments, it can be concluded that binding the container to a docker configured network did not facilitate the required communication between the application and the rest of the network components since user-defined networks do not allow for direct access to host interface(s).

VI. Evaluation Results

LoRa and LTE results were evaluated, while the implementation was tested in the experimental setup described above.

A. Long Range Results

In-depth evaluation of the LoRaWAN developed is divided into seven cases.

1) System Provisioning Time

First, the provisioning operation is evaluated (*cf.* Table II). The system is composed of three docker containers, namely RRU, BBU, and NS, holding all necessary GNU Radio Elements and instantiated over Compute Nodes #1 and #2. The RRU runs on Compute Node #1, while Compute Node #2 spawns both the BBU and NS. The Ethernet connection becomes the fronthaul of this LoRa C-RAN, while a virtual bridge interface on Compute Node #2 becomes the backhaul. Docker enables caching previously instantiated containers through the docker.io image subsystem. Therefore, service instantiation of those VNFs is a rapid operation, with less than a second completion time. Every test (*e.g.,* the *docker run* operation of the RRU) is repeated five times. The mean value and the standard deviation of the completion time are listed in the table. In all tested situations, the service instantiation time is considered very rapid and remains under 1 s. Saving docking images to file-system storage is costly and requires 20–30 s of processing time. The large size of the RRU and BBU images is the cause of this issue. The reason for that is that the preparation of those docker containers depends on the compilation of software provided as sources. In such a case, the development toolchain have to be installed (*e.g.,* gcc, g++, and cmake), and some libraries have to be delivered in their development versions, which makes the docker container heavy. As an example, RRU needs to compile gr-limesdr[19], which is a LimeSDR module for GNU Radio, while BBU has to compile gr-lora [35] with our changes. The restoration of images, in turn, is a quick operation and requires a couple of seconds to complete. It is worth noting, however, that saving and restoring images can be done in the preparatory phase of docker container migration. Therefore, saving, copying, and loading images do not pose significant delays in the docker.io migration process.

TABLE II: docker.io Processing Delays

Images	Run [ms]	Stop [ms]	Save [s]	Load [s]
RRU	235±10	50±1	22.06±0.57	1.70±0.12
BBU	228±29	42±2	23.01±1.60	1.32±0.13
NS	313±80	40±1	07.31±0.44	3.01±0.03

2) System Migration

The checkpointing experience of LoRa BBU was studied with CRIU (*cf.* Table III). First, the regular BBU application is started, which uses a TCP/ZMQ socket to receive I/Q samples from the RRU in the IF5 functional split and a UDP socket to communicate with the NS. The application can be checkpointed and restored successfully with CRIU, *i.e.,* all operations complete successfully, however, the TCP/ZMQ socket is not functional anymore, thus, the BBU application is not receiving any I/Q samples after restoration. The experiment is repeated in a containerized environment, where the container is checkpointed through the docker.io checkpointing subsystem, providing a similar outcome. Again, no I/Q samples are received after the BBU container restoration. Therefore, the TCP/ZMQ communication was replaced with a UDP-based socket communication on the fronthaul in RRU and BBU resulting in a UDP socket opened on the BBU for the

[18]https://github.com/jpetazzo/pipework

[19]https://github.com/myriadrf/gr-limesdr

fronthaul purposes. This allowed for successful checkpointing and restoration of BBU application. Furthermore, the communication on the RRU–BBU axis was functional. It allowed an end device to successfully report information towards the network NS on the upstream. Moreover, packets originated at the NS were successfully sent to the end device downstream in the LoRa Class A operation (*i.e.,* also after restoration).

TABLE III: BBU Evaluation of CRIU

Execution	Fronthaul Sockets	Checkpoint	Restore	Functional
Host	TCP/ZMQ	✓	✓	✗
Guest	TCP/ZMQ	✓	✓	✗
Host	UDP/grnet	✓	✓	✓
Guest	UDP/grnet	✓	✓	✓

When the guest functionality was confirmed after checkpointing and restoration (Table III), the performance statistics were gathered on the processes, which are displayed in Table. IV. Both checkpointing & restoration of the BBU were repeated five times. The mean value and the standard deviation of the completion time are listed in the table. It is worth noting that checkpointing and restoration are very efficient processes completing within a sub 2 s time frame. The total BBU downtime due to the service migration (*i.e.,* checkpointing+restoration) is evaluated at the level of sub 3 s.

TABLE IV: CRIU Processing Delays

Image	Fronthaul Sockets	Checkpoint [ms]	Restore [ms]
BBU	UDP/grnet	1,403±6	1,314±6

3) Demodulation and Decoding Processing Time

According to [37] end devices in LoRa networks, such as TTN, periodically report tiny data chunks with a periodicity ranging from 0–300 hours. The signal on the BBU is sampled with 1 MS/s (*i.e.,* Mega Samples per s), while the end device sends LoRa packets with a 125 kHz bandwidth, Spreading Factor (SF) 7, and Code Rate (CR) 1 PHY parameters. Note that with a 125 kHz BW and SF 7, every symbol coding SF bits, *i.e.,* chirp, lasts for 1.024 ms, while the lowest CR 1 introduces redundancy resulting in the 4/5 effective CR. Decoding is far more computationally expensive than encoding, since the signal must go through detection, synchronization, demodulation, deinterleaving, dewhitening, decoding, and packet reconstruction.The BBU implementation [35] is benchmarked in Fig. 12, which shows the processing time for tiny LoRa packets received. The figure displays the time, the BBU decoder stays in a given state, where:

0) detection,
1) synchronization to the signal,
2) the detection of the start of the frame delimiter,
3) pause (skipping samples),
4) header decoding,
5) payload decoding.

The processing time for such small packets is established at the level of less than 10 ms, *cf.* Fig. 12, in which the decoder goes from idle through all states 1-5 and comes back to the detection state again. This is good since for SF 7, one chirp lasts around 1 ms. Therefore, the processing time is shorter than the length of the frame preamble of around 12 ms. This allows for LoRa signal processing in real-time. Note that SF 7 has the shortest symbol duration in the LoRa PHY specification. Hence, it stresses the computing infrastructure the most, while chirps for higher SFs last longer, *i.e.,* when SF increases by one, the symbol duration doubles. In our experimental setup, the most considerable anomalies were encountered in the discovery of the frame delimiter, where for the 4-Byte packet, the receiver completely lost the synchronization with the transmitter, and 5-Byte packet required a longer duration to find the SFD. According to our benchmarks, this arises from a test verifying the correctness of time synchronization based on the threshold value received from a correlation coefficient between the instantaneous frequency of the locally generated chirp and the received chirp (please consult [35], [36] for a more detailed explanation). When the synchronization is dramatically low, the receiver breaks the synchronization (*cf.* the 2-to-0 state transition experienced by the receiver upon the 4-Byte packet processing). Furthermore, when the synchronization is too little, the receiver adjusts the synchronization, which can result in a longer processing time in state 2 (*cf.* 5-Byte packet).

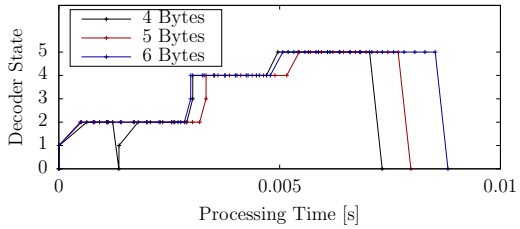
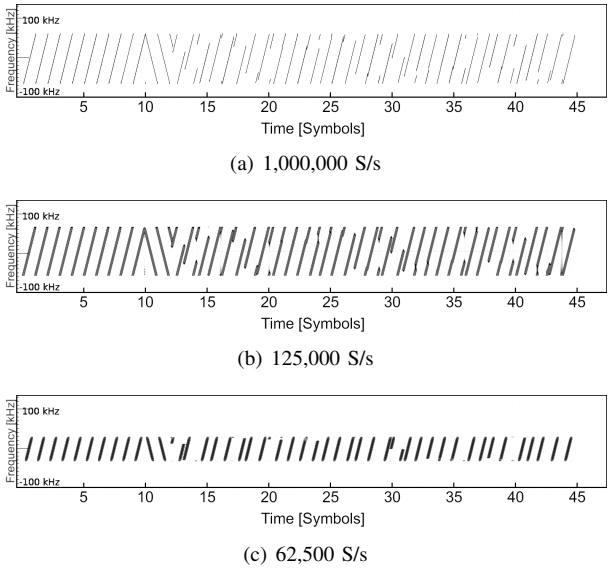


Fig. 12: LoRa Decoder Signal Processing Time

4) Fronthaul Network Utilization



(a) 1,000,000 S/s

(b) 125,000 S/s

(c) 62,500 S/s

Fig. 13: The Effect of the LoRa Signal Sampling Rate

Monitoring the fronthaul yields 335 Bps on idle. Once the C-RAN is instantiated, and the connection between RRU and BBU is established, the network utilization raises to a fixed

8 MiB/s. The theoretical value can be derived the following way: LimeSDR sends 1 MS/s of complex type (*i.e.,* 2×32 b), which results in the data traffic of 64 Mbps. Overhead of TCP and IP is 40 B (20 B each); the Maximum Segment Size (MSS) is, therefore, 1,460 B assuming the 1,500 B Maximum Transmission Unit (MTU). 64 Mbps requires 5480 TCP packets/s of MSS 1,460 B, which results in the total overhead of 1.76 Mbps. Hence, 64 Mbps (data) + 1.76 Mbps results in the 7.8 MiB/s fronthaul load.

The RRU constantly sends samples to the BBU despite the LoRa network utilization, therefore, the fronthaul load stays at the constant level. The load of the fronthaul may be, however, reduced. According to the Nyquist-Shannon sampling theorem [43], [53], a sufficient sampling rate f_s is needed to sample the LoRa signal according to the Nyquist-Shannon limit. The LoRa signal bandwidth in the experiment is set to 125 kHz, where the signal frequency varies between -62.5 kHz and 62.5 kHz with respect to the central frequency. Thus, the authors believe that the minimum sampling frequency required to decode a LoRa signal of 125 kHz bandwidth is 125 kS/s, which is 1/8 of the previous sample rate of 1 MS/s. To verify this experimentally, an example samples UL LoRa signals in the time and frequency domain for various sampling rates (*cf.* Fig. 13). It is noteworthy that the LoRa signal is correctly represented for both sampling frequencies of 125 kS/s and 1 MS/s, *cf.* Fig. 13(a)–(b). The signal varies between -62.5 kHz and 62.5 kHz and provides the clear visibility of particular CSS modulated chirps. On the other hand, the signal sampled with 62.5 kS/s, *cf.* Fig. 13(c), does not contain the same information as the other signals, *cf.* Fig. 13(a)–(b). It is, therefore, concluded that lowering the sampling rate below the Nyquist-Shannon limit of 125 kS/s should result in the signal being definitely unsuccessfully decoded. The implementation [35] used in this work properly decodes LoRa signals for 250 kS/s, 500 kS/s, and 1 MS/s. It does not work with the signals sampling rate of 125 kS/s. The most recent code version [35] does not even allow setting the sampling frequency below 250 kS/s. Following this, it can be concluded that the minimal fronthaul data load for the tested LoRa transceiver in the case of the 125 kHz channel is 2 MiB/s with sampling at the level of 250 kS/s (*cf.* Table V), as the decoder uses oversampling at the level of 2. In the case of better implementation, the sampling rate could be hypothetically further reduced to 125 kS/s as reported by [53].

TABLE V: Sampling Rates and Network Utilization

Samples per second	Max. Signal Frequency[20]	Network Utilization	Decode Success in Experiment
1,000,000	500 kHz	8 MiB/s	✓
500,000	250 kHz	4 MiB/s	✓
250,000	125 kHz	2 MiB/s	✓
125,000	62.5 kHz	1 MiB/s	✗

Various network delays for the outgoing ZMQ-based sample stream between the RRU to the BBU on the fronthaul interface were tested (*cf.* Table VI). Overall, the higher the delay, the lower the network utilization was measured on the fronthaul since the TCP does not maintain an appropriate throughout. Without additional delays, RX and TX, respectively, on the fronthaul are as expected at 8 MiB/s. However, when the delay on the fronthaul reaches $600\,\mathrm{ms}$, the network utilization drops to 2.32 MiB/s.

TABLE VI: Effect of Delay on Network Traffic and the Decoding Process

Fronthaul Delay [ms]	Fronthaul Load [MiB/s]	Decode Success
0	8	✓
300	8	✓
400	5.5	✓
600	3.6	✗

5) Cost of the Long Range Centralized Radio Access Network

LoRa GWs come at various price ranges. The low cost TTN GW costs around 70 US$, while the high-end counterpart sells for 300 US$. Since the bandwidth consumed by a regular GW is minimal, the cost of maintaining a regular LoRa GW is considered a one-time investment. Currently, SDR devices do not incur elevated costs, as a regular LimeSDR, which are used as part of our cloudified LoRa setup (*cf.* Section IV-A3), costs 300 US$. Considering the OPEX incurred by running a virtualized BBU over Amazon Web Services[21], the following parameters are used to calculate the cost of such a deployment: Compute (*i.e.,* Amazon EC2 instance), and data transfer, (*i.e.,* ingress and egress traffic), respectively.

In the case of an EC2 instance, and specifically when an *a1.medium* VM flavour which comes with 1 vCPU and 2 GiB of memory is selected, the costs could approach *0.0255 US$* per hour. In order to support the maximum possible signal bandwidth of 500 kHz in the US or 250 kHz in EU, the cloudified system requires 8 MiB/s and 4 MiB/s of Ingress traffic, in the two regions, respectively. Thus, for the European case 4 MiB per second equals 363 GB per day. Ingress traffic is free on AWS, giving a daily cost of 0.00 US$.

Egress data transfers, on the other hand, are attached with a cost. The amount of data sent depends on how many UL signals (here assuming Class A LoRa devices) trigger a DL signal on the DL, which can be of variable length in LoRaWAN. The 3 B DL payload is assumed to be fixed (excluding preamble, header, and CRC), SF 7, and CR 1, which yields an I/Q sample stream of around 248 kB (including preamble, header, and CRC) after coding and modulation and has a LoRa air-time of 25.86 ms. The EU specifies a 1% duty cycle for the 868.0–868.6 MHz frequency band [49]. With this, a DL signal can be sent every 3 s. This means that a maximum of 28,800 DL signals of this type can be sent by one GW a day. The Egress data transfer is, in such a case, limited to 7 GB a day, if duty cycles are respected, *i.e.,* $28,800 \times 248$ kB = 7 GB. TTN encourages end devices to avoid DL transmissions, since GWs are not designed to send and receive simultaneously. Assuming that about 30% DL messages are used, the egress traffic sets

[20]For complex/quadrature sampling used in this work, the baseband signal can be centered at 0 Hz, and spans from $-f_M$ to $+f_M$, where f_M is the Maximum Signal Frequency indicated in the table.

[21]https://calculator.s3.amazonaws.com/index.html

at the level of around 2.1 GB/day. The total monthly cost of a cloudified BBU is evaluated at the level of 24.07 US$, which includes the Amazon EC2 Instance at 18.67 US$, ingress traffic at 0 US$, and egress traffic at 5.40 US$, overall clearly acceptable. Furthermore, the OPEX is expected to go down in the future with the future development of cloud data centers and wired networks. Decreasing prices should, therefore, drive the development of LoRa C-RAN in the future.

6) Long Range DL Messages

Current LoRa modulator/demodulator implementations [35] focus on modulating/demodulating UL signals. To materialize a cloud RAN with a successful UL and DL communication, small modifications in the implementation are necessary. The LoRa encoder [35] currently available only provides UL messages (compatibility with, *e.g.,* SX1302 GW chips). To produce a DL LoRa signal, a DL transmission was first recorded from a regular GW to an end device. Differences between UL and DL transmissions are studied. By observing similarities in the UL, *cf.* Fig. 13(a)–(b), and DL messages, *cf.* Fig. 14, in the time and frequency domain, it is assumed that a DL transmission is complimentary to the UL message, where all up-chirps are converted to down-chirps and vice versa. The regular encoder implements down-chirps, *e.g.,* in UL headers upon SFD (*cf.* Fig. 13(a)–(b) for the time period between 10 to 12.25 symbols), which is provided through a so called complex conjugate (c.c.) of chirps in the modulator [36]. Therefore, a flag was added to the modulator [35], which flips all down chirps up and vice versa on the DL using the already existing functionality.

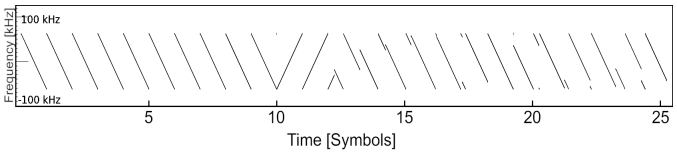
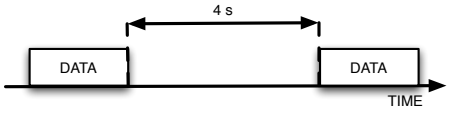


Fig. 14: LoRa DL Message

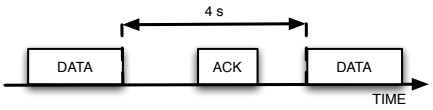
Finally, it was tested whether the DL signal generated is successfully received by the end device equipped with a standard SX1276 transceiver. The current implementation does not display good performance, while only a tiny fraction of packets are decoded on the DL. However, typically, well-tested message byte-streams (*i.e.,* encoder input) can be found, resulting in DL signals (*i.e.,* produced by the encoder) decoded by the regular decoder (*e.g.,* SX1276 on the end device) with high probability. Nevertheless, the regular encoder [35] does not deliver any packets to the end device, while a small fraction of DL messages issued by the modified encoder is successfully decoded by the SX1276 chip, which confirms the hypothesis (*cf.* Table VII).

TABLE VII: DL Decoding by the SX1276 Chip

Encoder	Successful Decoding by SX1276
Regular	✓
Modified	✗



(a) Regular Data Reporting



(b) Data Reporting using ACKs/Retransmissions

Fig. 15: Data Reporting Schemes

7) Long Range Baseband Unit Experienced Service Downtime

To test the influence of the BBU service downtime on the LoRa network, two transmission schemes were used as presented in Fig. 15. The first scheme illustrated in Fig. 15(a) refers to a regular LoRa Class A operation, in which an end device wakes up at any time, reports DATA packets on a regular time basis (*i.e.,* with the packet separation of 4 s), and goes to sleep again. The second scheme displayed in Fig. 15(b) is similar to the first one. An end device issues DATA packets with a time separation of 4 s between two distinct UL messages. The DATA packet issued has to be acknowledged with an ACK packet, *i.e.,* a DL transmission, by the NS within 4 s. If the ACK packet arrives, the subsequent DATA message can be issued. However, if the corresponding ACK message does not arrive at the end device, the previously sent DATA packet is re-transmitted again. This transmission scheme was planned as a simplified LoRa Class A operation. It is worth noting that LoRaWAN specifies two receive windows for Class A devices (*cf.* Section II-A). The first one typically opens 1 s after the UL transmission. If no DL transmission was received in the first window, another reception window could open 2 s after the initial transmission again. Therefore, the ACK-based protocol in this experiment is more lenient because when the initial DATA transmission is over, the device starts constantly listening for at least 4 s. If any DL ACK transmission targets the device within this time frame, the device will receive it. It, therefore, unifies two original reception windows of LoRaWAN Class A into a single longer reception window, relieving the network server from calculating the correct timing of DL transmission windows. If the corresponding ACK arrives within the 4 s window, a new packet can be transmitted as scheduled, *i.e.,* 4 s after the end of previous DATA transmission. Otherwise, the device retransmits the previously unacknowledged packet after 4 seconds expire. Currently, the experiment is carried out in a laboratory setup without any regard for duty cycles, as the ultimate goal is to model the outage time of future LoRaWAN deployments when the cloudified gateway goes through checkpointing and restoration. The appropriate LoRaWAN operation has to be, however, implemented in the future. It will support the timing of DL reception windows as well as duty cycles, which was

decided to be kept outside the scope of this paper.

The configuration of the experiment is the following. The end device and RRU are spaced by 1 m. A single channel at 868.5 MHz is used for UL and DL communication. The 125 kHz BW, SF 12, and CR 1 are set. The DATA packet size is 16 B in the simple reporting scheme and 19 B in the ACK/Retransmission scheme. The ACK packet size is equal to 3 B. The experiment comprises 4 trials. In one trial, 10 UL DATA packets are sent for both schemes. After the successful reception of the third packet, the LoRa BBU is checkpointed and immediately restored, which takes around 3 s to complete.

It is noted that the result of all trials is exactly the same. In the regular scheme, DATA packets 1–3 arrive at the NS, DATA packet 4 is lost on the UL as well as all remaining DATA packets 5–10 arrive at the NS. In the ACK/Retransmission scheme, all DATA packets 1–10 arrive at the NS. Upon transmitting DATA packet 4, the NS does not receive the first instance of the DATA or confirm DATA packet 4 with an ACK. Therefore, the end device does not receive the corresponding ACK, which triggers a retransmission of DATA packet 4. The second instance of DATA packet 4 as well as all remaining DATA packets 5–10 arrive at the NS.

TABLE VIII: Effect of Checkpointing/Restoring on the LoRaWAN Operation

Scheme	DATA pkts sent	DATA pkts received	ACK pkts sent	ACK pkts received
regular	10	9	-	-
ACK/ret.	10 (+1 ret.)	10	10	10

This experiment is presented in summary in Table VIII, which compares two data reporting schemes, *i.e.,* the regular one and the ACK/retransmission-based (ACK/ret). Four metrics are gathered, *i.e.,* the number of DATA packets sent by the end device, the number of DATA packets received by the NS, the number of ACK packets sent by the NS, and the number of ACK packets received by the end device. When the packets are sent 4 s apart from an end device, the number of dropped messages due to checkpointing/restoration is confirmed at 1. However, typical reporting time is highly elevated and spans several hours or days [37]. Therefore, the vast majority of devices will not notice the migrated BBU.

B. Long Term Evolution Results

In-depth evaluations of the architecture developed are provided for three key parameters.

1) Service Provisioning Time

TABLE IX: Service Provisioning Time

	eNodeB	HSS	MME	S/PGW
Average (s)	6	0.20	0.13	0.14
MAX (s)	6	0.21	0.17	0.16
MIN (s)	6	0.19	0.10	0.11

The RAN components (*cf.* Sec. V-B2) were instantiated by deploying them in docker containers, hosted across two different machines (BBU and RRU), essentially provisioning them as containerized services. Thus, service instantiation times of containers were measured (*cf.* Table IX), which indicated that the instantiation time of a RAN service took approximately $6\,\mathrm{s}$ on average across various runs. Part of the time required to instantiate the RAN is based on the instantiation of the USRP B210 SDR, which also takes a few seconds to start. This instantiation time could include the time required to load firmware into the USRP B210 board that provides the Radio Front-end for the setup. A similar metric was collected for EPC VM instances, *i.e.,* MME, HSS, S/PGW (*cf.* Table IX). It can be derived that the instantiation times of EPC components are relatively short, as compared to the eNodeB. The increased instantiation time of the eNodeB is likely due to the fact that the LTE software modem was compiled from sources, which makes the container heavy.

2) System Migration–Observations

To assess the effects of live migration on the network, the experiments monitored and evaluated the outcome of the network by emulating the migration process (*i.e.,* checkpointing and restoring the eNodeB application). From the observations on the migration process, it had been noticed that the RAN network could continue operations and remain synchronized with the rest of the network (*e.g.,* UE and eNodeB), despite downtime periods of up to $8\,\mathrm{ms}$, which is effectively the period of a single HARQ ACK. However, for longer downtime periods, such as those that would be required for a stop-copy-restore migration (*e.g.,* 1,314 ms needed to restore the BBU in LoRaWAN), the system synchronization breaks. This causes the eNodeB application to trigger a *ue release context* to the EPC MME[22]. As a result, the network-connected UEs lose their active network connections, and the average throughput of the system instantly drops to 0 Mbps. Note, however, that users remained attached to the network during the period, in which the eNodeB was down (or check pointed) and restored subsequently.

TABLE X: Fronthaul Throughput Rates based on Physical Resource Block (PRB) configuration

PRBs (bandwidth)	Fronthaul Throughput UL	Fronthaul Throughput UL
25 (5 MHz)	79.2 Mbps	80.8 Mbps
50 (10 MHz)	144.8 Mbps	146.4 Mbps
100 (20 MHz)	276.8 Mbps	278.4 Mbps

3) Network Performance

To determine the ability of the configured network to support various services, the average throughput was measured for different BBU cell configurations running on bare metal and in a container environment (*cf.* Fig. 16 and Fig. 17). Based on the cell configuration parameters used for the evaluation of the LTE network performance (LTE Release 8.6, Single Output Single Input (SISO), LTE FDD Band 7), the available and expected DL throughputs in the OAI-based network (*i.e.,* Release 8.6) for bandwidths of 5 MHz, 10 MHz, and 20 MHz using a Cat. 4 UE, are 17 Mbps, 34 Mbps and 69 Mbps, respectively[23]. Furthermore, the upper theoretical limit for user

[22]Based on the observation of the OAI eNodeB and EPC logs

[23]https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/doc/FEATURE_SET.md

TABLE XI: Fronthaul Load as a Function of Throughput and Bandwidth

Technology	Bandwidth	Sample Rate	I/Q Sample Size	Throughput	Functional Split	Fronthaul Load
LoRa UL	125 kHz	125 kS/s	64-bit	8 Mbps	IF5	64 bps/Hz
LoRa UL	125 kHz	125 kS/s	32-bit	4 Mbps	IF5	32 bps/Hz
LTE-FDD UL	20 MHz	30,720 kS/s	32-bit	278 Mbps	IF4.5	13.9 bps/Hz
LTE-FDD UL	10 MHz	15,360 kS/s	32-bit	146 Mbps	IF4.5	14.6 bps/Hz
LTE-FDD UL	5 MHz	7,680 kS/s	32-bit	80 Mbps	IF4.5	16 bps/Hz

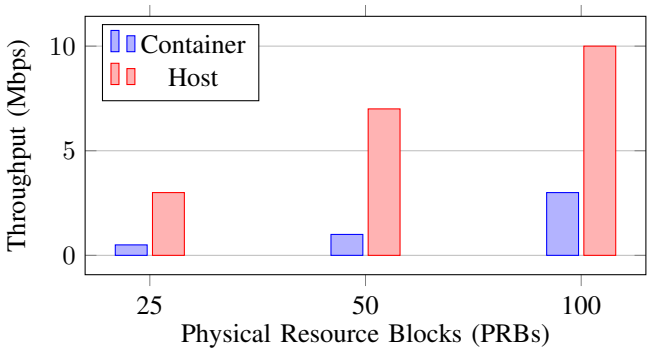


Fig. 16: Average Throughput (UL)

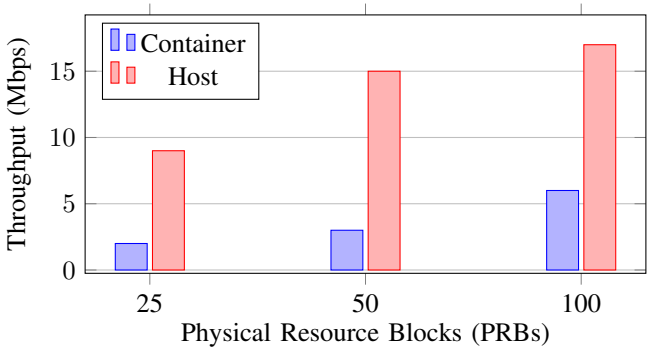


Fig. 17: Average Throughput (DL)

plane throughput is 75.376 Mbps[24] for an LTE FDD SISO configuration with 20 MHz channels. Nevertheless, while a lower UE performance was experienced and displayed by a host-based OAI RCC, this was not caused by the separation of eNodeB into two primary components, *i.e.,* RRU and RCC. Similar throughput (*i.e.,* UL and DL) was experienced on a UE connected to a non-cloudified eNodeB running on a dedicated PC, confirming that the rather limited performance is a result of the OAI platform. Furthermore, placing the RCC in the docker container had a degrading impact on the experienced throughput. However, the cause of this phenomenon needs to be investigated further.

Finally, despite the limited measured throughput, streaming High Definition (HD) 1080p videos is possible at the UE using the described network setup as well as browsing other generic websites with little to no delay. However, to ensure the network implemented could support strict requirements for services envisioned for 5G networks, such as enhanced Mobile Broadband (eMBB), massive Machine Type Communication (mMTC), and Ultra-Reliable and Low Latency Communication (URLLC) services, it will require modifications to different layers of the OAI RAN protocol stack (such the current Physical Layer implementation). Such considerations are outside the scope of the current work.

4) Front Haul Network Utilization

Table X shows fronthaul measurements for different network configurations. As expected, the fronthaul throughput (UL and DL) increases as the number of PRBs increases. It is observed that fronthaul throughput for 20 MHz LTE C-RAN network, based on the IF4.5 split (*cf.* Section II-D) reaches 276.8 Mbps UL and 278.4 Mbps DL, respectively. Those values agree with the fronthaul throughput experimentally verified by [5]. The theoretical IF5 throughput in LTE-FDD assuming a 20 MHz channel, 32-bit I/Q sample size (OAI with USRP B210), and the LTE basic sampling rate of 30.72 MHz [44] equals to 32 bits$\times$30.72 MHz = 983.04 Mbps. Based on the experimental setup used in this work, it is worth noting that the USRP B210 sample size (*i.e.,* for I or Q channels) is 12-bit wide[25]. OAI, in turn, works with a 16-bit sample resolution. Therefore, the designed OAI setup, which utilizes a USRP B210 as the SDR, uses 32 b to encode an I/Q sample. Due to CP removal and compression in IF4.5, the fronthaul throughput was reduced to 276.8 Mbps UL and 278.4 Mbps DL, respectively, which corresponds to approximately 28% of the initial IF5 throughput (*i.e.,* IF4.5 compression ratio of around 72%).

The LoRa C-RAN network, based on the IF5 split (*cf.* Section II-D) required at least 8 Mbps on the fronthaul interface for 125 kHz LoRa spectrum with 64-bit I/Q sample resolution. However, the fronthaul load can be further reduced by using a shorter sample resolution as in the case of OAI with USRP B210 (*i.e.,* 32-bit). Thus, the fronthaul load could be further reduced to 4 Mbps for 125 kHz channels. This effectively highlights the efficiency of the IF4.5 split in the LTE C-RAN in terms of throughput requirements compared to the IF5 split used in the LoRa setup. The LTE IF4.5 requires sending around 14–16 bps/Hz, while LoRa IF5 is at the level of at least 32 bps/Hz (*cf.* Table XI). Even if LTE samples with a little bit higher sampling rate than required (*i.e.,* 30.72 MS/s) instead of 20 MS/s for 20 MHz channels (due to certain particularities of the sub-carrier spacing and the channel width in LTE), the compression rate of IF4.5 split (*i.e.,* 72%) is so high that spectrum-wise LTE with IF4.5 split is more efficient in transporting I/Q samples than IF5 split in LoRa by a factor of around 57%. Therefore, the specification of an efficient split for LoRa communication is of high importance, as a significant amount of bandwidth can be spared on the fronthaul interface.

[24]This value reaches 100.8 Mbps on the PHY layer as reported in [44]

[25]https://www.ettus.com/wp-content/uploads/2019/01/b200-b210_spec_sheet.pdf

VII. Summary and Future Work

This new approach presented the design and experimentation of live migration of a containerized BBU. Besides its functional design and prototypical implementation, limitations in doing so are due to current deficits of the CRIU tool used by docker.io for live migration. Nevertheless, the demonstration of the first successful migration of a dockerized LoRa BBU with CRIU was possible due to changing all open sockets to UDP communications. While it would be possible to migrate the eNodeB application, particular concerns may arise, when the application is restored at the destination node, such as the complete re-synchronization of the entire network (EPC-RAN and RAN-UE).

Looking ahead, a recent proposal by the European Telecommunications Standards Institute (ETSI) to co-deploy Multi-Access Edge Computing (MEC) nodes at C-RAN sites motivates the study of more intelligent MEC/C-RAN resource management approaches. Thus, in future steps, contention-induced performance prediction of RAN VNFs, as a result of being possibly co-located with other VNFs or non-RAN applications/services, can be studied. The information predicted on RAN VNF performance can be used to proactively and dynamically determine the optimal functional split(s) between the C-RAN BBU and the RRUs. Such an approach will prevent performance degradation of RAN-related applications, such as the eNodeB. The adoption of Open-Radio Access Network (O-RAN) as an architecture based on openly available and commercial hardware to provide intelligent radio control in 5G and Beyond networks can also be investigated in the context of live migration of NFs. Thus, the work presented here can serve as a reference point for such future research. Specifically, novel approaches to support the migration of hardware-accelerated NFs in such an open and programmable network architecture can prove to be crucial for the successful deployment of O-RAN-based networks.

References

[1] 3GPP, "Study on New Radio Access Technology: Radio Access Architecture and Interfaces, (TR 38.801, Version 14.0.0, Release 14)," Mar. 2016.

[2] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[3] A. M. Alba, J. H. G. Velásquez, and W. Kellerer, "An Adaptive Functional Split in 5G Networks," in *IEEE Conference on Computer Communications Workshops (Infocom Wkshps'19)*, 2019, pp. 410–416.

[4] H. Ben Arab, "Virtual Machines Live Migration," Technical Rep. University of Passau, Germany, Mar. 2015, https://www.researchgate.net/publication/273574310_Virtual_Machines_Live_Migration, Last-Accessed: December 4, 2021.

[5] C.-Y. Chang, R. Schiavi, N. Nikaein, T. Spyropoulos, and C. Bonnet, "Impact of Packetization and Functional Split on C-RAN Fronthaul Performance," in *IEEE Conference on Communications (ICC'16)*, 2016, pp. 1–7.

[6] I. Chih-Lin, J. Huang, Y. Yuan, S. Ma, and R. Duan, "NGFI, the xHaul," in *IEEE Globecom Workshops (GC Wkshps'15)*, Dec. 2015, pp. 1–6.

[7] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines," in *2nd USENIX Conference on Networked Systems Design & Implementation (NSDI'05) – Volume 2*. USA: USENIX Association, 2005, pp. 273–286.

[8] F. Daquan, L. Lifeng, L. Jingjing, Z. Yi, Z. Canjian, and K. YING, "Ultra-Reliable and Low-Latency Communications: Applications, Opportunities and Challenges," *Science China Information Sciences*, vol. 64, no. 2, pp. 1–12, 2021.

[9] C. Delacourt, P. Savelli, and V. Savaux, "A Cloud RAN Architecture for LoRa," in *URSI GASS 2020*. Gent, Belgium: URSI, Sep. 2020.

[10] W. Ejaz, S. K. Sharma, S. Saadat, N. Muhammad, and N. Chughtai, "A Comprehensive Survey on Resource Allocation for CRAN in 5G and Beyond Networks," *Journal of Network and Computer Applications*, vol. 160, p. 102638, Mar. 2020.

[11] GNU Radio, "About GNU Radio," https://www.gnuradio.org/about/, Last-Accessed: December 4, 2021.

[12] K. Govindaraj and A. Artemenko, "Container Live Migration for Latency Critical Industrial Applications on Edge Computing," in *23rd IEEE Conference on Emerging Technologies and Factory Automation (ETFA'18)*, vol. 1, 2018, pp. 83–90.

[13] H. Guo, K. Wang, H. Ji, and V. C. M. Leung, "Energy Saving in C-RAN based on BBU Switching Scheme," in *IEEE Conference on Network Infrastructure and Digital Content (IC-NIDC'16)*, 2016, pp. 44–49.

[14] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A Survey of LoRaWAN for IoT: From Technology to Application," *MDPI Sensors*, vol. 18, no. 11, p. 3995, 2018.

[15] C.-Y. Ho, R.-G. Cheng, J.-W. Chen, and C.-S. Liu, "Open NB-IoT Network in a PC," in *IEEE Globecom Workshops (GC Wkshps'19)*, 2019, pp. 1–6.

[16] C.-L. I, J. Huang, Y. Yuan, and S. Ma, "5G RAN Architecture: C-RAN with NGFI," in *5G Mobile Communications*, W. Xiang, K. Zheng, and X. S. Shen, Eds. Cham, Switzerland: Springer International Publishing, Oct. 2017, pp. 431–455.

[17] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, R. Schmidt, and N. Nikaein, "Preventing RLC Buffer Sojourn Delays in 5G," *IEEE Access*, vol. 9, pp. 39 466–39 488, 2021.

[18] M. Kanj, V. Savaux, and M. Le Guen, "A Tutorial on NB-IoT Physical Layer Design," *IEEE Communications Surveys Tutorials*, vol. 22, no. 4, pp. 2408–2446, 2020.

[19] C. Kuilin and D. Ran, "C-RAN the Road Towards Green RAN," *China Mobile Research Institute, White Paper*, 2011.

[20] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 146–172, 2019.

[21] J. Li and J. Chen, "Supporting Low-Latency Service Migration in 5G Transport Networks," in *Optical Fiber Communications Conference and Exhibition (OFC'20)*, 2020, pp. 1–3.

[22] J. Liu, W. Xu, S. Jha, and W. Hu, "Nephalai: Towards LPWAN C-RAN with Physical Layer Compression," in *Proceedings of the 26th ACM Conference on Mobile Computing and Networking (ACM Mobicom'20)*. New York, NY, USA: Association for Computing Machinery (ACM), 2020.

[23] LoRa Alliance, "About LoRaWAN," https://lora-alliance.org/about-lorawan, Last-Accessed: December 4, 2021.

[24] Lora Alliance, "What is LoRaWAN," https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf, Last-Accessed: December 4, 2021.

[25] L. Ma, S. Yi, N. Carter, and Q. Li, "Efficient Live Migration of Edge Services Leveraging Container Layered Storage," *IEEE Transactions on Mobile Computing*, vol. 18, pp. 2020–2033, Sep. 2018.

[26] B. E. Mada, M. Bagaa, T. Tale, and H. Flinck, "Latency-Aware Service Placement and Live Migrations in 5G and Beyond Mobile Systems," in *IEEE Conference on Communications (ICC'20)*, 2020, pp. 1–6.

[27] S. Matoussi, I. Fajjari, S. Costanzo, N. Aitsaadi, and R. Langar, "5G RAN: Functional Split Orchestration Optimization," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1448–1463, 2020.

[28] N. Nikaein, R. Knopp, L. Gauthier, E. Schiller, T. Braun, D. Pichon, C. Bonnet, F. Kaltenberger, and D. Nussbaum, "Demo: Closer to Cloud-RAN: RAN as a Service," in *21st ACM Annual International Conference on Mobile Computing and Networking (MobiCom'15)*, 2015, pp. 193–195.

[29] N. Nikaein, R. Knopp, F. Kaltenberger, L. Gauthier, C. Bonnet, D. Nussbaum, and R. Ghaddab, "Demo: OpenAirInterface: An Open LTE Network in a PC," in *20th ACM Conference on Mobile Computing and Networking (MobiCom'14)*. New York, NY, USA: Association for Computing Machinery (ACM), 2014, pp. 305–308.

[30] N. Nikaein, E. Schiller, R. Favraud, R. Knopp, I. Alyafawi, and T. Braun, "Towards a Cloud-Native Radio Access Network," in *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, C. X. Mavromoustakis, G. Mastorakis, and C. Dobre, Eds. Cham, Switzerland: Springer International Publishing, Nov. 2017, pp. 171–202.

[31] L. Oliveira, J. J. Rodrigues, S. A. Kozlov, R. A. Rabêlo, and V. H. C. de Albuquerque, "MAC Layer Protocols for Internet of Things: A Survey," *MDPI Future Internet*, vol. 11, no. 16, pp. 1–42, 2019.

[32] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The Design and Implementation of Open VSwitch," in *12th USENIX Conference on Networked Systems Design & Implementation (NSDI'15)*. USA: USENIX Association, 2015, pp. 117–130.
[33] S. Popli, R. K. Jha, and S. Jain, "A Survey on Energy Efficient Narrowband Internet of Things (NBIoT): Architecture, Application and Challenges," *IEEE Access*, vol. 7, pp. 16 739–16 776, 2018.
[34] T. Rabia and O. Braham, "A New SDN-Based Next Generation Fronthaul Interface for a Partially Centralized C-RAN," in *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, 2018, pp. 393–398.
[35] P. Robyns, E. Marin, W. Thenaers, and C. Smith, "GNU Radio Blocks for Receiving LoRa Modulated Radio Messages Using SDR," https://github.com/rpp0/gr-lora, Last-Accessed: December 4, 2021.
[36] P. Robyns, P. Quax, W. Lamotte, and W. Thenaers, "A Multi-Channel Software Decoder for the LoRa Modulation Scheme," in *3rd International Conference on Internet of Things, Big Data and Security (IoTBDS'18)*. Setúbal, Portugal: SciTePress, Mar. 2018, pp. 41–51.
[37] E. Schiller, S. Rafati-Niya, T. Surbeck, and B. Stiller, "Scalable Transport Mechanisms for Blockchain IoT Applications," in *44th IEEE LCN Symposium on Emerging Topics in Networking*, Oct. 2019, pp. 34–41.
[38] E. Schiller, S. Weber, and B. Stiller, "Design and Evaluation of an SDR-based LoRA Cloud Radio Access Network," in *16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'20)*, 2020, pp. 1–7.
[39] E. Schiller, N. Nikaein, R. Favraud, K. Kostas, D. Stavropoulos, I. Alyafawi, Z. Zhao, T. Braun, and T. Korakis, "Network Store: Exploring Slicing in Future 5G Networks," in *10th ACM Workshop on Mobility in the Evolving Internet Architecture (MobiArch'15)*. ACM, Sep. 2015, pp. 8–13.
[40] E. Schiller, N. Nikaein, E. Kalogeiton, M. Gasparyan, and T. Braun, "CDS-MEC: NFV/SDN-based Application Management for MEC in 5G Systems," *Computer Networks*, vol. 135C, pp. 96–107, Feb. 2018.
[41] B. Seeber, Bastille Threat Research Team, IQ Donors, T. Telkamp, and C. Swiger, "GNU Radio OOT Module Implementing the LoRa PHY," https://github.com/BastilleResearch/gr-lora, Last-Accessed: December 4, 2021.
[42] Semtech, "How to Use LoRa Basics (TM) Station," p. 9, Sep. 2020, https://lora-developers.semtech.com/documentation/tech-papers-and-guides/how-to-use-lora-basics-station/download-how-to-use-lora-basics-station/, Last-Accessed: December 4, 2021.
[43] C. Shannon, "Communication in the Presence of Noise," *Proceedings of the Institute of Radio Engineers (IRE)*, vol. 37, no. 1, pp. 10–21, Jan. 1949.
[44] B. Stiller, E. Schiller, and C. Schmitt, "An Overview of Network Communication Technologies for IoT," in *Handbook of Internet-of-Things*, S. Ziegler and M. James, Eds. Cham, Switzerland: Springer, 2020, ch. 12.
[45] J. Tapparel, "Complete Reverse Engineering of LoRA PHY," Telecommunications Circuits Laboratory, EPFL, Lausanne, Switzerland, Tech. Rep., 2019, https://www.epfl.ch/labs/tcl/wp-content/uploads/2020/02/Reverse_Eng_Report.pdf, Last-Accessed: December 4, 2021.
[46] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming, and A. Burg, "An Open-Source LoRa Physical Layer Prototype on GNU Radio," in *21st IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC'20)*, May 2020, pp. 1–5.
[47] T. Telkamp, "LoRa Signals from a Low Orbit Satellite," https://twitter.com/telkamp/status/956900631985475586, Last-Accessed: December 4, 2021.
[48] The Things Industries, "The Things Network," https://www.thethingsnetwork.org, Last-Accessed: December 4, 2021.
[49] ——, "TTN Duty Cycle," https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html, Last-Accessed: December 4, 2021.
[50] R. S. Venkatesh, T. Smejkal, D. S. Milojicic, and A. Gavrilovska, "Fast in-memory CRIU for docker containers," in *Proceedings of the International Symposium on Memory Systems (ACM MEMSYS'19)*. New York, NY, USA: Association for Computing Machinery (ACM), 2019, pp. 53–65.
[51] D. Wubben, P. Rost, J. S. Bartelt, M. Lalam, V. Savin, M. Gorgoglione, A. Dekorsy, and G. Fettweis, "Benefits and Impact of Cloud Computing
[51] M. Waqar, A. Kim, and P. K. Cho, "A Study of Fronthaul Networks in CRANs–Requirements and Recent Advancements," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 12, no. 10, pp. 4618–4639, 2018.
on 5G Signal Processing: Flexible centralization through cloud-RAN," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 35–44, 2014.
[53] X. Xia, Y. Zheng, and T. Gu, "LiteNap: Downclocking LoRa Reception," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 2321–2330.
[54] C. Zeng, F. Liu, S. Chen, W. Jiang, and M. Li, "Demystifying the Performance Interference of Co-Located Virtual Network Functions," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 765–773.

Eryk Schiller received two M.Sc. degrees, one in Electronics and Telecommunications from the University of Science and Technology, and one in Theoretical Physics from Jagiellonian University, Cracow, Poland, in 2006 and 2007. He received a Ph.D. in Computer Science from the University of Grenoble, France, in 2010. He was a postdoctoral scholar at the University of Neuchâtel, Switzerland and the University of Berne, Switzerland. Since 2018, he has been a senior researcher at the University of Zürich UZH, Switzerland.

Jesutofunmi Ajayi received a B.Sc. (Hons) in Computer Science from Brunel University London, United Kingdom. He received an M.Sc. degree in Computer Science from the University of Berne, Switzerland, and is currently pursuing a Doctorate degree at the same university. His current research interests are Next-Generation Mobile Networks, Network Virtualization, and Edge or Fog computing.

Silas Weber received an M.Sc. Degree in Informatics from the University of Zürich UZH, Zurich, Switzerland. He is currently serving as a project manager within the Faculty of Law at the same university. His interests are in Next-Generation Mobile Networks, Long Range (LoRa), Containers, and Cloud Computing.

Torsten Braun received a Ph.D. degree from the University of Karlsruhe, Germany, in 1993. From 1994 to 1995, he was a guest scientist at INRIA Sophia-Antipolis, France. From 1995 to 1997, he worked at the IBM European Networking Centre Heidelberg, Germany, as a Project Leader and Senior Consultant. Since 1998, he has been a Full Professor in Computer Science at the University of Berne, Switzerland. He was Vice President of the SWITCH Foundation from 2011 to 2019. He received Best Paper Awards from IEEE LCN 2001, WWIC 2007, EE-LSDS 2013, IFIP WMNC 2014, ARMSCC 2014 Workshop, and the GI-KuVS Communications Software Award in 2009.

Burkhard Stiller received a Diplom-Informatiker (M.Sc.) degree in Computer Science and a Dr. rer.-nat. (Ph.D.) degree from the University of Karlsruhe, Germany. He chairs as a full professor the Communication Systems Group CSG, Department of Informatics IfI, the University of Zürich UZH since 2004. He held previous positions with the Computer Laboratory, University of Cambridge, U.K., the Computer Engineering and Networks Laboratory TIK, ETH Zurich, Switzerland, and the University of Federal Armed Forces, Munich, Germany. He received Best Paper Awards at IFIP Networking 2005, IFIP/IEEE DSOM 2007, AIMS 2012/2015/2016, IEEE APWiMob 2017, and IEEE ICBC 2021.