Design and Evaluation of an SDR-based LoRa Cloud Radio Access Network

Eryk Schiller, Silas Weber, Burkhard Stiller
Communication Systems Group CSG, Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
Emails: [schiller|stiller]@ifi.uzh.ch, silas.weber@ius.uzh.ch

***Abstract*—Long Range (LoRa) defines a popular modulation scheme based on the chirp spread spectrum technique. It is used in Low Power Wide Area Networks (LP-WANs) for the Internet-of-Things (IoT). Thus, this work here designs, specifies, implements, and evaluates a Cloud Radio Access Network (C-RAN) architecture for LoRa networks, while using *(a)* Software Defined Radios (SDR) to receive/send radio signals and *(b)* Docker to virtualize the setup. *(c)* A software modulator is developed to emit signals on the downlink targeting regular LoRa end-device receivers, such as Semtech SX1276 chips. Finally, the network, processing, and cost requirements of the C-RAN implemented are evaluated.**

I. Introduction

Low Power Wide Area Network (LPWAN) technology offers long-range communication with low power requirements in Internet-of-Things (IoT) use-cases. LPWAN battery powered devices can run for years. For instance, a node sending 100 Byte once a day using LoRa (Long Range) lasts for 17 years [12]. LoRa is a spread spectrum modulation technique and a wireless Radio Frequency (RF) technology for long-range and low power platforms. It has become the de facto technology of choice for IoT networks worldwide [6].

LoRaWAN is an open standard backed by the LoRa Alliance specifying the Medium Access Control (MAC) protocol built on top of the physical LoRa layer (LoRa PHY). LoRaWAN is designed in a bottom up way to optimize LPWANs for battery lifetime, capacity, range, and cost [7]. LoRaWAN operates in the unlicensed Industrial, Scientific, and Medical (ISM) radio band, which is a significant advantage in comparison to similar cellular solutions, such as the LTE Narrow Band (NB)-IoT.

In a typical LoRaWAN use case, an IoT device, such as a sensor, sends its data over the air. A LoRa gateway picks up the signal, decodes it, and forwards the data packet received over the Internet to the network server. If needed, a response message can be scheduled by the network server.

The Cloud/Centralized Radio Access Network (C-RAN) has been demonstrated to be highly beneficial [17]. In C-RAN systems, the software defined physical layer of a given radio protocol is implemented in software and runs in the cloud environment, *i.e.,* cloudification of PHY. The C-RAN architecture suffers from higher operational costs in comparison to typical hardware solutions, in which signal processing is handled by network access points (*e.g.,* LoRa gateways) equipped with special purpose Radio Frequency Integrated Circuits (RFIC). This paper introduces and evaluates an SDR-based LoRa C-RAN architecture, while focusing on costs associated with instantiating and provisioning cloudified LoRa networks, in which the LoRa PHY is cloudified. As a benefit, the LoRa C-RAN will support LoRa PHY protocol amendments without replacing existing physical infrastructures. Furthermore, LoRa C-RAN setups will explore dynamic provisioning and flexible scalability, once LoRa C-RAN will have become a reality.

The architecture of the current LoRa gateway may be divided into several separate components, *i.e.,* the RFIC-based LoRa PHY module, Hardware Abstraction Layer (HAL), Packet Forwarder, and backhaul interface. The functionality of the RFIC module may be significantly reduced, since several signal processing functions may be executed in the cloud environment (*i.e.,* not on the gateway) using a general purpose hardware. In such a cloudified case, a residual gateway could be left with marginal PHY functionality, since demodulation and decoding are performed in the cloud. Therefore, a gateway does not need any specialized hardware, *i.e.,* the SX1302 transceiver[1], found on regular LoRa gateways. Instead, the gateway is equipped with an antenna, an amplifier, Digital to Analog (DAC) and Analog to Digital (ADC) converters, and is addressed as the Remote Radio Head (RRH).

This paper delivers **three main research contributions**, which are based on a brief and general introduction into LoRa, LoRaWAN, and its applications. **1. Architecture and Specification:** This overview leads to implementation details of the novel C-RAN for LoRa. **2. Evaluation:** Major network-related requirements are evaluated experimentally with the prototype developed supporting one end-device. Third, as the LoRa PHY is closed source, there is no official documentation on how the LoRa PHY is implemented. **3. Extension of Existing Uplink Signal Encoder:** All existing implementations are reverse engineering attempts with a various degree of success and they focus on decoding LoRa signals transmitted by a regular hardware. However, the successful LoRa C-RAN is required to successfully decode uplink signals and also encode downlink signals. Thus, to achieve this the new extension developed provides the ability to generate downlink signals being compliant with regular LoRa end-devices using the Semtech SX1276 chip[2].

[1]https://www.semtech.com/products/wireless-rf/lora-gateways/sx1302
[2]https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276

The remainder of this paper is structured as follows. While Section II surveys LoRa and LoRaWAN as related work, Section III introduces the C-RAN architecture for cellular networks in terms of an architectural overview, a system specification, and implementation details of the new LoRa C-RAN developed. The steps needed to move from a traditional setup toward a C-RAN architecture are presented. Advantages as well as drawbacks of the C-RAN approach are discussed. Section IV describes C-RAN measurements, being complemented with a cost evaluation of the system. Finally, Section V draws conclusions and outlines future steps.

II. Related Work

The key aspects of LoRaWAN and LoRa PHY are presented in this section, however, a more detailed description of LoRa can be consulted in [18]. The main characteristics of C-RAN are also discussed but it is worth noting that to best of our knowledge, LoRa C-RAN (*i.e.,* cloudification of LoRa PHY) was not discussed in the literature to date. Furthermore, the details of major parameters relevant for comparative measurements are discussed as well.

A. LoRaWAN

A LoRaWAN network architecture is a star-of-stars topology. Gateways relay messages between end-devices and a central network server. Gateways are connected to the network server via IP connections, converting RF signals into IP packets and vice-versa [2]. End devices are not associated with any specific gateway. Messages are sent by a node in a destination-less manner and can be received by multiple gateways simultaneously. Each gateway forwards the message received to the network server, which filters redundant message instances, provides security checks, and forwards messages to the appropriate application server [7]. The network server is also responsible for scheduling downlink messages.

Three classes of end-devices are distinguished. *Class A* (uplink/downlink transmissions) is the default class supported by all LoRaWAN devices, in which end-nodes initiate communications. After an uplink transmission on an end-device, two downlink reception windows are opened to receive a response enabling the two-way server-device communication. LoRaWAN is an ALOHA-based protocol, where an end-device can initiate the transmission at any given moment of time. For Class A devices, downlink transmissions from the server have to wait for an uplink transmission from a given end-device and cannot be initiated directly by the network server. Class A devices show the lowest power consumption. A *Class B* (two-way communication with the deterministic downlink latency) device opens extra receive windows at explicitly scheduled times. This is achieved by time-synchronization using beacons, where gateways notify an end-device to open a reception window at a given moment of time. *Class C* (low latency, two-way communication) devices have always open reception windows except at the time when they transmit themselves. A downlink transmission can be initiated by the network server at any time assuming the device is currently not transmitting. Class C devices require more power, and are operated as plugged-in rather than battery powered devices.

B. LoRa PHY

LoRa is designed for long-range and low power communication, where only few bytes are transmitted per day. As an example, LoRa proves the signal reception from a low orbit satellite [5]. However typically, LoRa shows a range of 2-5 km in urban and 15 km in suburban areas [8]

Fig. 1. Uplink Transmission by the SX1276 Chip captured with the Lime SDR board[5]and displayed with Inspectrum[6]

1) Modulation: In North America and Europe the ISM band is located in the 902-928 MHz and 863-870 MHz spectrum range, respectively. LoRa signals in the ISM band are composed of "chirps" modulated using the Chirp Spread Spectrum (CSS) technique [10], [14]. A typical LoRa Bandwidth (BW) is at 500 kHz and 125 kHz in North America and Europe, respectively. Fig. 1 displays an example uplink LoRa transmission sent from an Arduino-based node[7] equipped with a LoRa shield[8] based on regular SX1276/SX1278 transceivers.

The signal presented consists of *up-chirps* and *down-chirps* spread among the available channel. Up-chirps start with a signal of low frequency and increase the frequency over time, while down-chirps (or conjugated chirps) start with the signal of high frequency and decrease the frequency over time (*cf.* Fig. 1).

The uplink signal begins with a so called preamble consisting of 10 unmodulated up-chirps. Those are then followed by two and "a quarter" unmodulated down-chirps that point out the end of the preamble and indicate the subsequent payload. The payload consists of symbols coding the header, the message, and a Cyclic Redundancy Check (CRC) used for final error detection.

2) Spreading Factor (SF): A varying SF (*i.e.,* $SF \in \{7 \dots 12\}$) influences on the air-time, when keeping the constant packet size and BW. Higher SFs typically mean longer range, however, the higher the SF, the longer the time on air.

3) Coding: LoRa signals are encoded to channel with a modified Hamming coding [14] of a given coding rate (CR), where CR denotes the fraction of data carrying the actual information. A channel code provides reliability to noise by introducing redundancy, which results in a longer binary sequence. Four different coding schemes exist, $Cx \in \{1, 2, 3, 4\}$, resulting in different $CR = 4/(4 + Cx)$.

[6]http://limemicro.com/products/boards/limesdr
[6]http://github.com/miek/inspectrum
[7]https://store.arduino.cc/arduino-mega-2560-rev3
[8]https://wiki.dragino.com/index.php?title=Lora_Shield

C. LoRa PHY Implementations in Software

Software-Defined Radios (SDR) provide signal processing components that are usually implemented in hardware. The most popular open-source signal processing framework is GNU Radio [1]. Seven steps in the receiver impact on a successful reception of LoRa signals: detection, synchronization, demodulation, de-interleaving, de-whitening, de-coding, and packet re-construction [3]. Four existing open-source SDR implementations of LoRa [3], [4], [19], [9] are able to successfully decode signals. The first implementation [3] uses a sub-optimal non-coherent CSS demodulator, which depends on a high Signal-to-Noise Ratio (SNR). The second implementation [4] provides an optimal coherent demodulator, however, de-whitening was not appropriately handled, which resulted in SF and CR-dependent whitening sequences. The third implementation [19] provides appropriate decoder and whitening operations fixing problems existing in [4]. This allows working in the low SNR region. Furthermore, the authors corrected the problems of Sampling Time Offset and Carrier Frequency Offset resulting in more reliable delivery. Other implementations, such as [9], are not documented and the setup is cumbersome [11].

D. Centralized-Radio Access Network

A Radio Access Network (RAN) is a major component in telecommunications, which provides the connection between an end-device, *e.g.,* a mobile phone, and the Core Network (CN) of the telecom provider. In a Cloud/Centralized-RAN (C-RAN), a cloudified BaseBand Unit (BBU) processing radio signals is hosted at a centralized location up to a few kilometers away from the RRH, *i.e.,* the BBU hotel. BBUs operate in a virtual environment and interact with physical resources directly or through hardware abstractions. Typically, BBUs run on a physical servers with each distinct BBU placed in a virtual environment. When LoRa C-RAN becomes reality, LoRa will be able to explore C-RAN benefits reported already in cellular systems. For example, the protocol amendments may be quickly introduced, while all components (*e.g.,* PHY, MAC) are written in software. Furthermore, on-demand provisioning using a pay-as-you-go system may be used, in which computing resources are allocated on demand resulting in decreased CAPital and OPeration EXpenditure (*i.e.,* CAPEX and OPEX).

Typically, analog electric signals may be sampled with a given sampling frequency (*i.e.,* samples per second) using an I/Q sample stream. An I/Q sample contains the real and imaginary part of the complex baseband signal derived from the passband signal at a given moment of time (*i.e.,* centered around the carrier frequency). Typically, communication systems are classified as baseband or passband systems. Baseband corresponds to sending a signal without modulation (*i.e.,* frequency shifting). In LoRa, this relates to frequencies around 125 kHz–500 kHz. Passband transmissions shift the baseband signal to a higher frequency, *e.g.,* 868 MHz in Europe. The passband signal may be shifted back to its original baseband frequency. The LoRa signal may be, therefore, effectively encoded with an I/Q sample stream, in which sampling respects the Nyquist-Shannon sampling theorem [16]. The main disadvantage of C-RAN is typically the high network capacity of the fronthaul network transporting the baseband In-phase and Quadrature (I/Q) sample stream exchanged between the BBU and RRH units.

BBUs processing I/Q samples are typically provided as containers, *e.g.,* Docker. This proved to be an adequate design choice in cellular networks, as containers offer near bare metal performance required by RAN functions [13].

III. C-RAN LoRa Architecture

To reach a minimal working environment for a LoRa C-RAN, such that comparative measurements are possible, the gateway functionality is divided into distinct RRH, BBU, and Network Server (NS) components. Furthermore, the environment is virtualized and runs on general purpose Commercial-off-the-Shelf (COTS) hardware. The RRH, BBU, and NS are interconnected, and a simple NS processes uplink messages and schedules downlink transmissions when/if required.

A. Architecture Overview

Fig. 2 displays the high level architecture with all components of the modular architecture involved. It contains three computing entities, *i.e.,* BBU, RRH, and NS. The RRH has an SDR attached, *e.g.,* through the USB port, to send and receive radio signals using the SDR radio chain.

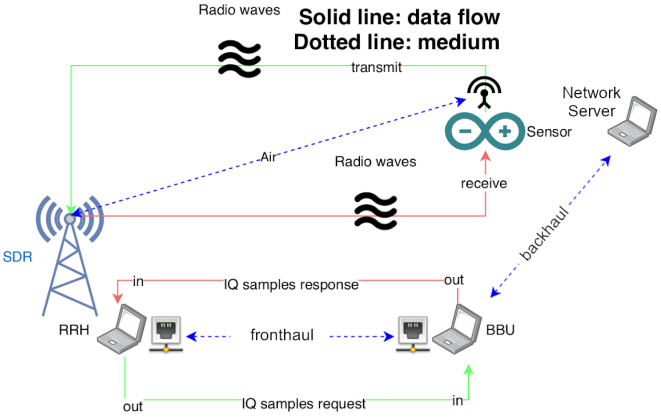

Fig. 2. High Level C-RAN Architecture

Regular devices generating and receiving LoRa-compliant signals are this architecture's clients. As such, regular sensors (*e.g.,* Arduino-based) equipped with legitimate hardware, *e.g.,* SX1276, may benchmark the reverse-engineered software-based LoRa PHY implementation.

B. End-to-End (E2E) Connectivity

Radio waves coming from sensors on the uplink are picked up by the RRH, which samples analog signals and sends the corresponding 32-bit I/Q sample stream over the fronthaul network to the BBU (*cf.* Fig. 2). The BBU entity runs a software-based LoRa modem as a Virtual Network Function (VNF). The virtual LoRa modem demodulates and decodes

the I/Q sample stream. The message decoded is sent to the NS for processing on the upstream using the backhaul interface. In case a response is requested, the NS originates a down-stream data packet toward a BBU through the backhaul interface. This response packet is encoded by the BBU and sent as resulting down-stream I/Q samples to the RRH on the fronthaul interface. The RRH provides the sample stream received toward the attached SDR, which in turn emits the analog radio signal through the outgoing radio chain so that the signal is transmitted back to the Arduino on the downlink. Finally, the Arduino device receives the LoRa waveform using the regular SX1276/SX1278 radio transceivers.

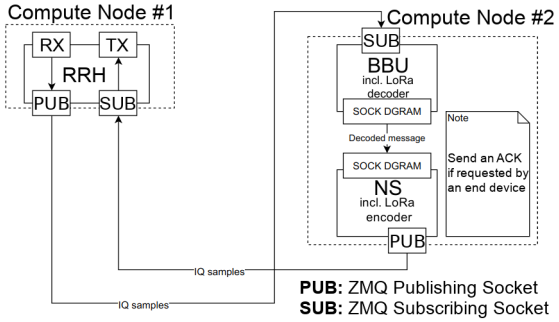

Fig. 3. C-RAN Socket Communications Between Components

C. Containerization and Orchestration

Utilized Docker containerization provides a uniform resource abstraction layer for all virtual entities, *i.e.,* RRH, BBU, and NS (*cf.* Sections. III-D, III-E, and III-F). Docker containers are more lightweight than full Virtual Machines (VM) and are better adapted for signal processing VNFs [13]. Docker provides the orchestration tool *docker-compose*, which stores the configuration of the entire system 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. Therefore, Docker is the foundation for the LoRa C-RAN specified and implemented in this work. Docker can also equally well deploy functions among open cloud-computing platforms, such as Openstack[9], which may improve the usability of the system.

Moreover, in the future, Docker will allow for LoRa PHY protocol amendments, while a new version of the container may be quickly developed incorporating a new software-based LoRa PHY available on the market. Furthermore, Docker will allow for on-demand system scaling as LoRa gateways may be instantiated and disposed to optimize the system according to the current situation. Those features are not yet, however, explored in this paper, while first, the cloudified LoRa system has to be developed, and its costs have to be evaluated.

D. Remote Radio Head (RRH)

The RRH (*cf.* Fig. 3) is a pure GNU Radio-based component [1]. Physically, it consists out of the computing node equipped with a Lime SDR[10] 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. RRH uses two RF components, *i.e., SDR Receiver* (RX) and *SDR Transmitter* (TX), which correspond to the physical RX and TX radio channels of the SDR device. The RX component of the LimeSDR provides an incoming signal as 32-bit wide I/Q samples and publishes them in a Zero Message Queue (ZMQ)[11]-based Publish (PUB) socket. The first task of the RRH is, therefore, to publish I/Q samples toward registered Subscribers (SUB), *i.e.,* BBUs. The second task is related to the processing of down-stream 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 Lime SDR, which emits corresponding LoRa waveforms through the air.

E. Base-Band Unit (BBU)

Physically, a BBU (*cf.* Fig. 3) is a native cloud-based computing resource, while logically it is a GNU Radio [1] environment running a LoRa decoder [3]. It is worth noting that the BBU does not contain any physical RF 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 to the local decoder. The decoder, in turn, processes LoRa signals and forwards messages decoded to the UDP message socket leading toward the NS. Moreover, the BBU shall provide the downstream UDP socket to receive a byte-stream from the NS, an extended version of the encoder component [3], and a PUB socket publishing I/Q samples on the downstream toward the subscribed RRH.

The message received from the NS on the downstream socket shall be provided toward the encoder, which in turn produces I/Q samples with the corresponding LoRa signal that are forwarded toward an RRH through the PUB socket. Currently, the downstream chain is not implemented, since the downstream packet is directly assembled by the NS and sent toward the RRH bypassing the BBU.

F. Network Server (NS)

An NS (*cf.* Fig. 3) 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 simplified architecture, the NS bypasses the BBU and directly provides the RRH with the corresponding I/Q sample stream on the downstream.

G. Communication Diagram

All communication between the components is implemented through sockets, as stated in Sections III-D, III-E, and III-F. The ZMQ messaging library is deployed here, since it offers N-to-N communication schemes using the Request-Reply or PUB/SUB paradigms. GNU Radio offers ZMQ blocks by

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

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

[11]https://zeromq.org/

default, while TCP source/sink blocks for socket-based communication are still available but deprecated. For the RRH-BBU communication, the PUB/SUB paradigm is applied.

H. Notes on the Implementation

Fig. 3 shows all PUB/SUB blocks alongside with the corresponding data flow. The RRH offers a PUB socket that publishes the I/Q sample stream generated by the SDR RX chain. Therefore, the SUB socket of the BBU subscribes to the RRH PUB socket to receive I/Q samples. Please note that a TCP connection is used, therefore, I/Q samples arrive in tact from the RRH RX radio chain at the BBU decoder [3]. A Python script plays the role of the LoRa NS. The NS receives the decoded LoRa messages sent over on this UDP socket and optionally streams down I/Q samples of the response message over a ZMQ PUB socket. The RRH SUB socket subscribes to the NS PUB socket, which closes the communication loop.

The advantage of ZMQ is that sockets may 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 RRH 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 of time.

IV. Measurements and Results

The experimental setup allows for in-depth evaluations of the architecture developed and its key parameters.

A. Hardware and Software Specification

The setup consists out of two regular laptops running Ubuntu 18.04[12] and following the specifications (*cf.* Fig. 3). The CPU is a 64 bit, 4-core Intel i7-7500U of the x86 family with the clock speed of 2.70 GHz. Moreover, laptops are equipped with 8 GB RAM of type DDR3 with the speed of 1,867 MT/s. Furthermore, a Samsung 860 EVO Solid State Drive of 250 GB is installed. Both laptops are armed with Ethernet interfaces using the full duplex 1,000 Base-T standard. The Ethernet cards are connected with a category 6 twisted pair, which materializes the fronthaul interface.

Additionally, a LimeSDR board equipped with a regular 2 dBi SMA antenna for for the 868 MHz frequency band uses a USB 3.0 port of Compute Node #1 node. Compute Node #2 is a regular computer without any additional RF elements. The end-device is an Arduino Mega board, with Dragino LoRa shield having the Semtech SX1276 radio transceiver. Currently, the focus is on the communication between one end-device and the cloudified system, as the system is in an early development stage. Furthermore, other contributions in this domain also support very limited setups [3], [4], [19].

[12]http://releases.ubuntu.com/18.04/

B. System Provisioning Time

First, the provisioning operation is evaluated (*cf.* Fig. 4). The system is composed of three Docker containers namely RRH, BBU, and NS, holding all necessary GNU Radio Elements and instantiated over Compute Nodes #1 and #2. RRH 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, therefore, service instantiation, *i.e.,* VNFs, is a rapid operation, with less than a second completion time. Two scenarios were tested, when BBU and NS, *i.e.,* BBU+NS, run within the same container or separately, *i.e.,* BBU and NS. In all situations, the service instantiation time is considered very rapid and remains under 1 s.

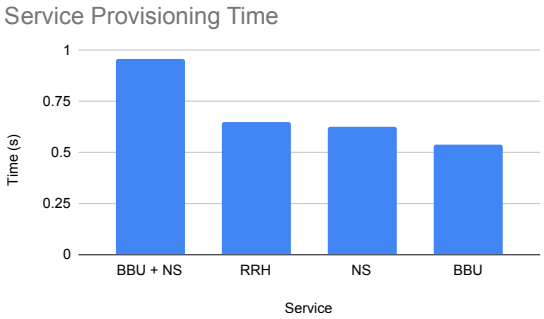

Fig. 4. Docker Provisioning Times

C. Demodulation and Decoding Processing Time

Decoding is considered a far more computing hungry process than encoding. Fig. 5 benchmarks the decoder on the BBU displaying the processing time for small LoRa packets received. Within LoRa networks, *e.g.,* The Things Network (TTN), end-devices periodically report small data chunks with a given periodicity between 0–300 hours [15]. The signal on the BBU is sampled with 1 MS/s (Mega Samples per second), while the end-device sends LoRa packets with a 125 kHz BW, $SF7$, and Cx=1. The processing time for such small packets is established at the level of around 10 ms (*cf.* Fig. 5), which is good, since for $SF7$, one chirp lasts around 1 ms, *cf.* Fig. 1, therefore, the processing time is shorter than the length of the packet preamble (*i.e.,* around 12 ms). This allows for a LoRa signal processing in real-time. Note that $SF7$ has the shortest symbol duration in the LoRa PHY specification, and, therefore, stresses the computing infrastructure the most, while, chirps for higher SFs last longer, *i.e.,* when SF increases by one, the symbol duration doubles.

D. Network Utilization

Monitoring the fronthaul yields 335 Bps on idle. Once the C-RAN is instantiated and the connection between RRH 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 Mb/s. Overhead of TCP and IP is 40 Byte (20 Byte each); the

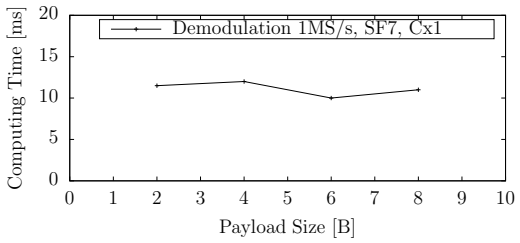

Fig. 5. LoRa Decoder Signal Processing Time

Maximum Segment Size (MSS) is, therefore, 1460 B assuming the 1500 B Maximum Transmission Unit (MTU). 64 Mbps requires 5480 TCP packets/s of MSS 1460 B, which results in the total overhead of 1.76 Mbps, therefore, 64 Mbps (data) + 1.76 Mbps results in 7.8 MiB/s fronthaul load.

The RRH 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 [16], a sufficient sampling rate f_s for a signal with bandwidth BW is given by $f_s \geq 2 \times BW$. The LoRa signal BW in the experiment is set to 125 kHz. Thus the minimum sampling frequency according to the sampling theorem is 0.25 MS/s, which is a quarter of the previous sample rate of 1 MS/s. Lowering the sampling rate below the Nyquist-Shannon limit results in the signal unsuccessfully decoded. It is, therefore, concluded that the minimal fronthaul data load for LoRa in the case of 125 kHz channels is 2 MiB/s with sampling at the level of 0.25 MS/s (*cf.* Table I).

TABLE I
SAMPLING RATES AND NETWORK UTILIZATION

Samples per second	Max. Signal Bandwidth	Network Utilization	Decode Success in Experiment
1,000,000	500 kHz	8 MiB/s	yes
500,000	250 kHz	4 MiB/s	yes
250,000	125 kHz	2 MiB/s	yes
125,000	62.5 kHz	1 MiB/s	no

Various network delays for the outgoing sample stream between the RRH to the BBU on the fronthaul interface were tested (*cf.* Table II). 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 600 ms value is reached, the network utilization drops to 2.32 MiB/s.

TABLE II
EFFECT OF DELAY ON NETWORK TRAFFIC AND THE DECODING PROCESS

Fronthaul Delay [ms]	Fronthaul Load [MiB/s]	Decode Success
0	8	yes
300	8	yes
400	5.5	yes
600	3.6	no

E. Cost of the LoRa C-RAN

LoRa gateways come with various prices. The low cost TTN gateway costs around 70$, while the high-end counterpart sells for 300$. The BW consumed by the gateway is minimal; it is a one time investment. Currently, SDR devices do not incur elevated costs either, as a regular LimeSDR costs 300$. Now let us consider, the costs incurred by running a cloudified BBU over Amazon Web Services[13]. The following three fields have to be considered: Compute, *i.e.,* Amazon EC2 Instance and Data Transfer. In the case of an EC2 instance, selecting *a1.medium* VM, which provides enough capacity, *i.e.,* 1 vCPU and 2 GiB of memory, costs 0.0255$ per hour. The Ingress traffic has to be 8 MiB/s to accommodate the maximum possible signal bandwidth of 500 kHz in the US or 4 MiB/s for 250 kHz in the EU; the ingress traffic is free, *i.e.,* daily cost 0$. Egress Data Transfer on the other hand does cost. In our calculation, we fix the 3 B downlink payload (excluding preamble, header and CRC), $SF7$, and Cx=1, which yields an I/Q sample stream of 248 kB (including preamble, header, and CRC) after coding and modulation. The computed downlink signal has an airtime of 25.86 ms. With the regular European duty cycle of 1%, the downlink signal can be sent every 3 seconds. This means a maximum of 28,800 downlink signals of this type can be sent by one gateway a day. The Egress data transfer is, thus, limited to 7 GB a day. Assuming the 30% downlink utilization, the egress traffic is at the level of 2.1 GB/day. The total monthly cost per a cloudified BBU is acceptable, *i.e.,* 24.07$, including the Amazon EC2 Instance: 18.67$, Ingress Traffic: 0$, and Egress Traffic: 5.40$. Furthermore, the price is expected to go down in the future with the development of cloud data centers and wired networks. Decreasing prices should drive the development of LoRa C-RAN.

F. LoRa Downlink Messages

The LoRa encoder [3] currently available only provides uplink messages (compatibility with, *e.g.,* SX1302 gateway chips). To produce a downlink LoRa signal, a downlink transmission was first recorded from a regular gateway (*cf.* Fig. 6) to an end-device. We studied differences between uplink and downlink transmissions. By observing similarities in the uplink and downlink messages, we assumed that a downlink transmission is complimentary to the uplink message, where all up-chirps are converted to down-chirps and vice versa (compare Fig. 1 vs. Fig. 6). The regular encoder is able to produce down-chirps in the header (*cf.* Fig. 1). Therefore, a flag was added to the modulator [3], which flips all down chirps up and vice versa on the downlink in comparison to the regular uplink encoder. In essence, this is provided through a so called complex conjugate of chirps in the modulator.

Finally, it was tested whether the downlink signal generated is successfully received by the end-device equipped with a regular SX1276 transceiver. The current implementation does not display good performance, while only a small fraction

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

Fig. 6. LoRa Downlink Message

of packets gets decoded on the downlink. Nevertheless, the regular encoder does not deliver any packets to the end-device, while a small fraction of downlink messages issued by the modified encoder is successfully decoded by the SX1276 chip, which confirms the hypothesis (*cf.* Table III).

TABLE III
DOWNLINK DECODING BY THE SX1276 CHIP

Encoder	Successful Decoding by SX1276
Regular	no
Modified	yes

V. CONCLUSIONS AND FUTURE WORK

Moving toward a C-RAN (Cloud Radio Access Network) in LoRa (Long Range) is now proven feasible as the implementation and evaluation performed have shown. To make this happen, LoRa gateways' functionality had to be split into separate RRH (Remote Radio Head) and BBU (Baseband Unit) components, where each can be containerized and run in separate, virtualized environments. However, this approach leads to a measurable amount of fronthaul load of approximately 2 MiB/s between the RRH and the BBU for the 125 kHz BW case as proven experimentally. Since SDRs (Software-Defined Radio) enable the reception and sending of radio signals., especially, the software modulator developed emits successfully signals on the downlink in support of regular LoRa end-device receivers. Nevertheless, such a C-RAN offers more flexibility in the long run against LoRa PHY (Physical Layer) hardware solutions, because amendments to the LoRa PHY can be provided in software, when required, without any infrastructural changes on a broader range. The evaluation of network, processing, and cost requirements of the new C-RAN implemented indicate besides a clear feasibility that communications are possible at a reasonable cost, including the technical efficiency in certain cases.

Thus, the three goals have been achieved successfully, the architecture and its specification, a detailed evaluation of a C-RAN for LoRa, and based on a reverse engineered LoRa PHY the extension of an existing uplink signal encoder to generate downlink signals being compliant with regular LoRa transceivers.

Next steps with respect to the signal processing are on the open-source-based modulation of LoRa signals, since it is still in early development stages. While the extension of the existing implementation to produce downlink signals was successful, the performance of the encoder is not yet at a fully satisfactory level, since signals encoded are not always decoded by end-devices using SX1276 transceivers or gateways using the SX1301 boards. Furthermore, the scalability and an on-demand provisioning of the architecture's prototype can be explored further for additional C-RAN setups.

VI. ACKNOWLEDGEMENTS

This paper was partially supported by (a) the University of Zürich UZH, Switzerland and (b) the European Union Horizon 2020 Research and Innovation Program under grant agreement No. 830927, namely the H2020 Concordia Project.

REFERENCES

[1] "About GNU Radio," https://www.gnuradio.org/about/, last visited 27.12.2019.
[2] "About LoRaWAN," https://lora-alliance.org/about-lorawan, last visited 27.12.2019.
[3] "GNU Radio blocks for receiving LoRa modulated radio messages using SDR," https://github.com/rpp0/gr-lora, last visited 27.12.2019.
[4] "GNU Radio OOT module implementing the LoRa PHY," https://github.com/BastilleResearch/gr-lora, last visited 27.12.2019.
[5] "LoRa Signals from a Low Orbit Satellite," https://twitter.com/telkamp/status/956900631985475586, last visited 27.12.2019.
[6] "What is LoRa," https://www.semtech.com/lora/what-is-lora, last visited 27.12.2019.
[7] "What is LoRaWAN," https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf, last visited 27.12.2019.
[8] F. Adelantado, X. Vilajosana, P. Tuset-peiro, B. Martinez, J. Melià-seguí, and T. Watteyne, "Understanding the Limits of LoRaWAN," no. Sep., pp. 34–40, 2017.
[9] J. Blum, "LoRa Modem with LimeSDR," https://myriadrf.org/news/lora-modem-limesdr, last visited 27.12.2019.
[10] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A Survey of LoRaWAN for IoT: From Technology to Application," *Sensors*, vol. 18, no. 11, p. 3995, 2018.
[11] A. Marquet, N. Montavont, and G. Z. Papadopoulos, "Investigating Theoretical Performance and Demodulation Techniques for LoRa," in *IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Jun. 2019, pp. 1–6.
[12] E. Morin, M. Maman, R. Guizzetti, and A. Duda, "Comparison of the Device Lifetime in Wireless Networks for the Internet of Things," *IEEE Access*, vol. 5, pp. 7097–7114, 2017.
[13] 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 *Proceedings of the 21st ACM Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15, 2015, pp. 193–195.
[14] P. Robyns, P. Quax, W. Lamotte, and W. Thenaers, "A Multi-Channel Software Decoder for the LoRa Modulation Scheme," in *Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security (IoTBDS)*. Setúbal, Portugal: SciTePress, Mar. 2018, pp. 41–51.
[15] E. Schiller, S. Rafati-Niya, T. Surbeck, and B. Stiller, "Scalable Transport Mechanisms for Blockchain IoT Applications," in *IEEE 44th LCN Symposium on Emerging Topics in Networking*, Oct. 2019, pp. 34–41.
[16] C. Shannon, "Communication in the Presence of Noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, Jan. 1949.
[17] B. Sousa, L. Cordeiro, P. Simões, A. Edmonds, S. Ruiz, G. A. Carella, M. Corici, N. Nikaein, A. S. Gomes, E. Schiller, T. Braun, and T. M. Bohnert, "Toward a fully cloudified mobile network infrastructure," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 13, no. 3, pp. 547–563, 2016.
[18] B. Stiller, E. Schiller, and C. Schmitt, "An Overview of Network Communication Technologies for IoT," in *Handbook of Internet-of-Things*, S. Ziegler and J. M., Eds. Cham, Switzerland: Springer, 2020, ch. 12.
[19] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming, and A. Burg, "An Open-Source LoRa Physical Layer Prototype on GNU Radio," *arXiv preprint arXiv:2002.08208*, 2020.