Reference architecture for slicing in LoRAWAN networks

Fabrice Guillemin, Renzo Navas, Alessandro Aimi, Tatiana Aubonnet, Tan-Guy Kerdoncuff, Stefano Secci, Yassine Hadjadj-Aoul, Stéphane Rovedakis, Amina Boubendir

Reference architecture for slicing in LoRAWAN networks

Project Deliverable D.1.1.1

INTELLIGENTSIA project

Executive summary

This document describes the basic functional architecture of the Intelligentisa project. This document reviews all the elements necessary to run a LoRAWAN network and the associated functions. The objective is to integrate the functional architecture into an orchestration framework, possibly by virtualizing some fundamental functions, such as LoRAWAN network Servers (LNSs). This requires to identify basic functional blocks, in particular telemetry exploiting metrics from the substrate virtualized infrastructure as well as supported LoRAWAN networks, and orchestration functions for the deployment and the optimization of LoRAWAN networks to achieve some performance objectives. In a first step, all these issues are considered without slicing considerations and in a second step, the concept of slicing in the context of LoRAWAN networks is introduced.

Authors: Fabrice Guillemin, Renzo Navas, Alessandro Aimi, Tatiana Aubonet, Tanguy Kerdoncuff, Stefano Secci, Yassine Hadjadj-Aoul, Stéphane Rovedakis,

Revised by: All members of the Intelligentsia project.

Approved by: Amina Boubendir

Date: May 31, 2021

Contents

List of Figures

Acronyms

AMF Access and Mobility Management Function.

BBU Base Band Unit.

CSMF Communication Service Management Function.

KPI Key Performance Indicator.

LNS LoRaWaN Network Server.

NFV Network Function Virtualization.
NRF NF Repository Function.
NS Network Service.
NSI Network Slice Instance.
NSMF Network Slice Management Function.
NSS Network Slice Subnet.
NSSAI Network Slice Selection Assistance Information.
NSSF Network Slice Selection Function.
NSSI Network Slice Subnet Instance.
NSSMF Network Slice Subnet Management Function.

PCF Policy Control Function.

SD Slice Differentiator.
SDN Software Defined Network.
SLA Service Level Agreement.
SMF Session Management Function.

UPF User Plane Function.

VIM Virtualised Infrastructure Manager.
VNF Virtualized Network Function.

Chapter 1

General Introduction

1.1 Motivation

Communication network infrastructures are nowadays undergoing a profound transformation related to the softwarization of all their basic components, from core network functions to radio access, including mobile devices and connected objects. Routers, switches, access control functions and handsets together with IoT devices are increasingly built as compositions of software components, each of them potentially following an independent development cycle but eventually working in a composite infrastructure that still ought to be as reliable and performing as possible.

Network virtualization and softwarization technologies enable, on the one hand, the cost-effective operation of network services in the form of network slices while ensuring SLA management; on the other hand, they open the door for the definition of interfaces in support of truly network automation. The research in network automation algorithms is today a green field, with neither established good practices nor sufficient state-of-the-art on models for these novel environments.

The advent of 5G together with the explosion of IoT devices are pushing network operators in the direction of designing flexible infrastructures that have to at the same time (i) be scalable with the number of devices [10], (ii) ensure service level agreements (SLA), (iii) be cost-efficient and (iv) grant high availability and low latency to communication services. As a matter of fact, the number of mobile and IoT devices is already exploding, with usage varying from the monitoring of garbage baskets in cities to bovines in mountains.

For low-latency and high reliability services, such as in the case of public-safety services, SLA management is fundamental in order to be able to discriminate among network slices with different sets of requirements. Although there are contributions in 5G to allow this differentiation in the core network, the integration of IoT radio SLA management in a network automation architecture is not specified today; indeed, even if radio resource blocks can be dynamically assigned for the data plane, the devices access (part of the control plane) is random, which limits these approaches when IoT devices massively access the network [7]. The problem becomes even more challenging in non-5G networks, such as LoRa, where the access is based on an ALOHA-type technique that does not allow resource partitioning by using traditional methods.

The Intelligentsia project aims at tackling this research field by specifying new functional elements, in particular virtual LoRaWAN Network Servers (for short, vLNS) and by exploiting as much as possible the new opportunities offered by virtualization, both for deploying on-demand virtualized network functions and for monitoring deployed ones. The associated monitoring infrastructure will subsequently be used to run Machine Learning algorithms in order to meet the requirements in terms of quality of the services supported by IoT networks. This global process is part of the global orchestration of the network.

1.2 State of the art

Network automation is not, per se, a new research area. Historically, about 10-20 years ago, both Academia and Industry have addressed challenges related to how to get distributed sets of agents to self-organize, to automatically discover themselves and the network states, and to operate necessary reconfigurations of the network. This was for example the focus of the FP6 AUTOI (Autonomic Internet) research project, and a number of research contributions falling in the area of autonomic networks. A complete survey of major contributions in the former area of autonomic networking is given in [3]. We can also cite standardization activities related to this area, as for instance the ones related to the GRASP (Generic Autonomic Signaling Protocol Application Program Interface) protocol architecture [4].

Nonetheless, these pioneering research activities are still lacking a stable reference technical architecture on top of which a decision-making framework could be developed and deployed at a large scale, for instance to solve routing or resource allocation optimization problems. With the advent of network virtualization, and in particular Network Functions Virtualization (NFV) and Software Defined Networking (SDN), the reference building blocks for the upcoming 5G and beyond 5G infrastructures are today quite clearly specified [10]. The relative maturity of NFV/SDN systems and platforms have moved the edge of industry specification efforts to the definition of the interfaces required for realizing network automation for real systems, somehow meeting the expectation of autonomic networking research appeared 20 years ago, but with a much clearer technology environment.

Two working groups at ETSI, the Zero-Touch Network and Service Management (ZSM) and Experiential Networked Intelligence (ENI), are addressing this need and have very recently produced a set of reference documents [5, 6]; similar activities exist at 3GPP. Besides standardization activities, open source network automation platforms have recently emerged, notably the Open Network Automation Platform (ONAP), chosen by many operators like Orange and AT&T as a reference platform for future network automation systems [7, 8, 9]. These platforms [11] and specifications are opening the way to a potentially very large set of network orchestration decisions for which there is a critical need for automation algorithms, and yet a clear method of determining how the state of a fully virtualized and programmable infrastructure, composed of a variety of software modules, should be modeled and inferred in runtime to support resilient and automated network orchestration.

In this global picture, the goal of Intelligentsia project is to propose orchestration and network automation solutions to address the special constraints of IoT services, in order to meet quality requirements of services supported by IoT networks.

1.3 Organization

This report is organized as follows: In Chapter 2, we introduce the various elements of the architecture. In Chapter 3, we describe the functional architecture without slicing. In the subsequent Chapter 4, we introduce the concept of slicing in LoRA networks. Some concluding remarks are presented in Section 5.

Chapter 2

Elements of the architecture

2.1 Global architecture

The objective of the Intelligentsia project is to address three main research directions:

- the specification of novel media access protocols and resource sharing policies able to support network slicing for IoT access networks in general, and LoRa networks in particular;
- the specification and the development of an access gateway dedicated to a domain of IoT devices and based on virtualization techniques to support new traffic engineering operations (virtualized LoRa Network Server, vLNS);
- the design of a network automation framework that incorporates novel learning algorithms to infer in real-time the state of the network and reconfigure accordingly the IoT access network, integrating the new proposed access gateway and IoT device behavior.

To reach these goals, it is necessary to specify a functional architecture. At first glance, the big picture of the envisioned architecture is depicted in Figure 2.1. The various elements composing the architecture are:

- IoT devices transmitting packets of data in the form of radio signals;
- radio gateways receiving and transmitting radio signals, demodulating and decoding them to restore information in the form of bits, which are subsequently transmitted in the form of IP packets to LoRaWaN Network Servers (LNSs);
- LNSs subsequently transmitting packets to the destination network through an IP network.

The major innovation of the project is to exploit the new possibilities offered by virtualization techniques to make LNSs virtual and to instantiate them on the fly depending on radio conditions and on traffic demand. This goal is perfectly in line with the general Virtualized Network Function (VNF) framework, which aims at making VNFs highly flexible and at instantiating them on demand [16]. In turn, this requires a network infrastructure capable of supporting VNFs. For instance, some equipment and functional elements are necessary:

- Data centers to host VNFs and their associated Virtualised Infrastructure Manager (VIM).
- An orchestration platform capable of onboarding VNFs, managing their life cycle (deployment, monitoring and deletion).

This requires that the network infrastructure is equipped with data centers located sufficiently close to devices and capable of exploiting the analytics provided by an ad-hoc monitoring platform. On the basis of these analytics, the orchestration platform has to make decisions in order to place the LNSs and possibly to upgrade them or duplicate them according to traffic conditions.

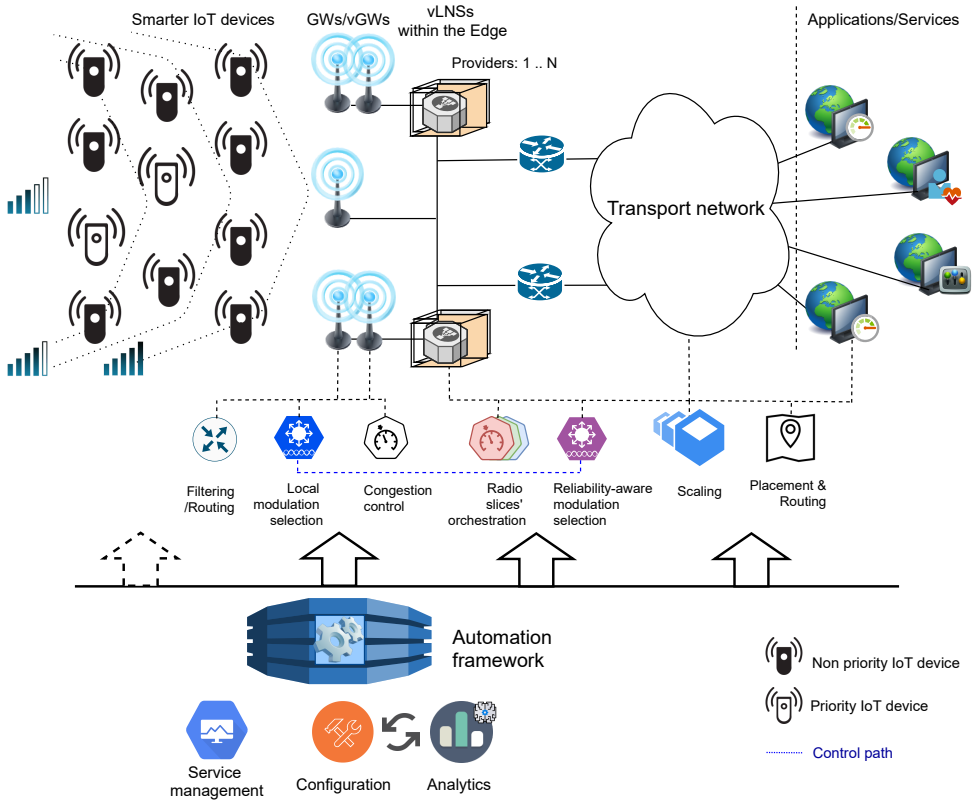

Figure 2.1: Global picture of the Intelligentsia architecture.

The above elements form the minimal set to support a virtualized IoT infrastructure. The Intelligentsia project aims at going one step further by integrating the concept of slice. A slice is composed of a set of devices transmitting information with some requirements in terms of quality (e.g., delay or loss sensitive transmission, availability of the service, etc.). This puts further requirements on the management of the infrastructure:

- resource allocation, which is slice oriented,
- monitoring which is capable of taking into account the slice level (e.g., specific Key Performance Indicators (KPIs), slice oriented probes, etc.);
- the orchestration platform should be able to offer APIs to access the performance of slices.

2.2 Elements of the functional architecture

2.2.1 IoT devices

The LoRaWAN specification defines three device types (Class A, B, and C), which determines for the devices (i) when downlink messages can be received, and (ii) the energy efficiency. All LoRaWAN devices must implement Class A, whereas Class B and Class C extend the specification of Class A.

Class A devices support bi-directional communications between a device and a gateway. These devices are most of the time in sleep mode, and they wake up randomly to send uplink messages. After an uplink transmission, the device then opens two receive windows at specified times (one and two seconds). The server can respond either in the first receive window, or in the second receive window, but should not use both windows.

Class A is the most energy efficient and results in the longest battery life time. Class B devices extend Class A by adding scheduled receive windows for downlink messages sent by

the server. A periodic beacon signal is transmitted by the gateway to synchronize the clock of the devices with the network server. This allows to periodically open receive windows by the devices. Class C devices extend Class A by keeping the receive windows open unless they are transmitting (the devices continuously listen for downlink messages), this allows for low-latency communication but it is more energy consuming than Class A devices. Class C devices require a constant power source.

The majority of literature only consider Class A LoRaWAN devices, because they are the most power efficient. As Class B and Class C devices inherit Class A behavior, it is possible to configure the mentioned parameters for all LoRaWAN classes. In the case of Class B devices, SF and channel can be configured as well for the periodic downlink reception windows with additional MAC primitives.

2.2.2 Radio technology

The term LoRaWAN represents an open standard promoted by the LoRa Alliance that defines Medium Access Control (MAC) and network management protocols on top of the Long Range (LoRa) Physical (PHY) layer, which is instead proprietary of Semtech. LoRa PHY is designed to operate on the unlicensed frequency bands which are defined by local regulations. All the described configurations are the ones available for the 863-870MHz band in Europe (EU863-870).

The innovative aspect of the LoRa PHY layer consists in its radio modulation technique. It is based on Chirp Spread Spectrum (CSS) technology and it allows us to select between one of the available Spreading Factors (SF) for a transmission. Different SFs make it possible to trade bitrate for range. Specifically, a transmission using the subsequent higher SF will take double the time and have half the bitrate. The difference in range is due to the fact that higher SFs make transmissions more robust. The range difference between SFs is not trivial to quantify, because it is highly dependent on the scenario we are in.

In LoRa, the term Data-Rate (DR) is used to identify a coupling of Spreading Factor (SF) and Bandwidth [kHz] used for a transmission. The SF-based data-rates defined by the LoRaWAN specifications are listed in Table 2.1. The usage of DR6 is not very common, as it constraints the number of available frequencies by using double the bandwidth.

Data-Rate	**Configuration**	**Transmission duration of a 59B packet [s]**
0	SF12 / 125 kHz	2.629
1	SF11 / 125 kHz	1.478
2	SF10 / 125 kHz	0.657
3	SF9 / 125 kHz	0.369
4	SF8 / 125 kHz	0.205
5	SF7 / 125 kHz	0.113
6	SF7 / 250 kHz	0.057

Table 2.1: LoRa TX data rates.

Moreover, transmissions happen on a Frequency [MHz]. By definition, a LoRa Channel is an entity associated to a Frequency and one or more allowed Data-Rates, usually the same for every Channel. The EU863-870 LoRaWAN supports a maximum of 16 channels. The three default ones are listed in Table 2.2. Network operators are free to add any frequency in the EU863-870 spectrum given that devices are instructed to use them in compliance with the restrictions defined by the ETSI [EN300.220] standard (see Section 7.2.3, Table 5).

Devices need to comply with law-imposed duty-cycle restrictions: uplink is 1% for every sub-band. This duty-cycle limits the percentage of time you can spend transmitting on a group of frequencies belonging to the same sub-band. In particular, it defines the time a device has

Modulation	Bandwidth [kHz]	Channel frequency [MHz]	LoRa DR/Bitrate	Duty Cycle
LoRa	125	868.10 868.30 868.50	DR0 to DR5 / 0.3-5 kbps	<1%

Table 2.2: Default channels for LoRa transmission.

to wait before being able to transmit again. For instance, if a device transmits for 1s on a 1% duty cycle freq., it will have to wait 99s before being able to transmit again on any freq. of the same sub-band.

Transmissions on different Frequencies do not interfere (i.e., they are orthogonal). Transmissions using different SFs on the same Frequency are instead semi-orthogonal, that is, each one of them suffers of minor interference from the others. In LoRa, each device is assigned a set of available Channels and it will pick one randomly each new transmission, therefore changing Frequency. This technique is called Frequency Hopping and it is claimed to help increase the robustness to interference of the whole system.

The SF or, more precisely, the DR to be used for UL transmissions is stored by the device and can be set dynamically by a MAC level message from the Network Server or by an Adaptive Data Rate (ADR) algorithm in the device. Concerning DL transmissions, the first receive window (RX1) uses the same settings of the UL transmission, while the second (RX2) by default uses freq. 869.525MHz (10% duty-cycle) and SF12.

2.2.3 Radio access gateways

Radio gateways are the infrastructure elements providing LoRa coverage to the end devices. They will receive uplink signals from the devices and transmit the correct LoRa frames to the network server. Uplink signals can be received by more than one gateway, so the LNS may have to deal with duplicated bits of information.

Gateways are usually equipped with a software called the packet forwarder: that software will transmit all frames received by the radio front-end in the form of a UDP stream. Packet forwarders from different manufacturers usually use payloads in a format defined by Semtech, that will contain the payload itself, as well as metadata such as the radio condition in which the packet was received. This UDP stream is also used to emit downlinks through the radio front-end, and to send regular statistics messages to the LNS.

Many LNS manufacturers have made the choice to run an additional agent on the gateway, to prevent from sending the non-secured UDP stream over public networks. This additional agent which is local to the gateway can then add authentication and ciphering to the payloads.

This agent can also provide additional features, such as connectivity checks between the LNS and the gateway, gateway radio (re)configuration, gateway maintenance connectivity, gateway firmware upgrade, gateway uplink cache in case of disconnection from the LNS.

Note that a gateway is also subject to duty cycle limitations, i.e., the fraction of time that the gateway may be busy (see TTN Duty Cycle), and may start dropping packets if a given application sends too many downlink packets (depending on implementation).

It is worth noting that most commercially available gateways are half duplex, meaning in particular that they are unable to receive uplink messages while transmitting data. This might also induce additional latency on downlinks if such a downlink arrives while the gateway is decoding an uplink.

The following elements may be relevant when considering a gateway

- Available Host resources (CPU, RAM, etc)

  - These host resources are relevant indicators of activity peaks when deviating from their nominal values.

- Available downlink duty cycle

  - As any other LoraWan node, gateways are only allowed to emit a given amount of time. Keeping track of how much of this ratio has been consumed will allow to detect cases where a gateway would drop downlinks due to exceeded duty cycle.

- Available time ratio for listening to uplinks

  - in some cases, some gateways are dedicated to the reception of uplinks, and never used for downlinks. Such behaviour could be dynamically controlled in order to react to either uplink or downlink bursts.

- Message processing time

  - This is the time between the instant when a downlink was received from the LNS, and the instant it is supposed to be sent. If the message is received too late, it will be dropped.

2.2.4 Virtual LoRaWAN Network Servers (vLNS)

LoRaWan typical architecture

The LNS is the interface between the LoRaWAN world and the generic IP world. Here is a non-exhaustive list of its features:

- Transmit the messages from the devices to the application servers, and from the application servers to the devices

- Respond to the Over The Air Activation (OTAA) devices' join requests to let them enter the network

- Select the most appropriate gateway for a given downlink message

- Configure the devices through MAC commands defined in the standard (e.g. set network, channels, configure data rates, etc) - see Appendix A.

Two implementations in particular should be mentioned in the context of this project:

- Chirpstack, a quite widely used open source LNS, with sufficient reliability and features to make it production-ready. This implementation has an *application centric* point of view, in the sense that devices will be grouped into profiles, which will be used to configure all the aspects of the communication chain, from device radio setting to application server integration.

- The Acklio LNS, a proprietary LNS developed by one of the members involved in the Intelligentsia project. Already used in industrial deployments, this product allows a fine tuning of devices regarding the radio aspects, while enabling the application server integration to be grouped into device profiles. Such device profiles allow the platform administrator to configure the data flow of a large number of devices (usually serving the same purpose) by routing their uplinks to one or more destination depending on the lorawan fport value in each frame. The product is also compatible with advanced device payload manipulation tools.

While the architecture of an LNS is by no means standardized the following functional blocks have been identified.

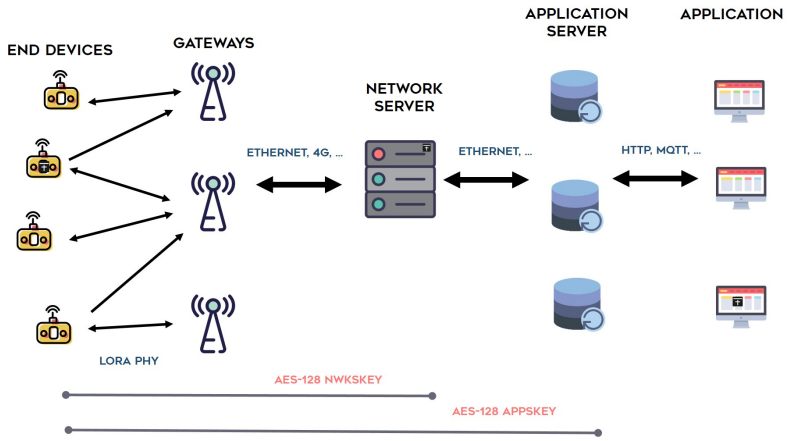

Figure 2.2: Typical LoRaWan architecture.

- **The Network Server** : This component handles all the LoRaWAN protocolar aspects, such as incoming and outgoing LoraWan frames or signaling. While needing minimal state-full data in order to have information on the device sessions, it could quite easily be scaled horizontally depending on the incoming LoraWan load. The minimal state-full data mentionned above are the LoraWan session data of devices (keys, devaddr, counters, etc), as the network server needs these data in their most recent version in order to process payloads. The simplest way of scaling here would be to keep such session in a database able to also scale horizontally (both the acklio LNS and chirpstack use Redis) and rely on this database cluster to warranty up-to-date session info.

- **Connected Device Platform** : also called **Application Server** in chirpstack and in some semtech representations, is the interface to the state-full data of the platform (in database), such as the list of registered devices, their session keys, radio and application profile, etc. It is also used as an entry point for administration, either to configure elements of the LoRaWAN chain (devices or gateways) through a web API, or to interface with application servers (for example through a REST API) to let the client application send downlinks to devices or query other aspects of the LoRa network.

- **The Join Server** : In some cases, it is possible to involve an external actor in the join process of the Over The Air Activation (OTAA) devices, and therefore in the way the payloads are being ciphered. When doing so, the final application server is also often involved. With such a setup, the **Join Server** will interact with the LNS when a device requires access to the network, and will be the element authenticating the device, but will only share with the LNS the keys that will let it route the LoRaWAN packet, but not the payload decryption one. It will, on the other hand, share the decryption key with the final destination server, thus achieving end-to-end payload encryption at the cost of a very tight coupling between the application server and the join server.

The typical architecture of a LoRaWan network without join server is depicted in Figure 2.2.
When we introduce the concept of join server, then we have the architecture displayed in Figure 2.3.

Virtualization of LNS

The three elements mentioned above, as well as their assorted databases can be deployed using containers. In the case of chirpstack, publicly available images can be pulled from dockerhub. Different container orchestration solutions can then be used, such as docker-compose and kubernetes. The different services are usually configured at startup time, with either env variables or configuration files. Such configurations are then usually fixed for the lifetime of the container.

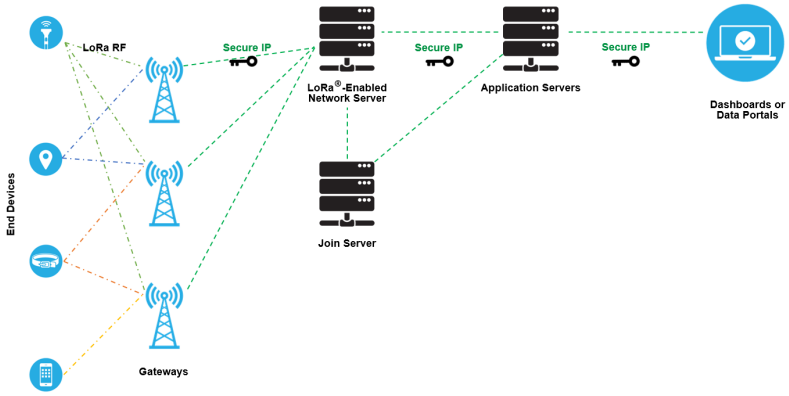

Figure 2.3: Typical LoRaWan architecture.

When scaling up to cope with an increase in traffic, the network server will be the service to duplicate as it handles all the crypto and protocol processing parts. Messages from the gateways are delivered to the network server containers through MQTT, so the MQTT broker will take care of load balancing between the different network servers subscribed to it.

There will however be a need for state sharing of the device session between the network server containers. Such state is stored in a redis database, that can also be scaled up in a multi instance cluster.

For both the network server and redis container, the way to scale up dynamically would be by interfacing with the docker orchestrator in order to increase or decrease the amount of containers for a given configuration.

Finer control over already running container might be necessary depending on use cases. Both the acklio LNS and chirpstack use gRPC for interprocess communication (modern open source high performance Remote Procedure Call). Such an API could also be exploited for this purpose.

In a sense, such container based deployments already respond to many aspects of a VNF, by opposition to embedded LNSs that are often shipped with physical gateways. However, such deployments schemes will mostly work on the separation of the communication chain in the functional communication blocks described above, their deployment, release and initial scaling. While these blocks are relatively clear functionally, even between different implementations, a finer definition of their behaviour could be beneficial, especially in terms of metrics and associated scaling strategies.

There is also some potential improvements on the gateway side where a physical gateway is linked to a single LNS. Virtualized lorawan gateways could be achieved by running more than one gateway agents on a single gateway. This would allow sharing of the radio resources. In particular, uplinks cost almost nothing to forward to their respective LNS when already received on the gateway. Downlink duty cycle is quite precious, and might need enforcement of how it is made available to different virtual gateways.

With regard to monitoring vLNSs, the following elements may be relevant when considering an LNS:

- Available Host resources (CPU, RAM, etc)
  - These host resources are relevant indicator of activity peaks when deviating from their nominal values.
- Duration between device uplink and customer server response
  - A class A device opens a reception window right after an uplink. In order to be able to use this reception window, it is important that all the cloud side processing (LNS

+ Customer server) is done before its expiry. Seeing how this interval evolve might help detect overload cases.

- Device session consistency (missing uplink frame counters)
  - Lorawan frames contain a frame counter incremented at each message. Detecting holes in a device's uplink messages may help detect coverage gaps.
- Device pending downlinks
  - Class A Devices can only receive downlinks after an uplink. So downlinks received by the LNS outside of this reception windows will be kept in queue. An evergrowing downlink queue would most likely be a symptom of poor device coverage, or badly tuned customer side application.
- Gateway overlapping
  - When an uplink is received by more than one gateway, the LNS will receive all of these. Keeping track of the number of gateways that receive uplinks from a given device, and with what radio values (RSSI, SNR) may help evaluate the density of the gateway deployment and how well a given area is covered.
- Message counters and processing time
  - Such counters will provide a direct estimation of the load an LNS is receiving.

2.2.5 Edge cloud

As IoT is traditionally developed at the edge of the network, an ideal placement of vLNS is an edge cloud. A testbed representing an edge cloud of an operator is considered in the Intelligentsia project and a provisional setting of the testbed setting at Cnam, in particular for Task 4.3 activities, is depicted in Figure 2.4; this testbed is intended to test the concepts of the project and is not the fianl test bed, which should involved real radio gateways and IoT devices.

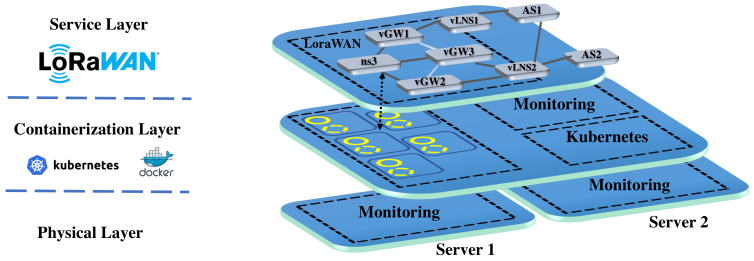

Figure 2.4: Provisional testbed setting for Task 4.3 activities.

This preliminary testbed is composed of two physical servers, where three virtual gateway nodes and two vLNS nodes would be deployed along with an ns3 node meant to generate emulated physical layer traffic. Such a setting is in fact to date already deployed for another use-case on which Cnam started working waiting for the virtualized network functions, and related software, of the virtualized LoRAWAN architecture to be defined.

In terms of forwarding, standard open vswitches hosted in containers could be controlled by a dedicated controller, hence adding additional monitoring points and contributing with related metrics to the machine learning framework. The actual location of the application servers will be determined in coordination with the partners, but likely they will be exterior to the Cnam testbed and connected through the Internet, possibly using a tunneling technique such as VXLAN or GRE or LISP.

What the current testbed does not include is an orchestration layer. The one of Kubernetes is currently disabled and the plan is to determine which components from ONAP, and/or ORAN and/or OpenSourceMano platforms could be integrated, as developed later in the document. Depending on these aspects, from 1 to 3 additional physical servers may be allocated to the testbed in 2022.

2.2.6 Monitoring

Monitoring has to be distributed and to adapt to the constantly changing needs of the infrastructure. This means that the monitoring system should support not only metric collection, but also aggregation, filtering and processing in a hierarchical fashion to enable :

- Scalability across the overall system.
- Introducing hierarchical analytical and actuation points to efficiently exploit metrics.
- Support a scalable decision making distribution.

Monitoring plateform based on Prometheus

Prometheus is an open source monitoring solution originally developed by soundcloud. It has since then become a reference in terms of infrastructure monitoring solution.

Prometheus monitoring tools provide a modular architecture to capture metrics. Those features provide a complete framework to handle different types of infrastructures separately, and to be able to analyse the end-to-end services in the architecture. Moreover, alerts are defined by using Prometheus Alert Manage. It provides level of control and management workflow, allowing the platform to warn management entities, as endpoints, of an alert in the platform.

In its usual mode, Prometheus periodically scrapes target using HTTP requests. The responses to these scrapes are a list of metric identifiers, and of their current value. Prometheus uses these metrics to build time series, that it enriches using metadata extracted from the host.

These time series can be used to build dashboards, but can also be queried using a dedicated language: **PromQL**. Such queries can then be used to generate alerts, for example if a metric goes above a given threshold.

Prometheus alerts are usually send to sysadmins by mail or other methods. In the context of intelligent self configuring network, however, the **Prometheus-am-executor** might be of interest. This tool of the Prometheus suite is able to receive Prometheus alerts, and automatically take action. A frequent use case consists, for example, of reacting to hard drive storage space alerts by clearing temporary files or rotating logs. One could also imagine using this feature to auto-scale a service based on its RAM usage, or traffic volume.

Preliminary experiments run at Cnam with a local NFV infrastructure composed of two physical servers and a dozen of virtualized network functions. In terms of monitoring, the collection of thousands of metrics at physical level, container level and VNF level leverages on Prometheus node-exporters (https://github.com/prometheus/node_exporter) as monitoring components for the physical server level, while Pods and containers are monitored through a Kubernetes embedded CAdvisor (https://github.com/google/cadvisor) agent. Both exporters are compliant with Prometheus data model and architecture so that feature metrics can be exported through GET requests at a specific polling frequency.

Metrics monitored

Within the framework of the Intelligentsia project, several metrics will be monitored in order to allow for the scaling of services and the network in the case of increasing or decreasing loads, and more generally to ensure the proper functioning of the proposed solutions.

These metrics can be classified into two main parts:

- metrics related to virtualized services (i.e., VNF),
- metrics related to algorithms for the optimization of the MAC layer of LoRaWan networks, through access improvement.

Virtualized infrastructure metrics. Service monitoring involves tracking the evolution of various metrics such as CPU, memory, file system, and network metrics, and so forth, at both VNF/container and physical server levels. It is also on the basis of these metrics that the infrastructure state will be determined and the orchestration decision will be taken, which may require, for example, the use of time series analysis for state classification or prediction of anomalous network conditions.

A reference dataset example for this area is the one documented in the SYRROCA repository `https://github.com/SYRROCA`. Metrics derived from server and container logs and also related to the traffic behavior were therein collected every 5 s, forming time-series of 17280 values per feature and per day during 21 days, while a virtualized network service was running. Collected features are explicitly typed as counters or gauges, to ease their pre-processing. To give an idea of the important magnitude in the number of features that can be reached working on a dataset of virtualized infrastructure metrics, Table 2.3 details the number of features per layer and resource group for the SYRROCA dataset. Such scales justify the usage of a machine learning approach to process them.

	CPU	Network	Memory	File-system	Total
Physical	370	290	40	260	960
Virtual	60	80	160	230	530

Table 2.3: Number of features per layer and resource type for the example SYRROCA dataset.

Access network related metrics. In this section, we focus on metrics that could be used to improve medium access in LoRaWAN networks. The metrics treated are not be exhaustive but give an idea of the factors that can be used to optimize the access to the network. All these metrics come from aggregating the metadata provided by packets at their reception. A detailed list is provided in Appendix B.

Even if the objectives are different, these different metrics can be considered for the auto-configuration of the devices, the prioritization of the access but also the orchestration of the radio interface through slicing. Moreover, contrary to Section 2.2.2, we will not limit the discussion to the devices of class A, because this class has many limitations and may not be sufficient to support advanced functions such as prioritization or network slicing. The following metrics are widely used in LoRaWAN Medium Access bibliography [17]:

- **Packet Delivery Rate (PDR)** [≡ **Data Extraction Rate (DER)**[9]]: The ratio of received packets to transmitted packets over a period of time.
- **Packet Error Rate (PER)**: The ratio of packets with CRC error to sent packets over a period of time. (A packet with CRC error was demodulated at the GW –it used resources–, but the error-detecting code did not pass. I.e, the packet has at least one bit of error, and can not be used)
- **Packet Loss Rate (PLR)**: The ratio of lost packets to sent packets over a period of time. (The GW never received these packets. It can be indirectly calculated with the "frame count" LoraWAN MAC metadata field at the NS. However, only the ED knows the PHY parameters of a lost frame)
- **Network Energy Consumption (NEC) [9]**: Energy spent by the network to successfully extract a message. (Power and time. Energy spent will depend on PHY parameters of the packets: e.g., SF and size)

- **Jain's Fairness index** [11]: $\frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$, where x_i denotes the *normalized throughput* of each device and n the total number of active devices in each "slice". Index varies between 0 and 1, with 1 being perfectly fair.

- **Per-Node Signal-to-Noise-Ratio (SNR)**: LoRa SNR, measured by GWs, of the last N packets received for a given ED[1].

Metrics can also be applied per relevant sub-sets of the LoRaWAN domain elements. For example, it is relevant to partition the metrics per:

- **Entities**: Network Server-Wide, Gateway, Node.

- **LoRa Physical Parameters**: Frequency, Data Rate (Spreading Factor).

- A combination of previous categories (intersection).

Finally, we propose two metrics that are not explicit in the bibliography:

- **Per Gateway-Metric**: *Spectrum usage* (radio-time).
  - Partitioned per frequency and data rate, over a period of time.
  - E.g.: DownLink, UpLink (UL) OK frame, UL CRC error frame, idle radio; Illustrated in Fig. 2.5.

- **Per Node-Metric**: *Energy Packet Delivery Ratio* (EPDR).
  - Energy spent to successfully deliver an N Bytes frame.
  - Will be useful aggregated per Data Rate (DR).
  - EPDR can only be calculated in node, but the ED will need information from the Network Server (e.g., the PDR per DR).

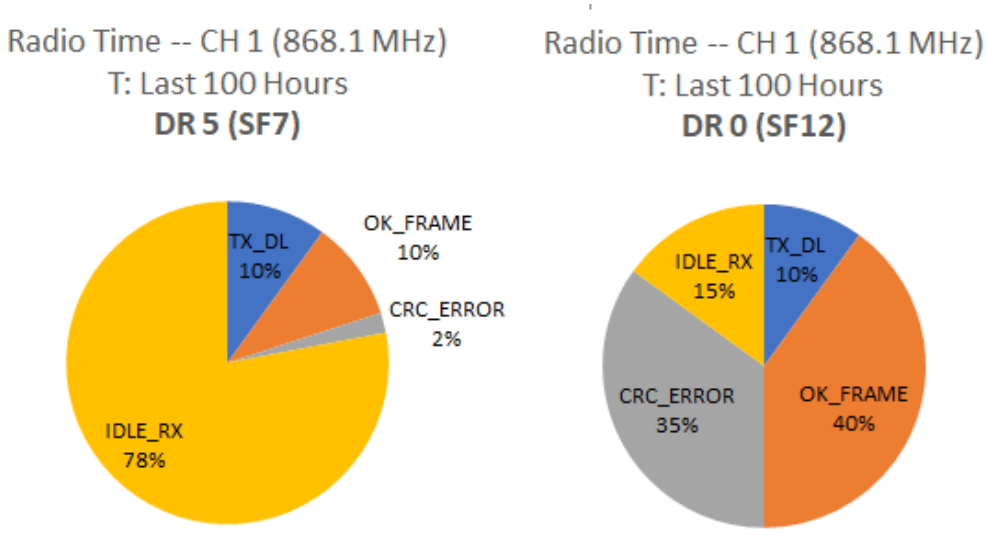

Figure 2.5: Possible spectrum-use metrics per Gateway

From a Network-perspective, spectrum-use metrics (as the GW-centric ones) will be primordial. As a final remark, it is worth noting that maximizing packet delivery ("throughput") in

[1]Network-side implementations of the Adaptive Data Rate (ADR) algorithm, like The Thing's Network's (https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate/index.html), use this metric (the mean value of last N samples) in order to estimate if the ED could lower its Data Rate ("link budget").

spite of everything else, is not an appropriate goal. "Fairness" considerations should be taken into account. To illustrate this statement: far-from-GW ED devices will have a higher-than-mean energy/spectrum usage to deliver a packet; yet, the network should prioritize them somehow. Otherwise, if raw-throughput is the only optimization goal, only EDs close to each GW will take priority. The metrics in this subsection will serve as inputs to other higher-layer actions (E.g., balancing fairness, throughput, and other considerations).

2.2.7 Orchestration

The term 'orchestration' is used in the context of virtualized networks and cloud networking to express the capability to deploy and (re)configure components that become easily programmable thanks to the softwarization of various infrastructure components. It is therefore important to specify the list of orchestration actions that can be envisioned in the framework of INTELLIGENTSIA project, to later determine which components the orchestration platform should encompass. That is, orchestration actions that are related to an the end-to-end network composed of:

1. LoRA devices and radio channel.
2. LoraWAN access gateways.
3. LoraWAN Network Servers (LNSs).
4. Application servers.
5. Transport network.

Orchestration actions

In terms of orchestration actions, we can at this stage identify the following possible orchestration space to be considered by the network automation framework.

1. *LoRA devices and radio channel.*

   The current practice in the management of IoT devices and related physical and MAC layer connection is to:

   - Configure Spreading Factor (SF) and bandwidth as one of the combinations offered by the data rate parameter;
   - Configure the transmission power.

   Both decisions are done accordingly to radio channel conditions determined at the LNS level with algorithms that are not formally specified, with some proposals at the state of the art such as [22, 27].

   In addition, the following reconfigurations are possible using MAC primitives:

   - Available uplink channels that can be used;
   - Maximum duty-cycle duration, aggregated over all sub-bands;
   - Number of default retransmissions per uplink packet;
   - Delay of the first downlink reception window after a transmission;
   - SF (in terms of data rate) and channel for both downlink reception windows.

   Even if they seem not to be exploited today, we could explore their usage and assess the utility to include them in the set of orchestration actions.

2. *LoraWAN access gateways.*

   Access gateways are anchored at the antenna facility. Their function is a basic digital signal processing of the received signal and to forward data to the LNS. Data forwarding is done by encapsulating the layer-2 frame into an IP packet format toward the LNS.

   About the digital signal processing involved, given the low bitrate at stake, it does not seem at this stage of particular utility to implement the radio signal processing in software to possibly locate it elsewhere in the core of the network as done for instance in software-defined cellular radio access systems.

   About the data encapsulation toward the LNS, under a virtualized network setting the gateway no longer resumes to an antenna with a quasi-static forwarder bridge, but to a programmable bridge requiring to be configured with:

   Adressing. As the vLNS location as well as the gateway-to-LNS assignments may change with time, the LNS address, identifier and location information may change accordingly and hence would need to be reconfigured as a function of the orchestration actions applying to the LNS (see below).

   Transport encapsulation. LoraWAN Frame to IP encapsulation may be needed for the back-hauling network, as for instance in case of slicing a type-of-service field (DiffServ Code Point) may be added, in case of a virtual network overlay protocol such as NSH et al.

   Bridging. The legacy basic gateway bridge could therefore possibly become a software-switch component able to perform the above mentioned encapsulation with address and transport header fields. In case of co-localised vLNSs and/or application server, additional port-forwarding and virtual bridge network configuration may need to be handled.

   Besides these actions, scaling ones do not seem to be useful given the low bitrate at stake at a given antenna, but we will be able to assess the importance of this aspect for high-density settings during the experimentation.

3. *LoraWAN Network Servers (LNSs).*

   The LNS architecture described above would require the following possible configurations:

   Placement. vLNS may be duplicated at different physical locations and different physical servers, in a serverless way as done by Kubernetes, to adapt to load and infrastructure conditions.

   LNS-gateway clustering. The assignment of radio gateways to LNS may change in time due to changing conditions related to, for instance, faults, interference and mobility phenomena at the gateway and device levels, and performance degradation or faults in the backhauling network.

   Scaling. In a virtualized setting, the computing power allocated to a LNS can change in time as a function of the load, i.e. the number of gateways handled by a LNS and the number of devices beyond these gateways.

   Bridging. For both uplink and downlink communications toward and from the application server and radio gateways, as the application server may be virtualized and possibly relocated in time, and the active radio gateways for a given device may also change in time as well.

4. *Application servers (ASs).*

Placement. Application servers may be duplicated at different physical locations and different physical servers, in a serverless way as done by Kubernetes, to adapt to load and infrastructure conditions.

LNS-AS clustering. in case of multiple application server instances, the assignment of LNS to application servers may change in time due to changing conditions related to, for instance, performance degradation or faults in the backhauling network.

Scaling. In a virtualized setting, the computing power allocated by an application can change in time as a function of thee load.

Bridging. For both uplink and downlink communications toward and from the LNS/vLNS.

5. *Transport network.*

   The local bridging reconfigurations needed at the gateway, LNS, application server locations (which may, in some envisioned setting even be physically the same), as listed above, already absolve network function chaining functions impacting the transport network forwarding. Besides this, the following orchestration actions are related to network-wide reconfigurations due to standard:

   - *Network path change.*
     To cope with network state change. This could be related to the usage of an MPLS-TE or an SDN architecture to have a more direct control on network path configuration by means of traffic engineering optimization.
   - *Load-balancing.*
     With given network paths, the network stack may need to adapt to other orchestration actions (placement, scaling) affecting arbitrary load-balancing policies as for instance function of the computing power of LNS and application servers.

Orchestration decision-making

The decision-making logic to handle these orchestration actions has to gather information from monitoring data. Available monitoring data and related filtering policies will be specified in Task 2.1.

Let us remember that one goal of the project is to assess, by means of both modeling and experimentation, how these orchestration actions would be integrated in a network automation framework. In particular, an expected outcome is to determine which actions would better be left under the control of autonomous and standalone distributed agents running at the different stages (e.g., device or LNS levels) and which could instead be run within a network automation framework encompassing active coordination among agents and/or full centralization. More likely a mix of distributed coordination and centralization among involved parties could reveal as being a good compromise trading reliability with efficiency.

The network automation framework itself could be relying on a variety of algorithmic approaches for executing the decision-making related to the orchestration actions, from online first-fit / best-fit algorithms classically adopted in node placement policies, to more sophisticated scheduling algorithms also relying on optimization approaches. In WP2, we will investigate the possibility to build a machine learning framework that works on the network data features collected by the monitoring elements, potentially forming a set of many thousands of metrics, hence calling for a machine learning approach able to scale with this complexity. WP2 will have to determine if there is a gain in integrating also the IoT access network elements (e.g., the IoT devices, gateways).

Orchestration platforms

It is the goal of Task 4.3 to specify the specific orchestration platform and software components we will use to implement the network automation framework, integrating the requirements

stemming from algorithms and protocols that will be defined in WP2 and WP3, accordingly to use-case specification from Task 1.2 and life-cycle management requirements from Task 1.3. The orchestration platform needs to be able to:

- integrate some if not all the project decision-making modules, at least those requiring a global view instead of a local view;
- communicate with the radio access network components, namely the LNS, the gateway and possibly even the IoT devices;
- configure the edge cloud and network function virtualization infrastructure, namely its scheduler for Kubernetes or specific orchestrator for NFV platforms;
- the programmable transport network, namely the network controller able to configure network bridges.

Our envisioned reference orchestration platform is ONAP (Open Network Automation Platform). We will determine at which extent ONAP can support the above mentioned orchestration actions or a subset, so that we may end up with complementary bricks or even alternative platforms when WP4 will start based on the state of the art of automation platforms in 2022.

Our current view is that the possible ONAP integration with the LoRA radio gateways should happen via the A1 interface currently envisioned for the interconnection with the Open Radio Access Network (ORAN) platform conceived for cellular access networks, and in particular the so-called Radio Intelligent Controller (RIC). Recently, the ORAN RIC was decomposed into a near-real-time scheduling entity sitting within the core of the ORAN architecture, and a non-real-time entity sitting within the orchestration platform. Indeed, both ONAP and ORAN releases are these days undergoing a fast update and we expect in 2022 to see novel bricks appearing in future corresponding releases, and in particular a featured newer version of the RIC modules beyond the A1 interface.

On the other hand, the LNS actually ensures the function of a radio controller, yet processing both the data-plane and the control-plane; as far as its distribution using multiple instances, virtualized or not, may reveal not to be interesting from a performance perspective, the communication with the orchestration layer could happen with the LNS. Instead, in case of distribution, this would be more difficult and the relay through a dedicated RIC handling only the control-plane, and guaranteeing data coherency seems more pertinent.

Therefore, our goal is to understand if we may envision to leverage on the ORAN RIC, by means of a dedicated RIC application, to link the orchestration platform with the LoRA access network, or if instead leveraging on an ad-hoc project-made radio controller. The decision will have to be taken depending on the importance that the different orchestration actions described above would take, and in particular which ones our study in WP2 will determine to be effective and useful to be integrated with the orchestration framework. At this stage it is indeed not straightforward to determine if the LoRA device and radio gateway orchestration actions would benefit from the global view provided by an orchestration platform as ONAP. Intuitively they should, given the relationship between the number of radio devices per LoRAWAN network node and the required computing resource scaling of the LNS and application servers, and the project use-cases may eventually show the importance of this multi-resource relationship.

As a contingency plan in case of problems, alternative orchestration platform approaches will also be considered, for example leveraging on OpenSourceMano orchestrator system or an ad-hoc orchestrator, possibly using Ansible for the configuration interface.

Chapter 3

Functional architecture without slicing

3.1 Global view

In view of the elements introduced in Chapter 2, the functional architecture considered in this project to model LoRaWAN networks is depicted in Figure 3 and is composed of

- a substrate network layer comprising the radio gateways and the cloud infrastructure for hosting the VNFs associated with the LoRa network (LNS and Application servers);
- a telemetry layer collecting analytics from the substrate network and exposing synthesized metrics (after filtering, aggregation, etc.) to the orchestration layer;
- an orchestration layer for the management: lifecycle management of the VNFs, resource allocation, etc.

The control plane of the LoRa network is executed by vLNS as illustrated in Figure 2.3. The LoRa enabled server processes the data plane (similar to the UPF of 5G control plane), the Application server ensures the control of sessions (the equivalent of the AMF and SMF functions of the 5G control plane) and the Join Server authenticates devices (similar to the AUSF function of the 5G control plane).

The VIM of edge and centralized cloud platforms will ensure the deployment of the containers hosting the VNFs in collaboration with the orchestration layer. The VIM of a cloud platform is in charge of managing the resources of the platform. The orchestration may move some VNFs or requires to scale up/down some functions when analyzing the measures from the network provided by the Telemetry layer aggregating LoRa analytics (radio metrics) and Prometheus analytics (from the cloud infrastructure). This loop is spanned over the Data Analytics Engines, the Optimization Function and Decision Engine of the orchestration layer.

In the following two sections, we give more information on two functional blocks: telemetry and optimization function as they will be instrumentalin the project.

3.2 Telemetry

Network automation include several provision of the service and its management. All the service elements included in the contract and their functional and non-functional aspects:

- Functional aspects include all the subscribed functionalities contained in the offer.
- Non-functional aspects include QoS (behaviour).

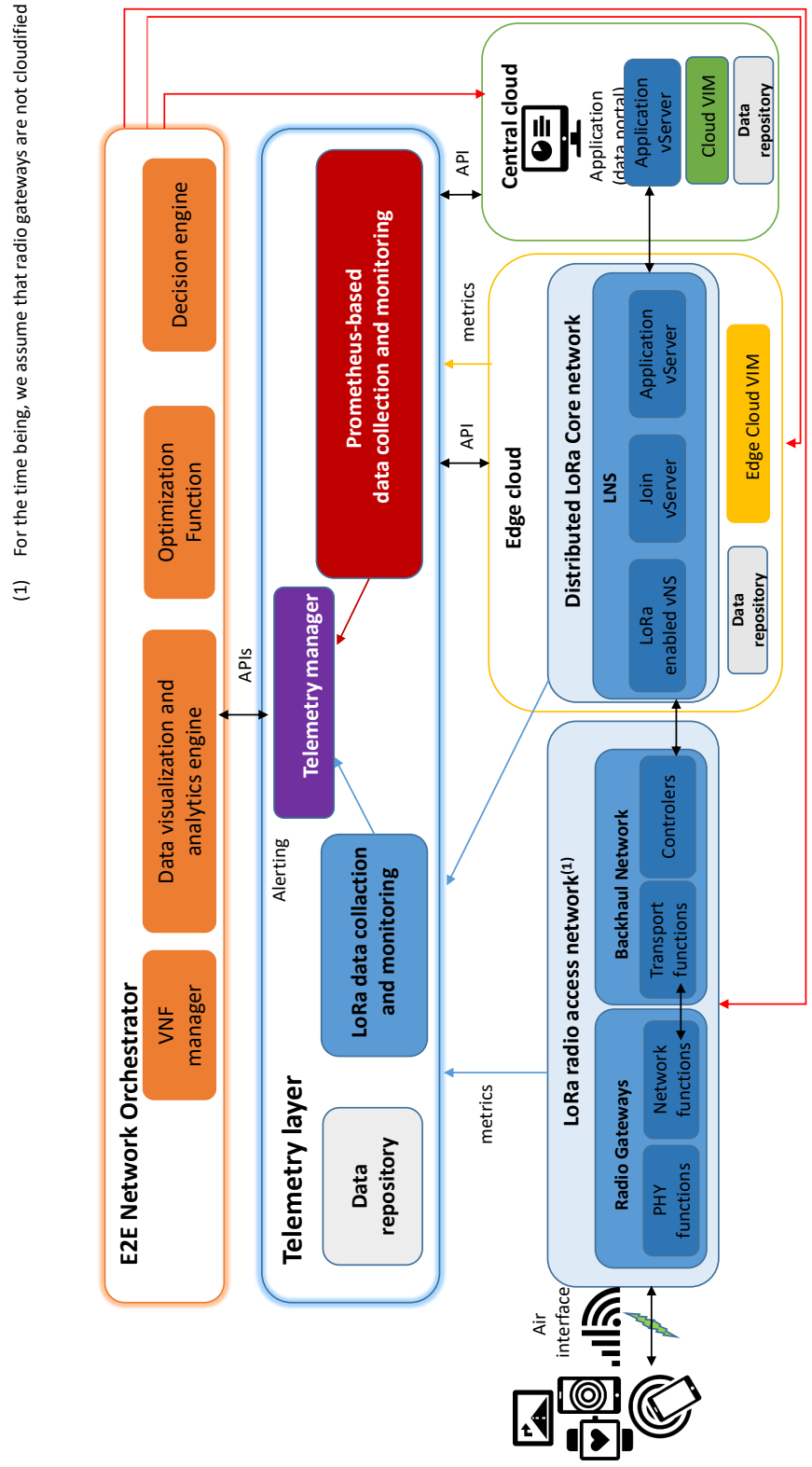

Figure 3.1: Reference functional architecture.

Monitoring represents the use of any available technical tool to assess permanently or for a given period of time a particular indicator, e.g. a server load or the response time for a service.

The monitoring requirements apply to the software service network on both, functional and non-functional aspects:

- **Functional monitoring.** provides the monitoring of devices and LoRa functions.
- **Non-functional monitoring** should be able to monitor and control their behaviours (non-functional aspects) using for example autonomic management approach. Placing the monitoring of QoS very close around each service component helps to detect exactly the malfunctioning component. Their generic nature allows a provider to apply them on any service.

We propose to measure a QoS of each component (hardware and software) allowing better diagnostic of various malfunctions whereas most existing tools monitor network traffic or CPU usage when they should monitor the functional component performance.

The various metrics have been introduced in Section 2.2.6. We include metrics from the virtualized infrastructure as well as the ones of LoRA. It is worth noting that a LoRa *packet* is the unity of interaction with the Medium. LoRa metrics derive from packets in terms of (a) *reception rates* and (b) associated *metadata* (LoRa-PHY at GW, and LoRaWAN-MAC at NS). The Telemetry layer must gather a time series of LoRa packet's metadata. This *fundamental* information suffices to compute any LoRA Medium Access-related metric that may be defined in the future. See Appendix B for detail on a LoRA packet's metadata and Subsection 2.2.6 (*Access network related metrics*) for concrete LoRa metrics examples (e.g., Packet Delivery Rate, Network Energy Consumption).

The metrics reported from the virtualized infrastructure as well as from LoRA elements are represented in Figure 3.2.

3.3 Resource allocation algorithms

We describe in this section the algorithms, which will be specified in other work packages.

3.3.1 Resource allocation algorithms on devices

Three are the key Physical parameters a LoRa End Device (ED) must fix at every LoRa packet to Send/Transmit: (1) Spreading Factor (SF), (2) central Frequency, and (3) Transmit (TX) Power. It is worth recalling that the abstraction of a *LoRa Data-Rate* (DR) includes a choice of SF, and the abstraction of a *LoRaWAN Channel* defines valid tuples of central Frequencies and LoRA DRs, see 2.2.2 for detail on these fundamental notions. Each node can set up the possible DR and TX power independently[1], or use the the standardized LoRaWAN's Adaptive Data Rate (ADR) mechanism (See LoRaWAN's standard [19] Section 4.3.1.1).

In this proposal, EDs will chose the fundamental Physical parameters using Reinforcement Learning techniques, in particular Bandits algorithms [18]. The *Bandit* algorithm will converge to a choice of Physical parameters (notably SF, but it generalizes to any combination of parameters, using the Bandit's abstraction of *Arm*). In order to converge (i.e., "learn"), the ED needs feedback from the LNS (i.e., DownLink packets with statistics). The feedback messages will include the Packet Delivery Ratio detailed per choice of Physical parameters (*Arms*) which, as stated before, will be used by the algorithm to converge. In Bandit's nomenclature, this feedback information is used to calculate the delayed *Reward* (i.e., a scalar value that synthesizes the optimization problem) of each *Arm*. While the Bandit's algorithms can be taken from literature with little modifications, the definition of the *Reward* is a fundamental challenge (i.e., a scalar value that captures –and solves– the problematic at hand).

[1] About the Frequency: the LoRaWAN standard suggest to uniformly randomize the choice.

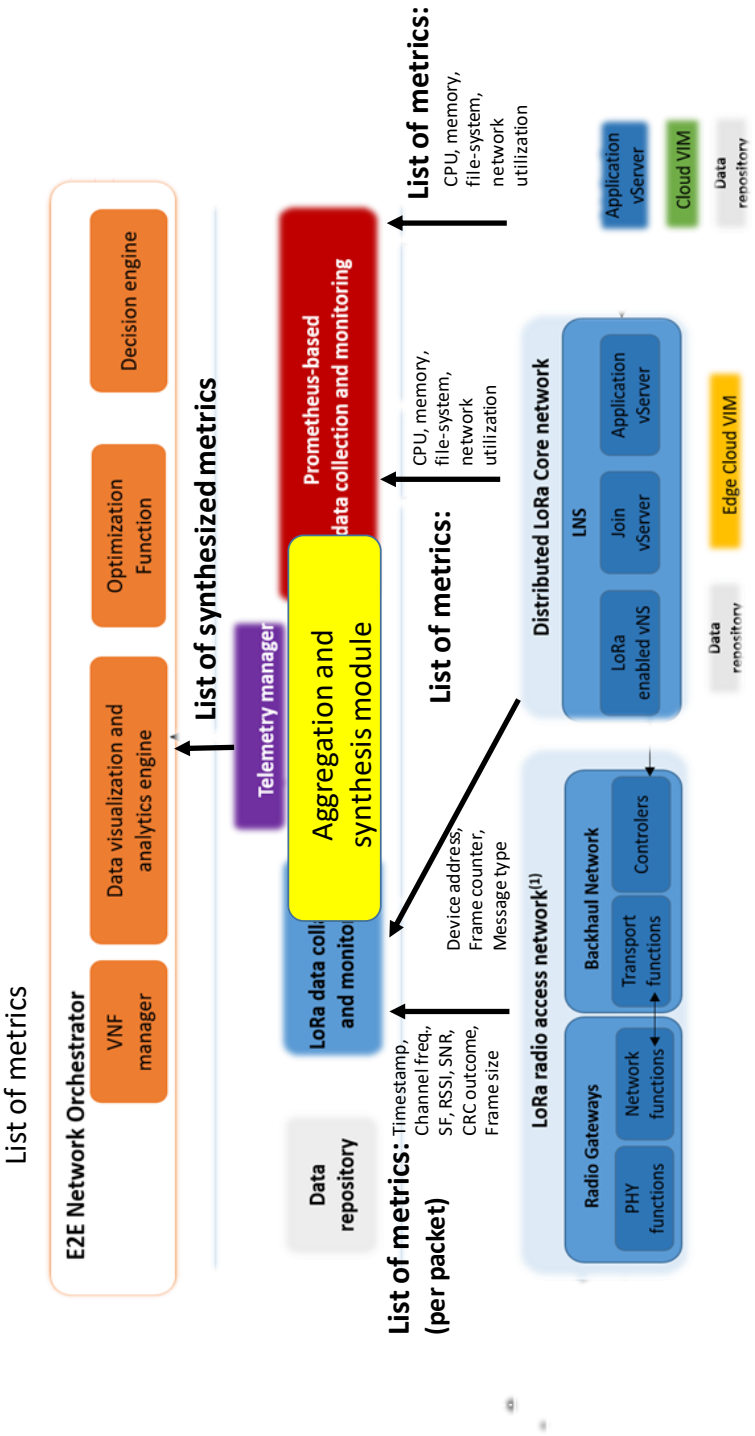

Figure 3.2: Reference functional architecture: exchanged metrics.

NOTE About Mobility Use Cases: In Use Cases with EDs "mobility" (i.e., highly dynamic, non-periodic, medium changes), long-term convergence is not possible nor desirable, this problematic is tacked by the field of *non-stationary* Bandit problems. The abstractions remain the same: *Arms* and *Rewards*. For example, the *convergence* concept used in the previous paragraph will apply but for shorter periods of time (i.e., we discretize the mobility problem in time, and can *converge* for these discrete parts of the problematic).

3.3.2 Resource allocation algorithms on LNS

Splitting the LoRaWAN access network into slices is much less natural than in the so-called cellular technologies. Indeed, in NB-IoT networks, for example, after a contention access phase, the allocation of resource blocks can be done deterministically (i.e. free from contention data submission).

In LoRaWAN networks, the story is quite different, since for maximum energy savings, it was decided to reduce the number of control messages to a minimum, which is also justified by the small size of the packets. To achieve such an objective, the devices do not necessarily have to associate with the network for each data transmission. Moreover, the devices do not even associate with a particular access point, implying the reception of the packets by several access points at the same time, which results in a higher reliability of the network.

The architectural modifications of LoRaWAN, very briefly described above, make all resource management extremely complicated at the gateway level. Resource management, on the other hand, can be done quite naturally at the LNS level, which has a slightly less local vision (compared to the gateways) of the use of resources, the reliability of the network, etc. The access resource partitioning algorithms, which will be proposed in the Intelligentsia project, will therefore be implemented at the LNS level.

In the context of the Intelligentsia project, and more particularly in WP3 (Task 3.3), resource partitioning solutions will be investigated. Since slicing cannot be done in a deterministic way, probabilistic solutions will be studied for the partitioning of wireless resources in time and space (i.e. frequency). This partitioning can be done according to different modes (by service level agreement (SLA), by traffic class, by tenant, ...), these will be studied in order to select the most relevant option(s) with respect to the project objectives.

The design of algorithms for access slicing will be done in Task 3.3. Different algorithms can be studied, in particular algorithms based on deep reinforcement learning.

3.3.3 End-to-end resource allocation

The Orchestration layer embeds functions in charge of optimizing the resources allocated in the LoRa network (radio gateways and Core network), the Edge and the Central Clouds.

The resource allocation algorithms executed at the Orchestration layer will rely on a set of functional metrics provided by the Telemetry layer. The objective of Task 2.1 is to define the different functional metrics used for state machine modeling and network automation.

To manage the resource allocation at the Orcherstration layer, we plan in a first step to classify the state of the network, then in a second step to select the actions that the orchestrator must transmit to the VIM of the Edge and Central Clouds as well as to the LoRa network (radio gateways and Core network).

Several types of algorithms will be used to learn and to infer in real-time the current state of the network and to select network orchestration actions to reconfigure the network elements. The classification of the state will be performed using standard clustering approaches based on regularly sampled memory and leveraging on dimensionality reduction on the set of features (e.g., k-means, principal component or linear discriminant analysis), as well as Long Short Term Memory (LSTM) networks. The design of these algorithms will be done in Tasks 2.2 and 2.3.

The selection of orchestration rules will be made based on reinforcement learning logic to define when a state transition should take place and which rule is better to activate for network

state regeneration. A catalogue of appropriate orchestration actions will be defined to reconfigure IoT device behavior, radio gateways for access control, Edge and Central Cloud for scaling and placement of vLNS and application server, and routing and placement of functions in the transport network segment. Reinforcement learning algorithms for the selection of rules to be used at the Orchestration layer will be developed in Task 2.4.

Chapter 4

Functional architecture with Slicing

4.1 Introduction

Network slicing is a big promise of 5G networks by enabling the customization of network features to customer needs. Slicing has been standardized by 3GPP (**references**) and implemented in some orchestration platforms like ONAP (**references**). Via slicing, a given customer can reserve resources in the network (bandwidth, IT resources, etc.) and customize network functions, for instance by requesting the network operator to implement specific and tailored network functions for its own usage. Such an approach is notably made possible by Network Function Virtualization (NFV), which enables VNFs to be customized and instantiated on demand.

A 5G slice can involve a private air interface (sub-licensed frequencies, frequencies reserved by the operator for a given customer, dedicated frequencies in the 2.6 GHz band, etc.), a private (virtualized) RAN, a private core network and private transport capacities or all these elements can be shared with the public mobile network, any combination of private and shared elements can in principle be possible. The 5G interface is "scheduled" in the sense that time slots and frequencies are alternatively allocated to the different connected UEs according to a scheduling algorithm. Hence, it is possible to allocate bandwidth to a UE on the 5G air interface even if the resources (time slots and frequencies) are not permanently allocated to the UE.

In the case of LoRaWan IoT networks, the scope of slicing is more restricted but also more complex. An IoT LoRaWan network involves end devices, the radio interface, the network server (LNS) and the application server. This last element depends on the IoT application and is specific to a given customer. We have seen in the previous chapter that an LNS can be virtualized and hence customized and instantiated on demand as any other VNF. In the present document, we shall not consider virtual radio gateways. Theoretically, radio gateways could be virtualized as in the case of Base Band Units (BBUs) in the framework of cloudRAN. We shall instead assume that a LoRa radio gateways are shared by all devices. We shall specify rules for distinguishing between devices.

In LoRaWan IoT network, the radio interface is much more complex to manage because it is based on ALOHA medium access. Transmission by EDs is not scheduled and transmissions of two packets by two different EDs can overlap and the two packets are eventually lost. Collision is an inherent limitation of the ALOHA medium access scheme, which is nevertheless easier to implement and requires less processing in the EDs.

Under the above assumptions, the realization of a slice relies on:

- customization of the vLNS,
- a fine tuning of the parameters of end devices spreading factor, power transmission, coding rate, etc.,
- on policies implemented in radio gateways and LNSs (frequency allocation, remote control of end devices, etc.),

- management of radio frequencies (reservation of some frequencies, allocation of spreading factors, etc.),

- modulation of the transmission power of devices.

A slice then relies on an adequate management of both radio resources and network resources (vLNS). We assume that the backhaul bandwidth is sufficient to transport information transmitted by EDs without loss.

4.2 Different levels of network slicing

As in 5G cellular networks, slicing in LoRaWan ~~IoT~~ can be based on several levels of isolation. SF, Transmission power

4.2.1 Strict isolation

To ensure strict isolation, it is possible to reserve frequencies for a group of devices. Moreover, with the virtualization of LNSs, it is possible to deploy a vLNS for a group of EDs. A vLNS can be deployed by an orchestrator as any other VNF.

While this approach guarantees isolation as much as possible and the quality of transmission by the EDs of the slice suffers from limited interactions with EDs outside of the slice (interference is still possible), this approach is resource consuming and certainly too much expensive, especially in dense areas where numerous EDs should have access to the LoRa spectrum. This is why we investigate in the next section, the case when slices share the radio spectrum.

4.2.2 Slices on shared radio spectrum

As mentioned in the previous section, vLNSs can be instantiated on demand and from an orchestration point of view, they appear as any other VNF. Some configuration is necessary in the radio gateways to send the packets to the ad-hoc vLNS.

A slice is usually defined for a given group of EDs and has objectives in terms of:

- quality of service expressed in Data Rate, latency to transmit a packet (including retransmissions), packet loss,

- availability.

There are a number of parameters which have a direct impact on the above metrics:

- TX data rate (Spreading Factor + Bandwidth),

- Transmission Power (TP),

- Number of transmissions for each uplink message,

- List of possible radio channels for uplink transmissions,

- Mask of of active radio channels for uplink transmissions.

These parameters can be tuned to distinguish between EDs and thus create slices.

4.3 Slicing in the existing literature

Although numerous works deal with the problem of network slicing, only very few works deal with slicing in LoRaWan networks [14, 11, 21, 15, 13, 12]. These works have a common co-author and follow a single methodology. The authors propose a three-step methodology:

- **Clustering:** The main idea here is to assign devices to slices according to some criterions. In this step the number of slices is also determined.
- **Throughput estimation:** The devices' throughput estimation is done using maximum likelihood, with the exception of [21] for which the mini-batch Gradient Descent and the unserved capacity are used to provide such an estimation.
- **Resources allocation:** In this step, the radio resources are reserved for the different slices, according to their needs.

Three slice types are introduced in [14] on the basis of quality and reliability as described in Table 4.1: Urgency and Reliability Aware (URA), Reliability Aware (RA) and Best Effort (BE). The values for latency should be considered as indicative and not firm objectives. A critical review of objectives should be performed in light of technological constraints. Nevertheless, one can keep in mind that there are globally three levels for latency: strict, elastic and a value in between. This roughly corresponds to the classical cases of QoS in networks: Expedited Forwarding, Assured Forwarding and Best Effort. It is also worth noting that these QoS classes have never been used in practice. This is why we shall develop a more pragmatic approach to QoS in the Intelligentsia project.

Table 4.1: IoT QCIs table.

QCI	Slice Name	Resource Type	Priority	Packet Delay Budget (ms)	PER %	Example Services
71	URA	GBR	1	100	10^{-3}	Real time, smart mobility
72	RA	GBR	2	200	10^{-3}	Real time, eHealth and home security
73	BE	nGBR	3	300	10^{-6}	Delay tolerant, smart agriculture

To meet the requirements in terms of quality and reliability, the authors in [14] propose to find an allocation of TP and SF in place of the classical ADR mechanism. This is done in two steps: first, channels are allocated to slices proportionally to the average throughput of their devices. Then, TP and SF are assigned to devices inside each slice using a multi-criteria decision analysis method based on Gaussian Mixture Models (GMM) and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS).

The objectives of the optimization are i) maximization of throughput and ii) minimization of delay, Packet Loss Ratio and energy consumption. Different weights are assigned to these objectives for each slice based on the QoS requirements. Then, the weights are used to find the best combination of TP and SF for each device in the slice.

The method is compared via simulation with a TP and SF allocation scheme that reflect the state of the art: SF is chosen so as to reach the closest gateway with the maximum data-rate possible, and then TP is set to me minimum value needed to still be in range. Results obtained by progressively increasing the number of devices show that the proposed slicing-based method is worse in terms of energy consumption, and comparable in term of packet loss, unsatisfied nodes in delay and unsatisfied nodes in throughput. Nevertheless, it is able to achieve slice isolation by exploiting the independent channels.

Excepting the papers introduced above, the concept of slicing is not explored in LoRaWan networks. Nevertheless, there are business opportunities identified by operators. As IoT becomes

more widely spread, the need for QoS and security is clearly identified by business units of operators, in particular Orange Business Services. Moreover, new mechanisms are appearing in EDs in order to reduce collisions inherent to the CSMA scheme used in LoRa. This is in particular the case of the Listen-Before-Talk (LBT) scheme [23], implemented by some Semtech products. It is nevertheless worth noting that LBT can only partially reduce collisions, as two EDs which are sufficiently distant can give rise to collisions at the radio gateway while they cannot detect each other.

In WP3, new methods of accessing the radio medium will be introduced to implement slicing on the basis of data analysis algorithms developed in WP2. Moreover, in WP3 we will not be limited to the vision of slicing as presented in this section, which corresponds rather to a classical vision of QoS support in networks. Indeed, several visions of network slicing are possible:

- **Per IoT device:** IoT devices sharing the same quality of service requirements, which correspond to the vision of the papers introduced in this section.
- **Per tenant:** Several devices' owners acting in the same network, which allows the support of multi-tenancy within the same infrastructure.
- **Per service:** Support of several services (with QoS requirements) within the same network. This corresponds to the classical view of network slicing, which generalizes the per IoT device view with customized constraints.

Other visions of the slicing may exist like the one ensuring fairness between slices or hybrid approaches mixing these different views. These different visions will be developed in WP3.

4.4 Impact of slicing on the functional architecture

In light of the previous section, we see that the introduction of slicing in IoT raises many issues:

- Identification of use cases: See Deliverable D1.2 where several use cases are identified. After discussion with OBS, there is clearly a need for studying the use case of waste management. This is the simplest case as EDs are static. More challenging use cases are identified in D1.2 where EDs are mobile (cattle, vehicles).
- Orchestration: negotiation and life cycle management of slices, notably the deployment of vLNS.
- Radio gateways: introduction of algorithms for controlling EDs in order to meet the requirements of slices; coordination between several gateways could be necessary.
- End Devices: introduction of algorithms for sharing the bandwidth of the LoRa cell according to the requirements of the various slices. Some parameters could be remotely controlled by the radio gateway and some algorithms could be implemented in the device to access the medium (LBT and other collision avoidance scheme).

4.4.1 Orchestration issues

With regard to orchestration, slicing in IoT raises the same issues as slicing in cellular networks, except that the VNFs are not the same:

- negotiation of the slice characteristics,
- provisioning and parameterization of the associated VNFs and their life cycle management (deployment, removal, update/upgrade, etc.),
- monitoring and MADE (Measure, Analyze, Decide, Execute) loop,

- parameterization of radio access gateways and possibly EDs for slicing support.

The 3GPP has released various 5G Technical Specifications (namely, TS 23.501 [2], TS 23.502 [3], TS 23.503 [4], TS 28.530 [5], TR 28.531 [6], TR 28.801 [1]) for supporting network slicing. Among the most relevant entities introduced by 3GPP, we can cite:

- Network Service (NS): A logical network composed of a chain of network functions.
- Network Slice Instance (NSI): A set of instances of network functions and the required cloud resources to execute them.
- Network Slice Subnet (NSS): A slice subnet is a network segments or sub-slice within a broader slice. For instance the access network within an end-to-end mobile network.
- Network Slice Subnet Instance (NSSI): A set of instances of network functions belonging to a network segment.

According to 3GPP, a single cellular device (UE) can support up to eight simultaneous connections to different slices. A slice shall be in fact identified by a Slice Differentiator (SD). The UE attachment to a given slice is then performed by the Network Slice Selection Function (NSSF), while using the Network Slice Selection Assistance Information (NSSAI), that refers to the expected slice performance (latency, bandwidth). During the registration procedure, the Access and Mobility Management Function (AMF) selects the adequate NSI among the enabled slices according to the user subscriptions. The NF Repository Function (NRF) is responsible of discovering the functions involved in the selected slice. The establishment of a session and data transmission is then given after the selection of the User Plane Function (UPF) function by the Session Management Function (SMF). The slice behavior is supervised by the Policy Control Function (PCF).

To the best of our knowledge, there are no NSSAI nor SD for IoT devices. The configuration of a slice in the context of IoT should then be based on other identifiers, for instance the MAC address. This means that for the negotiation of a slice in the context of LoRaWAN beyond security and QoS, EDs with their identifiers and location should be explicitly declared.

Now, the slice negotiation and management involve specific orchestration functions. 3GPP has introduced various entities for dealing with the network slicing management and orchestration, as follows:

- Communication Service Management Function (CSMF): It enables translating the performance requirements of a service to slice technical features (service level);
- Network Slice Management Function (NSMF): It performs the management and orchestration of NSI (slice level);
- Network Slice Subnet Management Function (NSSMF): Responsible of the management of NSSI (subnet level).

The CSMF offers a northbound interface to the core commerce layer in charge of the slice negotiation with the user. This will not be considered in this project.

As argued in [24], the CSMF and NSMF could be inside or outside the orchestration platform. of the slice. These two functions are within the scope of a supervision center (e.g., in the case of Virtual Private Networks). The trend in ONAP development is to insert these two functions into ONAP. In the present document, we identify these two functions without placing them explicitly inside or outside ONAP.

The orchestration platform will be in charge of the life cycle management of VNFs associated with LoRaWAN slices to meet security requirements and to configure radio gateways in order to meet QoS requirements. The concept of slicing has also a big impact on monitoring as the slice level has to be introduced when analysing and processing data.

4.4.2 Monitoring issues

In Section 2.2.6, a number of metrics have been identified to monitor the performance of the radio interface. The presence of slices lead to the introduction of an aggregation level when collecting and aggregating metrics. As a matter of fact, metrics should be labeled by taking into account the slice level when necessary.

This means that the NSMF has to pass information on the slice level to

- the various elements of the substrate layer (LoRa RAN, edge cloud, central cloud) in order to send slice labeled metric to the telemetry layer,
- the telemetry layer to analyse and aggregate data.

For monitoring slices, the telemetry layer should send sliced labelled and aggregated data to the NSMF, which in charge of maintaining the slices under utilization in order to meet the SLA requirements.

4.5 Slice enabled functional architecture

To take into account slicing, we are led to modify the functional architecture presented in Section 3.1. We basically introduce the NSMF and CSMF functions. Their relation with the Core Commerce is not detailed in the present document. Basically, according to the Open Digital Architecture specified by the TM Forum (see for details), the output of the Core Commerce layer is a Customer Facing Service (CFS) specification, which has to be converted into a Resource Facing Service, which is then translated into network services (VNFs, transport capacities including the air interface). The functional architecture taking account of slicing is depicted in Figure 4.1. It is worth noting that a slice in the case of a LoRa network is much simpler than a slice for a cellular network. The only VNF is the LNS, which can be shared or dedicated to the slice. The real issue in the case of LoRa network is the air interface. In cellular networks, the access to the air interface is regulated by the scheduler. In LoRa networks, EDs transmit according to the CSMA scheme and collisions and interference impact the delivery of packets sent by the EDs.

In Figure 4.1, we have illustrated in yellow the elements of the network impacted by the instantiation of a slice:

- The CSMF and NSMF functions of the orchestration platform translate the user request in terms of network service (vLNS and transport capabilities); we have placed these functions inside the orchestration platform but could be put outside;
- We have illustrated the case when a slice comprises its own Application server and portal together with a dedicated LoRa core network;
- Transport functions and controlers have to be slice aware and controlers may configure via Software Defined Network (SDN) interfaces some transport elements to reserve bandwidth or configure priority levels in the backhaul network;
- radio functions in the radio gateways as well in EDs to support slices and their SLAs.

We see that slicing impacts almost all the functions of the network, which have to be slice aware in order to take the right actions and decisions with regard to Service Level Agreements (SLAs) of slices.

The concept of slicing has also a major impact on the telemetry layer. The metrics coming from the radio and transport layers and the virtualized infrastructure may not be labeled with slice identifier. Additional modules have to be introduced in the telemetry layer in order to process data and then expose them to the orchestration layer. An additional module has to be introduced in order to make the connection between data and slices. These applies for data

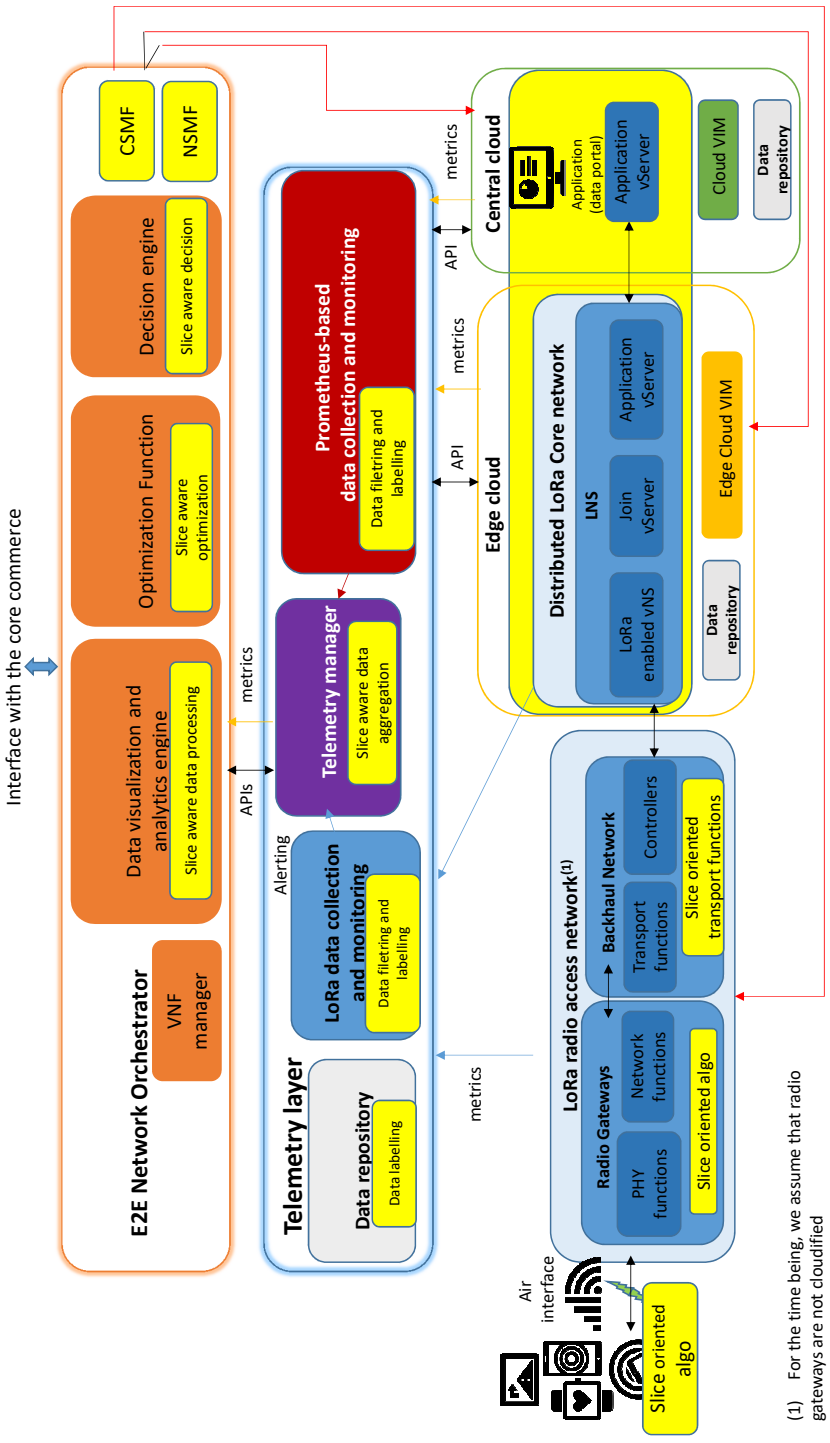

Figure 4.1: Reference functional architecture - slice aware functions are in yellow.

from the data planes as well as those from the virtualized infrastructure. This is illustrated in Figure 4.2.

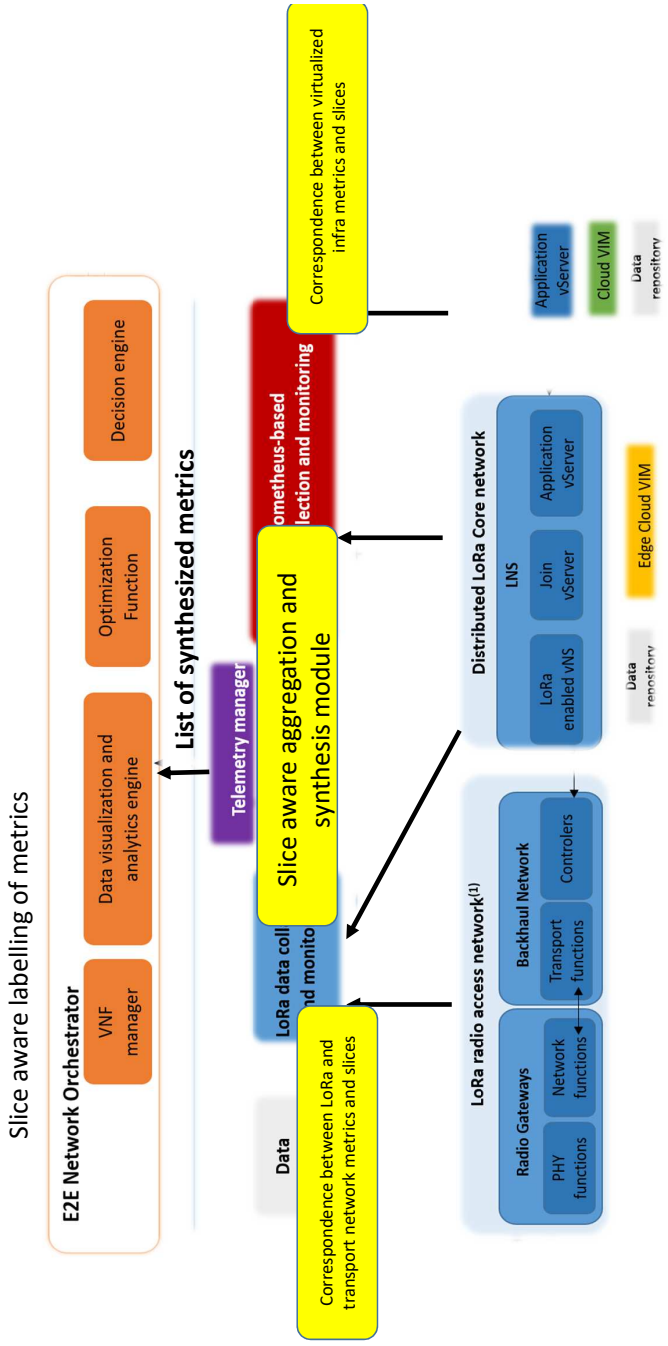

Figure 4.2: Reference functional architecture - slice aware functions are in yellow.

Chapter 5

Conclusion

We have introduced in this deliverable a functional architecture for the introduction of slicing in LoRaWAN networks. We have have notably identified the various functional blocks for supporting slicing as well as the metrics which could be used for maintaining the SLA of a slice.

The implementation of the slicing concept in LoRaWAN networks raises a number of questions and challenges that will be addressed in the different tasks of WP2 and WP3. We describe below some of these challenges:

1. **Definition of the Slicing concept for LoRaWAN networks.**

   - We plan to take into account three levels of quality of service (Urgency and Reliability Aware, Reliability Aware and Best Effort) which will correspond to 3 types of slices. It will be necessary to define the resources associated with each slice in the LoRaWAN access network, in particular the use and sharing of the 8 or 16 channels supported in EU868 between the slice types. Two approaches could be considered, strict segmentation of channels according to the type of slice or, on the contrary, sharing of part or all of the channels between the types of slice.
   - Other more advanced visions of network partitioning will also be considered. In these different visions, introduced above, we will consider a finer sharing of resources with the possibility to have a better spectral efficiency.
   - On the other hand, within each slice, several elements will have to be configured in order to guarantee the level of quality of service required by a maximum number of end-devices of the LoRaWAN network. We can mention in particular the choice of the transmission power, the Spreading Factor or the Coding Rate for changing the priority of the access to the network.

2. **Slice-aware components.**

   - LoRaWAN networks are composed of three segments: the air interface between end-devices and gateways, the segment between gateways and LNSs, and finally the segment between LNSs and the central cloud. For each of these segments, the hardware and software elements must be able to take into account the three types of slice considered in the project. To do this, it will be necessary to define algorithms for routing packets for each slice, to collect the information necessary for monitoring each level, and to modify the configuration of the parameters associated with slicing by the orchestrator taking also into account the overall network state. It will be essential to ensure that all the parameters on the three layers are properly configured to guarantee the implementation of end-to-end slicing.
   - On the other hand, it will be necessary to define the use and sharing of each hardware and software element between the slices. From a hardware point of view, it will be necessary to specify whether gateways and LNSs will be exclusively associated

with one slice or whether they can be used for several slices. From the software point of view, it is important to agree on the sharing of Application Server and Join Server components between the slices. The use of separate components between the slices allows better isolation and greater security of traffic routing between the slices. Conversely, sharing these components between slices makes it easier to maintain them and to provision or reconfigure slice through a single API. Moreover, a single shared component provides a consolidated view of the network state and facilitates the scaling decision according to the load.

3. **Gateways parenting to multiple LNS.**

   - The provisioning of vLNSs will be considered in the project in order to allow, depending on the choice of slicing implementation, either the assignment of distinct LNSs to the slices and/or a distribution of the load. Thus, in order to correctly route end-device packets, it will be necessary to manage the association and communication between each gateway and one or more LNSs depending on the type of packet (association of new end-devices or data packets for a type of slice).
   - On the other hand, it will be necessary to extend the definition and configuration of the association between gateways and LNS in the LoRaWAN network. Indeed, the gateways are in charge of filtering the packets which are not destinated for the LoRa network of the gateway, but also of transferring the packets to the good LNS. Therefore, it will be necessary to take into account the various types of slices considered in the project and to allow a dynamic (re)configuration of the associated LNS for a correct transfer of the packets.
   - LoRa gateways are by design half-duplex and cannot listen when transmitting data to end devices. To achieve the expected quality of service of a slice it could be envisaged to dedicate certain gateways only for UL or DL communications.

4. **Respect for the duty cycle.**

   - In LoRaWAN networks, each node (end devices and gateways) must respect a certain duty cycle rate for the use of each channel during UL or DL communications. LNS is in charge of taking into account this aspect in a LoRaWAN network since it is the only node that can have an overview of the network. However, the use of multiple LNSs makes difficult the consideration of duty cycle. To respect the LoRa standard, it will be necessary to delegate this control at the gateway level and take this aspect into account at the orchestrator level when setting up the end devices and gateways.

Appendix A

LoRaWAN orchestration parameters in detail

The purpose of this section is to present a structured set of parameters that could be used to represent a snapshot of the current configuration of a LoRa network. The effort will then be put on the design of an intelligent classification algorithm that can correlate such configurations to a set of abstract network states.

The following parameter list is obtained from the LoRaWAN specification document [8] and the LoRaWAN regional parameters document [20] provided by the LoRa Alliance. Here, by *parameters*, we mean the internal network settings that can be modified in real time during the network operation. Thus, they are inferred from the MAC commands provided by the LoRa specifications to modify the behaviour of the network, that is, the behavior of one of the connected End Devices (ED).

We consider class A end devices, in the following, as they are the most used, power-efficient and configurable, therefore being the true strength of the LoRa technology. However, in other WPs, we could also deal with the case of class B devices, which have other features that could be interesting for slicing. Moreover, for the time being, we focus on the version of the technology for the European 863-870MHz ISM Band, as different regions implement different MAC commands and different radio channels management policies to comply with local regulations.

Finally, in order to identify a structured set of parameters for the whole network, we could use a list containing, for each connected ED, the following sets of parameters. A list of the connected EDs could be easily maintained by the network server. We assume that such a list does not change, that is, no EDs are removed or added to the network during its operation. Therefore, parameters related to network join procedures are not considered.

EU868MHz Class-A LoRaWAN End Device parameters

Transmission window (TX) parameters

- *TX data-rate.* This parameter is set with the LinkADRReq MAC command and is encoded on 4 bits. The encoding is region-specific and is defined in the regional parameters document, as detailed in Table A.1.

  LoRa data-rates are an abstraction: in reality most of them correspond to a coupling of Spreading Factor (SF) and Bandwidth [kHz] (i.e., the parameters of LoRa modulation used to obtain a given data-rate).

[4]In this context it means: LoRa modulation technology
[4]Frequency Shift Keying
[4]Long Range Frequency Hopping Spread Spectrum
[4]Occupied Channel Width

DataRate	Configuration	Indicative physical bit rate [bit/s]
0	LoRa[1]: SF12 / 125 kHz	250
1	LoRa: SF11 / 125 kHz	440
2	LoRa: SF10 / 125 kHz	980
3	LoRa: SF9 / 125 kHz	1760
4	LoRa: SF8 / 125 kHz	3125
5	LoRa: SF7 / 125 kHz	5470
6	LoRa: SF7 / 250 kHz	11000
7	FSK[2]: 50 kbps	50000
8	LR-FHSS[3] CR1/3: 137kHz OCW[4]	162
9	LR-FHSS CR2/3: 137kHz OCW	325
10	LR-FHSS CR1/3: 336kHz OCW	162
11	LR-FHSS CR2/3: 336kHz OCW	325
12..14	Reserved for future use	-
15	Ignore field and keep current configuration	-

Table A.1: TX data rates.

- *TX power.* This parameter is set with LinkADRReq MAC command and is encoded on 4 bits. If it is set to a value higher than the device's maximum TX power, the ED will operate at its maximum TX power. The encoding is region-specific and is defined in the regional parameters document. In Europe, TX Power values are given by Table A.2.

TXPower	Configuration (EIRP)
0	Max EIRP
1	Max EIRP – 2dB
2	Max EIRP – 4dB
3	Max EIRP – 6dB
4	Max EIRP – 8dB
5	Max EIRP – 10dB
6	Max EIRP – 12dB
7	Max EIRP – 14dB
8..14	Reserved for future use
15	Ignore field and keep current configuration

Table A.2: TX power.

  EIRP refers to the Equivalent Isotropically Radiated Power, which is the radiated output power referenced to an isotropic antenna radiating power equally in all directions and whose gain is expressed in dBi.

- *Number of transmissions for each uplink message.* It is set with the LinkADRReq MAC command and by default to 1 (i.e., no re-transmissions by default). The valid range is [1:15] (i.e., a 4 bits integer), with 0 making the ED ignore the field and keep the current configuration.

- *List of possible radio channels for uplink transmissions.* Channels can be added or modified using the NewChannelReq MAC command. The EU863-870 LoRaWAN only supports a maximum of 16 channels. Each end device must include three default channels that

cannot[5] be modified and guarantee a minimal common channel set between end-devices and network gateways. Such channels are defined in the regional parameters document and are listed in Table A.3.

Modulation	**Bandwidth [kHz]**	**Channel frequency [MHz]**	**LoRa DR/Bitrate**	**Duty Cycle**
LoRa	125	868.10 868.30 868.50	DR0 to DR5 / 0.3-5 kbps	<1%

Table A.3: Default channels for LoRa transmission.

DataRates (DRs) numbers correspond to the ones detailed in the *TX data-rate* parameter. For each channel the following information is stored:

- *TX channel frequency.* For non-default channels this parameter is set with the NewChannelReq MAC command. It is encoded using a 24 bits unsigned integer. The actual channel frequency in Hz is the value×100, whereby values representing frequencies below 100 MHz are reserved for future use. This allows setting the frequency of a channel anywhere between 100 MHz to 1.67 GHz in 100 Hz steps. A value of 0 disables the channel. As we are detailing the technology for the European 863-870MHz ISM Band, frequencies supported by all EU868MHz end-devices are in the 863-870MHz spectrum.
- *TX channel min data-rate.* For non-default channels this parameter is set with the NewChannelReq MAC command. It is encoded with 4 bits and it follows the same convention of the *TX data-rate* parameter.
- *TX channel max data-rate.* For non-default channels this parameter is set with the NewChannelReq MAC command. It is encoded with 4 bits and follows the same convention as the *TX data-rate* parameter. It must be higher than the *TX channel min data-rate* parameter.
- *RX1 frequency.* It can be set with the DlChannelReq MAC command and defaults to the uplink frequency. It is encoded using a 24 bits unsigned integer and follows the same convention as a new *TX channel frequency.*

- *Mask of active radio channels for uplink transmissions.* This parameter is set with the LinkADRReq MAC command and is a subset of the list of possible uplink radio channels stored in the end device. The mask is encoded on 16 bits and each bit is set to 1 if the channel can be used for transmission. The bits represent the channels stored in the ED in order.

  Subsequent LinkADRReq MAC commands can be used to set the channels mask if more than 16 channels are allowed. If this is the case, the other fields set by this MAC command will be updated using the last command in the batch.

  Finally, there exists an option in the LinkADRReq MAC command to disable the channel mask entirely.

- *Max aggregated TX duty-cycle.* This parameter is set with the DutyCycleReq MAC command and corresponds to the TX duty-cycle over all sub-bands. The valid range for this parameter encoding is $d = [0 : 15]$ (i.e., a 4 bits integer) and the corresponding maximum duty-cycle is computed as: $\frac{1}{2^d}$.

[5]They can be modified in specifications v1.1

Receiving windows (RX*) parameters

- *RX1 delay from TX end.* This parameter is set with the RXTimingSetupReq MAC command and by default to 1 second. This parameter is encoded using 4 bits, corresponding to the delay in seconds. Therefore, the range is between 1 and 15 seconds, with the value 0 defaulting to 1 second.

- *RX1 data-rate.* This parameter is set with the RXParamSetupReq MAC command and defaults to zero. It is encoded using 3 bits, so its value ranges between $[0:5]$. The actual data-rate is a function of the uplink data rate and the 3 bit parameter (RX1DROffset) as given by Table A.4.

Upstream data rate	**Downstream data rate in RX1 slot**					
RX1DROffset	**0**	**1**	**2**	**3**	**4**	**5**
DR0	DR0	DR0	DR0	DR0	DR0	DR0
DR1	DR1	DR0	DR0	DR0	DR0	DR0
DR2	DR2	DR1	DR0	DR0	DR0	DR0
DR3	DR3	DR2	DR1	DR0	DR0	DR0
DR4	DR4	DR3	DR2	DR1	DR0	DR0
DR5	DR5	DR4	DR3	DR2	DR1	DR0
DR6	DR6	DR5	DR4	DR3	DR2	DR1
DR7	DR7	DR6	DR5	DR4	DR3	DR2
DR8	DR1	DR0	DR0	DR0	DR0	DR0
DR9	DR2	DR1	DR0	DR0	DR0	DR0
DR10	DR1	DR0	DR0	DR0	DR0	DR0
DR11	DR2	DR1	DR0	DR0	DR0	DR0

Table A.4: RX1 data rates.

- *RX1 frequency.* It depends on the channel used for the uplink TX.

- *RX2 data-rate.* It is set with the RXParamSetupReq MAC command and follows the same 4 bit convention of *TX data-rate*. The default value is 0.

- *RX2 frequency.* It is set with the RXParamSetupReq MAC command. It is encoded using a 24 bits unsigned integer and follows the same convention as a new *TX channel frequency*. The default frequency of the second receiving window is 869.525 MHz (10% duty cycle limitation).

Other interesting parameters (from Specifications v1.1, not present in 1.0.4)

- *ADR_ACK_LIMIT parameter of the end device's ADR backoff algorithm.* It is set with the ADRParamSetupReq MAC command and defaults to 64. It is set using a 4 bits integer: the parameter value in seconds can be obtained by elevating the coded integer to the power of 2.

- *ADR_ACK_DELAY parameter of the end device's ADR backoff algorithm.* It is set with the ADRParamSetupReq MAC command and by default to 32. It is set using a 4 bits integer: the parameter value in seconds can be obtained by elevating the coded integer to the power of 2.

Appendix B

LoRaWAN packet metadata

Here follows a detailed list of PHY layer metadata that is produced at gateway level once a packet is demodulated [26]. These metadata can be classified into two categories: radio transmission parameters (e.g., SF and channel frequency) and radio reception measurements (for instance, SNR and RSSI). The following table is taken from the Semtech's packet_forwarder gateway protocol documentation [25].

time	string	UTC time of pkt RX, us precision, ISO 8601 'compact' format
tmms	number	GPS time of pkt RX, number of milliseconds since 06.Jan.1980
tmst	number	Internal timestamp of "RX finished" event (32b unsigned)
freq	number	RX central frequency in MHz (unsigned float, Hz precision)
chan	number	Concentrator "IF" channel used for RX (unsigned integer)
rfch	number	Concentrator "RF chain" used for RX (unsigned integer)
stat	number	CRC status: 1 = OK, -1 = fail, 0 = no CRC
modu	string	Modulation identifier "LORA" or "FSK"
datr	string	LoRa datarate identifier (eg. SF12BW500)
datr	number	FSK datarate (unsigned, in bits per second)
codr	string	LoRa ECC coding rate identifier
rssi	number	RSSI in dBm (signed integer, 1 dB precision)
lsnr	number	Lora SNR ratio in dB (signed float, 0.1 dB precision)
size	number	RF packet payload size in bytes (unsigned integer)
data	string	Base64 encoded RF packet payload, padded

Moreover, once the packet arrives at the server, its MAC header is decrypted and a number of MAC layer parameters can be obtained. Some of the most relevant ones are the address used to identify the sender device, the frame counter, message type and parameters related to ADR. For a complete list look up the MAC header section in the LoRaWAN specifications [8].

Bibliography

[1] 3GPP TR 28.801 V15.1.0. Study on management and orchestration of network slicing for next generation network (Release 15), 2018.

[2] 3GPP TS 23.501 V16.0.2. System Architecture for the 5G System (Release 16), 2019.

[3] 3GPP TS 23.502 V16.2.0. Procedures for the 5G System (5GS), Stage 2, (Release 16), 2019.

[4] 3GPP TS 23.503 V16.2.0 . Policy and Charging Control Framework for the 5G System (5GS), Stage 2, (Release 16), 2019.

[5] 3GPP TS 28.530 V15.01.0. Management and orchestration; Concepts, use cases and requirements (Release 15), 2018.

[6] 3GPP TS 28.531 V15.1.0. Management and orchestration Provisioning (Release 15), 2018.

[7] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne. Understanding the limits of lorawan. *IEEE Communications Magazine*, 55(9):34–40, 2017.

[8] LoRa Alliance. LoRa Alliance Releases LoRaWAN® TS1-1.0.4 Specification; Simplifies Development, Deployment, and Interoperability, October 2020. Available online: https://lora-alliance.org/resource_hub/lorawan-104-specification-package/ (accessed on 10 December 2020).

[9] Martin C Bor, Utz Roedig, Thiemo Voigt, and Juan M Alonso. Do lora low-power wide-area networks scale? In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 59–67. ACM, 2016.

[10] Meriam Bouzouita, Yassine Hadjadj-Aoul, Nawel Zangar, and Gerardo Rubino. Estimating the number of contending IoT devices in 5G networks: Revealing the invisible. *Transactions on emerging telecommunications technologies*, (e3513):1–18, September 2018.

[11] Samir Dawaliby, Abbas Bradai, and Yannis Pousset. Adaptive dynamic network slicing in lora networks. *Future generation computer systems*, 98:697–707, 2019.

[12] Samir Dawaliby, Abbas Bradai, and Yannis Pousset. Distributed network slicing in large scale iot based on coalitional multi-game theory. *IEEE Transactions on Network and Service Management*, 16(4):1567–1580, 2019.

[13] Samir Dawaliby, Abbas Bradai, and Yannis Pousset. Network slicing optimization in large scale lora wide area networks. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 72–77, 2019.

[14] Samir Dawaliby, Abbas Bradai, and Yannis Pousset. Joint slice-based spreading factor and transmission power optimization in lora smart city networks. *Internet of Things*, 14:100121, 2021.

[15] Samir Dawaliby, Abbas Bradai, Yannis Pousset, and Roberto Riggio. Dynamic network slicing for lorawan. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 134–142, 2018.

[16] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[17] Rachel Kufakunesu, Gerhard P Hancke, and Adnan M Abu-Mahfouz. A survey on adaptive data rate optimization in lorawan: Recent solutions and major challenges. *Sensors*, 20(18):5044, 2020.

[18] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[19] LoRa Alliance. *LoRaWAN L2 1.0.4 Specification (TS001-1.0.4)*, 2020. Available online: https://lora-alliance.org/resource_hub/lorawan-104-specification-package/ (accessed on 10 December 2020).

[20] LoRa Alliance. *RP2-1.0.2 LoRaWAN Regional Parameters*, 2020. Available online: https://lora-alliance.org/resource_hub/rp2-102-lorawan-regional-parameters/ (accessed on 10 December 2020).

[21] Seifeddine Messaoud, Samir Dawaliby, Abbas Bradai, and Mohamed Atri. In-depth performance evaluation of network slicing strategies in large scale industry 4.0. In *2021 18th International Multi-Conference on Systems, Signals Devices (SSD)*, pages 474–479, 2021.

[22] The Things Network. The Things Network Docs: LoRaWAN Adaptive Data Rate. https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate.html. [Online; Accessed: 22-03-2021].

[23] Jorge Ortín, Matteo Cesana, and Alessandro Redondi. How do aloha and listen before talk coexist in lorawan? In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7, 2018.

[24] Veronica Quintuna Rodriguez, Fabrice Guillemin, and Amina Boubendir. 5g e2e network slicing management with onap. In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 87–94, 2020.

[25] Semtech. Lora network packet forwarder project. https://github.com/Lora-net/packet_forwarder. [Online; Accessed: 04-06-2021].

[26] Semtech. SX1301 Datasheet. https://www.semtech.com/products/wireless-rf/lora-gateways/sx1301. Accessed: 29-04-2021.

[27] Mariusz Slabicki, Gopika Premsankar, and Mario Di Francesco. Adaptive configuration of LoRa networks for dense IoT deployments. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2018.