Wireless Communications and Mobile Computing

Testbeds for Future Wireless Networks

Lead Guest Editor: Jorge Navarro-Ortiz
Guest Editors: Cristina Cervello-Pastor, Giovanni Stea, Xavier Costa, and Joan Triay

Testbeds for Future Wireless Networks

Wireless Communications and Mobile Computing

Testbeds for Future Wireless Networks

Lead Guest Editor: Jorge Navarro-Ortiz
Guest Editors: Cristina Cervello-Pastor, Giovanni Stea, Xavier Costa, and Joan Triay

Editorial Board

Contents

Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 2382471, 2 pages
https://doi.org/10.1155/2019/2382471

Editorial

Testbeds for Future Wireless Networks

Jorge Navarro-Ortiz,[1] **Cristina Cervelló-Pastor**,[2] **Giovanni Stea**,[3] **Xavier Costa-Perez**,[4] **and Joan Triay**[5]

[1] *Universidad de Granada, Granada, Spain*
[2] *Universitat Politècnica de Catalunya, Barcelona, Spain*
[3] *Università di Pisa, Pisa, Italy*
[4] *NEC Laboratories Europe, Heidelberg, Germany*
[5] *DOCOMO Communications Lab. Europe, Munich, Germany*

Correspondence should be addressed to Jorge Navarro-Ortiz; jorgenavarro@ugr.es

Received 2 May 2019; Accepted 5 May 2019; Published 16 June 2019

In the last years, the growth of wireless traffic has been enormous. For instance, according to the latest CISCO forecast, the number of mobile devices will grow from 8 billion in 2016 to 11.6 billion in 2021, including 3.3 billion M2M connections. Similarly, other wireless technologies such as Wi-Fi—with 541 million public hotspots by 2021—and LPWAN—accounting for 31% of M2M devices by 2021—will also experience a massive increase.

The scientific literature dealing with present and future wireless networks is indeed abundant. A large part of it focuses on formal concepts and theoretical architectures, which are evaluated via analytical models or simulation. Although theoretical works are indispensable for the advancement of knowledge and innovation, testbeds and prototypes are essential to demonstrate the correct operation and performance of these technologies.

Designing and constructing testbeds and prototypes is a non-trivial endeavor, which often requires joint expertise from different engineering areas (such as signal processing, electronics, networking, and software). Sound measurement and test methodologies are required and must be documented. The practical insight gained in setting up a testbed – such as lessons learned, dos and do nots – is often very relevant to researchers and practitioners. Moreover, testbeds and prototypes are in fact enablers of further research, acting as catalyzers of inter-institution collaborations.

This special issue is to explore recent advances and state-of-the-art research related to the development of testbeds and prototypes for relevant wireless and mobile technologies, including radio, software, and architectural aspects. The response to our call for papers for this special issue was satisfactory, and we finally selected a total of six high-quality papers, after a thorough peer-review process.

Software-defined Networking (SDN) is nowadays one of the main network architecture paradigms used for building experimentation testbeds. Even more, SDN principles are taken as a foundation to re-engineer networks. One good example in the present special issue is the paper entitled "Design and experimental validation of a software-defined radio access network testbed with slicing support", by Aikaterini K. Koutlia et al. The paper describes the key design and implementation aspects of an experimental testbed of a Software-defined Radio Access Network (SD-RAN) with slicing support. In the testbed, an SD-RAN Controller aggregates part of the RAN control plane functionality (e.g., slice-aware/multi-cell admission control) together with RAN slicing management functionality. The functional design of the testbed is in line with 3GPP Release-15 specifications related to information modeling and network slicing management. The testbed is used to demonstrate the provisioning of RAN slices (i.e., preparation, commissioning, and activation phases) and the operation of the implemented Radio Resource Management (RRM) functionality for slice-aware admission control and scheduling with commercially-available user equipment (UE). The clearer decoupling of control and forwarding planes in SDN, and the flexibility

that such approach brings enabling centralized network control, are further explored by Ricardo Santos et al. in the paper entitled "MmWave backhaul testbed configurability using software-defined networking". In the paper, the authors propose a Software-defined Networking approach to enable a wireless backhaul for small cells. The wireless backhaul is formed by a mesh of millimeter-wave links with multi-hop paths, which can be dynamically reconfigured to interconnect the cells. This solution enables the provision of a multigigabit and low-latency interconnection without using optical links, which may not be economically feasible for large-scale deployments. The proposed network architecture, named SOCRA, makes the SDN control plane responsible for configuring not only the data plane forwarding, but also the link configuration, antenna alignment, and adaptive node power-on/off operations. The performance evaluation of the implemented testbed shows that an optimal channel assignment between the mesh links can result in a 44% throughput increase, when compared to a sub-optimal configuration. In the implementation of the testbed, diverse types of commercially available equipment (such as mini-computers and power units) and open source software components are employed. Finally, Hai Xue et al. propose two packet scheduling schemes, First Come First Serve-Pushout (FCFS-PO) and FCFS-PushOut-Priority (FCFS-PO-P) to effectively handle the overload issue of multiple-switch SDNs targeting the edge-computing environment in the paper entitled "Packet scheduling for multiple-switch software-defined network in edge-computing environment". After developing analytical models for their operation, extensive experiments on a real testbed are carried out. The proposed schedulers outperform the classical FCFS-Block (FCFS-BL) in terms of packet waiting time. Furthermore, FCFS-PO-P achieves better performance in terms of sojourn and waiting time for various traffic conditions.

This special issue also includes interesting articles that employ the Network Functions Virtualization (NFV) paradigm to virtualize and orchestrate the required network services. Similar to SDN, NFV is one of the enablers for future wireless networks such as 5G. In the context of the Internet of Things (IoT), Jorge Navarro-Ortiz et al. present in the paper "A LoRaWAN testbed design for supporting critical situations: prototype and evaluation" a self-healing LoRaWAN network architecture to provide resilience when part of the equipment in the core network becomes faulty. Resilience is achieved by virtualizing and properly orchestrating the different network entities. Different options have been designed and implemented as real prototypes. Based on the performance evaluation, the authors claim that microservice orchestration along with several replicas allows the network administrators to seamlessly recover after a catastrophic situation. Focusing on Unmanned Aerial Vehicles (UAVs), Christian Tipantuña et al. introduce an optimal drone scheduling algorithm in paper "An NFV-Based Energy Scheduling Algorithm for a 5G Enabled Fleet of Programmable Unmanned Aerial Vehicles", which, by leveraging 5G and NFV capabilities, is able to perform an efficient energy-aware management of resources for network services provisioning. Simulation results validate the proposal and evaluate its performance in terms of both results (i.e., achieved metrics), and costs (i.e., the amount of resources needed for the execution of services in different scenarios). A testbed with several Raspberry Pi 3B and a specific power meter has been employed to measure power consumption under the simulated conditions. Future work includes the modelling of the services availability and the design of heuristic approaches.

Finally, one of the works highlights the importance of these testbeds and prototypes in a European funded project. In particular, in the paper "QoE evaluation: the TRIANGLE testbed approach", Almudena Diaz Zayas et al. describe a testbed, the test methodology, and the set of test cases developed within the TRIANGLE project, with the objective of testing and benchmarking mobile applications, services, and devices. The testbed can be remotely accessed through a web portal or through a Keysight Testing Automation Platform for advanced users. A set of test cases has been implemented that specify the measurements that should be collected to compute the Key Performance Indicators of the feature under test, which are then normalized into a standard 1-to-5 scale, the one typically used in Mean Opinion Scores (MOS). These synthetic MOS values are finally aggregated to obtain a synthetic MOS score in each test case. The paper includes a summary of these scores for the considered scenarios.

Conflicts of Interest

The editors declare that they have no conflicts of interest regarding the publication of this special issue.

Acknowledgments

The editors thank all of the authors for their submissions to this special issue. We hope that this special issue will further encourage research interests in the area of wireless network testbeds and prototypes.

Jorge Navarro-Ortiz
Cristina Cervelló-Pastor
Giovanni Stea
Xavier Costa-Perez
Joan Triay

Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 2361352, 17 pages
https://doi.org/10.1155/2019/2361352

Research Article

Design and Experimental Validation of a Software-Defined Radio Access Network Testbed with Slicing Support

K. Koutlia,[1] **R. Ferrús**,[1] **E. Coronado**,[2] **R. Riggio**,[2] **F. Casadevall**,[1] **A. Umbert**,[1] **and J. Pérez-Romero**[1]

[1]*Universitat Politècnica de Catalunya (UPC), Spain*
[2]*Fondazione Bruno Kessler (FBK), Italy*

Correspondence should be addressed to K. Koutlia; katkoutlia@tsc.upc.edu

Received 16 November 2018; Accepted 14 April 2019; Published 12 June 2019

Guest Editor: Joan Triay

Network slicing is a fundamental feature of 5G systems to partition a single network into a number of segregated logical networks, each optimized for a particular type of service or dedicated to a particular customer or application. The realization of network slicing is particularly challenging in the Radio Access Network (RAN) part, where multiple slices can be multiplexed over the same radio channel and Radio Resource Management (RRM) functions shall be used to split the cell radio resources and achieve the expected behaviour per slice. In this context, this paper describes the key design and implementation aspects of a Software-Defined RAN (SD-RAN) experimental testbed with slicing support. The testbed has been designed consistently with the slicing capabilities and related management framework established by 3GPP in Release 15. The testbed is used to demonstrate the provisioning of RAN slices (e.g., preparation, commissioning, and activation phases) and the operation of the implemented RRM functionality for slice-aware admission control and scheduling.

1. Introduction

5G systems are being designed to support a wider range of applications and business models than previous generations due to the anticipated adoption of 5G technologies in multiple market segments (e.g., automotive, e-health, utilities, smart cities, agriculture, media, entertainment, and high-tech manufacturing) and the consolidation of more flexible and cost-efficient service delivery models (e.g., neutral host network providers, Network as a Service, enterprise, and private cellular networks). Through the support of network slicing [1], 5G systems are expected to become flexible and versatile network infrastructures where logical networks partitions can be created (i.e., network slices) with appropriate isolation and optimized characteristics to serve a particular purpose or service category (e.g., applications with different access and/or functional requirements) or even individual customers (e.g., enterprises, third party service providers). This is especially relevant for the Radio Access Network (RAN), which is the most resource-demanding (and costliest) part of the mobile network and the most challenged by the support of network slicing [2].

System architecture and functional aspects to support network slicing in 5G Core Network (5GC) and Next Generation RAN (NG-RAN) have already been defined in the first release of the 5G normative specifications approved by 3GPP (e.g., network slice identifiers, procedures and functions for network slice selection, etc.) [3, 4]. Moreover, implementation aspects of network slicing in the NG-RAN have been studied from multiple angles, ranging from virtualization techniques and programmable platforms with slice-aware traffic differentiation and protection mechanisms [5–7] to algorithms for dynamic resource sharing across slices [8]. In this respect, we analysed in [9] the RAN slicing problem in a multicell network in order to show how Radio Resource Management (RRM) functionalities can be used to properly share the radio resources, and we developed in [10] a set of vendor-agnostic configuration descriptors intended to characterize the features, policies, and resources to be put in place across the radio protocol layers

of a NG-RAN node for the realization of concurrent RAN slices.

On the other hand, management solutions necessary for the exploitation of network slicing capabilities in an automated and business agile manner are at a much more incipient stage, particularly for what concerns the NG-RAN. In this regard, a network slice lifecycle management solution for end-to-end automation across multiple resource domains is proposed in [11], including the RAN domain for completeness, but not addressing it in detail. More focused on a 5G RAN; [12] proposes the notion of an on-demand capacity broker that allows a RAN provider to allocate a portion of network capacity for a particular time period, while [13] provides some insight on the need to extend current RAN management frameworks to support network slicing [1] and gives an extensive overview of related use cases. Further progressing on this topic, a functional framework for the management of network slicing for a NG-RAN infrastructure was introduced in [14], detailing the functional components, interfaces, and information models that shall be in place, together with a discussion on the complexity of automating the RAN provisioning process. More recently, specifications for a new service-based overall management architecture for 5G systems and network slicing has been concluded by 3GPP as part of Release 15 specifications [15, 16]. In this context, building upon the functional framework for RAN slicing management in [14] and consistently with the new service-based management architecture in 3GPP Release 15, this paper describes a Software-Defined RAN (SD-RAN) experimental testbed that allows RAN slices to be automatically provisioned through a RESTful Application Programming Interface (API). Moreover, the testbed implements RRM functions for admission control and scheduling able to treat differently connections belonging to different slices (referred to as *slice-aware* RRM functions in the following). For the characterization of the slicing features at management level, the latest 3GPP Release 15 information models are used as baseline and an extension is proposed to enable a more fine-grained characterization of the slice-aware RRM policies. The experimental testbed is implemented using open-source RAN distributions (srsLTE [17] and OAI [18]) and the 5G-EmPOWER platform [19]. The testbed is used to experimentally showcase and validate the operation of the slice provisioning phases (e.g., preparation, commissioning, and activation of RAN slices) offering the isolation level required, as well as the runtime operation of the implemented slice-aware RRM functionality. Unlike other existing prototypes and Proof of Concepts (PoCs) of RAN slicing features [20, 21], the use of the 5G-EmPOWER platform in our tested allows us to come up with a RAN slicing solution that can be ported to different RAN implementation.

The rest of the paper is organized as follows. Section 2 describes the overall solution framework for RAN slicing management and presents the proposed extension of the 3GPP information models for characterizing the RAN slices. Section 3 describes the experimental platform, discussing the design details of each of the main testbed components and the implemented slice-aware RRM functions. Section 4 focuses on showcasing the management and provisioning of RAN slices, while the operational validation and performance assessment of the testbed under a given RAN slicing configuration is addressed in Section 5. Finally, concluding remarks and future work are presented in Section 6.

2. Solution Framework for Automated RAN Slicing Provisioning

From a service perspective, 3GPP defines a network slice as a particular behaviour delivered by a 5G network. Such behaviour is identified by a Single Network Slice Selection Assistance Information (S-NSSAI) identifier within a Public Land Mobile Network (PLMN). From an implementation perspective, the realization of a network slice is referred to as a Network Slice Instance (NSI). A NSI consists of a set of network function instances and the required resources (e.g., compute, storage, and networking resources) that are deployed to serve the traffic associated with one or several S-NSSAIs. A NSI is composed of one or several Network Slice Subnet Instance(s) (NSSI(s)). For example, one NSI can be formed by a NSSI with the RAN functions, denoted in the following as a RAN Slice Instance (RSI), and another NSSI with the 5G core network functions.

Focusing on the RAN part, the implementation of a RSI offers different possibilities on how the NG-RAN infrastructure functions and resources are orchestrated, including how radio spectrum is distributed among RSIs (e.g., RSI-dedicated or shared spectrum). As a general case, let us consider a NG-RAN infrastructure where the base station functions (denoted as gNB for the 5G New Radio [NR] interface) are delivered as a combination of several network functions (e.g., gNB-Distributed Units [gNB-DU] and gNB-Central Units [gNB-CU] hosting different parts of the L3, L2, and L1 radio layer functions) and implemented either as dedicated hardware appliances (i.e., Physical Network Functions (PNFs), after ETSI terminology) or as Virtual Network Functions (VNFs) running on general-purpose hardware, e.g., Network Function Virtualization Infrastructure (NFVI). Under such scenario, a RSI can be realized as a particular configuration of a set of gNB-CU/DU(s) in terms of enabled radio protocol features and radio access capacity guarantees or limitations. The same set of gNB-CU/DU(s) is likely to be shared by several RSIs. On this basis, the support of RAN slicing management capabilities involves different management components as illustrated in Figure 1. The core functionality consists of a set of management functions, collectively referred to as RAN Slicing Management Function (RSMF) in Figure 1, in charge of the Lifecycle Management (LCM) of RSIs (e.g., creation, modification, and termination of RSIs). To that end, in line with the terminology and service-based management concepts adopted by 3GPP in Release 15 [16], the RSMF exposes a set of management services for the provisioning and monitoring of RSIs (e.g., management services to request the creation of a RSI based on templates, management services to create a measurement job for collecting the performance data of RSIs, etc.). In this respect, the RSMF plays the role of a producer of management services that can be accessed by one or multiple management

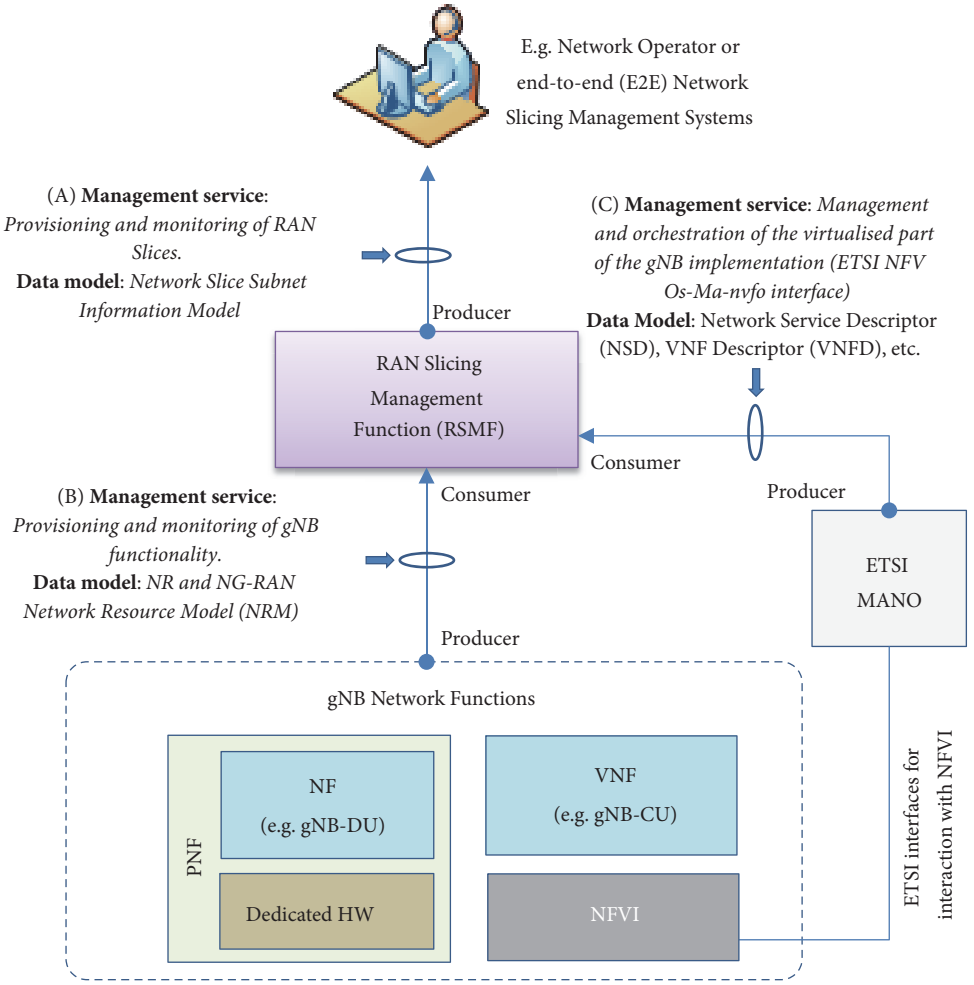


Figure 1: Functional framework for automated RAN slicing management.

service consumers. One consumer of the RSMF management services is the network operator (e.g., operator staff that access the RSMF through web-based interfaces or GUIs). Another consumer could be a management system in charge of the LCM of NSIs in which the RSIs are components of the end-to-end slices. In addition, a limited/restricted set of the management capabilities offered by the RSMF might be exposed to the external users (e.g., tenants using the slices) after enforcing the desired exposure governance.

On the other hand, in order to interact with the underlying infrastructure components and carry out the LCM of the RSIs, the RSMF has to be able to consume the management services provided by the set of PNFs and VNFs hosting the gNB functions. The management of the gNB functions can be realized via standardized interfaces (e.g., the new 3GPP service-based interfaces or legacy 3GPP interfaces such as Itf-N). Through these interfaces, the RSMF would allocate and configure the necessary L3, L2, and L1 radio access functions and resources (e.g., identifiers, resource reservations) for the creation and operation of RSIs within the gNB functions. Moreover, for the management of NFVI-related aspects of the VNFs hosting part of the gNB functions, the RSMF can consume the management interfaces provided by ETSI NFV Management and Orchestration (MANO)-compliant solutions, such as the Os-Ma-nfvo interface for LCM of network services and VNFs [22].

To support all the interactions across the aforementioned management interfaces, information models to represent the manageable characteristics of the L3, L2, and L1 radio access functions and resources are necessary. In this respect, 3GPP Release 15 specifications define the information models (i.e., managed object classes, attributes, and relations) of the components to be managed through the management services tagged as (A) and (B) in Figure 1. These information models are known as Network Resource Models (NRMs). Specifically, NRM definitions are provided for characterizing a Network Slice Subnet Instance (NSSI) (a NSSI is the central building block of a Network Slice Instance [NSI]; that is, a NSI is always composed of one or several NSSI(s) [15]. Note that a RAN Slice Instance (RSI) as defined in this paper is a particular realization of a NSSI for the NG-RAN functionality) through interface (A), together with NRM definitions for characterizing the gNB functions and its supported slicing features through interface (B). For the sake

Table 1: 3GPP defined attributes for network slicing configuration.

NSSI NRM (SliceProfile class)	
Attribute	**Explanation**
SliceProfileId	A unique identifier of the slice profile.
SNSSAI	Set of supported S-NSSAI(s) in the NSSI. Each S-NSSAI is comprised of a SST (Slice/Service type) and an optional SD (Slice Differentiator) field.
PLMNId	Set of PLMN(s) associated with the NSSI.
PerfReq	It specifies the requirements to the NSSI in terms of the scenarios defined in the TS 22.261, such as experienced data rate, and area traffic capacity (density) of UE density. Limitation of the attribute values is not addressed.
maxNumber ofUEs	It specifies the maximum number of UEs that may simultaneously access the NSSI.
coverageArea TAList	List of tracking area(s) where the NSSI can be selected.
Latency	It specifies the packet transmission latency (millisecond) through the RAN, CN, and TN part of 5G network
UEMobility Level	It specifies the mobility level of a UE accessing the NSSI. Allowed values: stationary, nomadic, restricted mobility, and fully mobility.
Resource SharingLevel	It specifies whether the resources to be allocated to the network slice instance may be shared with another network slice instance(s). Allowed values: shared, non-shared.
NR and NG-RAN (gNB) NRM	
Attribute	**Explanation**
SNSSAI	Set of supported S-NSSAI(s).
RRMPolicy	Represents the RRM policy, which includes guidance for split of radio resources between multiple slices the cell supports. The RRM policy is implementation dependent.

of having a comprehensive view of the scope of the NRM definitions, Table 1 outlines the main attributes in both NRMs that directly cope with the slicing features.

From Table 1, it can be noted that the slicing modelling established by 3GPP basically intends to define a minimum set of high-level attributes for network slicing management. This approach brings high flexibility as very few constraints are imposed on the specific configurations, though it only allows for a coarse-grained management capability, especially when it comes to the configuration of the radio resources in the gNB for RAN slicing. Therefore, building upon these high-level 3GPP models, a more fine-grained management of the RAN slices necessarily requires the extension or addition of new attributes for a more precise characterization of the behaviour of a RAN slice. In this regard, focusing on the problem of how to split the radio resources between multiple slices, the solution proposed in this paper develops the semantics of the *RRMPolicy* attribute of the gNB NRM, which is implementation dependent, to convey a more detailed configuration of the RAN slices. This is carried out by leveraging our previous work [10], in which a set of comprehensive configuration descriptors was proposed to parametrize the features, policies, and resources put in place across the L1, L2, and L3 radio protocol layers distributed across the set of PNFs and VNFs that jointly provide the full radio access functions.

An illustration of the formulation of the *RRMPolicy* attribute based on the proposed descriptors is depicted in Figure 2. It consists of three configuration descriptors, referred as *L3*, *L2,* and *L1 slice descriptors*, which are used to characterize the operation of the underlying radio protocol layers for the realization of the RAN slice. A brief description of these descriptors that is sufficient to base the subsequent design and implementation aspects is provided below. Further details on the different parameters included in the descriptors, together with the rationale behind, can be found in [10].

With regard to the *L3 slice descriptor*, L3 comprises the Radio Resource Configuration (RRC) protocol and RRM functions such as Radio Bearer Control (RBC), Radio Admission Control (RAC), and Connection Mobility Control (CMC) for the activation and maintenance of Radio Bearers (RB), which are the data transfer services delivered by the radio protocol stack. For each UE, one or more user plane RBs, denoted as Data RBs (DRBs), can be established per Protocol Data Unit (PDU) session, which defines the connectivity service provided by 5GC [3]. A *L3 slice descriptor* is necessary to specify the capacity allocation for the RAN slice (e.g., number and characteristics of the DRBs that can be simultaneously established), the RRM policies that govern the operation of the slice (e.g., DRB configuration policies), and the capability set of the RRC protocol in use (e.g., application type specific RRC messages).

As to the *L2 slice descriptor*, L2 comprises a Medium Access Control (MAC) sublayer for multiplexing and scheduling the packet transmissions of the DRBs over a set of transport channels exposed by L1. Moreover, L2 embeds a number of processing functions configurable on a per-DRB basis for e.g., segmentation, Automatic Repeat reQuest (ARQ) retransmissions, compression, and ciphering (i.e.,

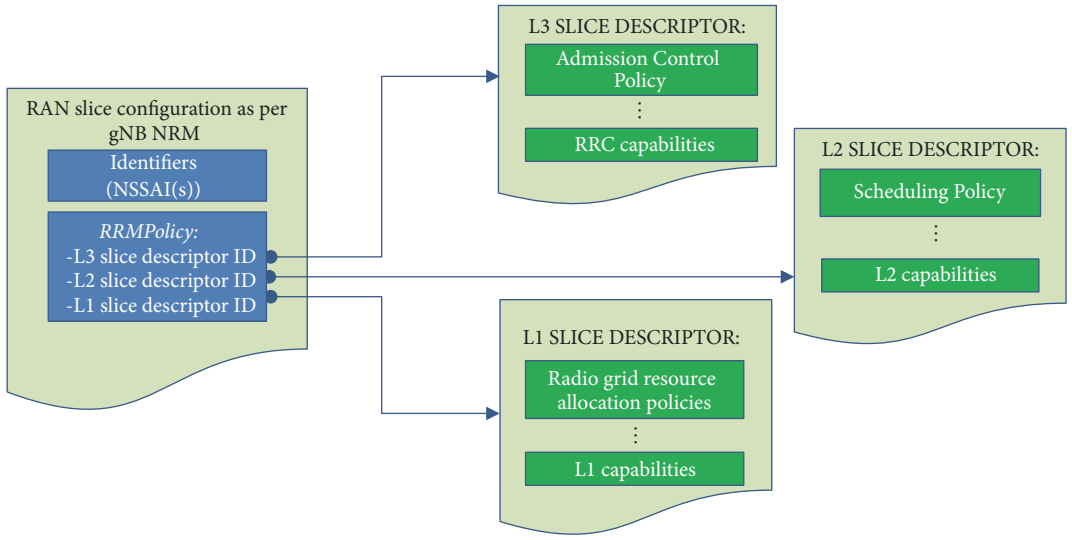


Figure 2: Implementation of the *RRMPolicy* attribute semantics to represent a set of descriptors for the configuration of the L3, L2, and L1 radio access functions for the realization of a RSI.

Radio Link Control [RLC] and Packet Data Convergence Protocol [PDCP]). In the NR specifications, an additional L2 sublayer named Service Data Adaptation Protocol (SDAP) is included to map the DRBs and the traffic flows managed by the 5GC, referred to as QoS Flows [3] SSI. Considering that the current MAC operation is based on individual UE and DRB specific QoS profiles, a *L2 slice descriptor* is necessary to define the packet scheduling behaviour to be enforced on the traffic aggregate of DRBs of the same slice and to specify the capability set of the applicable L2 sublayers processing functions.

With regard to the *L1 slice descriptor*, L1 provides L2 with transfer services in the form of transport channels, which define the way in which data is transferred (e.g., Transmission Time Interval [TTI], channel coding). L1 also establishes the corresponding radio resource structure of the cell radio resources (e.g., waveform characteristics and time/frequency domain resource structure). Considering that a RAN slice may require specific L1 transfer service capabilities (e.g., low latency shared transport channel) and/or specific radio resource allocation of the cell radio resources, a *L1 slice descriptor* is needed to specify both aspects.

3. Software-Defined RAN (SD-RAN) Experimental Testbed

A high-level view of the experimental testbed for showcasing and validating the solution framework for RAN slicing management presented in Section 2 is depicted in Figure 3. The testbed includes a disaggregated RAN implementation, in which some control-plane functions in the RAN (e.g., slice-aware/multicell admission control) are centralized in the form of a SD-RAN Controller. Together with the RAN control-plane functionality, the SD-RAN Controller also integrates the RSMF function, which is responsible for LCM of the RSIs. The SD-RAN Controller has been implemented as an extension of the 5G-EmPOWER Operating System (OS), which is an open-source experimental platform for 5G-service development and testing [19]. As illustrated in Figure 3, the SD-RAN Controller provides a RESTful API for the provisioning of slices and the so-called 5G-EmPOWER Northbound API for running control and management applications on top of the controller, such as the multicell/slice-aware admission control and multicell/slice-aware scheduling coordination functions explained later on.

The interaction of the SD-RAN Controller with the distributed functions of the RAN is performed through the 5G-EmPOWER Agent, which is a functionality embedded within the access nodes. The testbed used for the prototype is based on a research-oriented open-source SDR implementation of a LTE eNB (i.e., srsLTE [17]) and Ettus Research Universal Software Radio Peripherals (USRPs) b210, as depicted in Figure 3. Notice that the support of specific radio front-ends is up to the implementer of the network stack (i.e., in this case up srsLTE), and independent from the 5G-EmPOWER capabilities, which are compatible with any radio front-end supported by the stack. It is important to highlight that although the 5G-EmPOWER OS is able to support 4G and 5G networks, the validation of the prototype is at the moment limited to 4G since no open-source 5G stacks are currently available for experimentation. Accordingly, the eNBs are connected in the testbed to open-source Evolved Packet Core (EPC) (e.g., OAI [18] and NextEPC [23]) implementation to, jointly with the RAN, provide IP connectivity services to a number of off-the-shelf User Equipment (UE) terminals used for testing. The EPC, the eNBs, and the SD-RAN Controller are deployed on an Intel NUC with an i5 Intel processor and 16 GB of RAM memory running Ubuntu 18.04.1. However, alternatively, each of these components can be also deployed in independent off-the-shelf laptops equipped with the same processor family and at least 4GB of RAM. More details on each component are given in the following subsection.

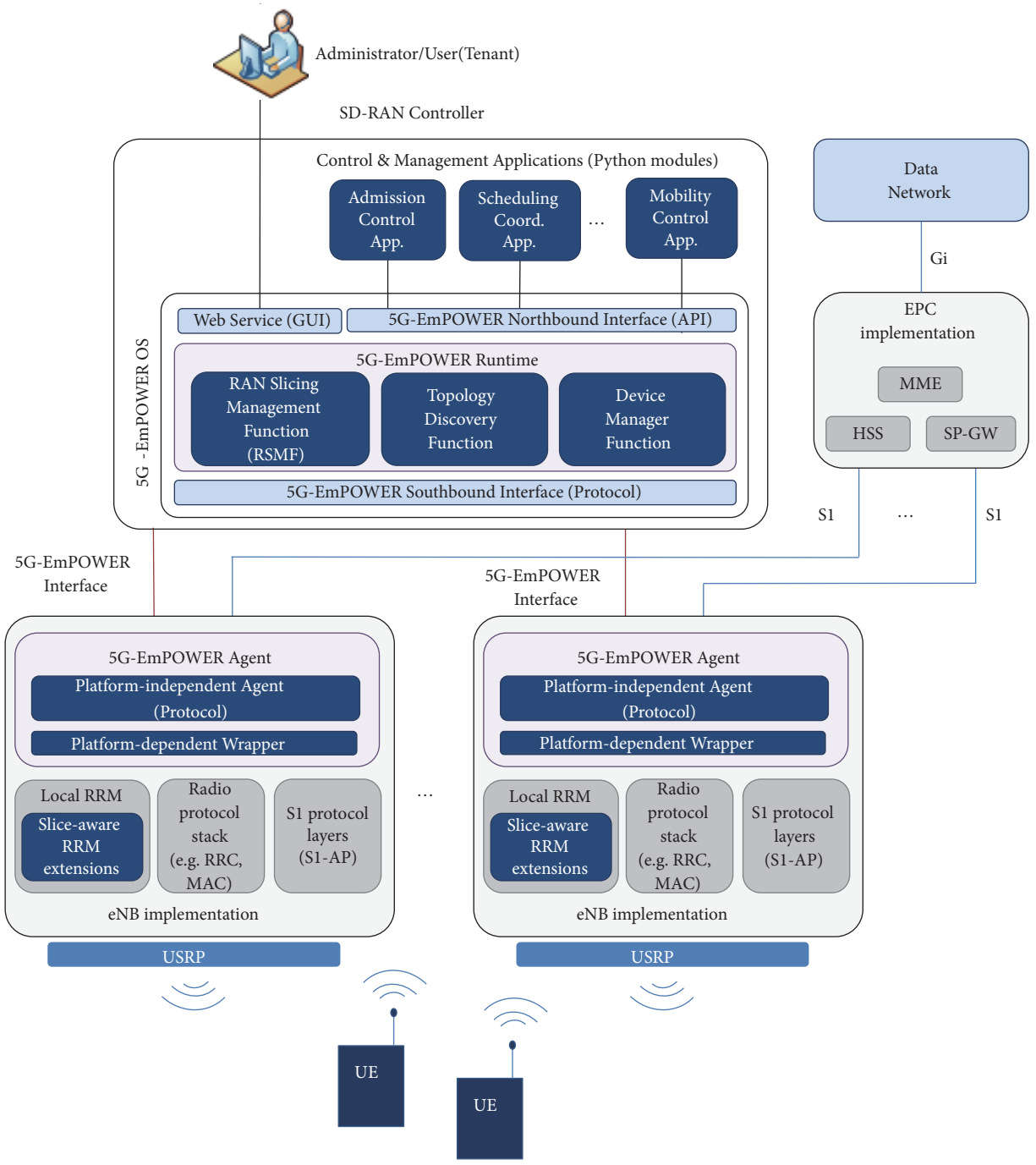


Figure 3: SD-RAN experimental testbed with RAN slicing management features.

3.1. The 5G-EmPOWER Operating System. The 5G-EmPOWER OS provides a framework for managing heterogeneous RAN nodes (e.g., 3GPP RAN nodes; Wi-Fi access points) based on the OpenEmpower protocol, together with a collection of built-in functionalities, services, and APIs that can be used to program control and management applications. The 5G-EmPOWER OS is implemented in Python using the Tornado Web Server as web framework.

The internal implementation of the OS follows a modular architecture. As a matter of fact, except for the logging subsystem (which must be available before any other module is loaded), every task supported by the 5G-EmPOWER OS is implemented as a plug-in (i.e., a Python module) that can be loaded at runtime. Modules can be built-in and launched at bootstrap time or started and stopped at runtime. Each module consists of a Manifest file containing the module meta-data (version, dependencies, etc.) and one or more Python scripts. Developers are free to decide how their network control and management applications are deployed. For example, the developed applications can implement

Table 2: Template with the descriptors for the specification of a RAN slice.

Attributes	Description
Identifiers	**RAN Slice ID:** Identifies the RAN Slice Instance
	PLMNList: Set of PLMN(s) served through the RAN Slice
	NSSAIList: Set of S-NSSAI(s) per PLMN served through the RAN slice
	CellList: Set of cells where the RAN Slice can be selected
RRMPolicy-L3 Descriptor	Admission Control Policy:
	Aggregated radio load [units: %][(QCI, ARP) pairs][cell-level, slice-level][Minimum, Maximum]
	Aggregated bit rate [units: b/s] [(QCI, ARP) pairs][cell-level, slice-level][Minimum, Maximum]
	Number of DRBs [(QCI, ARP) pairs][cell-level, slice-level][Minimum, Maximum]
	Number of UEs [cell-level, slice-level][Minimum, Maximum]
	Averaging Window: It represents the duration over which the radio load and bit rate shall be calculated
RRMPolicy-L2 Descriptor	Scheduling Policy:
	Inter-slice scheduling: [algorithm] optionally [resource units per slice: %]
	Intra-slice scheduling: [algorithm]

control capabilities per RAN slice or for the overall operation of the RAN. Moreover, an application can consist of a single or several modules. This approach is similar, in principle, to the Network Function Virtualization (NFV) paradigm, where complex services can be deployed by combining several VNFs. Likewise, in the 5G-EmPOWER OS, complex network management applications can be designed by deploying and combining different modules. This allows developers to dynamically set up a network monitoring application to perform a site survey or to roll out new features at runtime by selecting them from an "app store". In the implemented testbed, the 5G-EmPOWER OS includes three core modules, namely:

(i) *Device Manager Function*. It tracks the eNBs active in the RAN. This includes their IP address and their identifier, i.e., the eNB ID, the last seen date, and a list containing the capabilities of the eNB (e.g., Downlink/Uplink E-UTRA Absolute Radio Frequency Channel Number (DL/UL EARFCN)). The device manager exposes an API allowing applications to receive events when new eNBs join or leave the RAN.

(ii) *Topology Discovery Function*. This function is implemented as a collection of modules that allow the controller to collect static (with long term dynamics), as well as dynamic (short term) information of the network. The first type comprises information about how RAN nodes are interconnected (i.e., how the eNBs are interconnected with each other through, for example, X2 interfaces) in order to build a logical connection map of the network. This map is updated by the SD-RAN Controller upon the reception of events raised by the Agent when links are added or removed, case that does not occur in a frequent basis. The second type is related to periodic measurements from the eNBs and the UEs (e.g., RSRP/RSRQ). Notice that the notification period of these messages can be adjusted by the SD-RAN Controller depending on the network load. In addition, the eNB can aggregate in a single message the information of all the UEs attached to it, therefore not affecting negatively the scalability of the system. Let us notice that this long term and short term information can be used by control and management applications running on top of the SD-RAN Controller. Moreover, although not exploited in this work, the topology discovery modules may be fed with information from external sources (e.g., spectrum databases).

(iii) *RAN Slicing Management Function (RSMF)*. It is responsible for the entire slice lifecycle management, from provisioning to decommissioning. This module supports the instantiation of RAN slices whose operation is dictated by the *L3*, *L2,* and *L1 slice descriptors* defined in Section 2. In particular, Table 2 depicts the template proposed for the specification of a RSI and the configuration of the RRM policy for specifying the behaviour of both L3 through admission control and L2 through scheduling policies. Further details on the semantics of the RRM policy L3 and L2 attributes are given later on.

In addition to the RSMF, the support of RAN slicing features requires the deployment of control applications to cope with the slice-aware/multicell RRM functions necessary for the proper handling of the radio resources within and between slices. These control applications, whose functionality is explained below in a separate subsection, interact with the 5G-EmPOWER OS through a set of APIs (northbound API in Figure 3) designed with the express goal of shielding developers from the implementation details of the underlying wireless technology. In line with the implementation approach followed for the 5G-EmPOWER OS, the northbound APIs are provided as Python libraries so that the writing of new applications is facilitated. This design choice allows programmers to leverage a high-level declarative API, while being able to use any Python construct, such as threads, timers, sockets, etc.

Finally, the 5G-EmPOWER OS exposes a set of management services (e.g., slice creation, network inventory,

monitoring) to the platform administrator (e.g., operator role for the RAN), as well as to other potential users of the system (e.g., tenant role in multitenant RAN). This allows for an exploitation model where the platform administrator would own and operate both the radio infrastructure and the SD-RAN Controller, while the tenants (e.g., 3rd party service providers, verticals) would be consumers of the exposed management services (i.e., a restricted set of the management services authorized by the platform administrator), which could be integrated within their own management systems. The management services are offered through a RESTful API implemented by a Web Service module. This functionality is split into two submodules: the REST server and the front-end Graphical User Interface (GUI). The benefit of this approach is that the 5G-EmPOWER OS is not GUI dependent and any client that can consume a REST service can interact with the OS.

3.2. The 5G-EmPOWER Agent. The 5G-EmPOWER Agent is in charge of managing the LTE user plane. An eNB integrating the 5G-EmPOWER Agent within its subsystem can interact with the 5G-EmPOWER OS. The architecture of the 5G-EmPOWER Agent is composed of two parts written in C++: the platform independent 5G-EmPOWER Agent itself and the platform dependent Wrapper. The agent consists of (i) a protocol parser responsible for serializing and deserializing the OpenEmpower messages and (ii) two managers, one for single/scheduled events and one for triggered events. The types of events supported by the Agent are described in the next subsection. Finally, the Wrapper is responsible for translating OpenEmpower messages into commands for the LTE stack. Figure 3 sketches the structure of the 5G-EmPOWER Agent.

The Agent and the OpenEmpower protocol are generic and can be applied to any general eNB. However, the Wrapper must be written for a specific eNB implementation since it defines a set of operations that an eNB must support in order to be part of a 5G-EmPOWER-managed network. Such operations include, for example, getting/setting certain parameters from the LTE access stratum, triggering UE measurements reports, rising UE attach/detach events, issuing commands (e.g., perform a handover), and reconfiguring a certain access stratum policy. All these operations are invoked by the 5G-EmPOWER OS through the OpenEmpower protocol. Notice that the implementation of each of these features is responsibility of the eNB vendor and platform dependent, which makes it unrelated from the capabilities offered by the 5G-EmPOWER system.

The Wrapper is structured in as many submodules as the layers in the LTE access stratum plus an additional module for the RRC functions. Finally, it is important to highlight that the implementation of each of these submodules is responsibility of the eNB vendor and platform dependent, which makes it unrelated from the capabilities offered by the 5G-EmPOWER OS. At the time of writing, there is available a reference implementation of the 5G-EmPOWER Agent for OpenWRT-based Wi-Fi APs, for LTE small cells based on the srsLTE stack [17], and for a few commercial 4G/5G eNBs.

3.3. The OpenEmpower Protocol. As stated in the previous section, an agent is introduced at the eNB to implement the management actions defined by the OS layer. Communication between the agent and the OS layer happens over the OpenEmpower protocol. The 5G-EmPOWER OS provides a reference implementation of the OpenEmpower protocol; however implementation for other SDN platforms is also possible, e.g., ONOS or OpenDayLight. The OpenEmpower protocol allows remote management of RAN elements, while it makes no assumption about the type of RAN element, i.e., it can be used on Wi-Fi APs, LTE eNBs, or 5G gNBs.

The protocol is built on three major events or message types: single event, scheduled event, and triggered event. Their meaning is the following:

(i) *Single Events*. These are simple standalone events requested by the OS plane and notified back immediately by the agent. No additional logic is bound to such message and the OS decides the time to issue the next event. Examples include RAN element capabilities requests or handover requests.

(ii) *Scheduled Events*. These are events initiated by the OS plane and then executed periodically by the agent. Examples include the Physical Resource Block (PRB) utilization requests, which require the agent to periodically send a PRB utilization report to the OS plane.

(iii) *Triggered Events*. These events enable/disable a certain functionality at the agent. They specify a condition that, when verified, triggers a message from the agent to the 5G-EmPOWER OS. Examples include the RRC measurements requests.

All OpenEmpower protocol messages start with a common header that specifies the protocol version, the event type, the message length, the RAN element ID (e.g., eNB ID) and the cell ID, the transaction ID, and a 32-bit sequence number. The counter associated to the sequence number is independent for the connection between each eNB and the 5G-EmPOWER OS, and is incremented by one every time a message is generated by either an agent at the eNBs or the OS plane. The transaction ID is a 32-bits token associated with a certain request. Replies must use the same ID as in the request in order to facilitate pairing. This is necessary because all the communications using the OpenEmpower protocol are asynchronous.

The common header is followed by one of the three possible events headers. Each event header specifies the type of action, an operation code (opcode), and (in the case of a scheduled event) the event scheduling period. The opcode value depends on the particular type of action and can be used to indicate both error/success conditions or the type of operation (create, retrieve, update, or delete). Finally, after the event header, we can find the body of the message itself, which differs from action to action. Figure 4 sketches the structure of an OpenEmpower message.

3.4. Slice-Aware RRM Functions. The enforcement of the RRM policies defined to characterize the expected operation

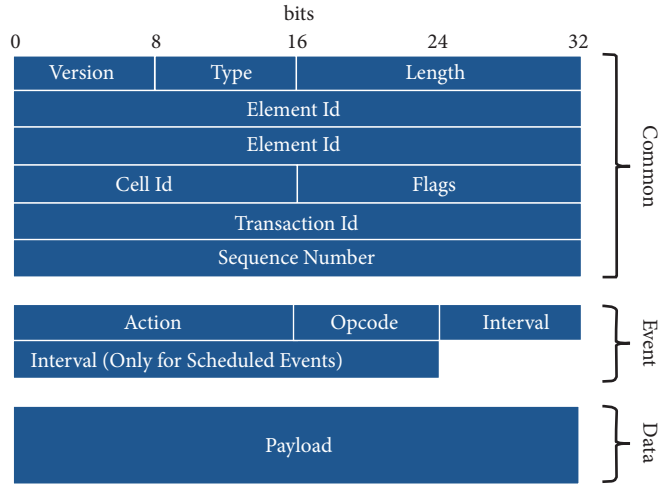


Figure 4: The OpenEmpower message structure.

of a RSI (see *RRMPolicy* attributes in Table 2) requires the implementation within the testbed of a slice-aware/multicell scope RRM functions for admission control (AC) and for scheduling.

More specifically, the AC function already supported in the eNB (srsLTE implementation) is extended in order to be compliant with the proposed descriptors and, as a whole, to oversee the capacity allocation and utilization per slice and across all the cells and decide on the acceptance or denial of the DRBs. The operation of the AC is dictated by the "Admission Control Policy" L3 attribute within the RSI template, whose detailed semantics have been presented in Table 2. As shown in the table, the capacity allocation is defined as a combination of the aggregated radio load, the aggregated bit rate and the number of established DRBs, each of them specifiable at a granularity down to particular pairs of QCI and ARP values and with the possibility of defining them per cell and/or per slice (multicell) level. In addition, the *L3 slice descriptor* also includes the number of connected UEs. All these parameters can be used to establish minimum capacity guarantees and/or capacity limitations.

The implementation of the AC function is carried out by extending the legacy RRM functions of the eNB with slice-aware AC capabilities at cell level and by a control application (*Admission Control Application* depicted in Figure 3) executed at the SD-RAN Controller with multicell AC capabilities. This split of the AC function between the eNB and the SD-RAN Controller reduces the amount of information exchanged between the eNB and the SD-RAN Controller in contrast to solutions in which the AC function could be entirely embedded in the eNB (i.e., fully distributed implementation) or entirely embedded in the SD-RAN Controller (i.e., fully centralized implementation). Notice that in a fully centralized solution with AC capabilities only at the SD-RAN Controller, much stress would be put on the eNB monitoring signalling to have a very accurate view of the radio load in each cell. On the other hand, on a fully distributed solution, with AC decisions only local to eNBs, a significant amount of information should be conveyed among eNBs in order to allow each of them to make decisions with a multicell/slice-aware scope.

One of the challenges met during the implementation of the distributed AC function has been the optimization of the message exchange delay between the centralized AC module and the eNB to support centralized decisions without causing a RRC timer expiration on the connected UEs.

On the other hand, in line with approaches such as the Network Virtualization Substrate (NVS) concept presented in [24] the scheduling function within the eNB implementation has been extended to cope with the resource allocation policies intended to regulate the distribution of the radio resources of a cell (i.e., PRBs) among the groups of UEs associated with different RAN slices. In particular, the portion of the radio resources to be assigned to a particular slice is dictated by the choice of the *L2 slice descriptor* (see Table 2), which is specified at SD-RAN Controller level and configured at the managed eNBs through the 5G-EmPOWER Agent. In this regard, the scheduling policy is defined through two attributes:

(i) *Interslice scheduling*. It defines the treatment that a slice is given with respect to the others. It is specified in terms of the algorithm used and, optionally, the percentage of resources (e.g., percentage of PRBs) allocated to each slice per TTI. The algorithm selected for interslice scheduling may be of various types. For example, when setting a classic Round Robin (RR) algorithm the slices are assigned the same portion of the available resources; however, this scheduler is not QoS-based and does not enforce any particular distribution of the resources. Conversely, by specifying a Weighted Round Robin (WRR) and the percentage parameter, it is possible to enforce the desired distribution of resources in terms of PRBs.

(ii) *Intraslice scheduling*. It defines how the UEs within a given/particular slice are scheduled. It is specified only in terms of the scheduling algorithm used in the particular slice. Different slices can be configured

with different algorithms. The sort of algorithms that can be selected currently are the common scheduling methods used in nonsliced settings, such as Round Robin (RR), Proportional Fair (PF) and max C/I [25]. These algorithms allocate (to the different UEs) the resources assigned to each slice by the SD-RAN Controller considering the QoS characteristics of the established DRBs (e.g., QCI) and the Channel Quality Information (CQI) reported by the UEs to the eNBs.

Notice that the configuration of the scheduling policy in the eNBs from the SD-RAN Controller is not only done at the commissioning phase of the RAN slice, but can also be triggered at runtime, allowing in this way a dynamic adaptation of the policy applied per cell in order to compensate potential traffic unbalances across the overall, multicell scenario. This capability is supported through the so-called *Scheduling Coordination Application* depicted in Figure 3

4. Provisioning of RAN Slices

The management of RAN slices in the testbed is structured in three phases: preparation, commissioning/decommissioning and operation. The preparation phase includes the registration of the eNBs in the SDN-RAN controller and the initial configuration setup prior to the creation of the slices in the eNBs. Commissioning and decommissioning actually refer to creation/termination of the RSIs, carrying out the necessary configurations and resource allocations over the affected eNBs. Finally, the operation includes the required actions to turn the RSI operational, i.e., serving UE's traffic. Each of the phases is described in detail in the following subsections.

4.1. RAN Slice Preparation Phase. Before provisioning RAN slices, the eNBs must be synchronized with the SD-RAN Controller. The signalling exchanged during the synchronization process is depicted in Figure 5 and is performed when an eNB running the 5G-EmPOWER Agent joins the network. To do this, the eNB must be registered at the 5G-EmPOWER OS. It should be noted that this task is carried out just once by the network administrator by introducing the eNB ID through the Web Service (GUI) in order to set it as a reliable 5G-EmPOWER-managed eNB.

After registration, the eNBs communicate their presence to the 5G-EmPOWER OS through the OpenEmpower protocol. This action is performed via the *Hello Request* message. In Figure 5 it can be seen how this request is repeated as a "heartbeat" message while the connection is maintained. Upon receiving it, the OS replies using a *Hello Response* message, and after that, it sends to the eNB a *Capabilities Request* message in order to retrieve the operational information from this specific eNB that is relevant to the controller. This information, allocated in the *Capabilities Response* message from the eNB, includes data such as the eNB ID, and the operational settings of the active cells (e.g., cell identifiers, channel numbers, channel bandwidth). Based on this, the 5G-EmPOWER OS can build and update the network-wide RAN map. In addition to this, the *Capabilities Response* also allows the eNB to inform the 5G-EmPOWER OS about whether it supports RAN slicing capabilities. Otherwise, the eNB is not a valid node to deploy RAN slices.

Finally, the OS solicits information regarding the UEs currently connected to the eNB through the *UE Report Request* message. Upon this, the eNB replies with a *UE Report Response*. This message provides the number of UEs with an active RRC connection to a cell handled by the eNB, along with the following information for each of these UEs: (1) the Cell Identifier (Cell ID), which provides a unique identifier of the specific cell to which the UE is connected; (2) the Non-Access Network (NAS) identifiers that could be extracted at the eNB, such as the International or Temporary Mobile Subscriber ID (IMSI/TMSI) and the Public Land Mobile Network Identifier (PLMN ID), which identify, respectively, the subscriber and the serving core network; (3) the Radio Network Temporary Identifier (RNTI), which is the temporary identifier allocated to the UE within a 3GPP RAN, and (4) information about the active DRBs, including the QCI and ARP. Of note is that the NAS identifiers will also include the S-NSSAI for 5G NAS signalling, though this is not currently implemented in the testbed that relies on the available 4G NAS signalling.

4.2. RAN Slice Commissioning and Decommissioning Phases. The commissioning phase refers to the process following the preparation phase to create a new RAN slice (RSI). In particular, the creation of the new RSI starts with a request through the GUI provided by the 5G-EmPOWER OS. At this point, the slice configuration can be selected. This is done by setting the desired values of the *L2* and *L3 slice descriptors* according to the format reported in Table 2. The SD-RAN Controller then verifies (and authorizes) that the selected values for the descriptors are valid and, if so, it instructs the eNB (through the 5G-EmPOWER Agent) to create the RSI through an *add slice request* message that includes the id of the slice (RSI_ID) to be created, along with the selected descriptor configuration. Upon receiving this message, the eNB registers the new slice and allocates the radio resources (i.e., resource units per slice [%]) according to the provided configuration. After these steps, the slice is ready for the activation, which is part of the operation phase described in the following section. The commissioning process is depicted in Figure 6. Finally, it has to be pointed out that since the SD-RAN experimental testbed is based on 4G technology, thus the EPC and UEs do not support slicing, a list of NAS identifiers (e.g., IMSI, TMSI) is set during the request process to be used by the eNB for the association of UEs with the corresponding RSI.

The lifecycle of a RAN network slice finishes with the decommissioning phase. At this point, the 5G-EmPOWER OS instructs the eNBs hosting the slice to release the resources allocated for serving its services. Notice that, from this moment the network slice is completely terminated and it is not available anymore. For that reason, in the case that any UE is attached to the slice, it will be automatically disconnected from that moment on. However, it should be noted that, upon a previous agreement between the network operators, these UEs could be migrated to another slice in order to maintain their services active. The communication

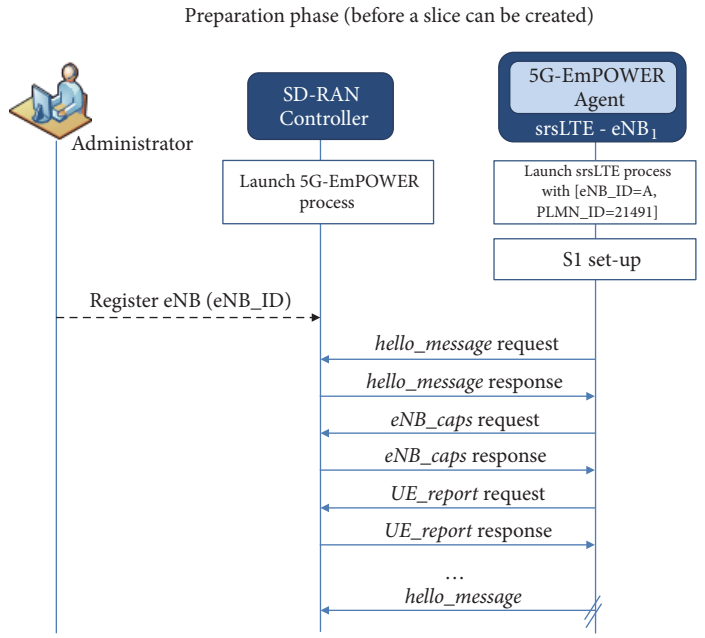


Figure 5: Preparation phase (before a RAN slice can be created).

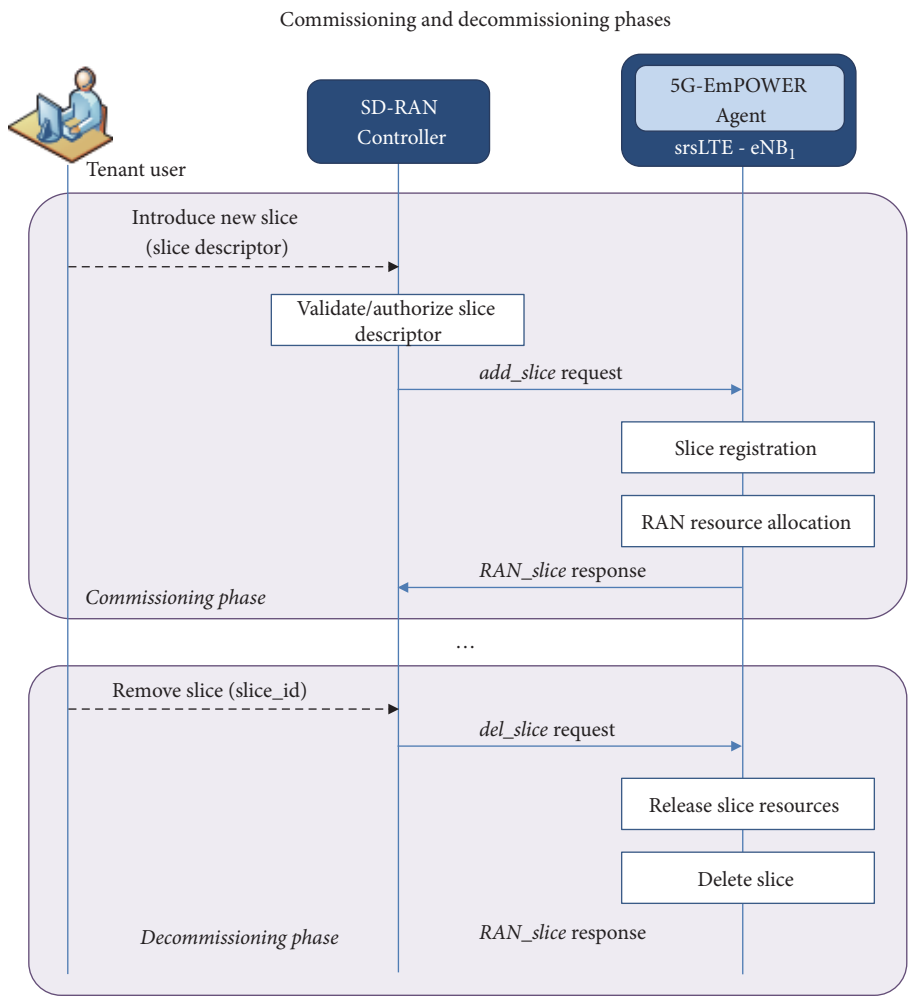


Figure 6: Commissioning and decommissioning phases for the RAN slice.

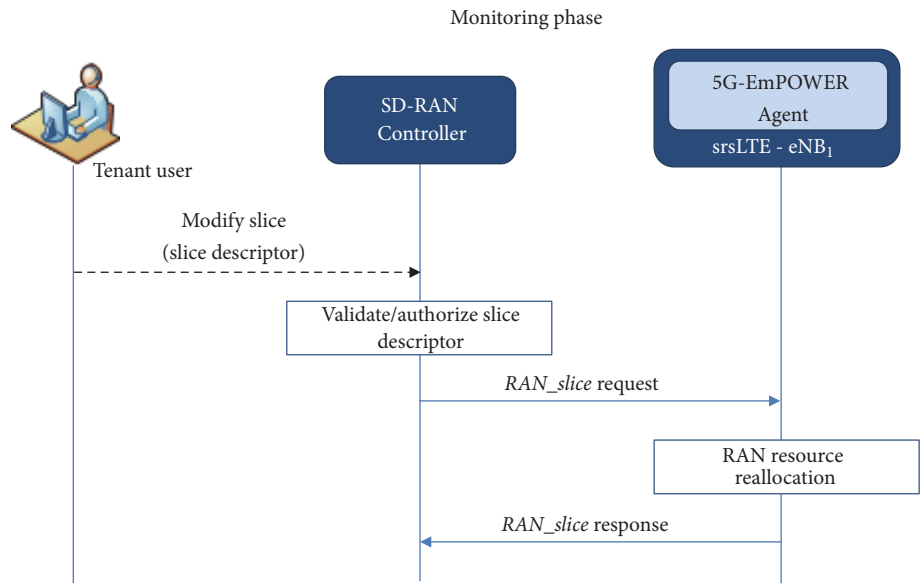


Figure 7: Monitoring step within the RAN slice operation phase.

performed between the SD-RAN Controller and the eNB for the slice decommissioning is also shown in Figure 6.

4.3. RAN Slice Operation Phase. This phase is composed of three major tasks: (i) activation; (ii) monitoring; and (iii) deactivation. The activation of the slice is carried out once the eNB has registered the slice and allocated the resources requested for such a slice. After this, the eNB (through the 5G-EmPOWER Agent) must provide the SD-RAN Controller with a complete view of the active slices through the *RAN slice response* message. Upon the confirmation from the eNB, the slice is active and can be requested by any UE. After the activation, a monitoring function in the SD-RAN Controller takes care of supervising the performance and resource utilization of the slice (e.g., Key Performance Indicators (KPI) monitoring). In this phase, modifications of the RSIs can be triggered through the GUI, introducing changes in the slice descriptors. After validation/authorization of the new configuration, the 5G-EmPOWER OS relies on the *RAN Slice Request*, a message that is sent to the eNBs providing an update in the slice, including information such as the AC policy, the UEs scheduling policy, and the percentage of resources assigned to a specific slice. Following this, the eNBs must apply the new configuration and, after the update, they must provide to the OS with an overview of the current status of the whole network through the *RAN slice response*. This message interchange is sketched in Figure 7. Finally, the operation phase also considers the deactivation of a Network Slice Instance, which could be later reactivated or decommissioned.

5. Testing of RRM Policies For Admission Control

The operation of the SD-RAN testbed after the commissioning and activation of the RAN slices is validated through a testing scenario designed to showcase the operation of the implemented slice-aware AC function. For this purpose, a RAN slice (RSI_ID=1) is provisioned with the following AC policy: *L3Descriptor.Number of DRBs[(∗,∗)][cell][min] = 1* and *L3Descriptor.Number of DRBs[(∗,∗)][slice][max] = 2,* with no differentiation per QCI and ARP. This AC policy results in a minimum capacity guarantee of one DRB per cell, regardless of the number of DRBs activated in the other cells of the slice. On top of that, if the traffic demand in one cell exceeds such minimum capacity, a maximum number of 2 DRBs is enforced for the whole set of cells serving the slice. Let us notice that these values have been chosen intentionally in order to force the UE rejection with the minimum number of devices as possible. For testing such a configuration, three commercial UEs (with test SIMs provisioned in the EPC) are switched on sequentially, get Internet connectivity through the SD-RAN testbed and start a video streaming application with downlink (DL) data rates of up to 3Mb/s. The testbed uses a channel of 10 MHz in the 2.6 GHz band, with PLMN ID=21491. During the connection of the UEs, the operation of the testbed is monitored, analysing the interactions between the testbed components and extracting specific details about the operation of the implemented AC function at both the eNB and the SD-RAN Controller.

Figures 8, 9, and 10 showcase the signalling messages exchanged between UEs, eNB, SD-RAN Controller, and EPC. For the sake of clarity, legacy 3GPP signalling is depicted with simple black lines, while the OpenEmpower protocol signalling is represented with blue bold lines. Moreover, the NAS messages, part of 3GPP signalling that are transparently transferred between UEs and EPC over RRC and S1-AP messages, are highlighted in green. All this information is collected from debugging and tracing capabilities that have been embedded within the eNB code.

As it can be seen from Figure 8, when UE#1 is switched on, a RRC connection is first established between the UE and

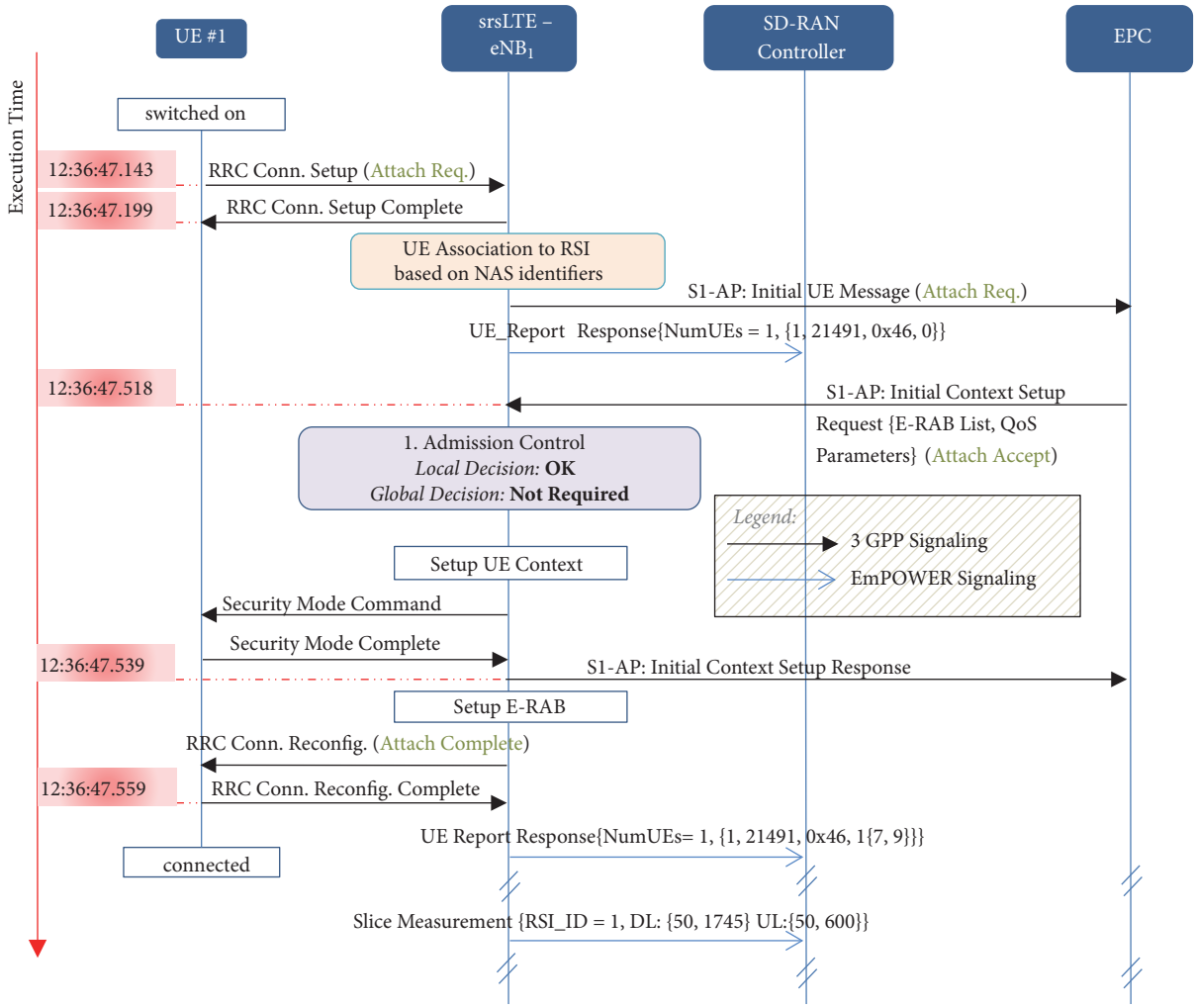


Figure 8: Admission Control Process for UE#1.

the eNB. This triggers an *Initial UE Message* that embeds an *Attach Request* message originated in UE#1 from the eNB to the EPC, as well as a *UE Report Response* message from the eNB to the SD-RAN Controller to notify about the presence of the new UE, which has been assigned the RNTI = 0x46. Note that at this point association between the UE and the RSI is solved by the eNB from the parameters (e.g., NAS identifiers) provided during the configuration of the RSI. As UE#1 is properly provisioned in the EPC database, an *Initial Context SetUp Request* is triggered to proceed with the creation of a UE context in the eNB and the establishment of the E-RAB for data plane connectivity. At this point, the AC function within the eNB is executed (see (1) in Figure 8) based on the RRMPolicy attributes (L3 slice descriptors) configured for the RSI_ID=1. In this case, since the minimum capacity guarantee is not exceeded (L3Descriptor.Number of DRBs[(∗,∗)][cell][min]=1) as no DRBs are activated yet for RSI_ID = 1, there is no need for interaction with the SD-RAN Controller. After the acceptance, the next action is to setup the UE context, activate the AS security, notify the EPC that the context has been setup successfully, internally configure the E-RAB within the eNB and finally send a *RRC Connection Reconfiguration* to UE#1 for DRB activation on the UE side. Finally, after the activation of the DRB (Setup E-RAB in Figure 8), a new *UE Report Response* is sent by the eNB to the SD-RAN Controller with information about the QoS parameters of the established DRB (i.e., QCI=7 and ARP=9 in this case). As illustrated in Figure 8, the duration of the overall process since the reception of the *RRCConnectionRequest* to the reception of the *RRCReconfigurationComplete* by the eNB takes around 416 ms. Also of note is that a *UE Report Response* message is sent by the eNB to the SD-RAN Controller each time a new RRC connection is turned up or down, while a *Slice Measurements* message, reporting the DL/UL PRB usage per slice is sent periodically. More particularly, the *Slice Measurements* message includes the RSI_ID, the DL/UL PRBs assigned to that slice, the used DL/UL PRBs during the measurement interval (i.e., 1 second) and other pieces of statistical information (not shown in the figures for clarity purposes). As depicted in Figure 8, the *Slice Measurements* message reports a downlink occupation of 1745 PRBs in one second, which corresponds to a 17.45% radio load due to the

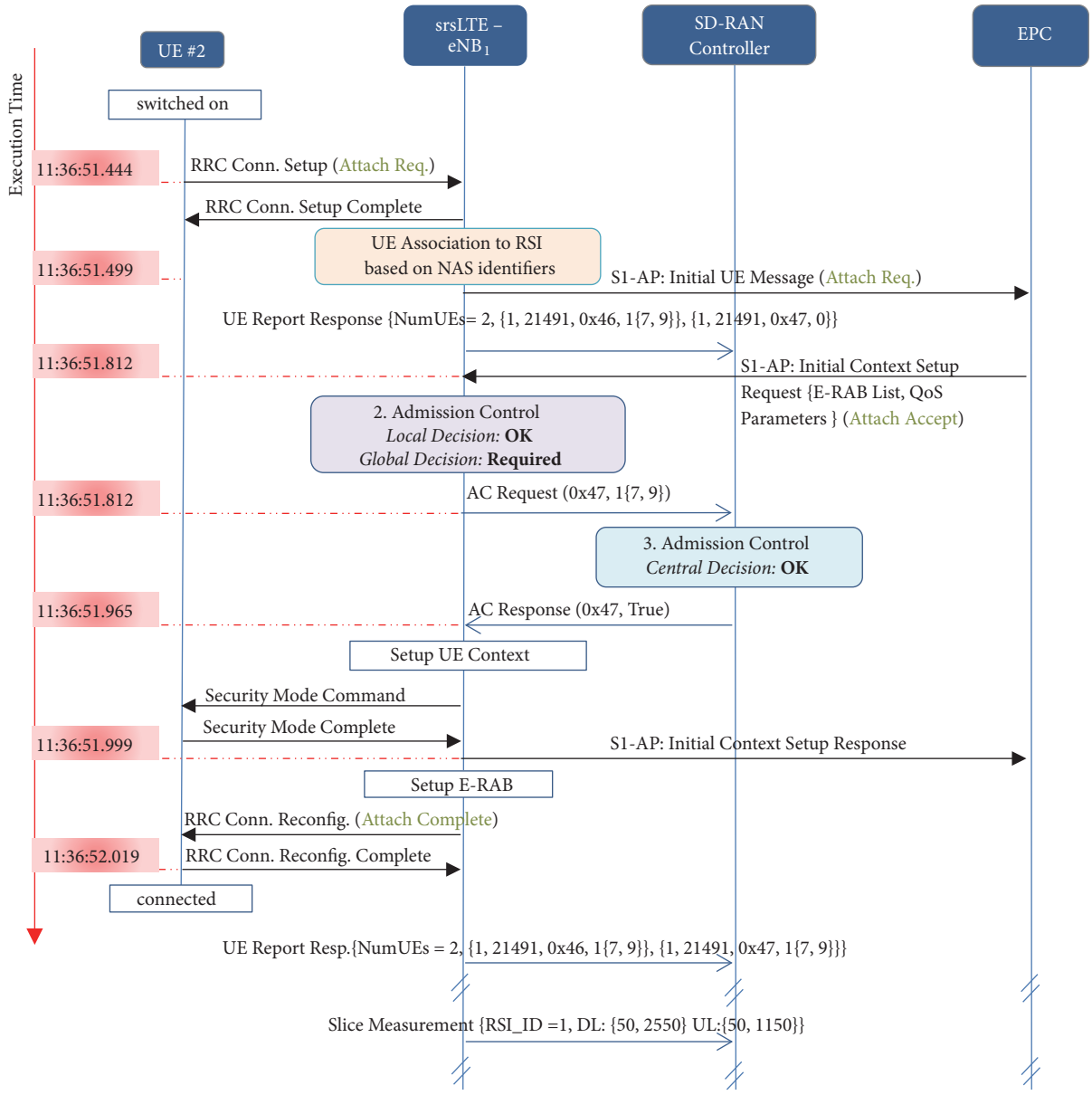


Figure 9: Admission Control Process for UE#2.

video streaming application (note that 50 PRBs are available in the 10 MHz cell).

When UE #2 is switched on, an identical process is followed till the triggering of the AC function in the eNB (see (2) in Figure 9). However, an interaction with the SD-RAN Controller is triggered in this case since the minimum capacity guarantee per cell has been reached with the previous activation of the UE#1 DRB in RSI_ID=1. Therefore, the admission decision depends now also on the total number of active DRBs within the slice (*L3Descriptor.Number of DRBs[(∗,∗)][slice][max] = 2*), which is known at controller level. For this reason, after the local decision made at the eNB, an *AC Request* message is sent to the SD-RAN Controller to trigger the centralized decision. This message includes the user RNTI (i.e., 0x47) and the details of the requested DRB (i.e., 1{7, 9} meaning 1 DRB with QCI=7 and ARP=9). Upon the reception of this message, the SD-RAN Controller checks the *L3Descriptor.Number of DRBs[(∗,∗)][slice][min] = 2* descriptor in order to accept or deny the DRB activation. Since in this case the number of active DRBs for RSI_ID=1 is equal to 1, it accepts the E-RAB establishment and sends the result (True) to the eNB with an *AC Response* message. The latter includes the user RNTI and the AC result. When the (positive) response is received by the eNB, the UE context is created, the EPC is notified, the E-RAB is established and finally the *RRC Conn Reconfiguration* is sent to the UE. In this case, the whole process takes around 575 ms due to the additional message exchange for the centralized AC control. Also of note is that now the resource utilization reported in the *Slice Measurements* message has increased to 25.5 % due to the traffic of the second user (UE#2).

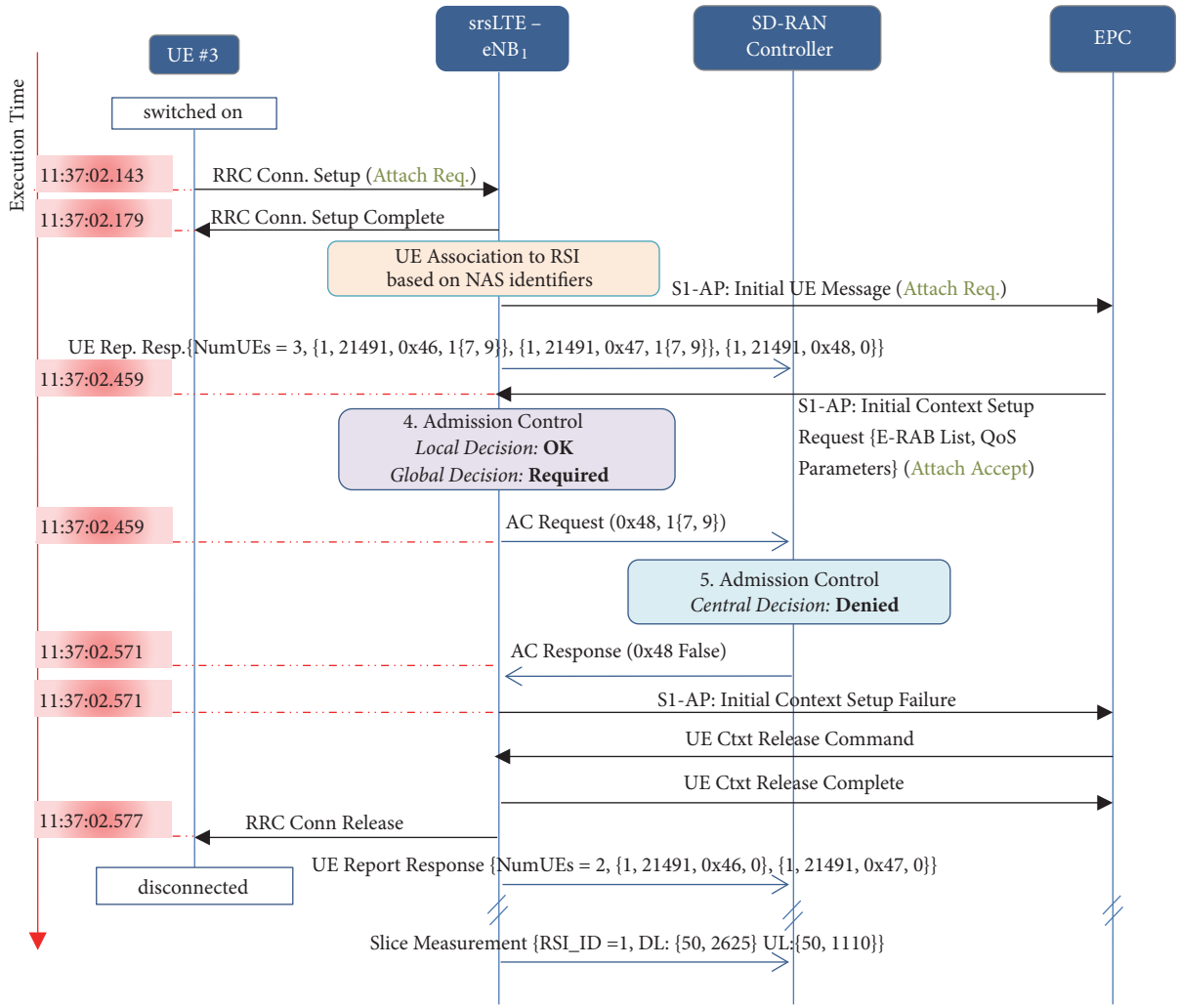


Figure 10: Admission Control Process for UE#3.

Finally, when UE #3 tries to attach to the slice, a similar process is followed (depicted in Figure 10) as for the case of UE #2, with the difference that here the connection is denied through the centralized decision since the maximum number of active DRBs per slice has been already reached. Consequently, an *Initial Context SetUp Failure* is sent to the EPC (instead of sending an *Initial Context SetUp Response*) and the process is terminated with the *Context/Connection Release* message exchange between the EPC, eNB and UE. In this last case, the execution time of the whole process is 434 ms, from which 112 ms are due to the *AC Request*/*AC Response* message exchange. Since connection of UE#3 was rejected, the resource utilization reflects the traffic of only UE#1 and UE#2.

The same test has been repeated 10 times following the same scenario setting and monitoring the message exchange delay in the eNB side. The average execution times measured for the establishment of the RRC connection and for completing the whole network registration setup, are given in Table 3, measuring separately the cases where AC decisions are made locally in the eNB and when centralized decisions in the SD-RAN Controller are also triggered (excluding the time required for the local decision). Moreover, the performance in the case of not using slicing is given as a benchmark.

As depicted in the table, similar execution times, of the order of 50 ms, are measured for the RRC connection setup in all the cases, since the slicing operation does not come with any additional processing to be carried during this procedure (differences in the average values observed between the three cases are only due to statistical fluctuations with the 10 measurements per case). On the other hand, for the whole network registration setup, it can be seen that centralized decisions lead to an increment in the execution time of the order of 200 ms. However, through the proper execution of each of the experiments it has been demonstrated that the addition of the slice-aware AC solution with distributed decision handling, presented in this work, is a feasible approach to follow and does not require any modification(s) to the standard settings of commercial UEs to cope with the extra time needed for the interaction between the eNB and the SD-RAN Controller.

Table 3: Impact on the performance of the LTE service.

	No slicing support (benchmark)		Slice-aware RAN (Local AC decisions)		Slice-aware RAN (Centralized AC decisions)	
	Avg (ms)	St. Dev. (ms)	Avg (ms)	St. Dev. (ms)	Avg (ms)	St. Dev. (ms)
RRC Connection SetUp	52	6.8	50	5.4	44	8.9
Whole network registration SetUp	410	39.1	428	20.8	619	79.2

6. Conclusions

This paper has presented a SD-RAN testbed that proves the feasibility and showcases the operation of (1) management services for the provisioning of RAN slices and (2) slice-aware/multicell scope RRM functions used to split the radio resources between RAN slices based on configurable RRM policy descriptors. The functional framework guiding the design of the testbed is in line with the latest 3GPP Release 15 specifications for network slicing management. In this respect, as a plausible realization of the *RRMPolicy* attribute included in the 3GPP information models, a template with a set of L3 and L2 descriptors has been proposed in this paper for a fine-grained specification of the RRM policy per slice at both cell and multicell levels. The testbed has been built leveraging open-source RAN distributions and the 5G-EmPOWER OS, which is the core component of the SD-RAN Controller providing the RAN slicing management functions. The main procedures and signalling exchanges supporting the preparation, commissioning, and operation phases for RAN slicing provisioning have been showcased. Moreover, the operation of the slice-aware RRM functions for admission control has been analysed in detail in a testing scenario with commercial UEs. The achieved experimental results have verified the correct operation of the proposed solution and more importantly, it has been demonstrated that the operation of the commercial UEs is not affected by the extra time needed for the interaction between the eNB and the SD-RAN Controller.

Future work is envisaged to implement RRM algorithms able to exploit the full potential of the RRM policy descriptors proposed in this paper and test their operation under more complex scenarios with multiple cells and a mix of applications demanding diverse QoS characteristics. Moreover, testing of more sophisticated RRM algorithms (e.g., [26]) can be pursued.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been supported by the EU funded H2020 5G-PPP project 5G ESSENCE under the grant agreement 761592 and by the Spanish Research Council and FEDER funds under SONAR 5G grant (ref. TEC2017-82651-R).

References

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[2] P. Rost, C. Mannweiler, D. S. Michalopoulos et al., "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[3] "System Architecture for the 5G System; Stage 2 (Release 15)," 3GPP TS 23.501 v15.3.0, September 2018.

[4] "NR; NR and NG-RAN Overall Description; Stage 2 (Release 15)," 3GPP TS 38.300 v15.3.1, October 2018.

[5] X. Costa-Perez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 27–35, 2013.

[6] S. Katti and L. Li, "RadioVisor: a slicing plane for radio access networks," in *Proceedings of the ACM 3rd workshop Hot Topics in Software Defined Networking*, pp. 237-238, Chicago, Il, USA, August 2014.

[7] A. Ksentini and N. Nikaein, "Toward enforcing network slicing on RAN: flexibility and resources abstraction," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 102–108, 2017.

[8] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Perez, "Multi-tenant radio access network slicing: statistical multiplexing of spatial loads," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3044–3058, 2017.

[9] O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti, "On radio access network slicing from a radio resource management perspective," *IEEE Wireless Communications Magazine*, vol. 24, no. 5, pp. 166–174, 2017.

[10] R. Ferrus, O. Sallent, J. Perez-Romero, and R. Agusti, "On 5G radio access network slicing: radio interface protocol features and configuration," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 2–10, 2018.

[11] A. Devlic, A. Hamidian, D. Liang, M. Eriksson, A. Consoli, and J. Lundstedt, "NESMO: Network slicing management and orchestration framework," in *Proceedings of the 2017 IEEE*

International Conference on Communications Workshops, ICC Workshops 2017, pp. 1202–1208, France, May 2017.

[12] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[13] I. da Silva, G. Mildh, A. Kaloxylos et al., "Impact of network slicing on 5G Radio Access Networks," in *Proceedings of the 2016 European Conference on Networks and Communications (EuCNC)*, pp. 153–157, Athens, Greece, June 2016.

[14] R. Ferrus, O. Sallent, J. Perez-Romero, and R. Agusti, "On the Automation of RAN Slicing Provisioning and Cell Planning in NG-RAN," in *Proceedings of the 2018 European Conference on Networks and Communications (EuCNC)*, pp. 37–42, Ljubljana, Slovenia, June 2018.

[15] "Management and Orchestration; Concepts, use cases and requirements (Release 15)," 3GPP TS 28.530 v15.0.0, 2018.

[16] "Management and Orchestration; Architecture Framework (Release 15)," 3GPP TS 28.533 v15.0.0, 2018.

[17] SRS Software Radio Systems, "srsLTE," 2012-2016, http://www.softwareradiosystems.com/tag/srslte/.

[18] OpenAirInterface, "5G Software alliance for democratising wireless innovation," 2018, http://www.openairinterface.org/.

[19] Create-Net, "5-G EmPOWER: Multi-access Edge Computing Operating System," 2018, https://5g-empower.io/.

[20] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proceedings of the 12th ACM Conference on Emerging Networking Experiments and Technologies, ACM CoNEXT 2016*, pp. 427–441, USA, December 2016.

[21] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, and X. Costa-Pérez, "OVNES: Demonstrating 5G network slicing overbooking on real deployments," in *Proceedings of the 2018 IEEE Conference on Computer Communications Workshops, INFOCOM 2018*, pp. 1-2, Honolulu, Hawaii, USA, April 2018.

[22] ETSI GS NFV-MAN 001 v1.1.1, "Network Functions Virtualization (NFV); Management and Orchestration," 2014.

[23] S. Lee, J. Park, and J. B. Lee, "NextEPC," 2018, http://nextepc.org/.

[24] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: a substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1333–1346, 2012.

[25] C. Deniz, O. G. Uyan, and V. C. Gungor, "On the performance of LTE downlink scheduling algorithms: A case study on edge throughput," *Computer Standards & Interfaces*, vol. 59, pp. 96–108, 2018.

[26] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *Proceedings of the 2017 IEEE Conference on Computer Communications, INFOCOM 2017*, USA, May 2017.

Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 8342167, 24 pages
https://doi.org/10.1155/2019/8342167

Research Article

mmWave Backhaul Testbed Configurability Using Software-Defined Networking

Ricardo Santos,[1] **Konstantin Koslowski**,[2] **Julian Daube**,[2] **Hakim Ghazzai**,[3] **Andreas Kassler**,[1] **Kei Sakaguchi**,[4] **and Thomas Haustein**[2]

[1]*Karlstad University, Karlstad, Sweden*
[2]*Fraunhofer Heinrich Hertz Institute, Berlin, Germany*
[3]*Stevens Institute of Technology, Hoboken, NJ, USA*
[4]*Tokyo Institute of Technology, Tokyo, Japan*

Correspondence should be addressed to Ricardo Santos; ricardo.santos@kau.se

Received 13 November 2018; Revised 16 February 2019; Accepted 13 March 2019; Published 8 April 2019

Guest Editor: Joan Triay

Future mobile data traffic predictions expect a significant increase in user data traffic, requiring new forms of mobile network infrastructures. Fifth generation (5G) communication standards propose the densification of small cell access base stations (BSs) in order to provide multigigabit and low latency connectivity. This densification requires a high capacity backhaul network. Using optical links to connect all the small cells is economically not feasible for large scale radio access networks where multiple BSs are deployed. A wireless backhaul formed by a mesh of millimeter-wave (mmWave) links is an attractive mobile backhaul solution, as flexible wireless (multihop) paths can be formed to interconnect all the access BSs. Moreover, a wireless backhaul allows the dynamic reconfiguration of the backhaul topology to match varying traffic demands or adaptively power on/off small cells for green backhaul operation. However, conducting and precisely controlling reconfiguration experiments over real mmWave multihop networks is a challenging task. In this paper, we develop a Software-Defined Networking (SDN) based approach to enable such a dynamic backhaul reconfiguration and use real-world mmWave equipment to setup a SDN-enabled mmWave testbed to conduct various reconfiguration experiments. In our approach, the SDN control plane is not only responsible for configuring the forwarding plane but also for the link configuration, antenna alignment, and adaptive mesh node power on/off operations. We implement the SDN-based reconfiguration operations in a testbed with four nodes, each equipped with multiple mmWave interfaces that can be mechanically steered to connect to different neighbors. We evaluate the impact of various reconfiguration operations on existing user traffic using a set of extensive testbed measurements. Moreover, we measure the impact of the channel assignment on existing traffic, showing that a setup with an optimal channel assignment between the mesh links can result in a 44% throughput increase, when compared to a suboptimal configuration.

1. Introduction

By 2021, mobile data traffic is predicted to grow to 49 exabytes per month, a sevenfold increase over 2016 [1]. With the present increase of mobile devices and respective traffic demands, the current mobile communication infrastructures will soon become resource-saturated. Consequently, fifth generation (5G) mobile networks need to provide multigigabit capacity and low latency connectivity at the access level, especially with the emergence of extremely demanding applications such as augmented reality, ultra-high definition video streaming, and self-driven automobiles [2]. To that end, multiple enablers for these requirements are proposed, including increased spectrum efficiency, network densification, and spectrum extension. Spectrum efficiency aims to improve the wireless radio resource usage, e.g., by coordinating multiple base stations (BSs) using schemes such as coordinated multipoint processing (CoMP) [3], improving modulation and coding schemes [4], or using massive multiple-input multiple-output (MIMO) techniques [5]. Network densification aims to create ultra-dense networks (UDNs) and spectrum extension explores the usage

of additional frequency bands for communication. More specifically, millimeter-wave (mmWave) band frequencies, located between 30 and 140 GHz, are promising solutions, due to the large amount of spectrum available, which can provide the necessary multigigabit capacity [6].

By fulfilling the 5G capacity requirements at the access level, the backhaul network needs to be designed so that it does not become the network bottleneck. The backhaul connects the BSs to the core network and is typically formed by fiber-cabled or point-to-point fixed microwave wireless links. However, interconnecting all the UDN small cells through fiber links is economically not feasible, which motivates the need of wireless backhaul solutions. mmWave-band links can be used to support the aggregated fronthaul traffic, as the small cells would be often located within close vicinity, forming multihop backhaul topologies [7]. However, due to network densification, a large number of small cells and backhaul links bring new challenges to the network management, due to the complexity of resource allocation problems, forwarding decisions in the backhaul, and mmWave-related connectivity problems. 5G standardization motivates split-plane HetNet architectures, where the control and data plane are split. For example, the macrocells can provide control plane connectivity while the data plane is mostly forwarding traffic through the small cells [8]. With a split-plane architecture, Software-Defined Networking (SDN) becomes an attractive solution for backhaul management. SDN is a networking paradigm where the control plane is decoupled from the data plane, logically centralizing the control plane at the SDN controller [9], which can be replicated and distributed for scalability and fault tolerance. With SDN, the controller has a global network view and can make decisions on the forwarding plane, based on the overall network knowledge.

Due to the dynamic nature of mobile communication traffic caused by diverse traffic demands during different times of the day, user mobility, and/or temporary network failures (for example, caused by long lasting obstacle blockage in mmWave links), it is important to be able to dynamically reconfigure the backhaul, in order to maintain the connectivity and adapt backhaul capacity to traffic demands. Such reconfiguration typically involves rerouting existing traffic to match new forwarding decisions and turning on more small cells to provide additional localized capacity on-demand. By adaptively powering off nodes and links that are not needed, the backhaul can also be managed in an energy efficient way. Such adaptation leads to significant changes in the topology and link configuration and ought to be seamless in order to minimize the impact on existing traffic. Consequently, a proper orchestration of the reconfiguration steps is required. From the SDN architecture perspective, these reconfiguration operations can be centrally triggered by a controller, in order to achieve different target policies defined by network operators, such as energy efficiency or capacity maximization.

While the reconfiguration of the mmWave wireless backhaul remains a fundamental aspect to consider, it is also important to build testbed environments in order to study the impact of such reconfiguration operations on real traffic. Existing testbed work is mostly focused on exploring physical layer aspects, such as beamforming in IEEE 802.11ad [10] networks [11] or propagation properties of mmWave frequencies [12]. The UE connectivity with access points (APs) using IEEE 802.11ad links has been also explored, focusing on the AP deployment [13] and lower-layer protocol tuning and its relation with higher-layer (i.e., TCP) protocols [14]. Within the wireless backhaul, several architectures consider the usage of mmWave-related technology for the backhaul links and its management to be done through SDN [15–17]. However, research on mmWave testbeds is still limited, especially when considering dynamic reconfigurable network aspects, and integrating multihop mmWave mesh topology management with SDN, in order to build future adaptive and reconfigurable SDN-based mmWave backhaul networks.

In this paper, we use the concept of SDN to develop dynamically reconfigurable mmWave mesh backhaul networks, where reconfiguration experiments can be orchestrated in a flexible way. In our approach, the SDN control plane exposes a high-level API that can be used by management applications to schedule and trigger various reconfiguration operations, which include the channel (re-)assignment of the links, the update of flow forwarding rules, the establishment of links with different neighbor nodes, the alignment of mmWave directional antennas, and powering on/off small cell backhaul nodes. We deploy an indoor testbed involving a SDN controller and four small cell multiradio mmWave mesh nodes, which is used to conduct controlled experiments to demonstrate the capabilities of our SDN-based reconfiguration orchestration platform. Based on various use cases, we orchestrate the alignment of mechanical steerable antenna elements with adaptive power reconfiguration and dynamic backhaul traffic rerouting, effectively coping with varying traffic demands. Our set of experiments is designed to evaluate the impact of various backhaul reconfiguration operations on existing user traffic. Moreover, we conduct experiments to show the impact of different channel assignment configurations in the wireless backhaul. Our results show a 44% throughput increase and 83 times lower latency for a scenario without cochannel interference, compared to when cochannel interference is present.

In summary, this paper provides the following main contributions:

(i) A comprehensive overview on wireless backhaul testbeds and related reconfiguration use cases

(ii) A detailed presentation of our designed SDN-based architecture for the wireless backhaul management

(iii) A thorough description of our developed mesh network testbed with steerable mmWave interfaces and power control units, which are configurable through a SDN control plane

(iv) A performance evaluation of the SDN-based reconfiguration of our testbed, focused on the quality of existing user traffic over different topology changes

The remainder of this paper is structured as follows. Section 2 explores the usage of SDN for managing a wireless backhaul, which is followed by our corresponding architecture, in Section 3. Section 4 describes the testbed used

to conduct our evaluation, presented in Section 5. Lastly, we present our conclusions and future work directions in Section 6.

2. Software-Defined Networking for Wireless Network Management

The centralized principles of SDN have motivated the design of new split-plane architectures, which can be beneficial for the network operation. With the centralization of the control plane, network programmability and the enforcement of global network policies become easier. Consequently, the network configuration tuning can be done by the SDN control plane, rather than individually at each network device. Considering the management of the forwarding plane, SDN allows the configuration of the managed devices with forwarding rules that match different packet header fields. The operator can then choose to install generic forwarding rules (e.g., matching traffic from a specific ingress port) or apply fine-tuned rules to distinguish specific flows (by matching their protocol source and destination port numbers, for example).

The SDN concept has been applied to different mobile network segments, mostly combined with network functions virtualization (NFV) of current mobile packet core infrastructures, e.g., long-term evolution evolved packet core (LTE EPC) [18]. Yet, the adoption of this type of architecture in the wireless backhaul has not been vastly explored. Therefore, hereby we discuss the placement of SDN functionalities for the wireless backhaul management, focusing on the reconfiguration aspects.

Wireless networks can be more complex to manage, due to additional configuration primitives that are not present within wired networks (e.g., neighbor selection, channel assignment, or transmission power configuration). Considering the transition between two different configuration states (set of active topology nodes and links, together with their configuration and traffic forwarding states), C1 and C2, while a basic forwarding rule update in wired networks only requires the specification of new forwarding rules to install, a reconfiguration within a wireless network requires the consideration of more steps. For example, it is required to firstly establish the new links from C2, and only when each link is ready, it is possible to install new forwarding rules and remove the unused links from C1. Other configuration primitives entail the assignment of channels to links, the power on/off operation of small cell mesh nodes, and the alignment of directional antennas to form links with new neighbors. In particular, when such link establishment takes a significant amount of time and no backup paths are available, the service quality for existing users can be significantly penalized. Without the coordination and proper ordering of these commands, the network can experience significant disruption of the existing traffic.

There are multiple options how an enhanced backhaul reconfiguration could be achieved:

(i) Using a distributed approach, the backhaul network would autonomously reconfigure itself based on, e.g., distributed protocols. The SDN control plane would only be responsible for the forwarding rule installation in the data plane. An example of such an approach is used in [19], which uses a distributed channel assignment approach. Its drawback, however, is that typically the routing impacts the traffic demands for a given channel, which impacts the channel assignment. In order to achieve optimized joint channel assignment and routing, additional coordination between distributed management protocols and SDN control plane would be required.

(ii) Using a centralized but legacy approach, the backhaul network would be managed by a separate network management and control plane, while the SDN control plane would be responsible for the establishment of forwarding rules. For example, in optical transport networks, a separate optical transport network control plane is connected to a centralized network management system (NMS). By using legacy protocols, the NMS is then responsible for wavelength assignment and optical path management [20], which impacts the available capacity at the routing layer. Again, suboptimal performance would be achieved if the legacy network management would perform its own decision logic regarding configuration and adaptation, independent from the SDN control plane, which would be responsible for the traffic steering. For optimal operation, a coordination between those two management systems would be required.

(iii) Using an integrated SDN-based approach, the SDN control plane would be responsible for the management of the wireless network configuration aspects, as well the establishment of forwarding rules. Such an integrated approach would enable SDN controlled traffic engineering and wireless link management. This significantly simplifies the joint optimization of traffic routing and wireless resource management (e.g., channel assignment). When using an integrated SDN-based approach, the configuration of wireless resources should be carried out by a set of distributed SDN controllers [21], for scalability and resilience reasons.

In this paper, we leverage the integrated SDN-based approach and we identify a set of important backhaul reconfiguration use cases in the next section.

2.1. Wireless Backhaul Reconfiguration Use Cases. Despite the centralized global network view, when considering a dense wireless backhaul, the reconfiguration decision making processes become difficult to compute. The high number of managed nodes and possible parameterization options increase the complexity of the calculation of new configuration states. This motivates the need for new algorithms and frameworks that optimize the backhaul operation for energy efficiency goals or other objectives. These frameworks can receive topology data as inputs from the SDN controller and apply optimization algorithms to calculate new network configuration states. As such optimization algorithms are difficult to

implement, they can be outsourced to dedicated frameworks and implemented in domain specific modeling languages (e.g., MiniZinc [22] or OPL [23]), and solvers such as Gurobi or CPLEX can be used to apply, e.g., branch-and-cut or heuristic algorithms to solve the optimization problems.

Within a dense wireless backhaul, often an UE can be within coverage of multiple small cells, alongside with one or more macrocells. Typically, the UE connects to the cell with the highest received signal strength. However, solely using this metric often leads to situations where some BSs are highly loaded while others are under-utilized, especially in hot-spot areas, where a high number of users can be located simultaneously. To avoid the multihop backhaul becoming the bottleneck, user association schemes that jointly optimize resources in fronthaul and backhaul (e.g., optimal routing and power allocation) are required [24]. Similarly, but considering the dual connectivity of the UE to both macro- and small cells, the overall backhaul throughput can be maximized by calculating an optimal traffic routing between the macro- and small cells [25].

Due to the densification of future 5G mmWave wireless backhaul networks, a high number of small cells contribute to a significant increase of the overall power consumption. Therefore, it becomes important to adaptively turn on/off small cells, in order to optimize the network for a power efficient operation, while still maintaining network connectivity. Deciding which cells should be turned on or off, combined with an efficient routing of existing traffic, becomes a difficult optimization problem to solve [26]. In addition, to minimize the backhaul energy consumption while maximizing the available capacity, the adaptive small cell powering on/off can be combined with an optimal user association [27]. Moreover, the backhaul small cells can rely on additional renewable energy sources, which can be used to optimize the overall energy consumption, alongside with adaptive powering of the backhaul small cells [28].

While it is known that mmWave links have a dynamic link margin due to random high path loss, shadowing, and blockages [29], when adding cochannel interference, forming stable mmWave links becomes even more challenging [30]. To solve such interference problems, interference coordination between the base stations is required. However, interference coordination on a per packet basis is difficult to achieve due to strict timing requirements, and is typically not available in off-the-shelf mmWave hardware (e.g., IEEE 802.11ad). Alternatively, interference coordination can be achieved through channel assignment schemes, which assign different frequencies to different links for a longer duration. As the optimal channel assignment depends on the traffic matrix, changes in traffic typically require a channel reassignment.

2.2. Software-Defined Wireless Architectures and Testbeds. The usage of SDN-based architectures has been thoroughly proposed for the management of wireless backhaul networks. Specific to SDN-based small cell backhaul networks, aspects such as out-of-band control channel, energy efficiency, dynamic optimization policies, resilient forwarding, and flexible path computation strategies should be taken into consideration [17]. The 5G-Crosshaul architecture [31] introduces a control infrastructure that is responsible for the management of heterogeneous fronthaul and backhaul networks, based on SDN/NFV principles, where different virtual network functions are managed according to an active orchestration of network resources. As mmWave mesh networks can also be formed in different environments, the authors of [32] proposed to use SDN to manage a network formed by unmanned aerial vehicles that are interconnected through IEEE 802.11ad/802.11ac links.

Recently, the application of SDN-based concepts has been studied in wireless testbeds. For example, in [33] it is showed that using centralized SDN-based forwarding reconfiguration can reduce the network disruption, compared to distributed routing protocols. The centralized architecture allows the SDN controller to quickly detect failures and react to it, while distributed routing protocols have higher response times due to the inherent convergence times. By using a SDN-based mesh testbed, authors in [34] show the benefits of a controller-triggered network reconfiguration, addressing interference-aware routing and flow load balancing scenarios. A SDN-empowered wireless small cell backhaul testbed is presented with WiseHAUL [35], featuring nine nodes that can have their forwarding tables configured by a controller. Similarly, the impact of OpenFlow-based resilience in mmWave mesh backhaul networks has been studied, showing how the backhaul can benefit from fast-failover resiliency and SDN-based reconfiguration, to avoid network disruption, caused by temporary failures [36]. Using the NITOS testbed, authors in [37] deployed a hybrid backhaul composed by mmWave and sub 6 GHz network interfaces, where the SDN controller can obtain link quality metrics related to the mmWave interfaces through OpenFlow protocol extensions. The framework of [38] uses an outdoor deployment of a mmWave mesh network with mmWave links spanning up to 185 meters to conduct network performance measurements in Berlin. Within the 5G-XHaul project, a city-wide SDN testbed with mmWave wireless backhaul links was deployed in Bristol [39], featuring a demonstration where different network slices are routed through paths formed by heterogeneous links. Additionally, Facebook's Terragraph [40] proposed a commercial solution for an mmWave-based backhaul, as an alternative to legacy copper and fiber networks.

Although different architectural solutions are proposed to use SDN for the management of wireless networks, dynamic and complex reconfiguration aspects of such networks have not been thoroughly considered. Therefore, additional experimental work using real-world SDN-based wireless testbeds, together with proper performance evaluation of key reconfiguration aspects, is necessary, in order to understand and quantify the benefits of SDN for the dynamic backhaul reconfiguration.

3. Architectural Considerations

The centralized aspects of software-defined wireless networks make such paradigm attractive for managing a wireless backhaul. However, it is important to address configurability aspects, if the backhaul configuration needs to be changed

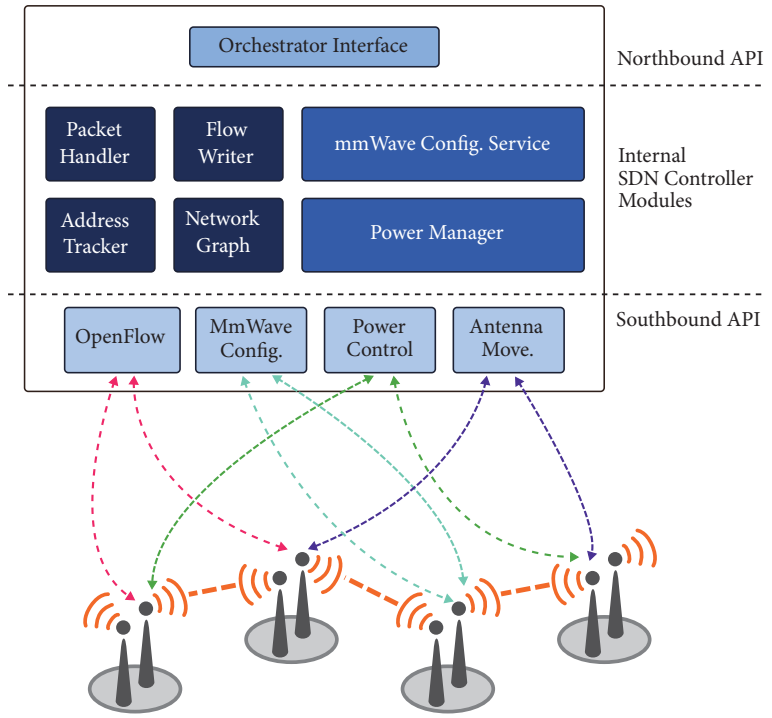


Figure 1: SOCRA architecture.

over time. With that, our goal is to provide an architecture that can orchestrate the overall wireless backhaul reconfiguration, taking advantage of the dynamic configuration aspects of wireless networks. Therefore, we present the SOCRA (Software-Defined Small Cell Radio Access Network) architecture, with its main functionalities:

(i) Provide a split-plane network design, using an out-of-band control channel

(ii) Enable the change of backhaul forwarding configurations, by specifying different routes, which can be individually tuned to match single flows

(iii) Adaptively power on/off the mesh nodes, in order to reduce the overall backhaul power consumption

(iv) Dynamically configure the wireless backhaul links and related configuration parameters, e.g., channel assignment and beam alignment

(v) Provide a high-level orchestration API to network operators, allowing them to modify the network configuration (e.g., align two mmWave interfaces and establish a link between them)

We consider an architecture where a SDN control plane is responsible for the management of a wireless mesh backhaul. The backhaul is a HetNet formed by multiple small cell nodes that are interconnected between mmWave wireless links, as depicted in Figure 1. The small cells can provide localized coverage to existing UEs with high capacity and forward the access-level traffic through the multihop mmWave links towards the core network. In addition, the small cells are located under the umbrella of macrocells (e.g., LTE eNodeBs), which can provide an out-of-band control channel, as well as a backup data plane to the backhaul. The SDN control plane manages the network by dynamically configuring the mesh nodes forwarding rules, wireless links and topology formation, and power configurations (power on/off nodes and interfaces).

To enable outdoor links that can span a large distance (e.g., up to 200 m), high antenna gain is required, especially for mmWave links. This is due to the high path loss at mmWave frequency bands, and additional oxygen attenuation [41]. Furthermore, for backhaul networks, a coverage of 360° is necessary. The high gain can be achieved either using large antenna arrays with at least 8x8 antenna elements, or using regular arrays and lenses [42]. To achieve full 360° coverage with digital beamforming, multiple radio units can be combined (e.g., four radio units, each covering 90°). In order to enable connections with multiple neighbors within the same sector, a radio unit can use point-to-multipoint (P2MP) connectivity [39]. While this increases the overall system flexibility, it also increases its complexity, as those nodes have to share the available bandwidth from a single mmWave interface through beam switching techniques, which has an impact on the total achievable throughput per node. For a mesh topology where small cell nodes need to receive packets and forward them to another neighbor towards the destination, this causes the reduction of capacity on all interconnected links.

When using reflect arrays or lenses to achieve high link gains, high gain beams are used on the transceiver and are focused at a single focal point, creating narrow "pencil" beams [43]. However, this requires a mechanical alignment of the

antennas and reflect arrays to form point-to-point (P2P) links, with the cost of eliminating P2MP capabilities, as they require a constant beam realignment between the connected nodes. On the other hand, each radio unit can create a full 360° coverage by rotating the antennas and reflect arrays. This allows all established links to be dedicated between two nodes, offering full link capacity, independent of neighboring nodes. The available link capacities can then be used as input parameters for optimization frameworks to calculate optimal solutions for traffic aware mesh backhaul reconfiguration. Additionally, resilience is provided, as multiple mmWave interfaces can be positioned in the same sector. Nonetheless, a mechanical alignment requires the SDN control plane to calculate the necessary angles between the backhaul nodes and align the radio units, before establishing the links. The alignment is time costly (i.e., order of seconds) and depends on the angle and rotational speeds of the mechanical elements. Yet, once all links are established, the backhaul topology is rather static and changes are only necessary due to e.g., permanent link blocking, hardware failures, or significant changes in traffic demands, which happen typically in the order of minutes or hours. As targeted in our design, load balancing and recovery of link failure can be handled on higher layers, using fast-failover among different neighbor nodes [36], leading to fast rerouting within the mesh topology.

3.1. SOCRA SDN Controller. The SOCRA SDN controller architecture is depicted in the top part of Figure 1 and is designed to enable the configuration of the wireless backhaul. Internally, it contains core SDN controller modules, which include the Packet Handler, Address Tracker, Flow Writer, and Network Graph. For scalability and resiliency, the controller ideally needs to be distributed using, e.g., a microservices architecture or other scalable multicontroller architectural principles. However, in this paper, for simplicity and testbed purposes, we limit the discussion to a single controller.

The Address Tracker keeps track of the locations of network hosts (e.g., UEs), and when they were lastly observed. The Network Graph maintains the backhaul topology, providing an internal interface to compute paths between backhaul nodes. The Packet Handler receives incoming packets sent to the controller by managed devices. When it receives a packet, if the destination can be resolved by the Address Tracker, it calculates a path between the source and destination nodes through the Network Graph and installs the corresponding forwarding rules using the Flow Writer.

In addition, the controller features modules specific to the wireless backhaul configuration. More specifically, it includes an mmWave Configuration Service (MCS) and a Power Manager module. The MCS handles the configuration of backhaul mmWave links, managing the creation/modification of backhaul links and respective beam/antenna alignment configuration. The Power Manager is responsible for managing and monitoring the backhaul nodes' power states, powering on/off backhaul nodes and accessing their respective power consumption.

To allow the communication between network management and orchestration applications, the controller exposes a set of REST Northbound APIs with its Orchestrator Interface. The provided APIs feature high-level configuration commands that can be used to configure the managed backhaul network, which are detailed in Section 3.3. The communication between the controller and the mesh nodes is performed by Southbound APIs, allowing the management of forwarding tables, mmWave link configuration and power management.

3.2. Small Cell Mesh Node. As part of the proposed architecture, the backhaul network is formed by multiple small cell mesh nodes. We present an overview of a small cell mesh node in Figure 2. Each node is composed of two main components, the PCU and the computation device, respectively.

The PCU is used to manage the power supply of the computation device, providing power consumption information and adaptively powering on/off the mesh node's computation device. The computation device is connected to a flexible number of mmWave radio interfaces and respective movement controllers. When using mechanical rotating antennas, the movement controllers are responsible for rotating and aligning the mmWave interfaces, according to the requested positioning. This positioning information contains the azimuth and elevation angles that the mmWave interface should transition to, within the supported range values (0° to 359.9° for azimuth and −15° to 15° for elevation), assuming that all the interfaces were initially calibrated to have the same alignment in their 0° positions, for both coordinates.

Internally, the mesh nodes contain different modules that handle the multiple types of configuration requests. The forwarding plane maintains the mesh node's forwarding tables and is responsible for processing incoming and outgoing packets from the mmWave interfaces, interacting with the mmWave driver and the operating system's network stack. The link configuration module is responsible for the configuration of the mmWave interfaces, setting up new links and handling the parameterization of existing links, e.g., channel assignment or transmission power, by interacting with the mmWave driver. The interface movement module communicates with the movement controller, configuring the mesh node antennas' positioning and alignment. Finally, the Power Management module is responsible for gracefully shutting down the computation device.

3.3. Management and Orchestration. As previously mentioned in Section 3.1, the Orchestrator Interface of the SOCRA SDN controller can receive configuration requests from management applications. These configurations allow the managed backhaul network to be configured, from a high-level perspective. Then, internally, the controller translates the received messages into the respective low-level configuration commands, which directly interact with the managed network devices. An overview of the provided RESTful API is presented in Table 1. The API can be divided into mmWave link, power, and forwarding rule management

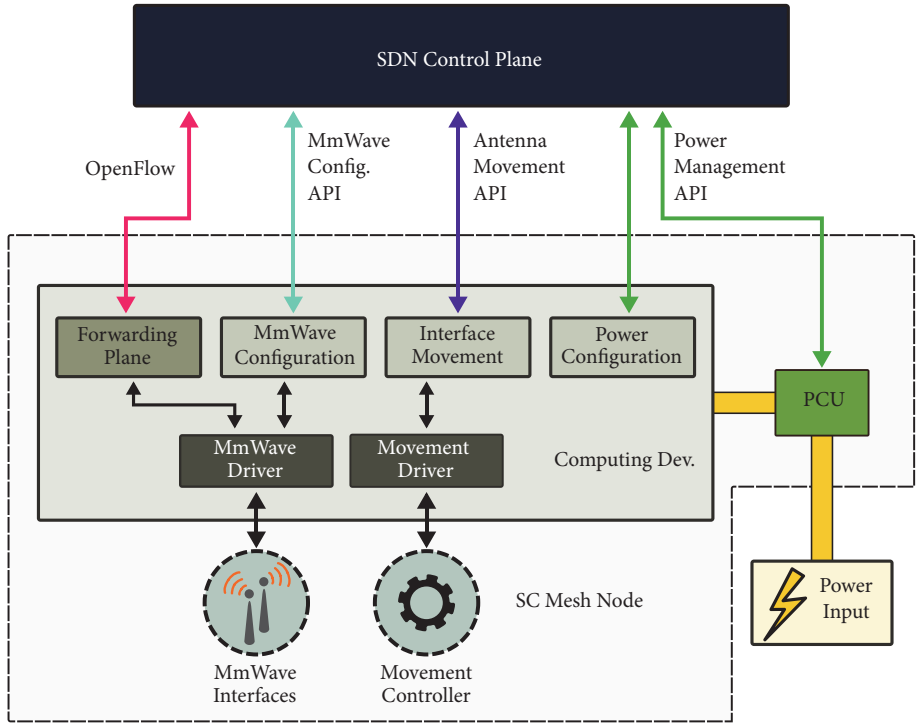


Figure 2: Small cell mesh node internal architecture.

Table 1: SDN controller orchestration REST API.

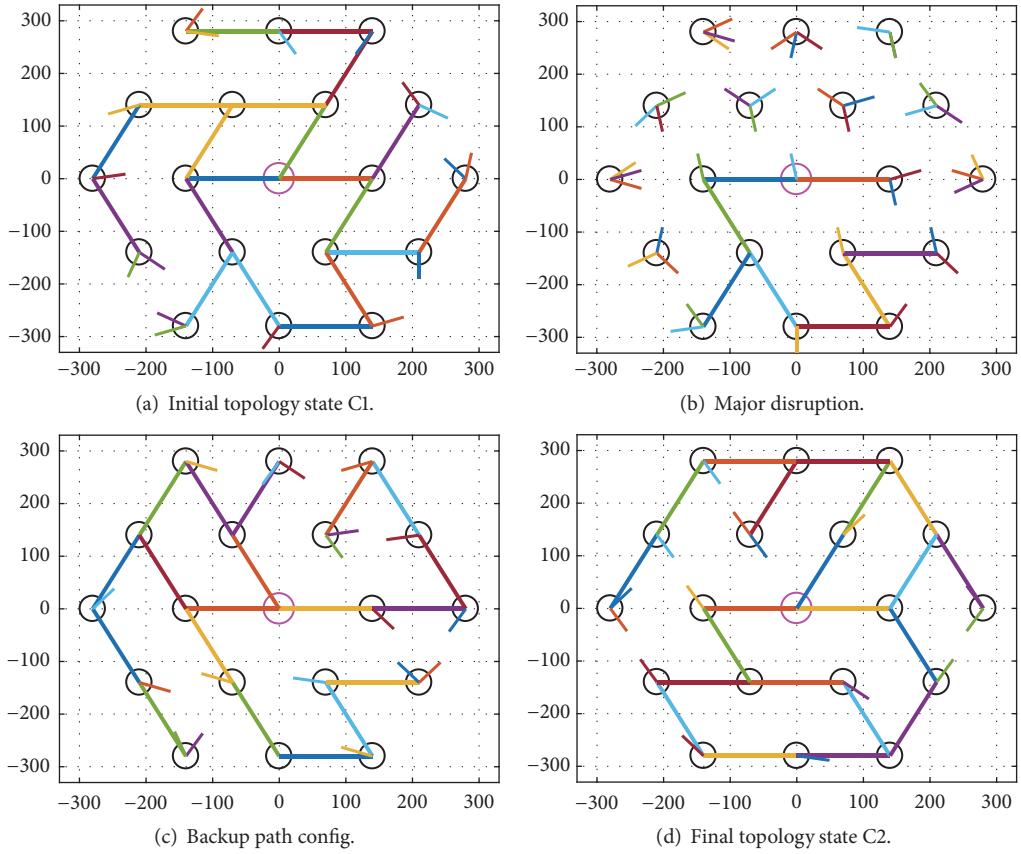
Command	Description
`mmwaveLinkConfig`	Requests the configuration of a link between two mmWave interfaces from two mesh nodes. The alignment details can also be provided as input.
`disableMmwaveLink`	Disables an mmWave interface from a mesh node.
`addFlows`	Adds forwarding rules between a source and destination node. The used path is internally calculated by the SDN controller.
`addPathFlows`	Adds forwarding rules between a source and destination node, specifying the used forwarding path.
`removeFlows`	Removes all forwarding rules from a node.
`powerOnNode`	Turns on a node power supply.
`powerOffNode`	Shuts down a backhaul node.

commands. The mmWave link commands allow the request of link configurations by specifying the related interfaces, link parameterization (e.g., used channel) and, optionally, the alignment values. The power configuration request messages abstract the power management operations, while the forwarding rule management messages provide an interface for specifying desired backhaul traffic forwarding rules.

Using these APIs, the SDN control plane can be coordinated by management applications, which can then configure the backhaul based on different use cases. We now present two scenarios where the proposed architecture can be used to reconfigure the mmWave mesh backhaul.

3.3.1. Use Case I: Optimal Steerable mmWave Mesh Backhaul Reconfiguration. In this use case, the orchestrator API of the SDN controller is used to adjust the backhaul topology, in order to cope with, e.g., changes in traffic demands or permanent node or link failures. Given a currently deployed topology state C1 (Figure 3(a)), the API can be used to orchestrate the necessary steps in order to implement a given new topology and link configuration state C2, where additional backhaul nodes are powered on, the topology is changed to form new links and forwarding rules are adjusted accordingly (Figure 3(d)).

Consequently, when a new link with a different neighbor should be established, the mmWave beam alignment is required to change. If the antennas need to be mechanically aligned, this operation is not immediate and, due to the directionality of the used transceivers, it is not possible to establish a new link until the interfaces are nearly aligned. Therefore, when a network interface that serves traffic needs to be realigned, the existing traffic needs to be rerouted via a different set of links/nodes, before the configuration operation starts, in order to avoid the disruption of ongoing



(a) Initial topology state C1.

(b) Major disruption.

(c) Backup path config.

(d) Final topology state C2.

Figure 3: Topology reconfiguration states, from an initial to a final setup, where the wireless interfaces need to be aligned.

network service. The ordering of this reconfiguration plays an important role, as it is desired to avoid the disruption of the backhaul operation (Figure 3(b)), while creating alternative backup paths, which allows existing traffic to not be affected by ongoing topology changes (Figure 3(c)).

To solve this wireless mesh backhaul reconfiguration problem, we previously developed a Mixed Integer Linear Program (MILP) based framework that computes the optimal steps of antenna realignments, link establishments and flow routing operations that needs to be done, when transition between two topology configurations C1 and C2 [44]. The system model builds a wireless backhaul topology with directional network interfaces that can be realigned over time. The constraints include maximum link capacity, possible links that could be formed, flow conservation and interface movement speed and alignment. The decision variables define the links, interface position and movement status (moving clock or counterclockwise), flow rates, and packet loss. The inputs contain the backhaul nodes, their respective positions, possible links and maximum capacity, necessary interface alignment angles, Internet-connected gateway nodes, and initial and final traffic demand matrices. The optimization goal is to minimize the total packet loss during the transition between the C1 and C2 topology configuration states. The total packet loss is quantified by the sum of all the backhaul nodes' packet loss over the total reconfiguration time.

The framework aims to reduce the packet loss by establishing backup links between unused network interfaces, before the ones used in the final topology configuration begin their alignment. Therefore, by providing a higher number of network interfaces per node, the total packet loss reduction can be improved. In addition, by providing additional reconfiguration time slots, it is possible to establish a higher number of backup links, which also contribute to the overall packet loss reduction.

In the evaluation section, we validate the main configuration primitives that our framework offers in order to implement the given use case. These primitives include (a) rotating the small cell network interfaces to adaptively change the backhaul topology, (b) dynamic backhaul mmWave link establishment, and (c) backhaul traffic forwarding. The configuration messages with these primitives can be exchanged between the framework and the SDN controller Northbound REST API, through JSON formatted messages. The topology data (e.g., node positioning and the set of possible links between the mesh nodes) can be provided by the SDN controller as input, which can be obtained from the controller's network graph. The computed framework solutions can be interpreted into specific configuration instructions that are sent to the SDN controller. For example, the interface movement variable values can be translated into single instructions with the involved interface and the destination alignment coordinates. Additionally, the link

status variable can be translated into link configuration and forwarding rule installation messages, whenever there is a change in the respective variable values.

3.3.2. Use Case II: On-Demand Powered Mesh Backhaul. Future small cell backhaul networks will often be formed by a dense mesh deployment of small cell nodes, generally in close vicinity, and interconnected by mmWave wireless links. Having all small cell nodes always powered on will lead to high operational costs due to their power consumption. For green backhaul operation, the backhaul nodes which do not serve any user traffic should be adaptively powered off, which changes the backhaul topology. For example, authors in [24] jointly solve the optimal UE association, traffic flow routing, power allocation and switch/off operation in order to minimize both access and backhaul power consumption.

In this work, we consider [26], which computes the overall topology on/off state, minimizing the total power consumption, while serving the required user traffic demands. The LTE macrocell provides coverage to the UEs and serves as mmWave gateway that can connect the surrounding small cells to the core network. The small cells can provide multigigabit capacity, as long as they can be connected to the macrocell through multihop paths. Therefore, the algorithm can activate small cells for relaying, even if they do not serve any user traffic. Additionally, the UE traffic can be multiplexed among different paths, splitting the required demand.

The constraints ensure that the traffic demand can be served, the maximum link capacity is not violated, and the traffic forwarding is only done on active APs. The decision variables define the traffic demand of each AP, the associated users with each AP, the amount of traffic sent on each link, for each flow, and the on/off state of the APs. The algorithm execution has three steps that (1) perform an initial on/off selection of the backhaul nodes, (2) create the necessary mmWave links, and (3) activate relay nodes. The algorithm computes new backhaul topologies and routing paths when UE traffic demands change. In addition, it reduces the backhaul power consumption by offloading the UE traffic through the LTE macrocell, therefore taking advantage of such HetNet architecture.

Overall, authors in [26] focus on the calculation of a new backhaul configuration state C2, given C1 and assuming a change of traffic demands. On the other hand, the use case presented in Section 3.3.1 is based on the reconfiguration between C1 and C2. Yet, the application of both use cases can be combined in order to (a) calculate a new backhaul topology and forwarding configuration C2, based on energy efficiency requirements, and (b) compute the necessary backhaul reconfiguration operations, to transition between the initial C1 and final C2 configuration states.

In the evaluation section, we therefore focus on seamless reconfiguration operations that involve multiple on/off operations in the backhaul nodes, the establishment of new links, and the update of the forwarding rules. Similarly as previously described, the SDN controller can request a new backhaul configuration state, by providing the topology information and existing traffic demands as input to the algorithm.

4. mmWave Mesh Backhaul Testbed

In order to evaluate our architecture and reconfiguration actions, we deploy a small cell wireless mesh backhaul testbed using mmWave links, which is integrated with our SDN controller and the small cell mesh node extensions (see Figure 4). The testbed is composed of a SDN controller, four small cell nodes (N1 to N4, respectively) and three nodes responsible for the traffic generation, i.e., Sender (S), Receiver 1 (R1), and Receiver 2 (R2). The backhaul mesh nodes are connected to an internal control network through a WiFi link and the receiver and sender nodes are connected to the same control network through an Ethernet link. The mesh nodes use the WiFi internal network for the SDN control channel; however our architecture supports the usage of other link types (e.g., LTE, or WiMAX) as control channel [45]. The small cell nodes are interconnected through four mmWave links (N1-N2, N1-N3, N2-N4, and N3-N4).

The testbed mesh nodes are positioned indoor, and the respective positioning layout is illustrated in Figure 5. The mmWave interfaces of each node are stacked on top of each other and mounted on tripods, with exception of N4, which has its interfaces also stacked, but attached to an existing platform in the lab room. As seen in Figure 5, the link distances between the mmWave interface pairs vary between 2.4 m and 3.8 m.

We use three separate machines to generate traffic (R1, R2, and S), which are connected to N4, N2, and N1, respectively, via a 10 gigabit Ethernet link.

We use the following three classes of commercial off-the-shelf minicomputers as computation devices that can support the necessary computation power to handle the high-throughput forwarding rates:

(i) *Class 1:* Intel NUC Kit D54250WYK
Intel i5 4250U, 16 GB RAM, SSD

(ii) *Class 2:* Intel NUC Kit NUC7i7BNH
Intel Core i7 7567U, 16 GB RAM, SSD

(iii) *Class 3:* Gigabyte BRIX GB-BKi7HA-7500
Intel Core i7 7500U, 16 GB RAM, SSD

In our experimental setup, the nodes N1, N3, and N4 are class 2 devices, and node N2 is a class 1 device. The SDN controller operates on a dedicated class 1 machine and is connected to the internal wired and wireless control networks. Both R1 and R2 are class 3 devices and S is a class 1 device. All testbed nodes use Ubuntu 16.04 with kernel 4.15.0-34. To orchestrate experiments and test the SDN-based reconfiguration capabilities of our testbed, we use an additional laptop connected to the control network. This laptop uses a REST client application to communicate with the SDN controller, issuing commands presented in Section 3.3. Additionally, N2 and N3 are equipped with TP-Link HS110 Smart Plug PSUs, used to dynamically switch them on or off, also connected to the internal Wi-Fi control network.

The power consumption of the used mesh nodes is listed in Table 2. The measurements were conducted under

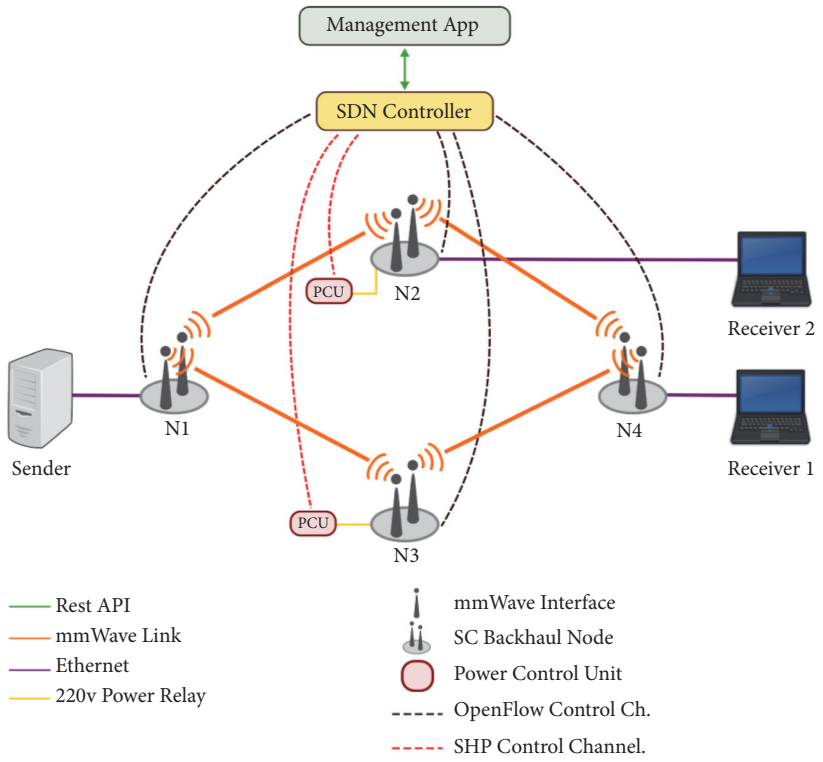


Figure 4: SDN-based mmWave mesh backhaul testbed.

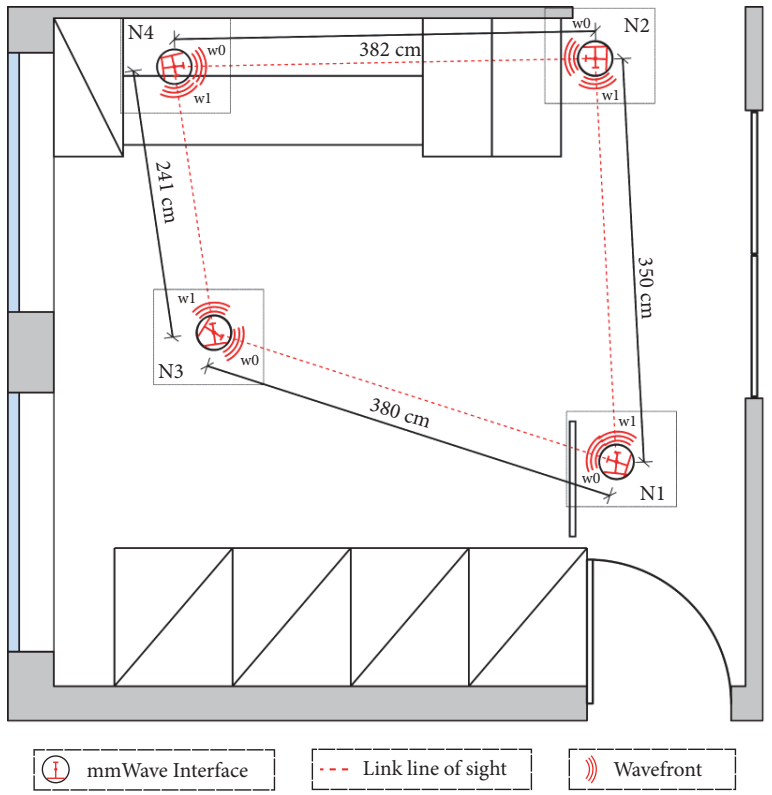


Figure 5: Testbed node positioning map.

Table 2: Power consumption measurements of mmWave small cell nodes.

Power state	Class 1	Class 2
Idle without mmWave	4.03 (± 0.22) W	8.01 (± 0.07) W
Idle with mmWave	7.42 (± 0.62) W	11.02 (± 0.13) W
Load without mmWave	17.74 (± 0.60) W	41.70 (± 0.79) W
Load with mmWave	20.12 (± 0.35) W	43.99 (± 1.08) W

Table 3: OpenFlow message extensions.

Message type	Description
`ofp_mmwave_config`	Requests the configuration of an mmWave link, which can include the used channel, MCS, beam alignment, SSID, security options, or power state.
`ofp_sc_features`	Provides initial information to the controller with the SC positioning, scanned neighborhood, and the respective PCU management IP.
`ofp_mmwave_stats`	Statistics periodically sent to the controller, with RSSI measurements over the connected small cell links.

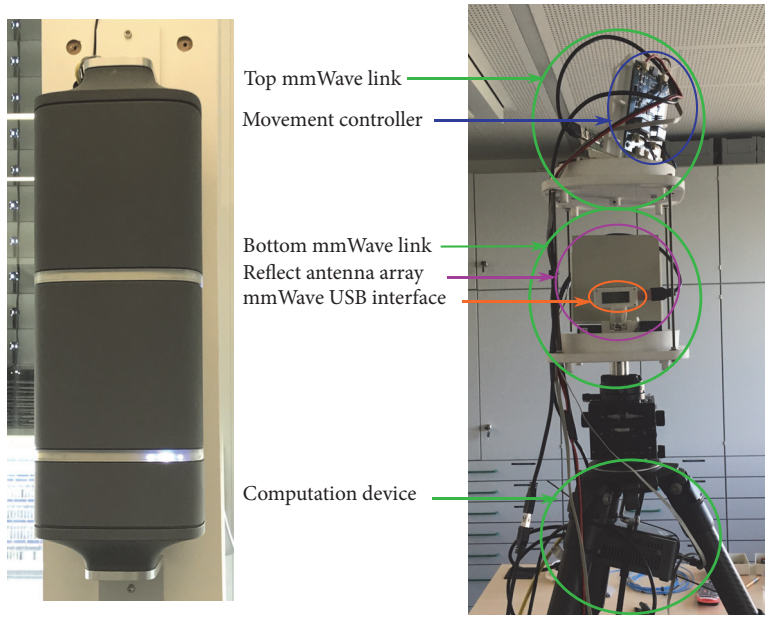
different computing requirements (idle and under high CPU load, with and without the mmWave interfaces connected to the node's USB ports), collecting the reported power consumption each second, during one hour, for every measured state. We measure the power consumption using a GUDE 8001 Power Distribution Unit, which has built-in power monitoring functionalities. In the idle power state, the computation device has the used software running, but without any additional computing processes running. To perform the CPU-loaded measurements, we use the `stress-ng` (http://kernel.ubuntu.com/~cking/stress-ng/) to maximize the device's CPU utilization. While it is observable that the power consumption is significantly higher with a high CPU load, if the backhaul topology is formed by thousands of small cell nodes, the aggregated power consumption of the idle nodes could reach substantial values, motivating the need of energy efficient topology management policies.

4.1. SDN Controller. We use a single SDN controller for the testbed to implement the SDN control plane (see Section 3.1) extending the OpenDaylight (ODL) L2Switch project (https://github.com/opendaylight/l2switch), which has a Packet Handler, Address Tracker, Flow Writer, and Network Graph components implementation.

The MCS translates high-level mmWave link configuration commands into individual device configuration messages, implemented through OpenFlow extensions (see Section 4.1.1). The configuration of a link between two nodes is done by setting the first node as access point (AP), and the second as station (STA). Based on the nodes' internal interface identifiers, an unique SSID for that link is created (e.g., `"of:4:1-of:3:2"`, if the new link is between interface 1 of node 4 and interface 2 of node 3). In addition, the involved interfaces' positioning parameters can also be provided as inputs.

The Power Manager interacts with the mesh node's PCU using an implementation of the TP-Link Smart Home Protocol southbound API (https://github.com/intrbiz/hs110). Additionally, the Power Manager sends shutdown requests to the computation device with the extended OpenFlow API. A high-level node power off command is internally orchestrated by (1) sending a graceful shutdown request to the computation device, and (2) sending a PCU power off request 5 seconds after (so the power supply is cut after the computation device's operating system is no longer running). Similarly, the Power Manager boots up the computation devices by turning the respective PCU's power on, as they are configured to wake on power.

4.1.1. OpenFlow Extensions. To increase the wireless backhaul configurability, we introduce new message types to the OpenFlow protocol. This approach has been previously considered also in, e.g., [37, 45, 46]. The extensions are implemented by extending ODL's OpenFlow Plugin, which is responsible for the serialization and deserialization of OpenFlow protocol messages. An abstracted list of the new OpenFlow messages can be found in Table 3. The `ofp_mmwave_config` message is sent by the SDN controller to address the small cell configuration, regarding (1) mmWave link configuration, (2) mmWave interface alignment, and (3) small cell power state (on/off). Whenever a new node connects to the SDN controller, the controller requests initial status data from the nodes, through a features request message. If the new node is a small cell mesh node, it includes that information in the features reply, then sending a `ofp_sc_features` message (requested by the controller), which contains information about its GPS coordinates, PCU IP address used for manage its power configuration and a list of scanned neighbors (BSSID, RSSI, and respective channel). If the node does not send a `ofp_sc_features` message, it is treated as a regular OpenFlow device by the SDN controller, providing backwards compatibility with the original OpenFlow specification. In addition, the controller periodically requests statistics from the existing mmWave links, which are returned using a multipart list of `ofp_mmwave_stats` messages.



(a) Enclosed node.

(b) Tripod mounted mesh node.

Figure 6: Small cell mesh node hardware.

4.2. Small Cell Mesh Nodes. Each small cell mesh node has a computation device that is connected to two mmWave interfaces. The nodes can be placed in a closed compartment, as seen in Figure 6(a). This enclosure provides a water and weather resistant case with flexible mounting options for outdoor experiments, e.g., on streetlights or walls, enabling an easy integration onto existing infrastructures. The used materials do not add significant attenuation to the mmWave transmission. Additionally, for the easiness of mobility and interaction with the node components, these nodes can be placed on adjustable tripods or similar support structures, which we used during our indoor experiments, as seen in Figure 6(b). The computation devices provide USB 3 ports, which are used to connect the mmWave interfaces, although additional device bus types can be integrated in future research (e.g., Thunderbolt 3 or M.2).

Each mesh node uses a modified version of Open vSwitch (OVS) 2.10.1, which integrates the enhanced OpenFlow API extensions. Internally, to process wireless-related commands, OVS exchanges messages with a Small Cell Agent (SCA) via a local UDP socket. The SCA software is written in Python and is used to communicate with the local hardware and software components used by the computation device. As seen in Figure 7, the SCA has multiple internal modules: The Beam Steering module uses a serial communication module to interact with the movement controller and sets the alignment of the mmWave interfaces. The Power Management module gracefully terminates all the running software, i.e., OVS and the other SCA modules, whenever the SCA receives a shutdown request message from OVS. Lastly, the Statistics and Link Configuration modules interact with the existing WiGig software in order to retrieve RSSI statistics and configure the existing links. Internally, the used software uses ported versions of the built-in `wpa_cli` and `wpa_supplicant` tools,

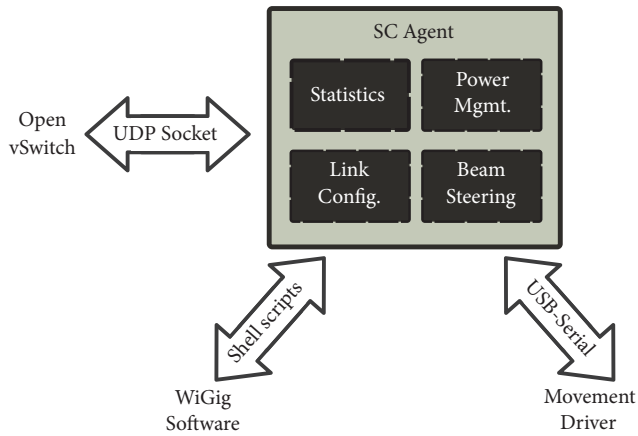


Figure 7: Small Cell Agent internal components.

which are adapted to operate with the mmWave module drivers (`wigig_cli` and `wigig_supplicant`, respectively).

To configure a mmWave link, the SCA orchestrates a set of procedures that (1) detach the involved interface from the OVS bridge (After inspecting the `wigig_supplicant` log, we discovered that the WiGig mmWave driver does not correctly process EAPOL authentication frames when the interfaces are added as OVS switch ports. As we do not have access to the driver source code, we were not able to fix this behavior.), (2) start a new `wigig_supplicant` instance, (3) perform a local link discovery routine, and finally, (4) reattach the interface to OVS. The local link discovery is conducted differently whether the interface is configured as AP or STA: when in STA mode, the SCA detects the new link by sending

Figure 8: mmWave USB interface.

ICMP packets through the involved interface every 0.1 s using `ping` tool until it receives a response. When in AP mode, the new link is detected when the SCA captures an ICMP packet on the new interface. To delete existing links, the SCA terminates the respective running `wigig_supplicant` instances.

4.3. Steerable mmWave Interfaces. Each mmWave interface used in the mesh nodes is formed by a USB 3 dongle with a 60 GHz transceiver, manufactured by Panasonic Inc., Japan, which can be seen, inside and outside its original enclosure in Figure 8. The dongles are based on WiGig/IEEE 802.11ad standards [47]. However, as they do not comply with the full protocol specification, the dongles do not support features such as digital beamforming or P2MP connectivity. The WiGig module uses the modulation coding scheme (MCS) 9 ($\pi/2$-QPSK with a 13/16 coding rate and a PHY data rate of 2502.5 Mbps, which is translated into an achievable MAC-layer throughput of $\approx$ 1.6 Gbps [48]). The module can operate on channel 2 and channel 3, among the 4 channels of IEEE 802.11ad, which operate between 59.40 to 61.56 GHz and 61.56 to 63.72 GHz (each with 2.16 GHz of bandwidth), respectively [49].

To provide the necessary attenuation to cope with the high path loss at mmWave frequency bands, the mmWave interfaces use a passive antenna reflect array, which offers beam forming capabilities. This array is made from specifically designed 12 $\times$ 12 cm printed circuit boards (PCB). It adds a gain of 26 dBi and narrows the beam to 6°, allowing a maximum intersite distance of 200 m, also greatly reducing the interference between adjacent nodes [50]. This narrow 6° beam requires a fine alignment of the mmWave interfaces, in order to form links between the nodes. For that reason, we integrate the mmWave dongle and reflective arrays in a steerable mechanical platform. The platform allows a full 360° horizontal movement freedom and 30° in the vertical direction. Respectively, the movement for each direction is controlled by a step motor, which is connected to a movement controller. This controller is connected to the computation device through a serial-to-USB interface, which can be used by the computation device to modify the antenna alignment. Figure 9 shows the front and back of a fully assembled mmWave interface. In the left, the front of the reflect array can be seen, along with the USB mmWave dongle, and both step motors below. Figure 9 (right) shows the back of the device, containing the movement controller PCB.

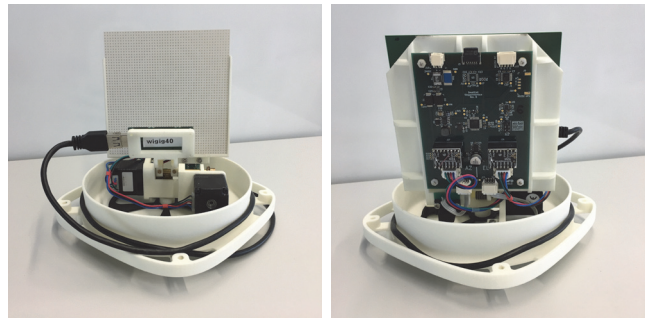


Figure 9: MmWave interface with a reflect array, assembled in a mechanical movable platform.

5. Testbed Evaluation and Discussion

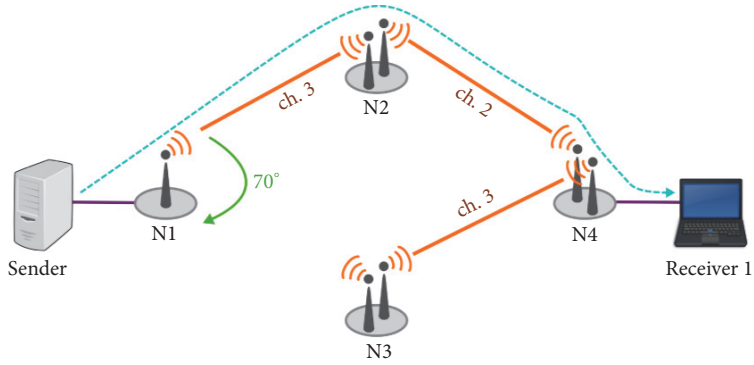
In this section, we evaluate several aspects of SDN-based mesh backhaul reconfiguration, using our mmWave multi-hop testbed. Our experiments aim not only to validate the developed backhaul reconfiguration primitives, but also serve to answer the following questions:

(i) What is the impact of topology changes and suboptimal channel assignment on existing traffic using a SDN-based mesh backhaul reconfiguration?

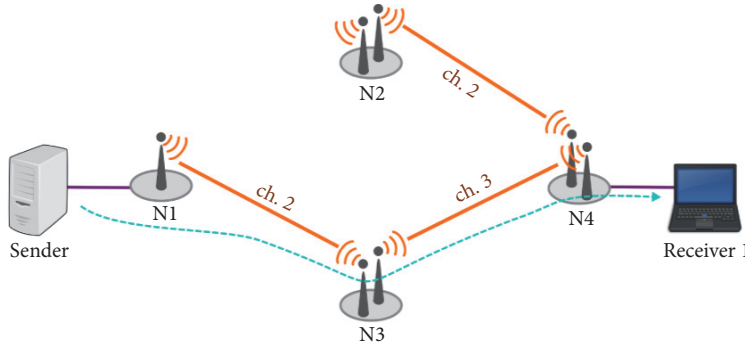
(ii) What is the impact of energy efficient small cell power on-off on existing traffic using a SDN-based mesh backhaul reconfiguration?

In the following subsections, we will analyze important key performance indicators such as delay, loss, and throughput when answering both questions for a variety of traffic demands. We begin with baseline experiments to identify the performance impact of different primitives used for the backhaul reconfiguration operations.

We use different traffic generation tools to conduct our experiments. `iperf3` (https://software.es.net/iperf/) generates UDP traffic between the sender and receiver nodes. The traffic flows are orchestrated remotely to run until the respective `iperf3` client applications are terminated. Each flow uses 7882 byte packets, matching the configured MTU in the mmWave interfaces. We vary the sending rate, according to our experiments. The `iperf3` server instance reports the throughput and loss values every second, which we later correlate with the configuration stages, to obtain the average and standard deviation values. We use the `ping` tool to measure the RTT by sending ICMP packets every 10 ms between the involved hosts. We use `tcpdump` to capture traffic during the whole experiment duration on the wired links from the receiver and sender nodes (as well as in the mesh node counterpart), and after a mmWave link is established (as it is not possible to start the traffic capture before) in all the mmWave interfaces of the involved testbed nodes. We implement scripts to correlate the trace files in order to dissect the RTT per link and per node, for identifying latency bottlenecks.



(a) Initial configuration. N1 has its mmWave interface aligned with N2.



(b) Final configuration. N1 aligns its interface with N3 and a new link is established.

Figure 10: Single backhaul link reconfiguration configurations.

5.1. Baseline Backhaul Link Reconfiguration. As a baseline measurement, we evaluate the impact of a single backhaul link configuration on existing traffic. To that end, we prepare a testbed setup where mesh node N1 uses a single mmWave interface. Initially, N1 has its interface aligned with N2 and a link is established between the two nodes (Figure 10(a)). In addition, the N2-N4 and N3-N4 links are also established. At the same time, the sender node S is sending an 800 Mbps UDP flow to R1 across the N1-N2-N4 path. The reconfiguration procedure consists in aligning N1's interface with N3 (through a 70° rotation), establishing a new N1-N3 link, and updating the previous installed forwarding rules to match the new link when it is detected by the SDN controller (Figure 10(b)). We repeat the experiment 15 times.

The links are aligned by requesting the N1 antenna to rotate to a new position that will align with the opposite link interface. The alignment values are calculated before the experiments, firstly according to the nodes' indoor positions, followed by the manual tuning of the respective interface alignment angles, in order to improve the link signal quality. The obtained azimuth and elevation values are then used in the configuration scripts as input during the experiments, although they can also be stored in the SDN controller. The forwarding rules are updated whenever there are new links formed by different interfaces. Since the involved links are wireless, it is necessary to rewrite the MAC addresses on each link, to match the source and destination MAC addresses from the associated STA/AP on the respective links [51, 52]. Before traffic is sent to the hosts, the small cell nodes rewrite the MAC addresses, matching the original source/destination addresses. Although the MAC addresses are modified on each link, the end-to-end forwarding paths remain identified by the source and destination IP addresses from the respective hosts.

Figure 11 shows the cumulative distribution function (CDF) for the different time intervals required to perform the configuration operations and reestablish end-to-end connectivity between the sender and receiver nodes. These include the mechanical interface alignment from the N1-N2 position to the N1-N3 destination, the internal link configuration by N1 and N3, and the detection of the new link by the SDN controller. The interface alignment time is measured by polling the mechanical controller status until the interface is no longer moving. The internal link configuration time is computed by calculating the maximum link configuration time between the AP and the STA. The link detection at the SDN controller is obtained by measuring the elapsed time from when the link configuration requests are sent to N1 and N3, until the new link is detected in ODL. Internally, ODL detects a new link when it receives a Link Layer Discovery Protocol (LLDP) packet from one of the corresponding interfaces (which are flooded every 5 seconds, or when a new switch port is added to a node) and updates its network graph. Lastly, the traffic interruption interval is obtained by calculating the maximum time where no packets are exchanged during the reconfiguration process. The results

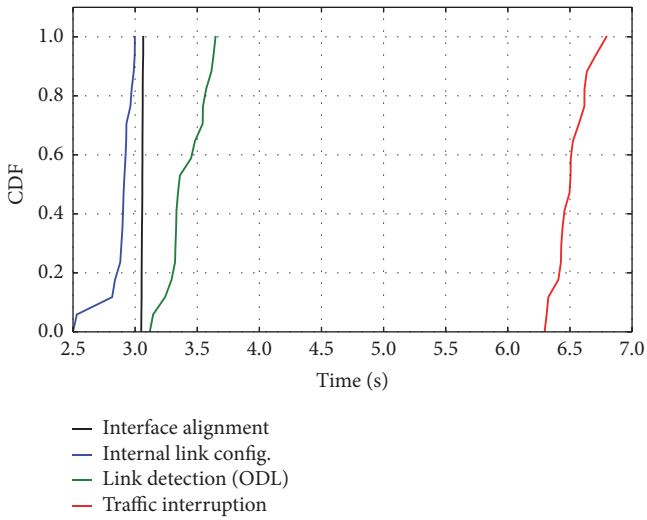


FIGURE 11: CDF for the link alignment time, link configuration time, link detection by the SDN controller and interruption time during the reconfiguration of N1's link, when having a single interface.

show that the alignment time is constant (≈3.06 s). The internal link configuration requires the lowest delay from all steps involved for reestablishing connectivity. Nonetheless, this operation still takes an average of 2.88 s, due to the overhead of restarting `wigig_supplicant`, as previously mentioned in Section 4.2. Note, that we only trigger the internal link configuration after the interfaces have been properly aligned. The new link is detected in the SDN controller at average around 3.41 s after the antennas have been aligned. This is because after the internal link is configured, the controller needs to receive the LLDP link detection messages and update the internal network graph, which, for this single link, takes around less than one second (interface detection, followed by generating, receiving and processing an incoming LLDP packet at the controller). The average of the total traffic interruption time (6.50 s) is shorter than the sum of the average alignment times and the link detection times by the SDN controller (6.46 s). Through the analysis of the packet trace files, we found that N1 can still transmit packets to N2 for some duration when the antennas start to rotate away, until the link breaks due to the complete misalignment. In addition, the mechanical platform initially increments its speed, moving slower in the beginning of the rotation. This causes the end-to-end connectivity interruption interval to be smaller than the total reconfiguration time.

This experiment demonstrates that the SDN controller is capable of reconfiguring the network and reestablishing end-to-end connectivity between the sender and receiver nodes. Due to the availability of only a single interface at N1, the backhaul reconfiguration leads to packet loss and negatively impacts existing traffic, in case the antennas need to rotate and links need to be reestablished with different neighbors. In the next sections, we evaluate the additional benefit of having multiple radios and antennas, which allows us to establish backup paths that can serve existing traffic while reconfiguring.

5.2. Optimal Steerable mmWave Mesh Backhaul Reconfiguration. To implement any wireless backhaul reconfiguration use cases from Section 2.1, it is crucial that the wireless backhaul can perform these reconfiguration operations with minimal disruption on existing UE traffic. Consequently, we ask the question: *what is the impact of topology changes and suboptimal channel assignment on existing traffic using a SDN-based mesh backhaul reconfiguration?*

To answer this question, we design a set of experiments where we use two different backhaul configurations (C1, C2) to serve a given traffic matrix. Moreover, these experiments aim to validate the related reconfiguration primitives that are used in network optimization frameworks (e.g., the previously primitives described in Section 3.3.2).

The experiments are orchestrated by a set of procedures that allow the transition from an initial topology state C1 to any given final C2 configuration. The main goal is to make this transition as seamless as possible, leading to minimum traffic disruption. The C1-C2 transition is done by (1) aligning the involved C2 mmWave antennas to their final positions, (2) configuring the new links from C2, (3) updating the new topology routes by rewriting OpenFlow rules, and (4) deleting unused links from C1.

In our experiments, the sender node has an active flow towards each receiver node. We consider two different scenarios: in the first one, the initial backhaul configuration C1 is able to handle the existing traffic demand, but one of the primary mmWave links that forwards the traffic from both nodes needs to be deactivated, causing the reconfiguration of the backhaul (antenna movement, neighbor establishment, rule update) to reroute all the traffic through new paths that are not initially configured, thus forming configuration state C2. In the second scenario, the initial backhaul configuration contains one bottleneck link that cannot forward the required traffic between the sender and two receiver nodes. The backhaul is then reconfigured to split the two flows through different paths. We investigate both optimal and suboptimal channel assignments in the mesh and its impact on the reconfiguration.

For both scenarios, the experimental methodology is similar. The testbed is initialized by configuring the N1-N2 and N2-N4 mmWave links using channel 3 and 2, respectively. Once the links are available, the initial forwarding rules are installed, routing the traffic between S and R1 nodes using the N1-N2-N4 path, and the traffic between S and R2 using the N1-N2 link (Figure 12). With the forwarding rules installed, the traffic and RTT measurements start and, approximately 20 s after, the backhaul reconfiguration is triggered by the SDN controller, aligning the interfaces to the new positions and configuring the new links. With the new links formed, the forwarding rules of the new configuration C2 are installed, replacing the initial ones. Afterwards, unused links from the initial topology are removed and the measurements continue until the end of the experiment. We repeat each experiment 15 times.

5.2.1. Scenario I: Reconfiguration under Low Traffic Volume. In this scenario, we aim to evaluate the reconfiguration of our testbed when transitioning between two configuration

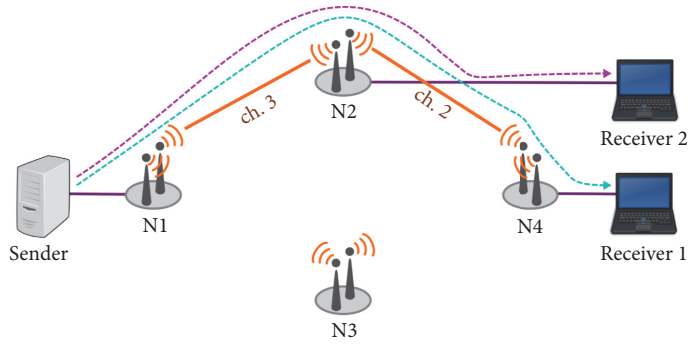


Figure 12: Initial mmWave mesh backhaul reconfiguration state C1. All testbed nodes are powered on and all backhaul traffic is routed through N2.

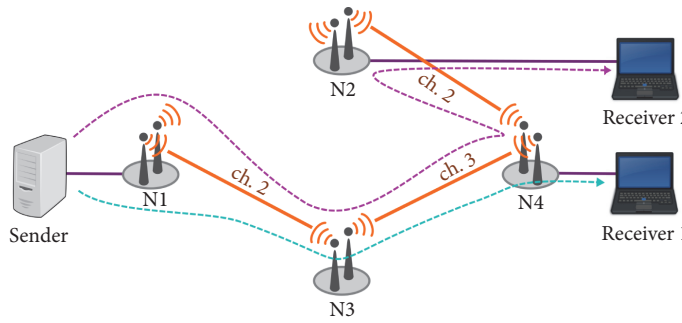


Figure 13: Final configuration state C2 of the low traffic volume scenario topology.

states C1 and C2, where the backhaul can provide the required demands in both initial and end states. Such a transition is necessary when the SDN controller detects a link failure persisting for a long duration, caused by, e.g., long-term blockage or hardware problems on the small cell transceivers. In this experiment, the sender node starts by sending two 500 Mbps UDP flows: F1 (to Receiver 1) and F2 (to Receiver 2). As the N1-N2 link is disabled in the final configuration state C2, the topology is reconfigured by setting up the N1-N3 (channel 2) and N3-N4 links (channel 3). When the new links are established, the traffic is rerouted to the new configuration C2, where the forwarding rules for F1 are configured to use the N1-N3-N4 path, and F2 is routed through N1-N3-N4-N2 path (Figure 13).

To avoid traffic disruptions, the rules are installed first in the nodes without ongoing traffic from active flows, then in the nodes where the initial links are being used. Therefore, forwarding rules for F1 are installed first in N3, updating N4 and N1 afterwards. The new F2 forwarding entries are installed in N3 and N4, then in N2 and N1. After the traffic is rerouted to use the new paths, the N1-N2 link is disabled and the testbed reaches the final configuration state C2. With our testbed, the calculation of the flow installation order is straightforward and can easily be hard-coded in the experimental scripts, but with larger scenarios, such flow migration is required to be computed considering the whole topology as input [53]. It is worth noting that during the initial configuration state C1, flow F1 is routed over two mmWave link hops and F2 over one hop, while in C2, flow

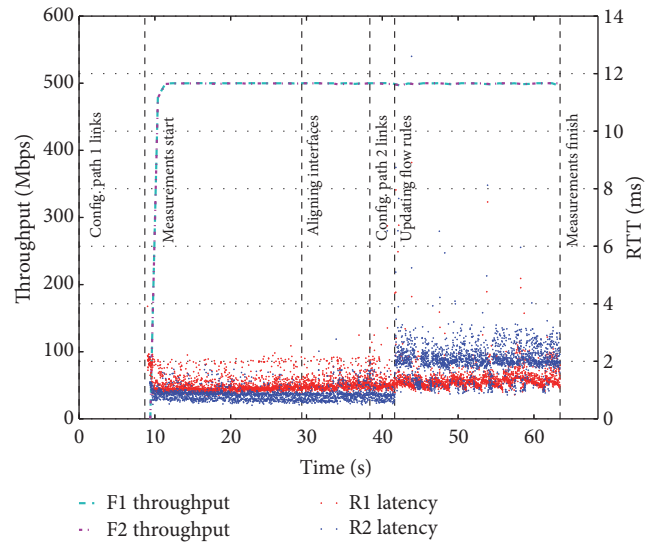


Figure 14: Throughput and end-to-end RTT over time with an SDN-based backhaul reconfiguration and two 500 Mbps flows between S and R1 and R2, respectively.

F1 experiences two wireless hops and F2 experiences three hops.

Figure 14 shows the RTT for both receivers R1 and R2 along with the throughput values of flows F1 and F2 over time. Complementary to the plot, the average and standard

Table 4: Average and standard deviation performance metrics during low traffic volume experiments.

Events	Host	Throughput (Mbps)	RTT (ms)	Loss %
Start - Alignment	R1	498.65 (±4.76)	1.23 (±0.37)	0.0
	R2	498.44 (±4.73)	0.87 (±0.32)	0.0
Alignment - Path 2 config.	R1	499.82 (±0.34)	1.21 (±0.29)	0.0
	R2	499.64 (±0.45)	0.87 (±0.29)	0.0
Path 2 config. - Fwrd. update	R1	499.68 (±0.55)	1.25 (±0.37)	0.0
	R2	499.55 (±0.46)	0.89 (±0.34)	0.0
Fwrd. update - End	R1	499.51 (±0.83)	1.51 (±0.49)	0.0
	R2	499.56 (±0.73)	1.70 (±0.57)	0.0

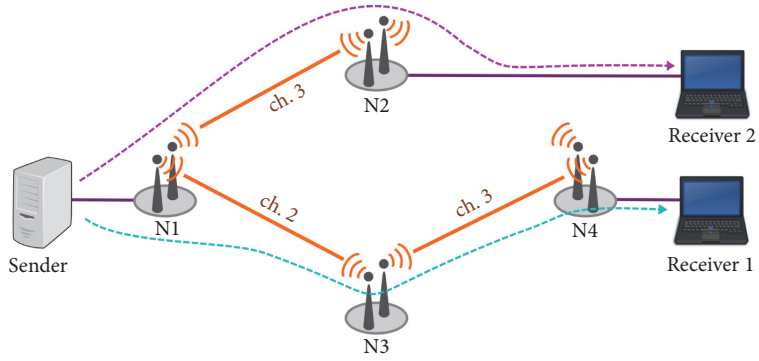


Figure 15: Final configuration state C2 of the high traffic volume scenario topology.

deviation (stdev) values of the RTT, throughput, and packet loss, for all the testing iterations and all configuration stages, are found in Table 4.

As the traffic demands of these experiments do not saturate the links, the impact on queuing is minimal for both nodes. R1 experiences higher RTT because it traverses more wireless hops in C1. After the links are aligned, after the second path is being configured (42 s), we observe punctual RTT spikes on both receiver nodes (up to nearly 10 ms). A more detailed analysis of these sudden delay spikes revealed interference effects, caused from having all the mmWave links active during this period and also due to the physical deployment of our nodes in the testbed. These interference effects motivated a more thorough analysis of the impact of channel assignment changes on mmWave links, which are discussed in detail in Section 5.2.3.

After the forwarding rules are updated, we can observe the impact of the new configuration C2 on the existing flows. While the average RTT of R1 stays the same due to the new forwarding configuration having the same number of hops (with an average short increase of ≈ 0.28 ms due to higher RTT measurements caused by the interference between the N2-N4 and N1-N3 links), the average RTT of R2 is increased to approximately 1.7 ms. This value is higher than the average RTT of R1 as the new forwarding path for F2 is composed by three mmWave hops, instead of the initial one-hop path.

Despite the variations of RTT between the two receiver nodes, the throughput of both flows F1 and F2 is not affected by the multiple reconfiguration operations. Therefore, we show that an orchestrated reconfiguration of the network under a low traffic volume can be achieved with minor impact on existing flows, without causing packet loss.

5.2.2. Scenario II: Reconfiguration with High Traffic Volume. When backhaul links are highly utilized, additional user traffic demand spikes (e.g., a sudden increase of users entering a stadium or a music arena) lead to persistent link congestion. In order to cope with an increased demand, new small cells can be powered on if available, routing traffic away from hotspots. With new demands, the backhaul orchestrator is required to compute a new backhaul configuration that can fulfill the increased traffic demand. To that end, we evaluate such a scenario in our testbed, following the same reconfiguration goals, as previously mentioned.

In this set of experiments, we setup two 900 Mbps UDP flows F1 and F2 between S and R1 and R2, respectively. The reconfiguration of the testbed is then triggered, configuring the N1-N3 and N3-N4 links. When the new links are formed, the forwarding rules of the flow F1 are updated to use the N1-N3-N4 path. Similarly as in the previous scenario, we install the new rules firstly in N3, and only after in N4 and N1. When the flow F1 is successfully rerouted, the N2-N4 link is then deactivated, having flow F1 routed through N1-N3-N4, and the flow F2 routed using the N1-N2 forwarding path (Figure 15).

For this experiment, the results of the bandwidth of flows F1 and F2, along with the RTT between the server and both receiver nodes, can be observed in Figure 16, over the elapsed experiment time, during one iteration. In addition, the respective average and stdev values, before

Table 5: Average and standard deviation performance metric values during the high traffic volume experiments.

Events	Host	Throughput (Mbps)	RTT (ms)	Loss %
Start - Alignment	R1	722.55 (±96.38)	41.59 (±11.29)	19.35
	R2	802.52 (±96.63)	42.77 (±8.90)	10.44
Alignment - Path 2 config.	R1	725.87 (±95.86)	44.89 (±1.54)	19.35
	R2	802.94 (±95.70)	44.73 (±1.63)	10.78
Path 2 config. - Fwrd. update	R1	740.66 (±97.12)	44.95 (±1.55)	18.04
	R2	796.17 (±96.82)	44.75 (±1.65)	11.76
Fwrd. update - End	R1	898.33 (±12.55)	1.67 (±3.32)	0.24
	R2	896.89 (±5.50)	1.16 (±4.17)	0.46

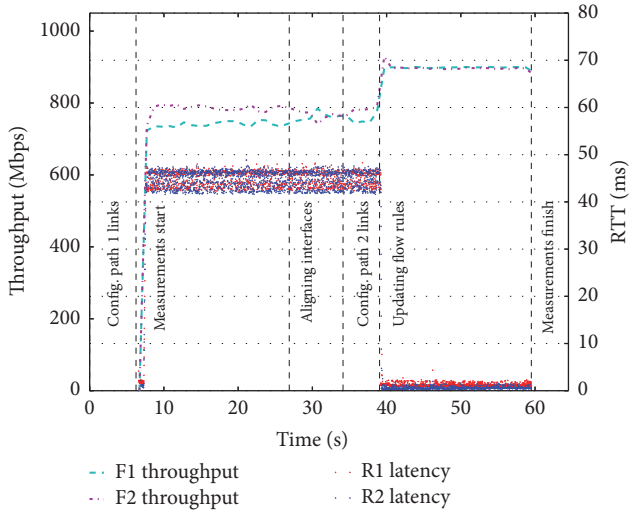


Figure 16: Throughput and end-to-end RTT over time with an SDN-based backhaul reconfiguration and two 900 Mbps flows F1 and F2 between S and R1 and R2, respectively.

and after the network reconfiguration, are presented in Table 5.

From the beginning of the traffic measurements (≈ 6 s), we observe the saturation of the N1-N2 link, as F1 and F2 receives less bandwidth compared to its target (900 Mbps). The aggregated throughput of both flows is capped to the ≈1.5 Gbps maximum available link capacity, resulting in an average packet loss of approximately 15% for both flows (18.9% in F1 and 11% in F2). The link congestion results in queue buildups, leading to bufferbloat and approximately 45 ms RTT for both receiver nodes (with exception of the interval between the measurements' start and the interface alignment, as this average value is slightly lower, due to the initial measurement values, before the N1-N2 link becomes congested).

After the new N1-N3-N4 path is configured and the F1 flow is rerouted, both F1 and F2 reach the desired throughput. The congestion on the N1-N2 link disappears, reducing the latency between S and R2 to around 1.2 ms. At the same time, the RTT between S and R1 drops to 1.7 ms. This increased latency, compared to the values for R2, is caused by the additional hop between the two nodes (S-N1-N3-N4-N1), compared to the S-N1-N2-R2 path. During this last experiment interval, we again observe interference between the mmWave links, which results in latency spikes (e.g., at 46 s) caused by a single high-RTT measurement (4 ms), followed by the decrease of the latency to the average values on the following received ICMP packets. As it can be seen from Table 5, the average packet loss for the last configuration stage is nonzero. This is because `iperf3` records packet loss every second, which does not perfectly align with the configuration stage changes.

5.2.3. Impact of Channel Assignment within the mmWave Backhaul. To investigate the effects of channel assignment on traffic and interference, we conduct a set of experiments with a suboptimal channel configuration on the used links. To that end, we reconstruct the experiments from Section 5.2.2, modifying the used channels in the mmWave link configuration: N1-N2 and N2-N4 were set to use channel 2, while N1-N3 and N3-N4 used channel 3. With this configuration, each of the two disjoint paths between N1 and N4 would be using the same channel on both links.

The throughput and RTT over time for a single experiment iteration are shown in Figure 17, while the average and stdev values for the 15 tested iterations can be found in Table 6. From the start of the measurements, until the update of the forwarding rules (43 s), both flows are routed using the N1-N2 link, which uses the same channel as the N2-N4 second hop of flow F1. Contrary to the previous experiments, where the N1-N2 link was fully utilized, both flows have an aggregated throughput of approximately 670 Mbps, which is 44% less than the full link utilization. Consequently, both nodes experience an increased packet loss (75.14% for F1 and 47.99% for F2), caused not only by the N1-N2 link saturation, but also by cochannel interference. In this scenario, as both links within a path use the same channel and both N2's interfaces are vertically stacked (having the transmitter and receiver radio in close physical proximity), when N2 receives a packet from N1 on one interface and N2 should forward a different packet to N4 in parallel, the transmitter radio at N2 it will not sense the reception from N1 due to the directional antennas. Consequently, it will send the packet in parallel to node N4. The high sending power of the transmitter interface N2 will lead to additional cochannel interference on the reception of the packet from N1 on the receiving radio on N2. This additional interference reduces the dynamic link

Table 6: Average and standard deviation performance metric values using a suboptimal channel assignment.

Events	Host	Throughput (Mbps)	RTT (ms)	Loss %
Start - Alignment	R1	201.05 (±72.44)	215.75 (±244.96)	73.60
	R2	445.69 (±131.86)	80.09 (±73.61)	49.82
Alignment - Path 2 config.	R1	189.58 (±72.67)	291.36 (±244.63)	76.50
	R2	479.71 (±132.65)	78.14 (±67.66)	45.87
Path 2 config. - Fwrd. update	R1	213.28 (±87.62)	263.26 (±202.98)	75.34
	R2	470.15 (±143.06)	81.75 (±69.47)	48.30
Fwrd. update - End	R1	413.24 (±112.18)	139.11 (±129.10)	53.57
	R2	890.01 (±27.12)	2.00 (±14.17)	1.33

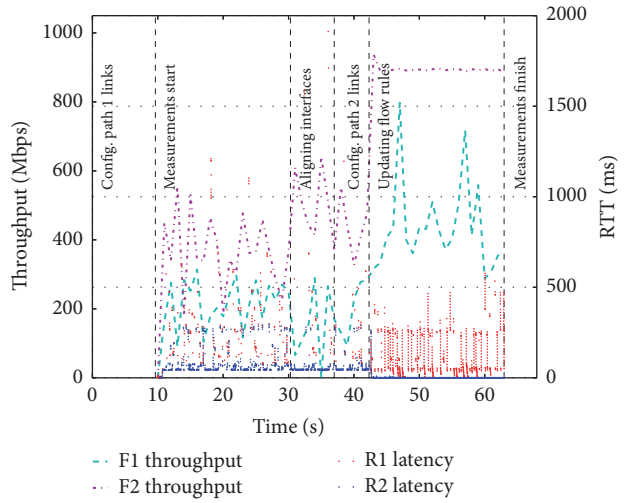


Figure 17: Throughput and end-to-end RTT over time for the SDN-based backhaul reconfiguration, using a suboptimal channel assignment.

margin of the mmWave links due to random high path loss, shadowing, and blockages, further leading to high random packet loss. Similarly, when N4 sends an ACK back to N2 and N2 sends back an ACK to N1 at the same time, the transmission of the ACK will interfere, leading to packet loss and reduced throughput. Note, also that the throughput of F2 is higher before the reconfiguration, compared to F1. F2 is routed over a single hop, while F1 is forwarded over two links, leading to a higher packet loss rate in F1.

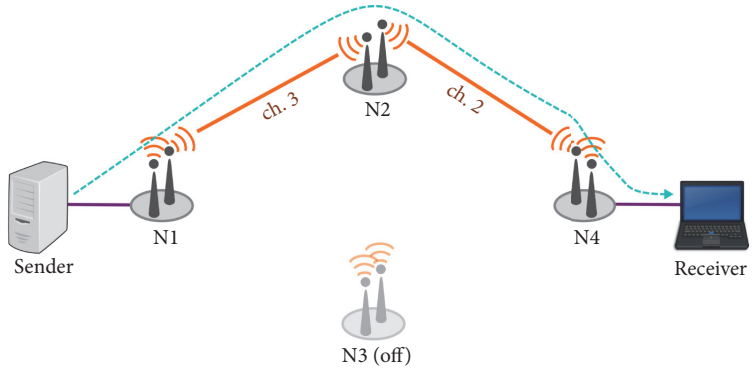
Regarding the RTT measured at the receiver nodes before the reconfiguration, we can observe the effects of increased queuing delay at N1-N2, as it increases shortly after the traffic congests the links. Yet, the average values are higher (212.96 ms more in R1 and 35.9 ms in R2) compared to the results from the previous scenario. While using a channel assignment configuration without significant interference does not impact the RTT values difference between the two receiver nodes (as it is primarily caused by a single link), in this set of experiments we observe an increase of approximately 177 ms between the measured RTT at R1 and R2, due to the delay of the N2-N4 link. This is because the additional cochannel interference leads to high packet loss, requiring the lower layer to frequently retransmit lost packets. This leads to significant queue buildups and bufferbloat due to cochannel interference.

After the network is reconfigured, the cochannel interference negatively affects the performance of flow F1, as it is forwarded over two hops having the same channel (ch. 3). The reasons for the low throughput and high packet loss for F1 are the same as described before. At the same time, F2 recovers from its throughput deficit and high latency, as the N1-N2 link is exclusively using channel 2 and it is solely used to forward the between the source and R2.

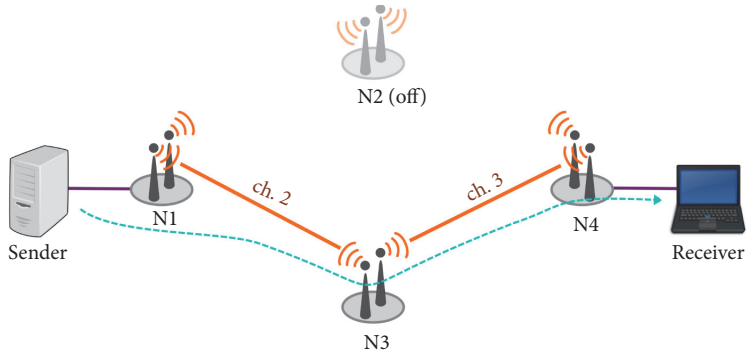
To conclude, it is possible to observe the negative impact of suboptimal channel assignment within the wireless backhaul, even when using our directional antennas and forwarding packets over more than one hop. The effects of cochannel interference cause a significant decrease of the existing flows' throughput and an increase in the measured RTT, when compared to an identical scenario where a better channel assignment results in less interference among the used links.

5.3. Adaptive On/Off Mesh Backhaul Operation. To reduce the overall backhaul power consumption, unused nodes should be powered off, leading to a change of backhaul configuration. Therefore, when transitioning between different configuration states, the SDN controller needs to be able to turn on or off the backhaul nodes, reconfigure the involved mmWave backhaul links, and update the existing forwarding rules, according to the newly formed topology. In this section, we evaluate the adaptive on/off configuration primitives, by seamlessly performing a reconfiguration of the backhaul jointly with the link realignment, link configuration and forwarding rule update commands, and the required combined orchestration by the SDN controller.

At the beginning of the experiment, N3 is powered off and all the remaining mesh nodes are switched on. We configure the N1-N2-N4 path initially (configuration C1), by installing the respective links and forwarding rules, and start a 800 Mbps UDP flow F1 between S and R1, as shown in Figure 18(a). At the same time, we start the RTT measurement between R1 and S. After 10 s, we power on N3, which takes approximately 33 s to boot. With all the mesh nodes turned on, we proceed to reconfigure the testbed to use the N1-N3-N4 path, using the same reconfiguration routines as in the previous described scenarios: firstly, aligning the involved mmWave interfaces of N1, N3, and N4, followed



(a) Initial and final configuration state C1.



(b) Intermediate configuration state C2 (N3 is powered on and N2 is shut down.)

Figure 18: Adaptive on/off backhaul topology and routing configurations.

Table 7: Average and standard deviation performance metrics during the adaptive on/off reconfiguration scenario.

Events	Throughput (Mbps)	RTT (ms)	Loss %
Start - Alignment	799.12 (±4.90)	1.29 (±0.25)	0.0
Alignment - Path 2 config.	799.75 (±0.64)	1.28 (±0.23)	0.0
Path 2 config. - Fwrd. update	799.04 (±2.38)	1.29 (±0.36)	0.0
Fwrd. update - Path 1 config.	799.77 (±0.86)	1.27 (±0.24)	0.0
Path 1 config. - Fwrd. update	799.55 (±0.87)	1.29 (±0.29)	0.0
Fwrd. update - End	799.63 (±0.73)	1.36 (±1.87)	0.0

by the configuration of the new links and the update of the forwarding rules to match the new path. Ten seconds after, we disable the links in the N1-N2-N4 path and power off N2, leaving the mesh network operating again with three nodes powered on, as seen in Figure 18(b) (configuration C2). N2 is powered on again after 15 s, and after it is operational, we re-establish the connectivity on the N1-N2-N4 path, rewriting the flows to its original configuration until the end of the experiment (configuration C1).

As it can be seen in Table 7, the receiver does not experience any packet loss during any of the experimental phases. Consequently, the throughput for the existing flows between the two nodes maintains the desired rate. Figure 19 shows the RTT and throughput values over one iteration, where the network reconfiguration events are marked over vertical dashes. The plotted RTT values are separated by different lines.

While there is no significant deviation from the overall measured RTT values during the experiment, similarly to the previous experiments, it is possible to see punctual RTT spikes whenever all the backhaul links are active due to interference (shortly after 64 s and 138 s, respectively), caused by MAC-layer link establishment messages being exchanged. Shortly after the update of new forwarding rules (around 71 s and 146 s), we observe a short interruption on the latency measurements on the mmWave links, while the total RTT does not get affected. A closer inspection of the collected trace files reveals that, while the forwarding rules are being updated, the end-to-end delay is successfully measured at R1 (by sending and receiving the respective ICMP request and reply packets). However, a number of ICMP requests are sent by R1 over the newly configured path, while the replies are sent over the old path, as the forwarding rules in N1 are the last ones to be installed. Whenever the request and

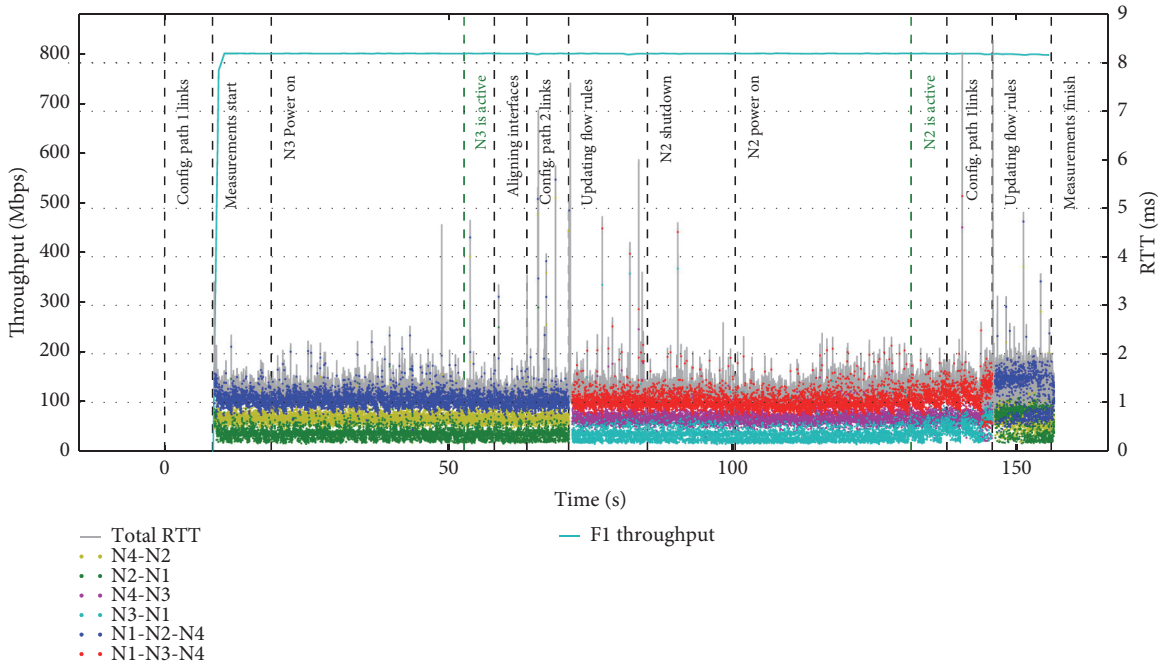


Figure 19: Throughput and per-hop RTT over time during a single experiment for the adaptive on/off topology reconfiguration scenario. The total end-to-end RTT measured at R1 is shown by the grey line, while the RTT over the mmWave links is plotted with different colored dots. The first path (N1-N2-N4) is plotted in blue, while the second one (N1-N3-N4) is plotted in red. The remaining colors are used to individually display the RTT on each mmWave link.

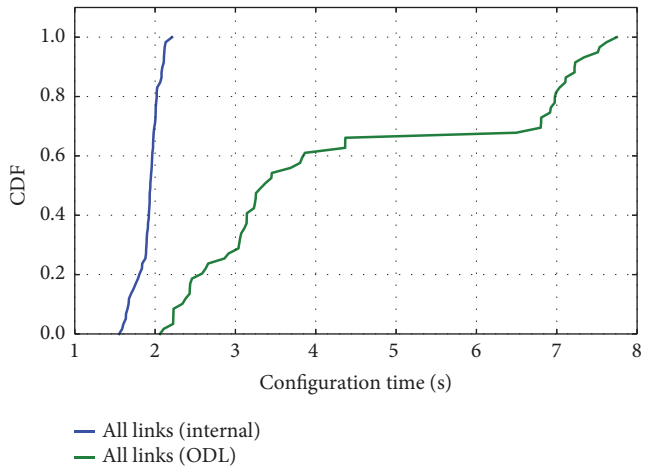


Figure 20: CDF for the internal link configuration time and link detection time by the SDN controller for all the testbed links, during the adaptive on/off testbed experiments.

reply packets are split over different links, it is not possible to calculate the RTT individually per link, as they would require both packets to compute this metric.

Similarly to the baseline link configuration experiments, the interface alignment times are also stable, i.e., 4.92 s (±0.053) for a 160° rotation of N3's interface to the N3-N1 link, and 3.48 s (±0.012) for a 90° rotation of N3's interface to the N3-N4 link. Figure 20 shows a CDF of the internal link configuration time by the mesh nodes, alongside the link detection time by the SDN controller (as measured in Section 5.1), for all the configured link pairs, during this set of experiments. While the internal link configuration time is consistent, averaging 1.92 s with stdev 0.149 s, the detection of the new links at the SDN controller varies between 2.06 s and 7.75 s. This variation is caused by the scheduling of LLDP packets by the controller for the backhaul links, as it occurs every 5 seconds, or when a new switch port is added. However, when configuring a new link and the mmWave interface is added back to OVS by the SCA, if both interfaces from that link are not ready (i.e., both interfaces are not configured in OVS in both nodes), the sent LLDP packet is not received by the corresponding interface. Therefore, only

the next LLDP packet can be transmitted (after 5 seconds), if the link is ready (e.g., if a LLDP packet is sent at t=1.8s and the link is ready at 1.9s, the next LLDP packet is only transmitted at t=6.8).

5.4. Reflections. Our experiments demonstrate how SDN enables the orchestration of different backhaul reconfiguration mechanisms, even under the presence of complex reconfiguration operations, i.e., a series of interface alignment, link establishment, power on/off management, and rerouting steps. As we have seen, a proper orchestration leads to a seamless network operation and uninterrupted end-to-end user connectivity. By being able to adaptively power on/off the network nodes, the backhaul can be easily reconfigured to achieve energy efficiency goals and contribute towards the goal of green network operation.

However, we also need to consider how resiliency can be applied jointly with this type of energy efficient network reconfiguration. Specifically, if all the unused backhaul nodes in a given topology configuration state are powered off, any kind of failure, even temporary, would disrupt the network operation. Therefore, a balance needs to be made between energy efficient and resilient operations of the mesh backhaul, where nodes can be adaptively powered on/off, while providing backup links and/or paths. Nonetheless, having both fast-failover resiliency and adaptive on/off backhaul reconfiguration mechanisms available in the network would allow the operators to fine tune the backhaul operation, according to the desired policies.

6. Conclusions

In this paper, we present the SOCRA architecture, which uses the SDN control plane to manage a small cell wireless multihop mesh backhaul network. The SDN control plane is responsible for all the configuration-related operations, including channel assignment, interface alignment using mechanical rotating directional antennas, powering on/off the small cells, and managing the forwarding states. The proposed architecture provides an orchestration interface that can be used to communicate with external optimization frameworks, which can use the framework to optimize the backhaul for different operational goals (e.g., energy efficiency or resiliency).

We implemented an SDN controller and a multiradio mmWave small cell backhaul node and deployed a testbed in a mesh topology formed by small cell nodes, which can interpret the reconfiguration commands issued by the SDN controller and translate them into appropriate actions. The nodes are equipped with multiple mechanical steerable mmWave interfaces that can rotate and align with each other, dynamically forming new links, according to the SDN controller's instructions. The testbed was used to validate different reconfiguration primitives, which included the realignment of the mmWave interfaces, the dynamic configuration of the backhaul links, on/off powering of the small cell nodes, and the update of the forwarding rules. We evaluated the reconfiguration of our testbed when transitioning between different configuration states using different traffic scenarios and channel assignments. Our measurement results indicate that it is possible to reconfigure the backhaul without a significant impact on existing UE traffic, if there are available backup paths that can be used for temporary traffic routing.

As future work, we intend to investigate the impact of the backhaul channel assignment using different positioning and distances of the backhaul nodes and interfaces. Moreover, we will extend our backhaul optimization framework to consider the used channels in the formed links and develop fast-heuristics. The fast-heuristics guide the backhaul orchestrator, providing an ordered sequence of reconfiguration operations to perform, in order to minimize the impact on the end-to-end performance.

Data Availability

The experimental log and trace files used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Parts of this work have been funded by the Knowledge Foundation of Sweden through the project SOCRA. This work is also funded by the European Commission (EC) H2020 and the Ministry of Internal affairs and Communications (MIC) in Japan under grant agreements 723171 in the EC and 0159-0149, 0150, 0151 in the MIC.

References

[1] C. Cisco, "Cisco visual networking index: global mobile data traffic forecast, 2016–2021," *Cisco White Paper*, 2017.

[2] S. Telecom, "SK telecoms view on 5G vision, architecture, technology, and spectrum," *5G White Paper*, 2014.

[3] Q. Cui, H. Wang, P. Hu et al., "Evolution of limited-feedback CoMP systems from 4G to 5G: CoMP features and limited-feedback approaches," *IEEE Vehicular Technology Magazine*, vol. 9, no. 3, pp. 94–103, 2014.

[4] S. Nagul, "A review on 5g modulation schemes and their comparisons for future wireless communications," in *Proceedings of the 2018 Conference on Signal Processing and Communication Engineering Systems (SPACES)*, IEEE, pp. 72–76, 2018.

[5] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, 2014.

[6] T. Rappaport, S. Sun, R. Mayzus et al., "Millimeter wave mobile communications for 5G cellular: it will work!," *IEEE Access*, vol. 1, pp. 335–349, 2013.

[7] W. Feng, Y. Li, D. Jin, L. Su, and S. Chen, "Millimetre-wave backhaul for 5g networks: challenges and solutions," *Sensors*, vol. 16, no. 6, p. 892, 2016.

[8] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design considerations for a 5G network architecture," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 65–75, 2014.

[9] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[10] IEEE Std, "802.11ad, Part 11: wireless lan medium access control (MAC) and physical layer (PHY) specifications amendment 3: enhancements for very high throughput in the 60 GHz band," *IEEE Computer Society*, 2012.

[11] T. Nitsche, A. B. Flores, E. W. Knightly, and J. Widmer, "Steering with eyes closed: Mm-wave beam steering without in-band measurement," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM)*, IEEE, pp. 2416–2424, 2015.

[12] T. S. Rappaport, F. Gutierrez, E. Ben-Dor, J. N. Murdock, Y. Qiao, and J. I. Tamir, "Broadband millimeter-wave propagation measurements and models using adaptive-beam antennas for outdoor Urban cellular communications," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 4, pp. 1850–1859, 2013.

[13] S. K. Saha, H. Assasa, A. Loch et al., "Fast and infuriating: Performance and pitfalls of 60 GHz WLANs based on consumer-grade hardware," in *Proceedings of the 15th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON 2018*, IEEE, pp. 1–9, Hong Kong, June 2018.

[14] H. Assasa, S. K. Saha, A. Loch, D. Koutsonikolas, and J. Widmer, "Medium access and transport protocol aspects in practical 802.11 ad networks," in *Proceedings of the 19th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2018*, IEEE, pp. 1–11, Greece, June 2018.

[15] X. Li, R. Ferdous, C. F. Chiasserini et al., "Novel resource and energy management for 5g integrated backhaul/fronthaul (5G-crosshaul)," in *Proceedings of the 2017 IEEE International Conference on Communications Workshops, ICC Workshops 2017*, IEEE, pp. 778–784, France, May 2017.

[16] A. D. La Oliva, X. C. Perez, A. Azcorra et al., "Xhaul: toward an integrated fronthaul/backhaul architecture in 5G networks," *IEEE Wireless Communications Magazine*, vol. 22, no. 5, pp. 32–40, 2015.

[17] R. Santos and A. Kassler, "A SDN controller architecture for small cell wireless backhaul using a LTE control channel," in *Proceedings of the IEEE 17th International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–3, 2016.

[18] V.-G. Nguyen, A. Brunstrom, K.-J. Grinnemo, and J. Taheri, "SDN/NFV-based mobile packet core network architectures: a survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1567–1602, 2017.

[19] J. Kim, S. Kim, and Y. Joo, "Distributed channel assignment algorithm based on traffic awareness in wireless mesh networks," *Wireless Personal Communications*, vol. 95, no. 4, pp. 4983–5001, 2017.

[20] S. Gringeri, N. Bitar, and T. Xia, "Extending software defined network principles to include optical transport," *IEEE Communications Magazine*, vol. 51, no. 3, pp. 32–40, 2013.

[21] A. Tzanakaki, M. Anastasopoulos, D. Simeonidou et al., "5G infrastructures supporting end-user and operational services: The 5G-xhaul architectural perspective," in *Proceedings of the 2016 IEEE International Conference on Communications Workshops, ICC 2016*, pp. 57–62, Malaysia, May 2016.

[22] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack, "Minizinc: Towards a standard cp modelling language," in *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, pp. 529–543, Springer, 2007.

[23] P. V. Hentenryck, *The OPL Optimization Programming Language*, MIT Press, Cambridge, MA, USA, 1999.

[24] A. Mesodiakaki, E. Zola, and A. Kassler, "Joint user association and backhaul routing for green 5G mesh millimeter wave backhaul networks," in *Proceedings of the 20th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM 2017*, pp. 179–186, USA, November 2017.

[25] W. Kim, "Dual connectivity in heterogeneous small cell networks with mmwave backhauls," *Mobile Information Systems*, vol. 2016, Article ID 3983467, 14 pages, 2016.

[26] H. Ogawa, G. K. Tran, K. Sakaguchi, and T. Haustein, "Traffic adaptive formation of mmwave meshed backhaul networks," in *Proceedings of the 2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, pp. 185–191, 2017.

[27] G. K. Tran, H. Shimodaira, R. E. Rezagah, K. Sakaguchi, and K. Araki, "Dynamic cell activation and user association for green 5G heterogeneous cellular networks," in *Proceedings of the 26th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC 2015*, pp. 2364–2368, China, September 2015.

[28] M. Miozzo, L. Giupponi, M. Rossi, and P. Dini, "Switch-on/off policies for energy harvesting small cells through distributed q-learning," in *Proceedings of the Wireless Communications and Networking Conference Workshops (WCNCW)*, IEEE, pp. 1–6, 2017.

[29] T. Bai, A. Alkhateeb, and R. W. Heath, "Coverage and capacity of millimeter-wave cellular networks," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 70–77, 2014.

[30] W. Feng, Y. Wang, D. Lin, N. Ge, J. Lu, and S. Li, "When mmwave communications meet network densification: a scalable interference coordination perspective," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 7, pp. 1459–1471, 2017.

[31] S. González, A. de la Oliva, X. Costa-Pérez et al., "5G-Crosshaul: an SDN/NFV control and data plane architecture for the 5G integrated fronthaul/backhaul," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1196–1205, 2016.

[32] G. Secinti, P. B. Darian, B. Canberk, and K. R. Chowdhury, "SDNs in the sky: robust end-to-end connectivity for aerial vehicular networks," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 16–21, 2018.

[33] D. Sajjadi, M. Tanha, and J. Pan, "A comparative study of channel switching latency for conventional and SDN-based routing in multi-hop multi-radio wireless mesh networks," in *Proceedings of the 13th IEEE Annual Consumer Communications and Networking Conference, CCNC 2016*, pp. 330–334, USA, January 2016.

[34] A. Betzler, F. Quer, D. Camps-Mur, I. Demirkol, and E. Garcia-Villegas, "On the benefits of wireless SDN in networks of constrained edge devices," in *Proceedings of the 2016 European Conference on Networks and Communications, EUCNC 2016*, pp. 37–41, Greece, June 2016.

[35] J. Núñez-Martnez et al., "Wisehaul: an SDN-empowered wireless small cell backhaul testbed," in *Proceedings of 2016 IEEE 17th International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, IEEE, pp. 1–3, 2016.

[36] R. Santos, H. Ogawa, G. K. Tran, K. Sakaguchi, and A. Kassler, "Turning the knobs on openflow-based resiliency in mmwave small cell meshed networks," in *Proceedings of the Globecom Workshops (GC Wkshps)*, IEEE, pp. 1–5, 2017.

[37] D. Giatsios, K. Choumas, P. Flegkas, T. Korakis, and D. Camps-Mur, "D5.1 Evaluation of SDN functionalities over heterogeneous RAN technologies at UTH testbed," 5G-Xhaul Project, 2017.

[38] J. Mack and P. Cabrol, "EdgeLink™ mmw mesh transport experiments in the berlin 5G-crosshaul testbed," in *Proceedings of the 2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, IEEE, pp. 1–6, Farmingdale, NY, USA, May 2017.

[39] K. Choumas, D. Camps-Mur, J. Aleixendri et al., "D5.3 demonstration and evaluation of the 5G-xhaul integrated prototype," *5G-Xhaul Project*, 2018.

[40] "Terragraph, solving the urban bandwidth challenge," 2018, https://www.terragraph.com.

[41] Y. Niu, Y. Li, D. Jin, L. Su, and A. V. Vasilakos, "A survey of millimeter wave communications (mmWave) for 5G: opportunities and challenges," *Wireless Networks*, vol. 21, no. 8, pp. 2657–2676, 2015.

[42] A. Artemenko, A. Maltsev, A. Mozharovskiy, A. Sevastyanov, V. Ssorin, and R. Maslennikov, "Millimeter-wave electronically steerable integrated lens antennas for wlan/wpan applications," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 4, pp. 1665–1671, 2013.

[43] I. Uchendu and J. R. Kelly, "Survey of beam steering techniques available for millimeter wave applications," *Progress in Electromagnetics Research B*, vol. 68, pp. 35–54, 2016.

[44] R. Santos, H. Ghazzai, and A. Kassler, "Optimal steerable mmwave mesh backhaul reconfiguration," in *Proceedings of the GLOBECOM 2018 - 2018 IEEE Global Communications Conference*, IEEE, pp. 1–7, Abu Dhabi, United Arab Emirates, December 2018.

[45] R. Santos and A. Kassler, "Small cell wireless backhaul reconfiguration using software-defined networking," in *Proceedings of the 2017 IEEE Wireless Communications and Networking Conference, WCNC 2017*, San Francisco, CA, USA, March 2017.

[46] A. Patro and S. Banerjee, "Outsourcing coordination and management of home wireless access points through an open api," in *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM)*, IEEE, 2015.

[47] T. Tsukizawa, N. Shirakata, T. Morita et al., "A fully integrated 60GHz CMOS transceiver chipset based on WiGig/IEEE802.11ad with built-in self calibration for mobile applications," in *Proceedings of the 2013 60th IEEE International Solid-State Circuits Conference, ISSCC 2013*, pp. 230-231, USA, February 2013.

[48] K. Takinami et al., "Design and experimental evaluation of 60ghz multiuser gigabit/s small cell radio access based on IEEE 802.11ad/WiGig," *IEICE Transactions on Communications*, vol. E100.B, no. 7, pp. 1075–1085, 2017.

[49] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer, "IEEE 802.11ad: Directional 60 GHz communication for multi-gigabit-per-second Wi-Fi," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 132–141, 2014.

[50] T. Visentin, *Reflectarray antennas with high gain and high bandwidth in the 60 ghz domain [Master, thesis]*, Technische Universität, Berlin, Germany, 2014.

[51] H. C. Yu, G. Quer, and R. R. Rao, "Wireless sdn mobile ad hoc network: from theory to practice," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, pp. 1–7, May 2017.

[52] M. Rademacher, F. Siebertz, M. Schlebusch, and K. Jonas, "Experiments with openflow and IEEE802. 11 point-to-point links in a WMN," in *Proceedings of the International Conference on Wireless and Mobile Communications (ICWMC)*, 2016.

[53] P. Danielis, G. Dán, J. Gross, A. Berger, and G. Dán, "Dynamic flow migration for delay constrained traffic in software-defined networks," in *Proceedings of the IEEE Global Communications Conference (Globecom 2017)*, 2017.

Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 1684906, 12 pages
https://doi.org/10.1155/2019/1684906

Research Article

A LoRaWAN Testbed Design for Supporting Critical Situations: Prototype and Evaluation

Jorge Navarro-Ortiz[1], **Juan J. Ramos-Munoz**[1], **Juan M. Lopez-Soler**[1], **Cristina Cervello-Pastor**[2] and **Marisa Catalan**[3]

[1]*Department of Signal Theory, Telematics and Communications, University of Granada (UGR), Granada, Spain*
[2]*Department of Network Engineering, Universitat Politècnica de Catalunya (UPC), Castelldefels, Spain*
[3]*Fundació i2CAT, Barcelona, Spain*

Correspondence should be addressed to Jorge Navarro-Ortiz; jorgenavarro@ugr.es

Received 29 November 2018; Accepted 7 February 2019; Published 21 February 2019

Academic Editor: Laurie Cuthbert

The Internet of Things is one of the hottest topics in communications today, with current revenues of $151B, around 7 billion connected devices, and an unprecedented growth expected for next years. A massive number of sensors and actuators are expected to emerge, requiring new wireless technologies that can extend their battery life and can cover large areas. LoRaWAN is one of the most outstanding technologies which fulfill these demands, attracting the attention of both academia and industry. In this paper, the design of a LoRaWAN testbed to support critical situations, such as emergency scenarios or natural disasters, is proposed. This self-healing LoRaWAN network architecture will provide resilience when part of the equipment in the core network may become faulty. This resilience is achieved by virtualizing and properly orchestrating the different network entities. Different options have been designed and implemented as real prototypes. Based on our performance evaluation, we claim that the usage of microservice orchestration with several replicas of the LoRaWAN network entities and a load balancer produces an almost seamless recovery which makes it a proper solution to recover after a system crash caused by any catastrophic event.

1. Introduction

The Internet of Things (IoT) is one of the hottest topics in communications today. Although previous forecasts may have overestimated the growth of connected IoT-devices, it is clear that the current and short-term market revenues are impressive: from $151B in 2018 up to $1,567B by 2025. Current IoT-devices vary from 6 to 9 billion devices (e.g., the current number of IoT devices in 2018 is 7 billion, according to IoT-analytics [1]) whereas forecasts estimate from 20 to 30 billion IoT devices by 2020 (e.g., Ericsson figure is 28 billion by 2021 [2]).

Many of the IoT services follow the category of massive Machine-Type Communications (mMTC), one of the three major 5G use cases (in addition to enhanced mobile broadband and ultrareliable and low latency MTC). Since mMTC communications assume a massive number of devices, in most of the cases battery-powered and in a high number of locations and environments, its main requirements are low-power communications and a wide range coverage.

Two main technologies which fulfill these requirements are being used for these applications: cellular evolution and Low Power Wide Area Networks (LPWAN).

Regarding cellular evolution, the Third Generation Partnership Project (3GPP) has tried to adapt the existing mobile standards for the requirements of IoT devices. In this way, cellular IoT standards utilize the existing mobile network infrastructures in an effort to integrate both worlds. Some of the main cellular technologies for IoT are Extended Coverage Global System for Mobile communications (EC-GSM), LTE Cat-0 (new low complexity Long Term Evolution device, defined in 3GPP Release 12), LTE-M, and narrow-band IoT (NB-IoT). NB-IoT addresses the specific requirements of mMTC but, unlike LTE Cat-0 and LTE-M devices, requires a specific frequency band different from those used for LTE or LTE-Advanced.

To the contrary of cellular technologies, LPWAN technologies have been born with IoT requirements from the very beginning. Previous attempts such as local or mesh networks can accommodate part of IoT services, such as low battery consumption and optimization for low data rates but are not intended for global coverage. Some of the most popular LPWAN technologies are LoRaWAN, SigFox, RPMA, and NWave. They offer long range (up to several tens of kilometers), very low power consumption (years of battery operation), and very low bandwidth (tens of kbps) and utilize license-exempt frequency bands. Another advantage of LPWANs is that they require a much lower investment compared to mobile networks, allowing new players to compete with current Mobile Network Operators (MNOs). For this reason, many MNOs (e.g., KPN, Orange, SK Telecom, Bouygues Telecom, Swisscom, and SoftBank) have started to deploy LoRaWAN (Long-Range Wide Area Network) to complement their current cellular networks deployments.

In this article, we propose the design of a LoRaWAN testbed for supporting critical situations, i.e., an IoT testbed that shall be able to automatically recover if part of its network infrastructure is destroyed. Since, in the case of LoRaWAN, the radio equipment is cheap and can be easily replaced, we will focus on the core network infrastructure.

This testbed will be integrated with the demonstrator from the 5G-City [3] project. 5G-City is a Spanish coordinated research project among five universities and research centres (Universitat Politècnica de Catalunya, Universidad de Granada, Fundació i2CAT, Universidad Carlos III, and Universidad del Pais Vasco), which aims at providing an adaptive management of 5G services to support critical events in cities. In that sense, 5G-City is focused on one of the most difficult situations for current and future communications systems: unexpected events that affect a relatively large amount of mobile users which are concentrated in a small area, such as traffic jams due to congestion or accidents, disasters, or any other emergency situations that may affect a large number of users. An overview of the 5G-City demonstrator is shown in Figure 1.

One of the objectives of the 5G-City research project is the design of a virtualized 5G network for massive IoT and broadband experience. Within the 5G context, different wireless technologies will coexist as technological alternatives for the interconnection between information producers and consumers.

In the case of IoT, one of the wireless technologies that will be included in the 5G-City demonstrator will be the LoRaWAN network prototype proposed in this paper. For that purpose, this prototype will be integrated with the 5G-City 4G/5G network demonstrator following our previous work in [4].

The literature that defines the state of the art regarding the evaluation of LoRaWAN and LoRa testbeds in real deployments is very rich in quality and quantity. A number of papers have been devoted to measuring LPWAN performance metrics in both indoor and outdoor deployments, as well as in rural, urban, and suburban scenarios.

Almost invariably all of these works focus on coverage measurements. Particularly, different evaluations are reported, ranging from covered distance [5–8] to percentage of packets successfully delivered [9], in different scenarios and operation conditions, i.e. different payload sizes [10] and different impairments or spreading factors [11, 12], in high-density urban environments with many multifloor buildings [13].

In [14] a comparison of different LoRaWAN testbeds is provided in terms of several metrics (RSSI, SNR, and distances covered). Reference [15] evaluates the packet transmission time for different spreading factors. Additionally, [16] evaluates the average LoRaWAN throughput as a function of the spreading factor for different payload sizes.

However, none of the reported papers evaluates the LoRaWAN resilience dimension under a critical situation, i.e., the elapsed time for recovery and packet losses impact after a system crash. Note that these quantitative evaluations will definitively help to determine the LoRaWAN suitability for IoT deployments under critical circumstances.

The main objective of this paper is the proposal of a self-healing LoRaWAN network architecture in order to provide resilience under critical situations such as earthquakes, fires, or hurricanes. Under such conditions, part of the network equipment may become faulty. By virtualizing the different entities in the core network, i.e., converting them into VNFs (Virtual Network Functions), we will be able to reduce costs, increase flexibility, and provide resilience. We have implemented different options for the virtualization of the LoRaWAN core network entities which will be compared in terms of time for recovery, packet losses, and resource usage.

For this objective, the rest of the article is organized as follows. Section 2 provides a LoRaWAN technology overview, including main transmission characteristics and architecture. Section 3 describes LoRaWAN implementation issues. Particularly, both NFV (Network Function Virtualization) orchestration options using microservices and virtual machines are explained. Section 4 explains our testbed. It includes the hardware setup and the software platform details. Section 5 identifies the evaluated four use cases, and it includes the obtained results. Finally, Section 6 sums up the paper and provides the main conclusions.

2. LoRaWAN Overview

LoRaWAN [17] is a standardized Low Power Wide Area Network (LPWAN) which uses LoRa [18] or FSK modulations. LoRa is a proprietary modulation owned by the French company Semtech. This modulation is based on CSS (Chirp Spread Spectrum) and it features the same low-power characteristics of FSK but increases the coverage range. The bandwidth of a LoRa signal can be 125, 250, or 500 kHz, and different spreading factors (SF) can be used to achieve a trade-off between data rate and coverage.

The spreading factor is defined as $SF = \log_2(R_C/R_S)$, where R_C is the chip rate and R_S is the symbol rate. Since the chip rate is constant for a fixed bandwidth ($R_C = BW$ chips/sec, where BW is the bandwidth), a higher SF implies a lower data rate but increases the transmission range due to a higher robustness. Codes from different SFs are orthogonal,

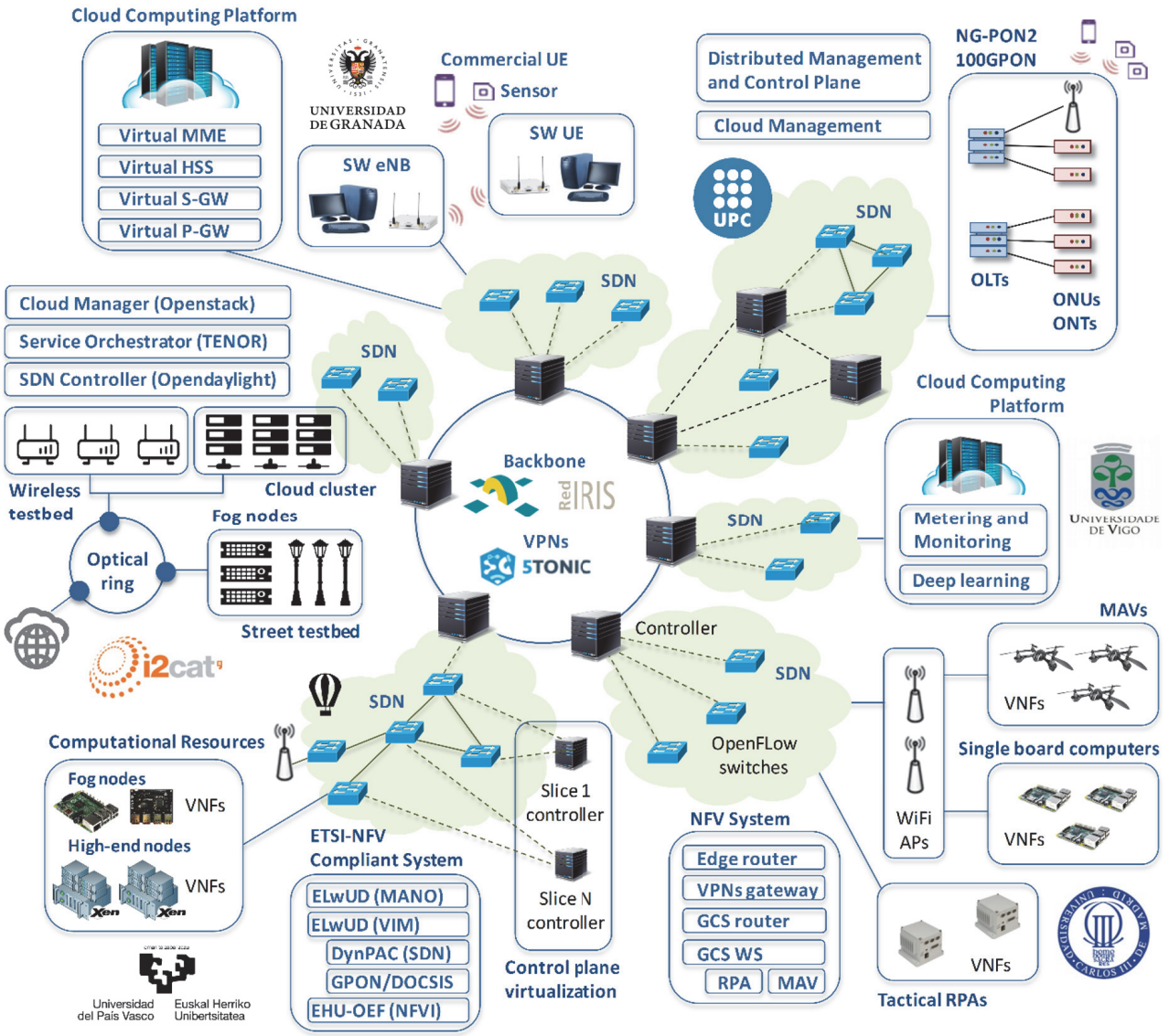


Figure 1: Testbed of the 5G-City research project [3].

so multiple frames can be simultaneously transmitted on the same channel as long as they utilize different SFs.

LoRaWAN is an open standard managed by the LoRa Alliance. LoRaWAN defines the Medium Access Control (MAC) layer on top of the LoRa physical layer. It also defines the system architecture.

The MAC layer utilizes a duty cycle to reduce the probability of collisions in a simple and hardware-efficient manner. Depending on regional regulations [19], this duty cycle can be, e.g., 1%, meaning that a LoRaWAN node can only transmit 1% of the time, thus affecting its maximum data rate. This limitation makes the SF selection have a high impact on the transmission rate since it is determining the Time on Air (ToA). ToA can be computed as

$$ToA = T_{preamble} + T_{payload} = T_{preamble} + T_S \times n_{payload}, \quad (1)$$

where $T_{preamble}$ and $T_{payload}$, respectively, are the preamble and payload transmission time, whereas $T_S = 2^{SF}/BW$ is the symbol period.

In the case of European regulations, the duty cycle is 1% and the combination between SFs and bandwidths produces the different data rates (DR) included in Table 1. This table also includes the ToA and the minimum time between consecutive frames (i.e., to fulfill the duty cycle limitation) assuming an application payload of 12 bytes.

The architecture of a LoRaWAN network is based on a star topology, as shown in Figure 2. This figure shows the different entities in a common LoRaWAN deployment. It includes the Radio Access Network (RAN) and the Core Network (CN). The RAN is composed of nodes and gateways, which act as base stations forwarding the frames received from the radio interface to the core network entities. The CN is composed

Table 1: Parameters for the different LoRaWAN DRs.

DR	SF	BW (kHz)	data rate (bps)	ToA (ms)	Time between frames (s)
0	SF12	125	250	1482.8	148.3
1	SF11	125	440	823.3	82.3
2	SF10	125	980	411.6	41.2
3	SF9	125	1760	205.8	20.6
4	SF8	125	3125	113.2	11.3
5	SF7	125	5470	61.7	6.2
6	SF7	250	11000	30.8	3.1

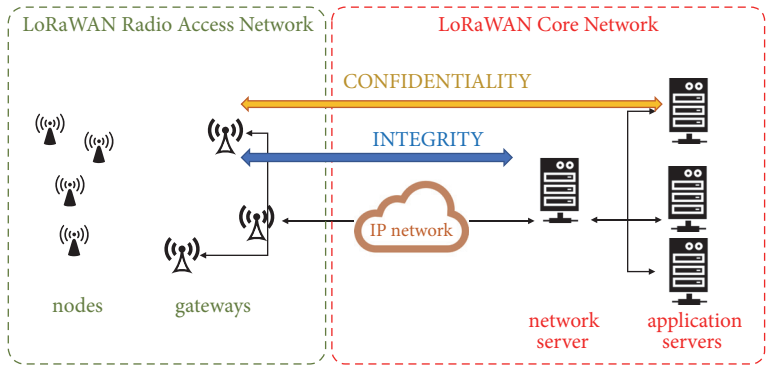


Figure 2: LoRaWAN network architecture.

of an IP network and two types of servers: network and application servers.

All the frames forwarded from the gateways to the network server are integrity protected (thanks to a Message Integrity Code (MIC) generated with a network session key, `NwkSKey`), whereas privacy is kept up to the application server (thanks to the encryption of the payload with an application session key, `AppSKey`). The network server sends packets to the appropriate application server, which handles the customer application and processes the customer data. Since this architecture achieves end-to-end security, the IP network infrastructure can be from a different provider.

In order to exchange the required session keys (`NwkSKey` and `AppSKey`), the LoRaWAN standard defines two activation methods when the node is attached to the network: Activation by Personalization (ABP) and Over-the-Air Activation (OTAA).

In the first case, the developer shall include this information in both the nodes (i.e., stored in their firmware) and the servers. Thus, no signalling is needed. In the second case, the node shall send a `JoinRequest` frame with a device identifier (`DevEUI`), an application identifier (`AppEUI`), and a random challenge (`DevNonce`). Upon receiving this frame, the gateway shall send a `JoinResponse` frame with the device address (`DevAddr`), a network identifier (`NetID`), and another random challenge (`AppNonce`). With these data and a preshared key (`AppKey`), both the node and the servers are able to derive the same `NwkSKey` and `AppSkey`, which are used for the subsequent transmissions. Both activation types are summarized in Figure 3.

LoRaWAN allows nodes to have bidirectional communications with gateways although asymmetric, since uplink transmissions (from nodes to gateways) are strongly favored. Three types of devices are defined (classes A, B, and C) with different capabilities. Class A is the most energy efficient and must be supported by all nodes. Class A nodes use pure ALOHA for uplink access, and they can only receive a downlink frame after a successful uplink transmission. This class is intended for battery-operated sensors. Class B nodes utilize beacons sent from the gateway to determine whether they have to receive downlink frames or not, using scheduled receive windows at a predictable time without the need of successful uplink transmissions. This class is intended for battery-operated actuators. Finally, class C nodes are always listening to the radio interface except when they are transmitting. Due to its power consumption, class C is intended for main powered actuators. As commented, class A is mandatory for all LoRaWAN nodes, and the three classes may coexist in the same network.

3. Implementation of a Virtualized LoRaWAN Network Architecture

This section presents the two options which have been implemented for the virtualization and the automatic orchestration of a LoRaWAN network. As commented, the first proposal is based on microservices using the Kubernetes platform. The second implementation is based on virtual machines, using OpenStack along with its modules for the automatic deployment of the LoRaWAN services.

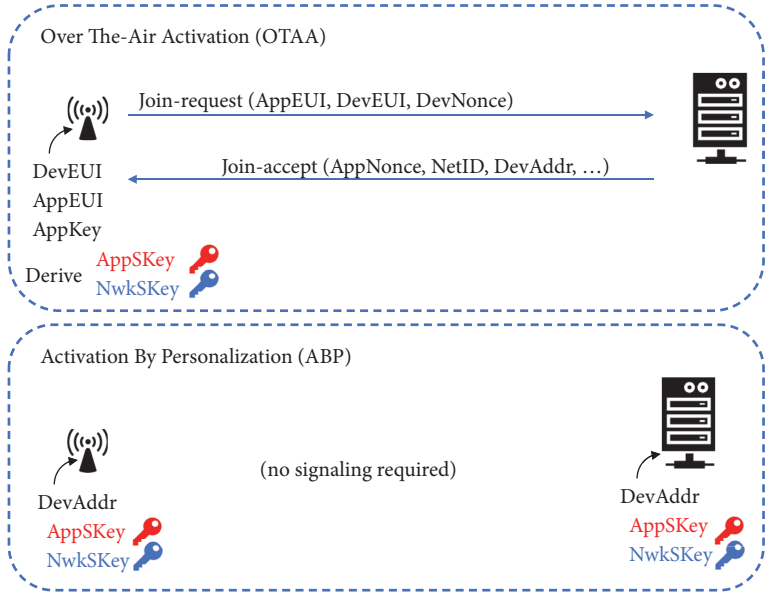


Figure 3: LoRaWAN activation types.

3.1. LoRaWAN NFVs Orchestration Using Microservices. Due to the architecture of a LoRaWAN network and the lightweight functionalities of the different entities, they can be deployed as microservices. With the success of containerization technologies, such as Docker [20], microservices can be realized as containers that result to be extremely fast to start up and can be easily deployed.

In order to implement a LoRaWAN network which supports autorecovery in the case of an emergency situation such as an earthquake, fire, hurricane, or any other situation that may destroy part of the core network infrastructures, the usage of a microservice orchestration platform may suit these requirements due to its efficiency in terms of CPU, memory, and storage consumption compared to virtual machines [21–23].

In particular, we propose to utilize the Kubernetes platform [24] (also named K8S) for the container orchestration. Figure 4 presents the proposed architecture. The details about the hardware and software that implement the 5G-City Kubernetes cluster are described in Section 4.

The LoRaWAN network and application servers are based on the LoRaWAN Server Project [25], an open-source project that provides the components for building LoRaWAN networks. Figure 5 summarizes the interaction between the different dockers and the exposed services to allow external connectivity. The docker images that implement the LoRaWAN network and application servers, along with the required services, have been modified and stored in a personal repository to support the communication with Kubernetes.

3.2. LoRaWAN NFVs Orchestration Using Virtual Machines. In the case of the implementation using virtual machines, due to its popularity, large community, high availability of modules, and being open-source, we have opted for using OpenStack. Our OpenStack testbed is based on the Rocky release and has been installed using the DevStack scripts.

Apart from the primary OpenStack services (Keystone for the identity service, Glance for the image service, Nova for the provision of compute instances or virtual machines, Neutron for network connectivity, and Horizon for the dashboard user interface), Heat and Ceilometer have been installed for the orchestration and the telemetry service. The chosen hypervisor is KVM (Kernel-based Virtual Machine) using QEMU Copy-on-write (qcow2) as the virtual machine image format.

For comparison purposes, the virtual machine images are based on CentOS 7 cloud images, similarly to the Kubernetes deployment. For the same reason, the LoRaWAN network and application servers have also been installed using the LoRaWAN Server Project [25]. Our prototype using OpenStack is depicted in Figure 6.

For our OpenStack testbed, we have developed a module which automatically starts the provision of resources for a new instance, which is then launched. This procedure is triggered once that the original instance is destroyed due to, e.g., a catastrophic event. The module utilizes the API provided by the heat orchestrator and the metrics from ceilometer.

4. Testbed Proposal

This section describes both the hardware and software used for the design of our LoRaWAN network prototype.

4.1. Hardware Setup. The radio access network of our LoRaWAN network prototype is composed of 5 gateways and 12 nodes. The gateways are LiteGateways from iMST [26], which utilize one Raspberry Pi connected to an iMST ic880A LoRaWAN concentrator and 868MHz antenna (see Figure 7).

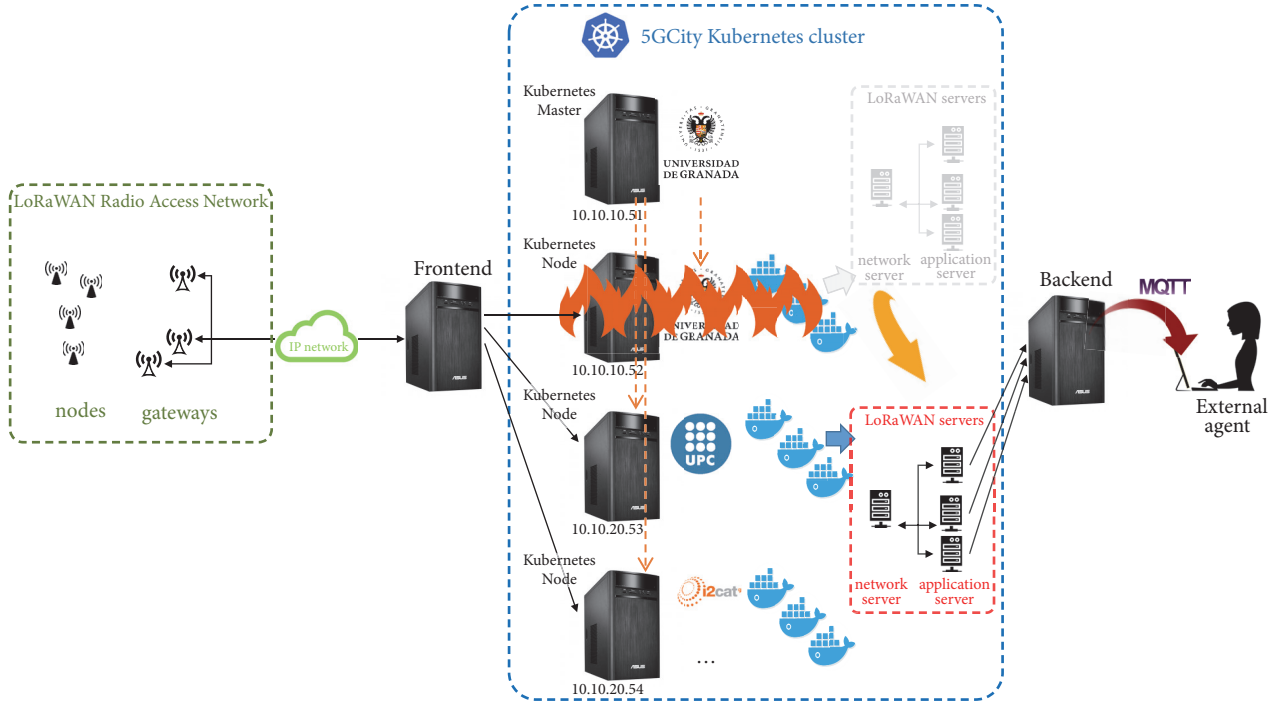


Figure 4: Proposed network architecture based on microservices.

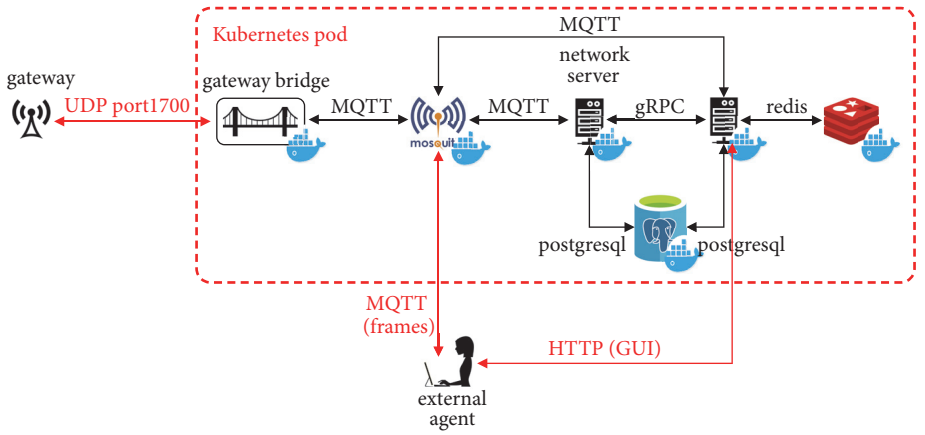


Figure 5: Kubernetes pod (group of colocated containers that are tightly coupled and need to share resources) for LoRaWAN deployment.

The nodes are TTGO-LoRa32 devices (see Figure 8) which are based on the ESP32 microcontroller with a Semtech's SX1276 LoRa transceiver.

Our core network prototype is composed of two servers with an Intel Core i7-7820X CPU (8 cores operating at 3.6 GHz) and 32 GB of RAM located in University of Granada (UGR). These two servers act as the master node of the Kubernetes cluster and the first worker node (minion-1). In addition, we have other two servers located in Barcelona at Universitat Politècnica de Catalunya (UPC) (based on a six-core Intel i7-5820K operating at 3.3 GHz) and Fundació i2CAT (based on a Intel Xeon E312xx (Sandy Bridge) with 32 cores operating at 2.5 GHz), which acts as the second and third worker nodes (minion-2 and minion-3), respectively.

The IP network is a direct Ethernet connection between the gateways and the master server located at University of Granada, which also implements the frontend using one NGINX ingress controller.

4.2. Software Configuration. The nodes are programmed using the Arduino framework, based on the IBM's LMIC library [27]. Before transmitting a LoRaWAN frame, the nodes connect to a server (named *experiment manager*) to ask whether it shall transmit or not, thus allowing us to control the network load. In addition, the transmission parameters are also commanded from the server. These parameters include the spreading factor and the time between frames, which is composed of a constant term and a random term.

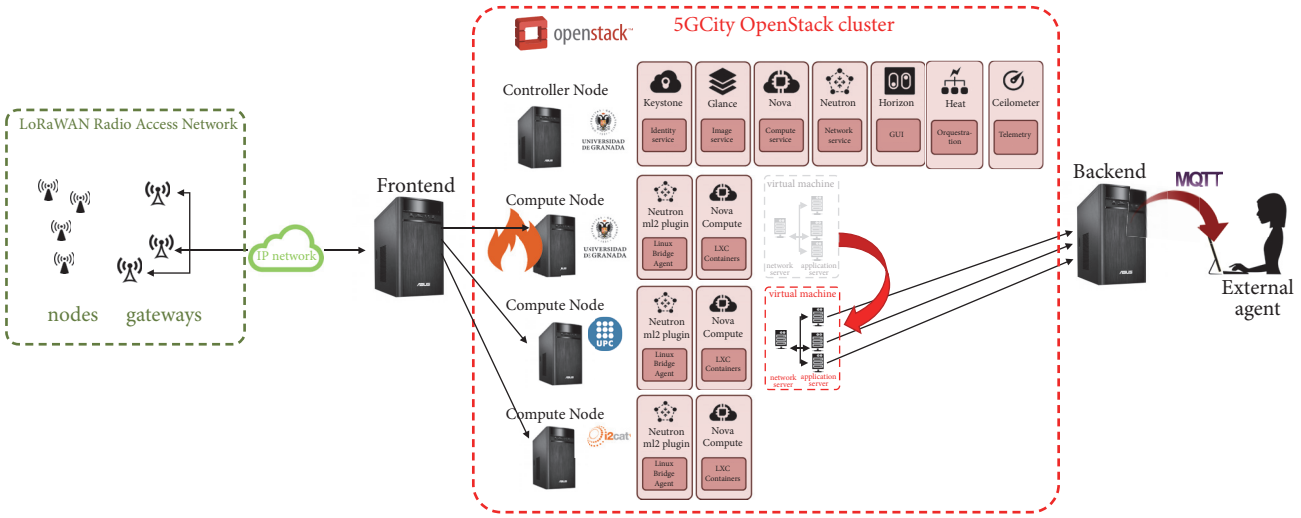


Figure 6: Network architecture based on OpenStack and virtual machines.

Figure 7: LoRaWAN gateways used in the proposed testbed.

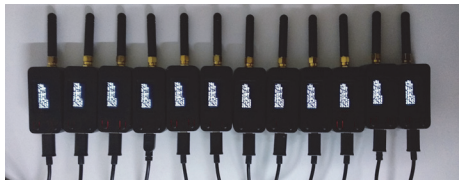


Figure 8: LoRaWAN nodes used in the proposed testbed.

Figure 9: Detail of a LoRaWAN node.

Figure 9 includes an example of the "TTGO #1" node, which is connected to our server through the Wi-Fi network "5gcity-testbed" with the IP address shown. The last transmission parameters were the usage of spreading factor 7 (SF7) and a time between frames with a fixed part (f) of 10 seconds plus a random part (r) between 0 and 10 seconds. It shall be noted that the ESP32 contains the hardware to generate true random numbers whenever an RF subsystem is running (i.e., Bluetooth or Wi-Fi is enabled) [28].

In order to avoid that the connection between the nodes and the experiment manager may impact the results of the experiments, e.g., due to additional delays, all the nodes connect to the Wi-Fi network only when they are switched on; i.e., there is no connection establishments during the experiments. The mean response time, between the request from the node and the response from the server, has been measured around 85 ms, which is much lower than the time between LoRaWAN frames (which has a minimum value of 6.2 seconds according to Table 1 for SF7 and 125 kHz and has never been lower than 10 seconds in the performed experiments). Besides, the server collects stats from the nodes since it knows when they are going to transmit a frame and with which parameters.

As previously stated, the gateways are based on the Raspberry Pi platform with the iC880A concentrator. The software is based on the reference gateway implementation from TTN-ZH (Zurich community of The Things Network) [29], which has been configured to use our own LoRaWAN network servers. The gateways are also connected to the experiment manager, which collects the logs related to their LoRaWAN activity.

Our Kubernetes cluster is based on Kubernetes version 1.5.2. For portability and reproducibility purposes, both master and worker nodes have been virtualized and executed using VirtualBox version 5.2.22. The host operating system is Ubuntu Server 16.04.05 (64 bits), and the guest operating system is CentOS 7.5.1804 (64 bits) running with 1 core and 2 GB of RAM. We utilize Vagrant version 2.1.5 in order to automatize the virtual machines deployment, and Ansible version 2.6.4 to automatize the installation and configuration of the required packages.

In order to simplify the requirements for connecting the worker and master nodes, a VPN was created using OpenVPN version 2.4.6 using client certificate authentication. In

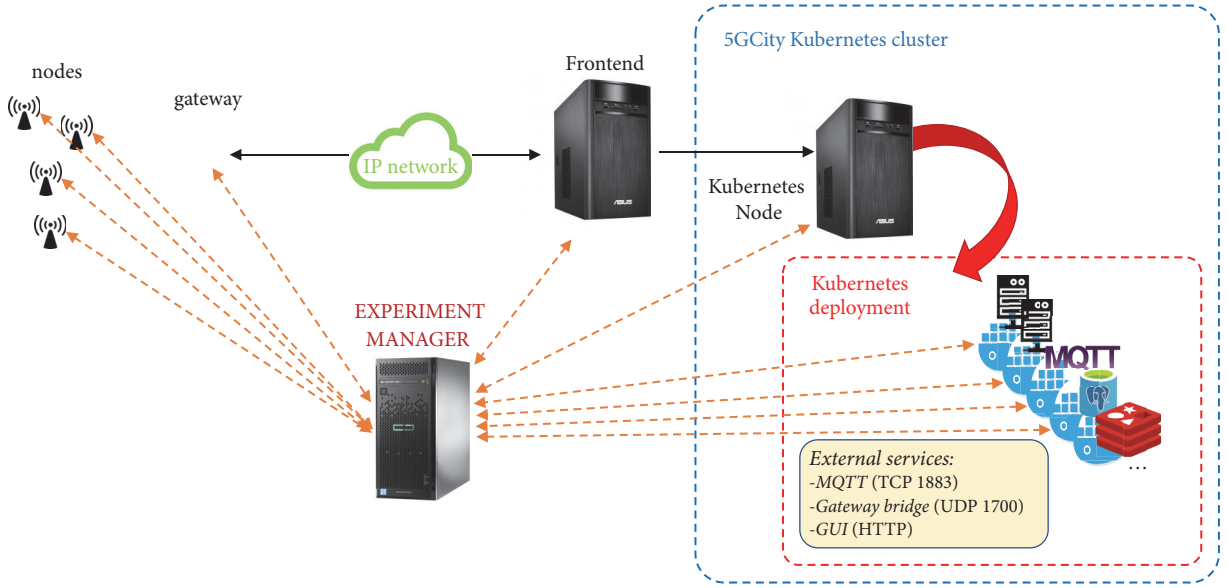


Figure 10: Experimentation testbed.

this way, only the master node is required to have a public IP address. In addition, it eliminates the possibility of issues due to firewall rules. In our VPN, the master node acts as the OpenVPN server whereas the worker nodes act as OpenVPN clients. The workers also collect tcpdump traces on the TCP/UDP ports that are used by services related to the LoRaWAN deployment, which are later sent to the experiment manager.

The dockers that implement the LoRaWAN deployment are also connected to the experiment manager, which starts/stops the services depending on the experiment and collects the required logs and stats.

The experiment manager connects to the different entities using SSH connections, which enable us to execute commands (e.g., to start or to stop a particular service), to upload files (e.g., a configuration file), or to download files (e.g., logs, tcpdump traces, or stats). In the case of the LoRaWAN nodes, they connect to the experiment manager after transmitting one LoRaWAN frame to request the transmission parameters for the next frame. If the connection manager commands the node not to transmit, it will ask again after 10 seconds. Figure 10 shows a simplified view of the testbed for experimentation.

The NGINX ingress controller has been configured with the default values for load balancing the different external services, i.e., the UDP port 1700 (which is used by the lorawan-gateway-bridge container), the TCP port 1883 (which is used by the MQTT broker), and the TCP port 443 (which is used for the HTTPS-based GUI). The main parameters are *max_fails*=3 and *fail_timeout*=30s, meaning that the backup server will be used after the main server has failed to respond to at least 3 packets in a period of 30 seconds.

Based on this experiment manager, we have developed a framework based on scripting to generate different scenarios for both the radio access network and the core network and to automatically collect statistics, which will be used in the next section for the experimental evaluation.

5. Use Cases and Experimental Results

Considering the two options that we have followed for the virtualization of the LoRaWAN network entities, i.e., using microservices (Kubernetes) and virtual machines (OpenStack), the following use cases have been tested:

(i) *UC1*: Kubernetes deployment with one replica and default parameters: the chosen version of Kubernetes considers a deployment (i.e. a set of pods and services available externally) to be unavailable after five minutes. Thus, a new replica will be deployed after this time.

(ii) *UC2*: Kubernetes deployment with one replica and an eviction pod timeout of 30 seconds. Instead of waiting the default 300 seconds, this use case attempts to react faster to possible unavailability of worker nodes due to a catastrophic situation. We did not select a lower timeout value (e.g. 3 seconds) in order to avoid new replica deployments due to temporary network fluctuations.

(iii) *UC3*: Kubernetes deployment with two replicas: in this case, the replica at UGR is chosen initially, and the replica available at UPC/i2CAT will be used for backup.

(iv) *UC4*: OpenStack deployment with one replica and default parameters: similar to the first use case but using the OpenStack platform.

The reason of selecting these four use cases is twofold. First, testing and comparing different configurations using

microservices have been proved to be suitable for the deployment of LoRaWAN servers. For that purpose, we want to compare the usage of Kubernetes with default values (UC1, with a timeout of 300 seconds), with two replicas in order to achieve a solution (almost) without service interruption (UC3) and an intermediate situation (UC2). Second reason is to compare both Kubernetes (with containers, UC1) and OpenStack (with virtual machines, UC4) with their default configurations.

Since we want to simulate a high-loaded IoT scenario, nodes will transmit 12-byte frames using SF7 and 125 kHz, leading to a minimum time between frames of 6.2 seconds due to the duty cycle (see Table 1). Since we want transmissions to be uncorrelated, the time between frames are composed of a fixed part, f=10 seconds, and a random part, r, which follows a random uniform distribution between 0 and 10 seconds. This also avoids the problem of collisions due to the simultaneous powering on of the nodes.

With these values, the average time between frames is 15 seconds, i.e., leading to a load of $12 nodes/(15 sec/frame/node) = 0.8$ frames/sec, which is similar to approximately 1000 nodes transmitting one frame every 20 minutes. These frames will be received by only one gateway, meaning that it will be high-loaded since other works (e.g., [30]) suggest that the maximum load that a LoRaWAN gateway can support without frame losses is around 0.1 frames/sec. This is the maximum load that can be generated with the real equipment in our testbed. It is left for future work to include a load testing tool which would allow us to evaluate the performance of our testbed under stress conditions (e.g., in [31], the authors emulate the load generated from 14,000 nodes).

In the proposed use cases, the main performance indicators are the recovery time, i.e., the time that elapses from the failure in one worker node until another worker node executes the pod with the LoRaWAN deployment, and the lost frames during that recovery. It shall be noted that the lost frames will depend on the gateway load, and the results shown in this section are given for the aforementioned load of 0.8 frames/sec. Additionally, we also want to show the different requirements, in terms of CPU and memory, between the usage of a Kubernetes cloud or OpenStack.

Figures 11 and 12 depict the main performance indicators related to the recovery of the LoRaWAN core network after an equipment failure due to, e.g., a catastrophic situation. All the use cases (UC1 to UC4) are included. These *box-and-whisker* charts include a box bounded by the first and third quartile and lines that extend to the minimum and maximum, respectively. The median is also included as the line that divides the box.

As shown, UC1 takes between 5 and 6 minutes (with an average of 321 seconds and an standard deviation of 13.6) to recover due to the default value of the pod eviction timeout (300 seconds). In the case of UC2, we have reduced this timeout to 30 seconds, leading to a recovery time of around one minute (average of 64 seconds with an standard deviation of 14.7). To conclude with the Kubernetes-based use cases, UC3 has an almost negligible recovery time. This is because two replicas are already executed, and the second

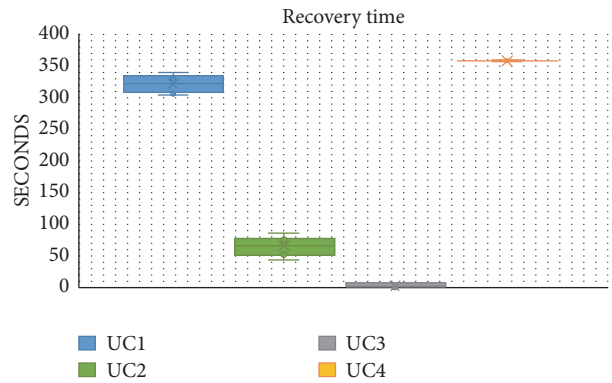


Figure 11: Recovery time after server failure.

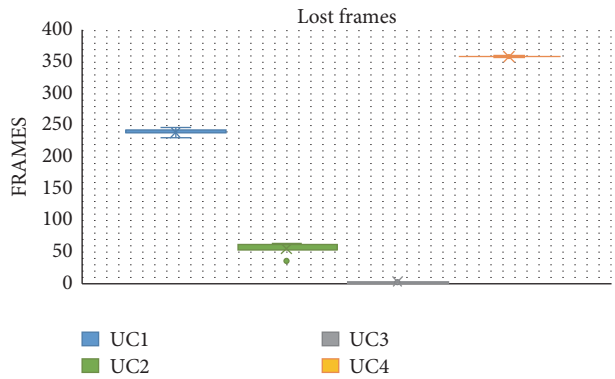


Figure 12: Frames lost during recovery.

one takes over when the first one fails. Since we utilized the default values for the NGINX frontend, only three packets are required to change from one replica to another. The rate of these packets depends on the transmissions from the LoRaWAN nodes and some periodic packets, being the recovery time of 4 seconds with a standard deviation of 2.4. As it was expected, the number of lost frames is approximately proportional to the recovery time.

In the OpenStack use case (UC4), the developed module waits 5 minutes (like the default timeout value for Kubernetes) before provisioning and launching the new instance, leading to a total recovery time of around six minutes (with an average of 358.1 seconds and a standard deviation of 0.72). As in the previous use cases, the number of lost frames is almost proportional to the recovery time.

Next, we compare the usage of resources of both options. For Kubernetes, we employed cAdvisor [32], a tool that provides the resource usage and performance characteristics of running containers. The resources used by the different containers that compose the Kubernetes deployment for LoRaWAN under a load of 0.8 packets/second are summarized in Table 2.

The results from Table 2 show that the CPU usage is almost negligible (lower than 0.01%) and the total memory usage of the LoRaWAN deployment is 39 MiB.

Table 2: Usage of resources for the containers of the LoRaWAN deployment.

Container	CPU %	MEM %	MEM (MiB)
lora-app-server	0.00	0.60	11.3
loraserver	0.00	0.40	7.6
lora-gateway-bridge	0.00	0.20	4.0
mosquitto	0.00	0.00	1.5
postgres	0.00	0.70	13.0
redis	0.00	0.00	1.6

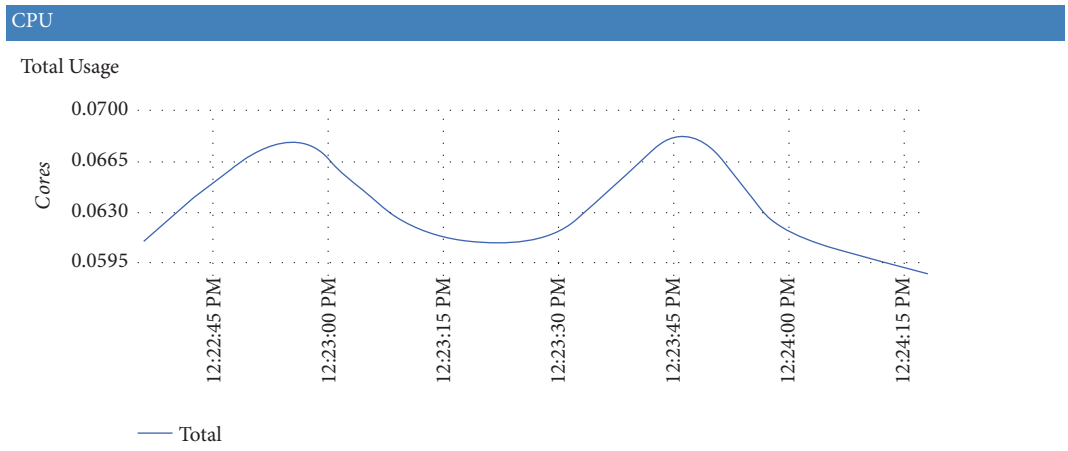


Figure 13: Total CPU usage including all Kubernetes processes.

Figure 13 presents the total CPU usage for one of the worker nodes of the Kubernetes cluster. As shown, Kubernetes (and docker in general) utilizes very few resources in terms of CPU, between 6% and 7% of one core. The main consumers are processes related to the management of the Kubernetes cloud (kubelet with 2.8%, dockerd-current with 0.5% and kube-proxy with 0.4%).

In terms of memory, around 1.6 GB are used by the worker node. The processes that reserve more memory are also related to Kubernetes (java with 392 MB, kubelet with 75.9 MB, dockerd-current with 51.5 MB, kube-proxy with 39.9 MB, and flanneld with 26.8 MB).

In the case of OpenStack, the total memory employed by the worker node is 8.45 GiB when no instances are deployed and 288 MiB more when one virtual machine with the LoRaWAN is executed. In terms of CPU, worker node consumes 1.28 CPU cores. This means that, in our given scenario, OpenStack requires more than 5 times the memory needed by Kubernetes and more than 18 times in terms of CPU consumption.

6. Conclusions

In this paper, we propose the usage of a microservices platform such as Kubernetes for the deployment of a LoRaWAN network infrastructure. Based on its orchestration capabilities, the proposed framework is able to support catastrophic situations and to rapidly recover from equipment failure in the core network. To evaluate the performance of this solution, a prototype testbed of a complete LoRaWAN network has been implemented. By using an experiment manager, we have been able to automatize the node traffic generation and the automatic recollection of stats, as well as the presence of failures. We have evaluated our implementation in terms of time to recover, lost frames, and resource usage. After the conducted evaluation, we claim that the usage of several replicas of the LoRaWAN core network entities and a load balancer, which automatically changes between servers in a fast and efficient way, produces an almost seamless recovery, what makes it a proper solution to recover after a system crash caused by any catastrophic event.

For future work, based on our previous analysis [33, 34], we plan to mathematically model the implemented VNFs in order to estimate the performance for a given configuration and to derive when to scale out by requesting more resources.

Data Availability

The data has been generated from live tests in our LoRaWAN testbed. Logs or any other information is available upon request to the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund (Projects TEC2016-76795-C6 and EQC2018-004988-P).

References

[1] K. L. Lueth, "State of the IoT 2018: Number of IoT devices now at 7B –market accelerating, 2018," https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[2] A. Nordrum, "Popular Internet of Things forecast of 50 billion devices by 2020 is outdated," *IEEE Spectrum's General Technology Blog*, 2016, http://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated.

[3] C. Cervello-Pastor, "5G-City: Flexible management of 5G services oriented to support urban critical situations, project TEC2016-76795-C6 of the Spanish Ministry of Economy, Industry and Competitiveness, 2017-2019".

[4] J. Navarro-Ortiz, S. Sendra, P. Ameigeiras, and J. M. Lopez-Soler, "Integration of LoRaWAN and 4G/5G for the Industrial Internet of Things," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 60–67, 2018.

[5] X. Xiong, K. Zheng, R. Xu, W. Xiang, and P. Chatzimisios, "Low power wide area machine-to-machine networks: Key techniques and prototype," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 64–71, 2015.

[6] A. Cenedese, A. Zanella, L. Vangelista, and M. Zorzi, "Padova smart city: an urban internet of things experimentation," in *Proceedings of the 15th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '14)*, pp. 1–6, Sydney, Australia, June 2014.

[7] T. Petrić, M. Goessens, L. Nuaymi, L. Toutain, and A. Pelov, "Measurements, performance and analysis of LoRa FABIAN, a real-world implementation of LPWAN," in *Proceedings of the 27th IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, PIMRC 2016*, pp. 1–7, Spain, September 2016.

[8] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios," *IEEE Wireless Communications Magazine*, vol. 23, no. 5, pp. 60–67, 2016.

[9] J. Petäjäjärvi, K. Mikhaylov, A. Roivainen, T. Hänninen, and M. Pettissalo, "On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology," in *Proceedings of the 14th International Conference on ITS Telecommunications, ITST 2015*, pp. 55–59, Denmark, December 2015.

[10] M. Aref and A. Sikora, "Free space range measurements with Semtech Lora™; technology," in *Proceedings of the 2014 2nd International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS-SWS)*, pp. 19–23, Odessa, Ukraine, September 2014.

[11] T. Wendt, F. Volk, and E. Mackensen, "A benchmark survey of long range (LoRaTM) spread-spectrum-communication at 2.45 GHz for safety applications," in *Proceedings of the 2015 16th IEEE Annual Wireless and Microwave Technology Conference, WAMICON 2015*, pp. 1–4, USA, April 2015.

[12] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia, and N. Strachan, "Evaluation of LoRa and LoRaWAN for wireless sensor networks," in *Proceedings of the 15th IEEE Sensors Conference, SENSORS 2016*, pp. 1–3, USA, November 2016.

[13] P. J. Radcliffe, K. G. Chavez, P. Beckett, J. Spangaro, and C. Jakob, "Usability of LoRaWAN technology in a central business district," in *Proceedings of the 2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pp. 1–5, Sydney, Australia, June 2017.

[14] J. M. Marais, R. Malekian, and A. M. Abu-Mahfouz, "LoRa and LoRaWAN testbeds: A review," in *Proceedings of the IEEE AFRICON 2017*, pp. 1496–1501, South Africa, September 2017.

[15] N. Vatcharatiansakul, P. Tuwanut, and C. Pornavalai, "Experimental performance evaluation of LoRaWAN: A case study in Bangkok," in *Proceedings of the 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, S. T. P. Chantamunee and S. Doung-in, Eds., pp. 1–4, Institute of Electrical and Electronics Engineers Inc., NakhonSiThammarat, Thailand, July 2017.

[16] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A Study of LoRa: long range & low power networks for the internet of things," *Sensors*, vol. 16, no. 9, article 1466, 2016.

[17] LoRaWAN™ 1.1 specification, "LoRa Alliance, Specification, Oct. 2017," https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.

[18] LoRa™ modulation basics, "Semtech Corporation, Application Note AN1200.22, May 2015," https://www.semtech.com/uploads/documents/an1200.22.pdf.

[19] LoRaWAN™ 1.1 regional parameters, "LoRa Alliance, Specification, Jan. 2018," https://lora-alliance.org/sites/default/files/2018-04/lorawantm_regional_parameters_v1.1rb_-_final.pdf.

[20] Docker, "Docker, enterprise-grade container platform," https://www.docker.com.

[21] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu, and W. Zhou, "A comparative study of containers and virtual machines in big data environment," in *Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 178–185, San Francisco, CA, USA, July 2018.

[22] M. Chae, H. Lee, and K. Lee, "A performance comparison of linux containers and virtual machines using Docker and KVM," *Cluster Computing*, 2017.

[23] A. M. Joy, "Performance comparison between Linux containers and virtual machines," in *Proceedings of the 2nd International Conference on Advances in Computer Engineering and Applications, ICACEA 2015*, pp. 342–346, India, March 2015.

[24] Google, "cadvisor (container advisor)," https://github.com/google/cadvisor.

[25] O. Brocaar, "LoRa server v2.3.0," https://www.loraserver.io/, 2018.

[26] IMST, "Lite gateway," https://shop.imst.de/wireless-modules/lora-products/36/lite-gateway-demonstration-platform-for-lora-technology.

[27] M. Kooijman, "IBM's LoRaMAC-in-C," https://github.com/matthijskooijman/arduino-lmic.

[28] "ESP32's random number generation," https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/system.html#_CPPv210esp_randomv.

[29] "TTN-ZH's iC880a-gateway," https://github.com/ttn-zh/ic880a-gateway/wiki.

[30] D. Bankov, E. Khorov, and A. Lyakhov, "On the limits of LoRaWAN Channel Access," in *Proceedings of the 2016 International Conference on Engineering and Telecommunication (EnT)*, pp. 10–14, 2016.

[31] Q. Zhou, K. Zheng, L. Hou, J. Xing, and R. Xu, "X-LoRa: An Open Source LPWA Network," http://arxiv.org/abs/1812.09012, 2019.

[32] Cloud Native Computing Foundation, "Kubernetes, production-grade container orchestration," https://kubernetes.io/.

[33] J. Prados-Garzon, P. Ameigeiras, J. J. Ramos-Munoz, P. Andres-Maldonado, and J. M. Lopez-Soler, "Analytical modeling for Virtualized Network Functions," in *Proceedings of the 2017 IEEE International Conference on Communications Workshops, ICC Workshops 2017*, pp. 979–985, France, May 2017.

[34] J. Prados-Garzon, J. J. Ramos-Munoz, P. Ameigeiras, P. Andres-Maldonado, and J. M. Lopez-Soler, "Modeling and Dimensioning of a virtualized MME for 5G mobile networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 4383–4395, 2017.

Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 4734821, 20 pages
https://doi.org/10.1155/2019/4734821

Research Article

An NFV-Based Energy Scheduling Algorithm for a 5G Enabled Fleet of Programmable Unmanned Aerial Vehicles

Christian Tipantuña,[1,2] **Xavier Hesselbach**,[1] **Victor Sánchez-Aguero**,[3] **Francisco Valera**,[4] **Ivan Vidal**,[4] **and Borja Nogales**[4]

[1]*Department of Network Engineering, Universitat Politècnica de Catalunya, Spain*
[2]*Department of Electronics, Telecommunications and Information Networks, Escuela Politécnica Nacional, Ecuador*
[3]*IMDEA Networks Institute, Spain*
[4]*University Carlos III of Madrid, Spain*

Correspondence should be addressed to Christian Tipantuña; christian.jose.tipantuna@upc.edu

Received 16 November 2018; Revised 29 January 2019; Accepted 7 February 2019; Published 20 February 2019

Academic Editor: Giovanni Stea

The fifth generation of mobile networks (5G) is expected to provide diverse and stringent improvements such as greater connectivity, bandwidth, throughput, availability, improved coverage, and lower latency. Considering this, drones or Unmanned Aerial Vehicles (UAVs) and Internet of Things (IoT) devices are perfect examples of existing technology that can take advantage of the capabilities provided by 5G technology. In particular, UAVs are expected to be an important component of 5G networks implementations and support different communication requirements and applications. UAVs working together with 5G can potentially facilitate the deployment of standalone or complementary communications infrastructures, and, due to its rapid deployment, these solutions are suitable candidates to provide network services in emergency scenarios, natural disasters, and search and rescue missions. An important consideration in the deployment of a programmable drone fleet is to guarantee the reliability and performance of the services through consistent monitoring, control, and management scheme. In this regard, the Network Functions Virtualization (NFV) paradigm, a key technology within the 5G ecosystem, can be used to perform automation, management, and orchestration tasks. In addition, to ensure the coordination and reliability in the communications systems, considering that the UAVs have a finite lifetime and that eventually they must be replaced, a scheduling scheme is needed to guarantee the availability of services and efficient resource utilization. To this end, in this paper is presented an UAV scheduling scheme which leverages the potential offered by NFV. The proposed strategy, based on a brute-force search combinatorial algorithm, allows obtaining the optimal scheduling of UAVs in time, in order to efficiently deploy network services. Simulation results validate the performance of the proposed strategy, by providing the number of drones needed to meet certain levels of service availability. Furthermore, the strategy allows knowing the sequence of replacement of UAVs to ensure the optimal resource utilization.

1. Introduction

Recent evolution in Unmanned Aerial Vehicles (UAV) boosted by the miniaturization of electronic and sensors have allowed the use of UAVs in different civilian applications. Their shrinking size in combination with price reductions has increased the popularity of these devices both in the amateur community as well as in professional applications. Accordingly, we are now witnessing the fast deployment of a new categorization in the UAV area: Small Unmanned Aerial Vehicles (SUAV), commonly known as drones (that will be the preferred name in this article), which are low-cost devices with reduced payload capacities, restricted communication range, and limited battery time, but still powerful enough so as to carry small computers on board.

Drone applications are spreading throughout a plethora of different fields covering from smart agriculture scenarios to road traffic monitoring, public safety, sensor information retrieving, or even unmanned cargo. In general, these use cases are normally scheduled as relatively fixed missions of

standalone drones [1]. This paper, in particular, is focused on the energy management challenge such that, although is obviously present in standalone drone short missions, its complexity is exponentially exacerbated when dealing with multidrone long-term operations.

This research work has been done within the framework of the Spanish research project 5G-City (5GCity is a coordinated national project (2017-2019), funded by the Spanish Ministry of Economy and Competitiveness with the following partners: Universidad Carlos III de Madrid, Universidad de Granada, Fundacio i2Cat, Universitat Politecnica de Catalunya, Universidad de Vigo, and Universidad del Pais Vasco) that focuses on the provisioning of solutions for unpredictable critical events such as natural disasters or crowded events that produce damage and faults in the conventional network infrastructure (i.e., legacy infrastructure needs to be reinforced or is simply not available). The use of multidrone network is apparently a cost-effective solution which enables a fast and agile deployment in hard-to-reach locations and can straightforwardly be integrated into existing networks and adapt to unexpected changes. This flexibility can be certainly improved with the usage of Network Function Virtualization (NFV) 5G technology enabled drones as we have shown in [2]. However, drones have also several challenges that should be addressed. The weight a drone can carry determines the payload equipment and the size of the on-board battery. In consequence, we deal with low-resource payload (e.g., Single Board Computers) equipment and small batteries that provide limited service time. Because of these limitations, we need multidrone systems to cover areas of significant size and a fleet of reserve drones for replacements in order to provide long-term services. Apart from connectivity requirements, such as latency and bandwidth, a communication system provided by drones needs innovative management solutions (e.g., NFV) that enable the use of resources and energy efficiently.

For this reason, this article presents a strategy for the efficient management of resources in a communications system that provides services or network functions through the deployment of drones. The proposed solution leverages the potential offered by NFV and the 5G capabilities. In the context of the proposal, 5G technology is used to meet connectivity requirements, such as very low latency and high bandwidth in order to guarantee a correct migration procedure and also to provide communication between the different components in the system. Instead, NFV is in charge of the management tasks in the system. Specifically, in order to carry out the management tasks related to the replacement of drones and allocation of drones to services, an energy-aware scheduling algorithm has been developed, which is the main contribution of this paper.

The proposed algorithm, based on a brute-force search combinatorial method, explores all possible combinations of drones and service with the aim of providing the exact or optimal scheduling of drones to execute services. This exact allocation of drones over time ensures the continuity of services during a finite time interval, while leading to the optimal resource utilization. Apart from the replacement sequence, the algorithm can inform the total number of drones (or batteries) to use to reach a certain level of availability. In addition, within an NFV scope, the drone scheduling strategy can be considered as a network service.

To validate the performance of our solution, two small-scale scenarios have been analyzed, one Generic and one Realistic, whose results can be applied in the planning of design stages in a variety of real use cases, such as services in emergency or natural disasters and relief services in search and rescue missions. In addition, the information obtained with our implementation is also useful as a baseline to develop mathematical models and faster suboptimal or heuristics methods for real-time practical implementations.

The rest of the article is organized as follows: Section 2 provides a general overview of the related work. In Section 3 the main problem is formally presented. Section 4 presents the drone scheduling procedure and the complexity analysis. The performance evaluation is described in Section 5 and results are illustrated in Section 5.2. Finally, Section 6 concludes the article.

2. Related Work

In the last years, drone uses have evolved from the basic on-board video camera applications to a wide range of novelty functions such as drones acting as first responders in an accident or drone swarming intelligence to provide network services. To conduct these assignments efficiently, on account of drones' limitations, the use of 5G technologies such as NFV or SDN seems essential as they will enable an accurate operation. In particular, in this article, NFV is used to exemplify the execution of the proposed algorithm. There are several examples of the use of NFV in the UAV domain in the literature. In [3] an UAV platform provides to external controllers the opportunity to adapt the telemetry monitoring. In [4] is presented an NFV programmable infrastructure that enables the agile unification of services and functions, which may be determined by the operator of the UAVs at deployment time. NFV is used to decouple the drone hardware infrastructure from the control layer that virtualizes the infrastructure resources for the higher layers [5]. NFV is also used to enable multimission drones and supports a flexible deployment of network services [2]. Finally, NFV allows the migration of Virtual Network Functions (VNF) [3], which are the responsible units for providing the network functionality through the software implementation. The VNF migration enables an agile and flexible execution of the network services encompassing those VNFs that can be accommodated by the drones. Basically, the migration of VNFs consists of moving a virtual machine from one drone unit to another. There are different migration types: nonlive migration, where the VNF is down and it is moved to a different compute node, and live migration, where the VNF is running throughout the migration. Well-known tools that are key in the NFV framework development, such as OpenStack (OpenStack: https://docs.openstack.org/ocata) or VMware (VMware: https://www.vmware.com/es.html), support migration. The use of NFV is reinforced by the appearance of multidrone systems. Drones can run different

VNFs, endowing a huge versatility to the drone swarm. Nonetheless, virtualization is a resource intensive process, and because of the limited on-board equipment, it is necessary to use solutions such as LXC Linux Containers (Linux Containers: https://linuxcontainers.org/) to provide a similar environment as a Virtual Machine (VM) but reduce the overhead that comes with running a separate kernel and simulate all the hardware. It should be noted that the migration of container-based VNFs presents an additional challenge since this virtualization unit is stateless and in principle cannot be migrated. However, there are recent works in which they aim at addressing migration issues with containers like CRIU (a project to implement checkpoint/restore functionality for Linux: https://github.com/checkpoint-restore/criu). For this reason, in this article, because of the selected VNFs, the migration process is not mandatory. Routing VNFs recover their status proactively, collecting all the necessary routes in a few seconds but, in complex network scenarios, the use of VNF migration is crucial for correct operation.

Different alternatives for drone communication have been proposed in [6] being the WiFi in ad hoc mode one of the most popular solutions. Regarding routing strategies, there are also several options that extend from mobile ad hoc networks (MANETs) and vehicular ad hoc networks (VANETs) and also some innovative proposal like Software Defined Network (SDN).

The main focus of this paper is set on the battery power consumption. All the drones are normally equipped with a Single Board Computer as payload (Raspberry Pi 3B (Raspberry Pi 3B: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/) (RPi) in our case), with an autonomous battery giving rise to an independent service of the drone. In standard drone applications (drone flying as expected), flight-engines consume most of the energy [7] while the power consumption by network services is practically negligible, so that the service time is limited by the drone battery time (around 20 minutes following the technical specifications (DJI Phantom 3 Pro: https://www.dji.com/es/phantom-3-pro)). Even so, when a drone has a static position, it tends to land whenever possible to save energy (a drone that is providing a WiFi access point service does not necessarily have to be flying). In this case, service time will be limited by the SBC battery time and network services should be taken into account and will play an important role in modeling battery consumption.

Regarding power consumption in mobile and portable devices, there are different examples studying the impact of hardware components on the energy consumption [8, 9] and also the impact related to wireless communication [10]. In [11] is presented a method for wirelessly charging the drone battery when it lands, without the need to remove it and replace it. Ground task automation has come to the attention of researchers during the past few years [12, 13] reducing the human operators at the Ground Control Station (GCS).

In addition, in order to efficiently manage the available resources (e.g., energy), various techniques, mechanisms, and procedures have been developed. One of the most widely used is the combinatorial analysis, in which all possible combinations of resources to be used are analyzed. In this proposal, this mechanism is used to analyze all possible combinations of drones to run services. Considering a procedure similar to that described in [14], the proposed technique, by analyzing the whole set of possible cases, ensures the best (exact) result by providing the information of specific resources (drones and batteries) to be used. This optimal scheduling of drones guarantees an efficient use and management of available energy at every moment.

3. Problem Statement and System Model

In this section, first the statement of the problem is formally presented in Section 3.1. Then, the system model and notations are described in Section 3.2 followed by the definitions in Section 3.3 and the performance metrics in Section 3.4. Finally, the assumptions are presented in Section 3.5.

3.1. Problem Statement. Maintaining a certain degree or level of availability can become an important and even critical consideration in the deployment of network services. Especially in communication systems provided by drones, whose capacities in terms of processing and energy may have limitations, the efficient use and management of resources must be guaranteed in order to provide or maintain a desired level of availability. Therefore, this metric is an important factor in the design, planning, and deployment phases, considering that some applications may demand specific values for their operation.

In order to provide network services, by leveraging the connectivity capabilities offered by 5G networks and within an NFV context, a set of programmable drones can run VNFs, and, thus, provide the required services. In this sense, a fleet of programmable drones can offer different network services simultaneously, such as routing tasks, Internet connectivity, video surveillance services, telemetry, and multimedia services. To ensure proper coordination and management of the devices that implement the VNFs or services, it is necessary for an entity or component to perform the corresponding management tasks. In this way, and in an NFV environment, the core management entity, i.e., the orchestrator, can perform the orchestration and management of available resources [15].

Besides, because the provision of services provided by drones is constrained to their autonomy or battery duration, an efficient energy management scheme is of paramount importance in both short- and long-term applications. In this regard, a policy or scheme that allows the coordination and replacement of drones, to keep the service in an active state while ensuring a certain level of availability, is essential. As a result of all aforementioned, this work presents a scheme or management system for the deployment and replacement of drones, in which an optimal scheduling algorithm is implemented in order to guarantee the continuity of services, i.e., a level of availability, during a finite time interval.

The proposed scheme is shown in Figure 1 and is composed of two components: (i) a set of drones, which are in charge of executing the VNFs and that constitute the Network Function Virtualization Infrastructure (NFVI)

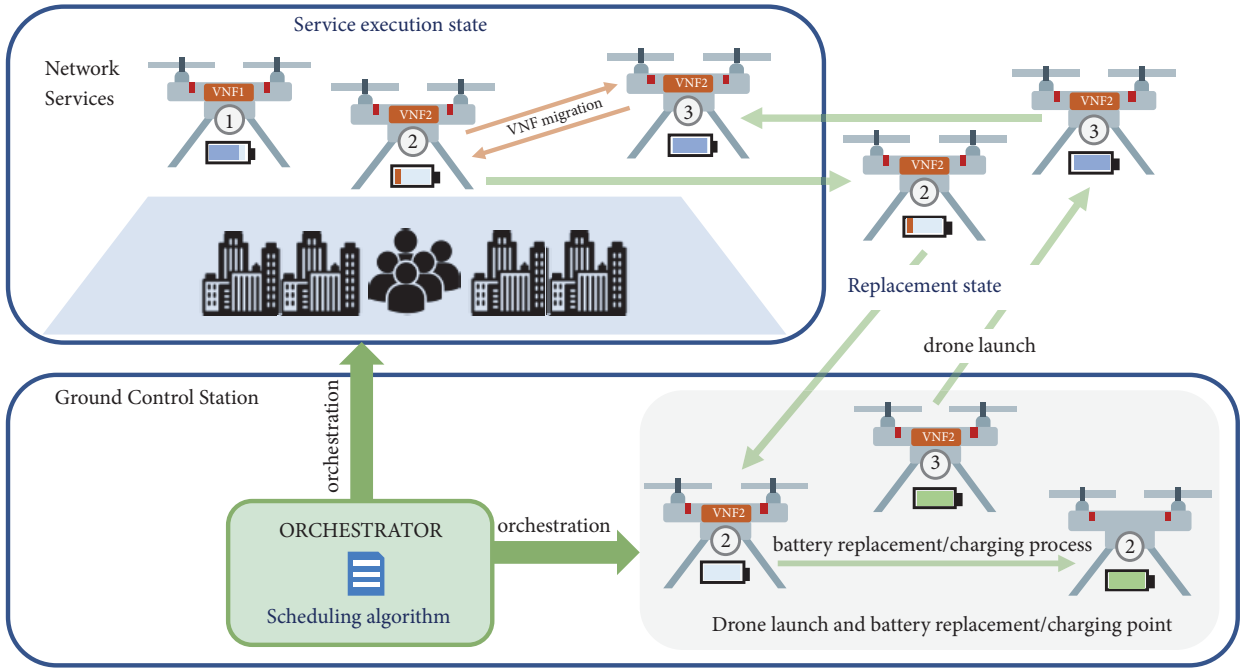


Figure 1: Overview of the proposed approach.

and (ii) a GCS, where the elements and entities responsible for monitoring, control, and management of resources and network services are located. The latter is precisely the component (NFV orchestrator) where the developed algorithm is intended to be executed. In this regard, the complete system can be considered as a network service, which in an NFV environment and using the connectivity benefits provided by 5G such as very low latency and high bandwidth can offer the optimal or exact drone scheduling for services execution. In addition, because 5G is considered in the design of the system, the proposed solution can also be categorized as a novel 5G use case.

The goal of the proposed algorithm is to carry out an optimal drone scheduling over time, in order to maximize the use of available resources, drones, while providing a reliable communications system guaranteeing continuity in the execution of services. In addition, the information provided by the algorithm can be used as a toolkit in mission planning. Apart from the replacement sequence, the algorithm can inform the services availability level obtained with the deployment of a given number of drones, or in turn the results can be used to know the number of drones that must be deployed to obtain a given service level.

The proposed scheme is characterized based on two different states, which are described as follows.

(1) Service Execution State. In this state the drones, which are equipped with processing and communication devices, execute the VNFs to provide the demanded services. For its operation, the drones are battery powered. Therefore, the autonomy time or drone lifetime is constrained to the capacity of the power supply, the energy consumption of the services to be executed, and the consumption of all the elements that allow the operation of the device.

In the proposed approach, the drones can execute the VNFs or services while they are in flight, as shown in the example of Figure 1. Also, for strategic reasons and with the aim of extending the service lifetime, it is possible to consider scenarios in which not all drones remain in flight. For example, for certain applications, such as the provision of connectivity services, some drones after their launch may land on specific locations. In the latter scenario, the service provided by the drones on the ground is not limited to the time that the drone can remain in the air. Even in this case, it is possible to consider the use of a secondary energy source to further lengthen the time of service provision. In any of the proposed scenarios, flying drones or drones on land, the algorithm guarantees the optimal drone scheduling overtime over time. Of course, depending on the scenario, for example, if all the drones are flying, the drone replacement procedure should be performed more frequently.

(2) Replacement State. In this state the drones do not provide the services. However, this phase is necessary to guarantee both the migration (transition) of VNFs and the replacement or recharging of drone batteries. Since the battery duration has a finite lifetime, it must be recharged or replaced, the replacement being a more useful and practical option in most cases, due to the agility involved in the process.

In the proposal, the management system located at GCS has all information about available resources (drones and batteries) and service requirements (power demanded by each service and the total required availability time) because all is provided by the users of the system. Through the

execution of a scheduling algorithm, the system is able to provide the optimal allocation of drones to cover the demanded services. The scheduling algorithm is executed in the NFV domain, specifically in the NFV orchestrator as depicted in Figure 1.

For the computation of the optimal allocation of drones, the algorithm considers both the consumption related to the service (*service execution state*) and the necessary power to perform the replacement process (*replacement state*). In the proposal, these power consumptions are represented by time variables, a *service time* related to the *service state* and a *replacement time* associated with the *replacement state* (see Figure 3). Thus, a percentage of the total battery capacity (majority) is assigned to service execution and the rest is dedicated to the execution of the replacement tasks (service migration). The time variables associated with the operating states of the system are described in detail in Section 3.3. Regarding the *service time*, this value depends on the power demanded by the service and can vary greatly from one service to another; for example, considering the same battery capacity, a drone running a service that demands high power consumption will have a shorter *service time* compared to another running a service whose power consumption is lower. On the other hand, the *replacement time* is composed of the time needed to perform the migration of the VNFs, from old drone (low battery level) to new drone (high battery level), and the time associated with the round trip flight to (old drone) and from (new drone) the GCS. Since the replacement time is not directly linked to the execution of the service, the value of this parameter could be similar or the same for the different services.

In order to guarantee the continuity of the services in the proposed system, the service migration process starts when the drone that is going to be replaced is active; i.e., when it is running the service, specifically, the migration process begins at the end of the *service time* or at the beginning of the *replacement time* (for a better understanding see Figures 3 and 5(e)). After the migration process has been carried out, i.e., when the *replacement time* is over, the replaced drone returns to the GCS; at this time this drone is no longer active but is still part of the system. Once the drone reaches the ground, its battery is replaced or recharged so that, according to the indications received by the GCS, it can be assigned for the execution of another service.

According to the aforementioned, in the system, the replacement time is sufficient to guarantee the transition of the services as well as the launching and landing of the drones. In addition, regarding the migration process, among the important aspects to consider are the service hand-off processes, from one drone to another, and the exchange of information associated with this procedure. Regarding the latter, in the proposal the exchange of information is accomplished thanks to features such as high connectivity and low latency time provided by 5G technology. Instead, the procedure related to the service transition is a process linked to the type and features of each service; therefore, although this is an interesting topic, it is not addressed in the article since it is out of scope of the proposal. However, it is worth mentioning that Section 5 presents the results of the application of the proposal in a real case whose values of both the *service state* and the *replacement state* (including the migration process) have been obtained through measurements.

At all times, the management system coordinates the resources that must be allocated (drones to be launched from the ground), because based on the initial information of services and drones, as well as the computations performed by the algorithm, the system is able to estimate the number of available drones, the status of the services, the sequence of replacement to be performed, and the availability level reached. Hence, the characterization of the system through the *service state*, the *replacement state,* and their corresponding time variables enables the system to operate with the appropriate margins so that the services can be executed continuously during a required time interval while the resource utilization is optimized.

For a better description and understanding of the different states of the proposed energy management scheme, an example is presented below. In Figure 1 is considered an application environment composed by two VNFs, which are expected to be active during a finite time interval. To this end, the system initially uses two drones, drone 1 and drone 2, which execute VNF1 and VNF2, respectively. As time goes by, the management system evidences that drone 2 is draining its battery due to the consumption of the service and the consumption related to its flight. In response to this, and before the drone stops providing the service or in the worst case it stops working and collapses to ground, the system coordinates the sending of another drone. In this case drone 3 is selected, whose energy level is adequate to guarantee the execution of the VNF 2 for a subsequent time interval. At the moment that drone 3 is located at a suitable distance for the establishment of communication with drone 2, the migration of VNF2 from drone 2 to drone 3 is performed, so that the service is not interrupted and remains available. Subsequently, drone 2 returns to the ground station to recharge or replace its battery, so that it can be ready for a new allocation. Thus, drone 2 is available to run the VNF2 or a different VNF, and it depends on the decision that is made by the scheduling algorithm and the corresponding management system.

During all the time of operation of the service, all the actions both on land and in the air are coordinated by the management and orchestration systems. In summary, the replacement state includes the launching of the new drone (with high battery level) from the ground station, the return of the old drone (with low battery level) to the ground station, and the service migration process (VNF migration).

In addition, from the example described above, it can be observed that to guarantee a continuous execution of the service and a total availability level (100%), the number of available drones must be at least one unit greater than the number of services. In the example it is verified that, to guarantee the continuous operation of VNF1 and VNF2, it is necessary to use 3 drones, drone 1 (VNF1), drone 2 (VNF2), and drone 3 (VNF2).

Also, as aforementioned, the replacement state may include the battery replacement or recharge of it. In the first

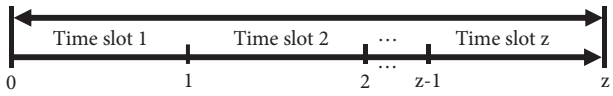


Figure 2: Time representation.

case, the battery replacement is a process that can generally take less time; for example, in a search or rescue mission, it is possible to use a limited number of drones and a large number of batteries. Meanwhile, in the second case, the battery charging commonly is a slower process, but necessary if the drone is tampering resistant, or if the number of available batteries is limited.

In the proposal, regarding the replacement state, for practical reasons, has been considered the battery replacement procedure. Nonetheless, the algorithm developed has the flexibility to consider a battery charging procedure. In fact, within the characterization of the system, the battery charging phase could be considered as an additional state, the *battery charging state*.

In summary, the proposed strategy bases its operation on a drone scheduling algorithm, which allows knowing how many drones are going to be used, how they should be replaced, and when the replacement should be made.

3.2. System Model. The drone scheduling algorithm is intended for providing the information of the optimal drone scheduling over time. In the proposal, the time variable has been divided into time slots, as shown in Figure 2. Thus, a drone is able to run a service over time (service execution state) during one or several slots according to its capabilities (available energy) and the features of the services that it can run. Similarly, the drone replacement state can last one or more time slots depending on the features of the drones (battery replacement procedure) and the scope of application of network services.

A summary of notations that describe the drone scheduling strategy is shown in Table 1. Then, these parameters are defined in Section 3.3.

3.3. Definitions

(1) *Expected availability time* (T_A^E): also defined as service availability time, it represents the time interval where the services are expected to be active/available.

(2) *Reached availability time* (T_A^R): time interval during which the services are active/available.

(3) *Number of services* (NS): the set of services or VNFs that are executed by the drones during a certain time period.

(4) *Initial time of service* j (T_{init}^j): time instant from which the service j is available/active, initial time in which the service availability is analyzed.

(5) *Power demanded by the service* j (P_d^j): power demanded by each service j to be executed. Each service may demand a different amount of power. In general, this parameter represents the consumption demanded by the services, and in practical implementations its units can also be given in terms of electric current (e.g., [mA]).

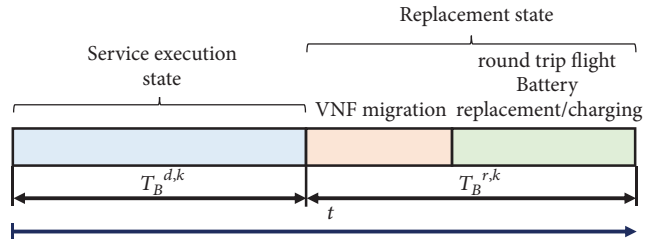


Figure 3: Time variables of the drone scheduling strategy.

(6) *Number of drones available* (ND): set of drones that are part of the system.

(7) *Battery capacity of drone* k (C_B^k): it represents the amount of energy that can be stored in a battery of each drone k. Moreover, in practical implementations this capacity can be expressed in terms of electric charge, i.e., electric current per time units (e.g., [mAh]).

(8) *Drone battery lifetime* ($T_B^{d,k}(P_d^j)$): each drone k with a battery capacity (C_B^k) can execute a service that demands a power level (P_d^j) during a time period $T_B^{d,k}$. This relationship can be expressed as follows:

$$T_B^{d,k}\left(P_d^j\right) = \frac{C_B^k}{P_d^j} \tag{1}$$

This time variable represents the time interval linked to the *service execution state*.

(9) *Battery replacement time* ($T_B^{r,k}$): this time variable is linked to the *replacement state* of a drone k. The ($T_B^{r,k}$) includes the time associated with the sending of the new drone (drone with high level of energy supply), the time demanded to perform the migration process of the services, and the time needed for the old drone (drone with low level of energy supply) to reach the ground station (charging point).

A pictorial representation of the time variables related to the two states that characterize the system is shown in Figure 3.

3.4. Metrics. To assess the performance of the drone scheduling algorithm, two metrics have been defined.

(1) *Services availability* (A_v): also defined as the total availability of services and expressed as a percentage, this metric shows the ratio between the time that all the services are available and the expected availability time. If the (T_A^E) = (T_A^R), i.e., all the services are available during all the time required, the (A_v) = 100

Table 1: System parameters.

Parameter	Description	Comments/Units
T_A^E	Expected availability time	Time units
T_A^R	Reached availability time	Time units
A_v	Service availability	Percentage, $A_v \in \{0, \ldots, 100\}$
A_vS	Service availability per services	Percentage, $A_vS \in \{0, \ldots, 100\}$
NS	Number of services (VNFs)	Integer number
S_j	Service identifier	$j \in \{1, \ldots, NS\}$
T_{init}^j	Initial time of service j	Time units
P_d^j	Power demanded by the service j	Power units
ND	Number of available drones	Integer number
D_k	Drone identifier	$k \in \{1, \ldots, ND\}$
C_B^k	Battery capacity of drone k	Power x Time units
$T_B^{d,k}(P_d^j)$	Battery lifetime of drone k for service j	Time units
$T_B^{r,k}$	Battery replacement time of drone k	Time units

%; otherwise, this value will be lower. The service availability can be expressed as

$$A_v = \frac{T_A^R}{T_A^E} \cdot 100\% \tag{2}$$

(2) *Services availability per services* (A_vS): this metric provides the information of the mean availability value of all services. A service j can reach an availability level equal to $A_{v,j}$, if this value is small compared to the availability of the other services, the A_v value will also be small and equal to $A_{v,j}$. For this reason, the (A_vS) metric is defined, because it is less restrictive and weights all availability values, in order to provide information on the behavior of all the services that are part of the system. As a consequence, the A_vS value will always be greater or at most equal to the A_v value.

The service availability per services is defined by

$$A_vS = \frac{\sum_{j=1}^{NS} A_{v,j}}{NS} \cdot 100\% \tag{3}$$

3.5. Assumptions. The following assumptions are made for the practical implementation of the algorithm:

(1) In practical implementations each programmable drone can execute more than one VNF concurrently. However, to simplify the analysis, in the proposed scheduling scheme, each drone k can run only one VNF or service j. This consideration is valid, since the execution of several services in the same drone would correspond to the consumption of different power levels. Thus, the processing of only one VNF and the analysis of its consumption could represent a summarized value of all the services that are executed in the drone.

(2) Similar to the previous consideration, the strategy considers that a service j can only be executed by one drone k at the same time. This is with the aim of simplifying the analysis in the distribution of drones and services.

(3) In the proposed system, any drone has the ability to execute any VNF. Likewise, any battery can power any available drone. In this sense, all available resources, drones and batteries, can be reused when demanded. It is clear that the services execution is limited to the capabilities of drones and the features of services, as previously discussed in Section 3.1.

(4) In the proposal it is considered that all services work simultaneously, i.e., all services are available as long as the system has the resources for their execution.

(5) The ND must be at least equal to NS. However, as discussed in Section 3.1, to ensure the execution of services without interruption, ND should be at least greater than or equal to $NS + 1$ ($NS \geq ND + 1$). If $ND < NS$, then $A_v = 0\%$ and $A_vS = 0\%$. The aforementioned consideration is mandatory in the initial execution or first drone allocation process, after this stage the algorithm is continuously evaluating the amount of available resources. Therefore, in the following allocation processes ND could be smaller than NS, in which case the algorithm analyzes the requirements of services to perform the corresponding allocations.

(6) In the *replacement state*, the drone that is replaced arrives at the ground station, with a very low battery level (fully discharged battery). For the replacement process, a battery that has previously been charged up to 100% of its capacity is used (fully charged battery). Similarly, if the complete drone must be replaced and not just its battery, the device that replaces it will be equipped with a battery charged to the maximum level. This consideration is also valid for the drones that are assigned for the first time; i.e., the drones used in the first allocation process have their batteries fully charged. In addition, the system has enough batteries to guarantee the replacement process of all the drones that demand them.

(7) In the proposal it is assumed that communication requirements such as very low latency and high bandwidth capabilities are provided by 5G technology. Moreover, the level of connectivity provided by 5G allows for proper communication and coordination between the different components within the system.

4. Drone Scheduling Strategy

In this section, first the drone scheduling procedure is presented in Section 4.1; then, the complexity of the problem is discussed in Section 4.2.

4.1. Drone Scheduling Algorithm Procedure. The drone scheduling strategy consists of systematically computing the optimal set of available of drones to execute the services. To this end, the strategy follows the guidelines described in the *execution* and *replacement* states.

The scheduling process starts with the individual analysis of the execution of each service for each available drone ($T_B^{d,k}(P_d^j)$) then goes through the following three phases to obtain the optimal allocation of drones to run services.

(a) *Computation of Combinations.* In order to find the exact or optimal allocation of drones to run services, the algorithm, based on a brute-force search combinatorial method, explores all possible combinations of drones and services. Once all the possible combinations are obtained, the algorithm determines those that meet the system requirements (valid combinations). Subsequently, this set of combinations is sorted in descending order according to the A_v metric. At the end of this phase, the best combination of drones (the first combination in the list from the top) is selected. In this context, this best combination represents the optimal set of drones whose services execution produces the highest A_v value in the system.

(b) *Resource Allocation and Services Evaluation.* Once the best combination of drones and services is obtained, the drones are allocated to their corresponding services. Afterwards, the A_v and A_vS metrics are computed; specifically, the A_vS metric is computed because the A_v metric was already obtained in the previous phase. If the A_v reached is equal or greater than the desired value, i.e., $T_A^R \geq T_A^E$, the algorithm stops its execution; otherwise, it analyzes the current availability level of all services ($A_{v,j}$) and the available resources (drones that have not been used) to proceed with the next allocations.

The analysis of the available resources is carried out in the following phase, while the analysis of availability per services is part of this phase and corresponds to the services evaluation, which is a procedure performed in order to reach the highest possible A_v or T_A^R value (both parameters completely equivalent), from the second allocation process. In this regard, based on the information of the last allocation made, the algorithm lists the services in descending order according to the $A_{v,j}$ reached, so that this information can be used in the computation and subsequent allocation of the best combination of drones. In specific, the objective of this process is to provide additional information to the algorithm in order to allocate the drones with the highest battery capacity to the services with the lowest current $A_{v,j}$ values. In summary, the services evaluation contributes that the drones are allocated starting with the service with the lowest $A_{v,j}$ value. The mechanism described in this phase ensures an increasing A_v and efficient resource utilization.

(c) *Verification of Available Resources.* Throughout the scheduling process, the algorithm must know the status of the executed services (T_A^R) and the information of the resources in the system. Especially from the second allocation process, the algorithm has to identify the resources used and available in order to perform the computation of combinations and the subsequent allocation of resources. In this regard, the algorithm has to evaluate at each time the information of the drones in the system, considering that this information consists of drones used, drones that have not been used, drones that must replace their battery, and drones whose battery has been replaced and are ready for a new allocation.

In an iterative process, the algorithm follows the phases described above and continuously calculates the best scheduling of drones to execute services. This procedure is carried out constantly until any of the two stopping criteria is met. The first criterion is the A_v value reached; if after an allocation process $A_v = 100\%$, the algorithm stops its execution. The second stop criterion is related to the number of available drones in the system, considering that this number is made up of drones that have not been used (not allocated yet) and drones whose battery has been replaced (or charged). In the event that the system does not have the necessary resources (drones) to perform the corresponding allocations, the algorithm stops its execution, under this condition $A_v \neq 100\%$ will be achieved. Finally, the algorithm provides the information of the (T_A^R), A_v, and A_vS reached.

The developed algorithm guarantees the best drone scheduling for services execution over time, by analyzing all possible drones-services combinations. However, the problem tends to growth as the *NS* and *ND* increase, which can be a problem if the capacity or processing time are constrains within the system.

The phases discussed above are implemented in the algorithm through different steps. The drone scheduling algorithm is explained in Figure 4 and each step is described in detail based on the example depicted in Figure 5. In this example $T_A^E = 7$ [time slots], and, for simplicity, the *NS* and *ND* are limited to 2 and 4, respectively. A pictorial representation of the required services is shown in Figure 5(a). Moreover, the example considers a $T_B^{r,k} = 2$ [time slots]; the first time slot corresponded to the VNF

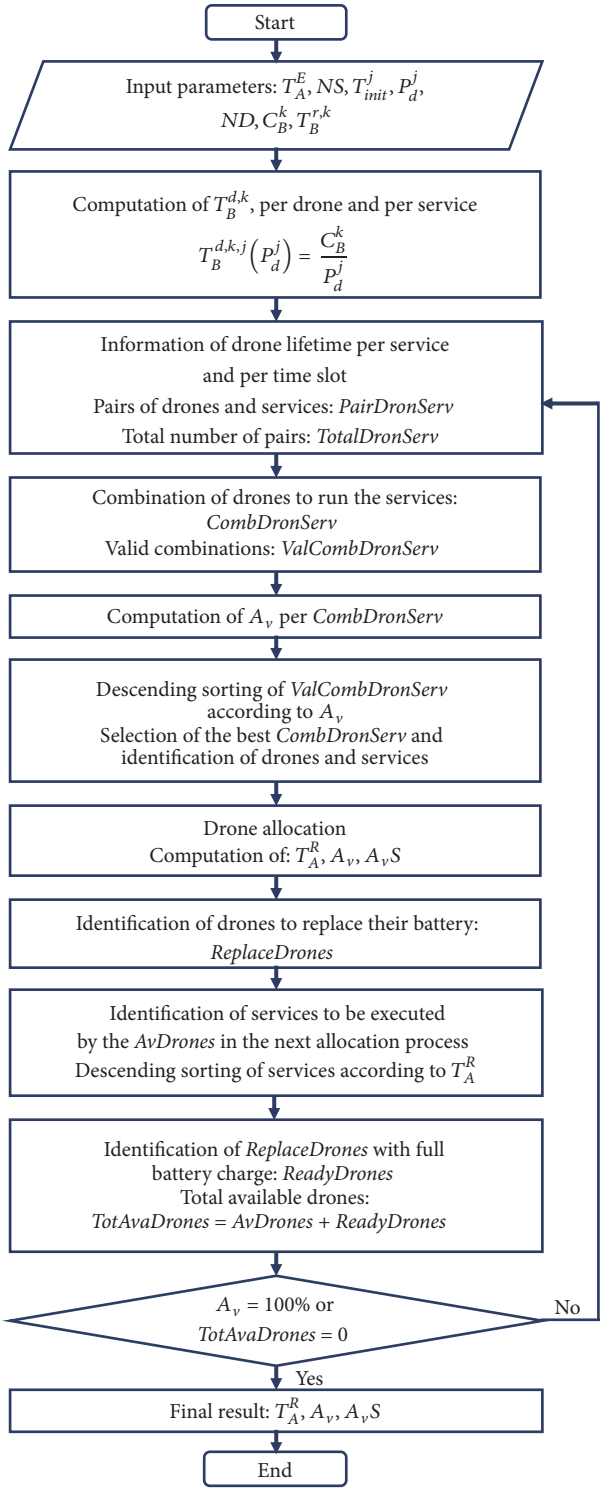


Figure 4: Algorithm for drone scheduling.

migration process and the round trip time of the drone, and the second time slot referred to the time for battery replacement. In addition, to better understand the proposed strategy in Table 2 is provided a summary of parameters related to the processing of the combinations and the analysis of the drones in the system. The different steps that are part of the algorithm are explained as follows.

(1) Input parameters: the input parameters comprise the T_A^E value and the information of services and drones, as shown in Tables 3(a) and 3(b), respectively.

(2) Computation of $T_B^{d,k}(P_d^j)$, per drone and per service: for the computation of these values, (1) is used.Table 4(a) presents all possible values of battery lifetime per drones $(D_1, \ldots, D_4)$ and per services (S_1, S_2).

(3) Information of drone lifetime per service and per time slot: in this step is provided the same information displayed in Table 4(a) but disaggregated in pairs of drones and services (*PairDronServ*), as shown in Table 4(b). Further, in this step the information of $T_B^{d,k}(P_d^j)$ is presented in time slots, following the discrete time model discussed in Section 3.2. At this point, the algorithm provides the information of all possible drone options to execute the services individually. The total number of pairs of drone-services (*TotalDronSer*) is given by

$$TotalDronServ = NS \cdot ND \tag{4}$$

In the proposed example, with $NS = 2$ and $ND = 4$ the $TotalDronServ = 8$ pairs of drones-services, from *PairDronServ* 1 to *PairDronServ* 8, see Table 4(b). Therefore, in this table are represented all possible service lifetime values depending on the drones to be used. For instance, if S_1 would be executed by drone D_4 (*PairDronServ* 7), the service lifetime would be $T_B^{d,4}(P_d^1) = 1$ [time slot]; instead, if S_1 would be run on drone D_1 (*PairDronServ* 1), the service lifetime would be several times greater and equal to $T_B^{d,1}(P_d^1) = 4$ [time slots].

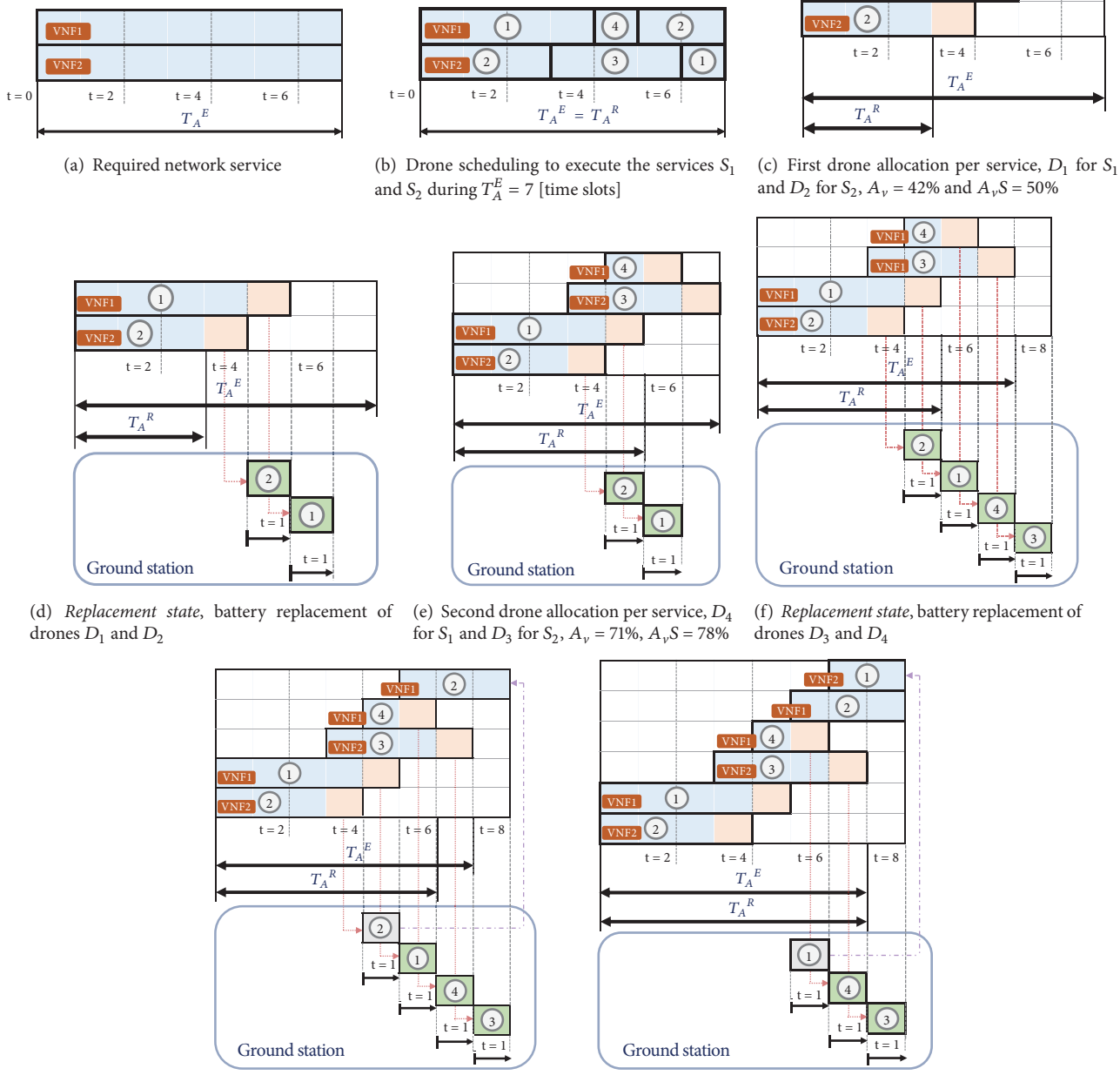
The information of each *PairDronServ* will be used in the next step of the algorithm to analyze the joint action of drones to run services, in order to obtain the maximum possible A_v and A_vS values in every allocation process.

(4) Combination of drones to run the services: since all services run simultaneously, the different combinations of drones and services (*CombDronServ*) must be analyzed. Hence, the algorithm from a group of ND drones must obtain a set of all possible combinations of drones to run NS services (*AllCombDronServ*). In this context, the total number of combinations (*NumAllComb*) to be processed is given by the analysis of $NS \cdot ND$ pairs (*TotalDronSer*, see Table 4(b)) taken NS at a time and can be expressed as

$$\begin{aligned} NumAllComb &= \binom{ND \cdot NS}{NS} \\ &= \frac{(NS \cdot ND)!}{NS! \cdot (NS \cdot ND - NS)!} \end{aligned} \tag{5}$$

The *NumAllComb* obtained in this step is critical, because it contributes largely to the growth of complexity of the problem. For instance, $NS = 8$ and $ND = 10$ produce over 28 billion of combinations to be processed.

In accordance with the criteria adopted for the drone scheduling strategy, see Section 3.5, not all *AllCombDronServ* are valid. For instance, in the example



(a) Required network service

(b) Drone scheduling to execute the services S_1 and S_2 during $T_A^E = 7$ [time slots]

(c) First drone allocation per service, D_1 for S_1 and D_2 for S_2, $A_v = 42\%$ and $A_vS = 50\%$

(d) *Replacement state*, battery replacement of drones D_1 and D_2

(e) Second drone allocation per service, D_4 for S_1 and D_3 for S_2, $A_v = 71\%$, $A_vS = 78\%$

(f) *Replacement state*, battery replacement of drones D_3 and D_4

(g) Third drone allocation per service, D_2 for S_1, $A_v = 85\%$ and $A_vS = 92\%$

(h) Fourth drone allocation per service, D_1 for S_2, $A_v = 100\%$ and $A_vS = 100\%$

FIGURE 5: Example of the drone scheduling algorithm.

(see Table 4(b)), the combination of *PairDronServ* 1 (D_1, S_1) and *PairDronServ* 2 (D_1, S_2) is not valid, because the same drone (D_1) is assigned to different services (S_1, S_2) at the same time. Likewise, the combination of *PairDronServ* 3 (D_2, S_1) and *PairDronServ* 5 (D_3, S_1) is also not valid, because the same service (S_1) is executed by two drones (D_2, D_3) concurrently. In contrast, the combination of *PairDronServ* 1 (D_1, S_1) and *PairDronServ* 4 (D_2, S_2) is valid, because once selected the drones D_1 and D_2 can be allocated to the services S_1 and S_2, respectively. Considering this, the total number of valid combinations (*NumValComb*) is given by

$$NumValComb = \frac{ND!}{(ND - NS)!} \tag{6}$$

In the example, $NS = 2$ and $ND = 4$ produce $NumAllComb = 28$ combinations and $NumValComb = 12$ combinations, the latter shown in Table 4(c).

(5) Computation of A_v and A_vS per CombDronServ: using the information of *ValCombDronServ* in Table 4(c) and

Table 2: Parameters related to the processing of combinations and drones in the scheduling algorithm.

Parameter	Description
PairDronServ	Pair of a drone and a service. A pair is used to describe the individual analysis of the execution of a service S_j running on a drone D_k
TotalDronServ	Total number of pairs of drones and services
CombDronServ	Combination of a set of drones to run a set of services. A combination of drones and services, which is commonly referred to as a combination, is composed of different pairs of drones and services
AllCombDronServ	Set of all possible combinations of drones and services
NumAllComb	Total number of all possible combinations. This number is given by (5)
ValCombDronServ	Set of valid combinations of drones and services. The *ValCombDronServ* is a subset of *AllCombDronServ*
NumValComb	Total number of valid combinations. This number is given by (6)
SortCombDronServIDs	Sorted list of the identifiers of the analyzed combinations. To obtain this list, the combinations are sorted in descending order according to the A_v reached
ReplaceDrones	Set of drones whose battery must be replaced
AvDrones	Set of drones that have neither been used nor allocated in the system
ReadyDrones	Set of drones whose battery has been replaced. These drones can be used for a new allocation process
TotAvaDrones	Total number of drones that are available in the system. This number is given by (10)

Table 3: Information of services and drones.

(a) Information of services

S_j	T_{init}^j [*Time slots*]	P_d^j [*Power units*]
1	0	1
2	0	1

(b) Information of drones

D_k	C_B^k [*Power x time slots*]	$T_B^{r,k}$ [*Time slots*]
1	4	2
1	3	2
3	3	2
4	1	2

the $T_B^{d,k}(P_d^j)$ value, given in terms of time slots, in Table 4(b), it is possible to compute the A_v and A_vS metrics, for each of the *CombDronServ*. Table 5(a) shows the different A_v and A_vS values for each *CombDronServ* of Table 4(c). In this table, the metrics have been rounded to the lower bound. An example of the computation of these metrics for the *CombDronServ* 1 is provided below. In the case of *CombDronServ* 1 (composed by *PairDronServ* 1 and *PairDronServ* 4), D_1 is allocated to S_1 during $T_B^{d,1}(P_d^1) = 4$ [time slots], while D_2 is allocated to S_2 during $T_B^{d,2}(P_d^2) = 3$ [time slots].

$$A_{vComb1} = \frac{3\,[time\ slots]}{7\,[time\ slots]} \cdot 100\% = 42.86\% \tag{7}$$

$$A_vS_{Comb1} = \frac{4\,[time\ slots]\,/7\,[time\ slots] + 3\,[time\ slots]\,/7\,[time\ slots]}{2} \cdot 100\% = 50\% \tag{8}$$

(6) Selection of the best CombDronServ: the *ValCombDronServ* are sorted in descending order according to their A_v value (see Table 5(a)), using a quick sort method. The algorithm provides a list with these values and the combination with the best value; the first in the list is selected. Even if there is more than one better combination, the first one in the list is always selected. In the example, the sorted list of all *ValCombDronServ* is

$$Sort\ CombDronServ\ IDs := \{1, 2, 4, 5, 7, 9, 3, 6, 8, 10, 11, 12\} \tag{9}$$

where the *CombDronServ* 1 ($ID = 1$) is the best combination, while the *CombDronServ* 12 has the lowest A_v level. After selecting the best *CombDronServ*, in the example *CombDronServ* 1, the algorithm proceeds to identify the drones and services belonging to that combination, as shown in Table 5(b).

(7) Drone allocation and computation of T_A^R, A_v, and A_vS: in this step, with the information of the best combination, the algorithm allocates the drones to their corresponding services over time. As shown in Table 6(a), D_1 is allocated to S_1 and D_2 is allocated to S_2. Figure 5(c) illustrates this allocation procedure, and in this figure is not only represented the

Table 4: Information of drones, services, and combinations.

(a) Information about battery lifetime of drones for services S_1 and S_2

D_k	$T_B^{d,k}(P_d^1)$ [*Time slots*]	$T_B^{d,k}(P_d^2)$ [*Time slots*]
1	4	4
2	3	3
3	3	3
4	1	1

(b) Information of pairs of drones and services and drone lifetime per time slot

No.Pair	D_k	S_j	*Drone per time slot*
1	1	1	1 1 1 1
2	1	2	1 1 1 1
3	2	1	2 2 2 0
4	2	2	2 2 2 0
5	3	1	3 3 3 0
6	3	2	3 3 3 0
7	4	1	4 0 0 0
8	4	2	4 0 0 0
		Time Slot	1 2 3 4

(c) Combinations among available drones to run the services

No.Comb.	*Combination* (*No.Pair*)	*Drones*	*Services*
1	1, 4	1, 2	1, 2
2	1, 6	1, 3	1, 2
3	1, 8	1, 4	1, 2
4	2, 3	1, 2	2, 1
5	2, 5	1, 3	2, 1
6	2, 7	1, 4	2, 1
7	3, 6	2, 3	1, 2
8	3, 8	2, 4	1, 2
9	4, 5	2, 3	2, 1
10	4, 7	2, 4	2, 1
11	5, 8	3, 4	1, 2
12	6, 7	3, 4	2, 1

execution state ($T_B^{d,k}(P_d^j)$, blue color) but also the *replacement state* ($T_B^{r,k}$, orange and green colors). Then, the performance metrics A_v and A_vS are computed. In this particular example the values reached are $A_v = 42\%$ and $A_vS = 50\%$. This is the initial allocation of drones, and since none of the stop criteria have been met, i.e., $A_v \neq 100\%$ and the system has available resources (drones D_3 and D_4), the algorithm continues its execution process.

(8) Identification of drones to replace their battery (ReplaceDrones): the algorithm must constantly monitor the drones used, so that when they finish their execution (T_A^R), they have to change to the *replacement state* (*ReplaceDrones*). In the example, in the first allocation drones D_1 and D_2 have been used, and, as can be seen in Figure 5(d), drone D_2 must start its replacement process in time slot 4; instead, D_1 must perform this procedure in time slot 5.

(9) Identification of services to be executed by the available drones (*AvDrones*) *in the next allocation process:* to continue

Table 5: Computation of metrics for *ValCombDronServ* and selection of the best combination.

(a) Computation of A_v and A_v for all *ValCombDronServ*

No.Comb.	*Combination* (*No.Pair*)	A_v(%)	A_vS(%)
1	1, 4	42	50
2	1, 6	42	50
3	1, 8	14	35
4	2, 3	42	50
5	2, 5	42	50
6	2, 7	14	35
7	3, 6	42	42
8	3, 8	14	28
9	4, 5	42	42
10	4, 7	14	28
11	5, 8	14	28
12	6, 7	14	28

(b) Information of pairs of drones and services belonging to the best combination

No.Pair	D_k	S_j	*Drone per time slot*
1	1	1	1 1 1 1
4	2	2	2 2 2 0
		Time Slot	1 2 3 4

with the drone scheduling process, from the first allocation, an analysis of the priority of the executed services is carried out. This level of priority is given based on the (T_A^R) parameter of the services. Thus, a service with a lower (T_A^R) value will have a higher priority level to be processed in the next drone allocation step. In this way, the algorithm will allocate the drones with the highest ($T_B^{d,k}(P_d^j)$) values to the services with the highest priority levels. The priority in the execution time of the services is analyzed at the end of the first allocation process, since the start time of all the services is the same ($T_{init}^j = 0$).

The priority information of the services is used in the computation of the combinations and in the selection of the best combination, from the second allocation. This process is carried out to guarantee an efficient and uniform allocation of the drones; otherwise, a specific service could achieve a T_A^R value much higher than the others. This situation must be avoided since the services must be executed simultaneously with the objective of always reaching an A_v as large as possible. In the example (see Figure 5(d)), S_2 ($T_A^R = 3$ [time slots]) has higher priority level compared to S_1 ($T_A^R = 4$ [time slots]). Therefore, later in the computation of all combinations this information will be considered so that the selection of the best combination allows a drone allocation (*AvDrones* = 2, *D*3 and *D*4) that favors the execution of S_2, as shown in Figure 5(e).

The priority level of services is also useful in the case that $ND < NS$, a situation that may occur after the first drone allocation. In this particular case, the priority allows to know the services that must be attended, with the available

resources. In this scenario, the combinations are computed with the number of available drones.

(10) Identification of total available drones (TotAvaDrones): the total number of drones available in the system include drones that have not been used, which in the case of the example are D_3 and D_4(*AvDrones*), and drones whose battery has been replaced and are ready (*Ready*D*rones*) to be used when the system demands them. The *ReadyDrones* are the *ReplaceDrones* that have completed the *replacement state*. In the example proposed, at this point, the first drone allocation has been performed; therefore, the system does not have *ReadyDrones*. Under these conditions, the drones available are D_3 and D_4. In the next stages, the algorithm could have *ReadyDrones* available, once they exceed the *replacement state*.

The total number of available drones (*TotAvaDrones*) in the system can be expressed as

$$TotAvaDrones = AvDrones + ReadyDrones \quad (10)$$

(11) Iterative drone allocation: in an iterative process of computation of combinations, selection of the best combination, drone allocation, priorities of services and metrics, the algorithm continues its execution until either of the two stopping criteria is met ($A_v = 100\%$ or $TotAvaDrones = 0$). Continuing with the example, in the second iteration the algorithm allocates D_3 to S_2 and D_4 to S_1, as shown in Table 6(b) and in Figure 5(e). The results after this procedure are $A_v = 71\%$ and $A_vS = 78\%$. In a similar way to the process carried out at the end of the first iteration, once the services have been completely executed by drones D_3 and D_4, these devices pass to the *replacement state*. Specifically, this status will be reached at time slot 5 for D_4 and at time slot 6 for D_3, as represented in Figure 5(f). For practical reasons, the time window in proposed example has been limited to 8 [time slots].

After the second iteration has been completed, the algorithm checks the total number of available drones. At this point, given that $AvDrones = 0$, the algorithm checks if any of *ReplaceDrones* has become *ReadyDrones*. In the example, the unique drone that meets this condition is D_2, which after the corresponding computations is allocated to S_1, as shown in Table 6(c) and in Figure 5(g). At the end of the third iteration $A_v = 85\%$ and $A_vS = 92\%$. Once the third iteration is finished, the scheduling process continues in the same way as in the previous iterations and according to the steps described in the Figure 4. The fourth iteration is the last in the example. In this iteration $D1$, whose battery has been replaced, is allocated to S_2, as seen in Figure 5(h) and in Table 6(d). Resulting in the final values: $T_A^R = T_A^E = 7$ [time slots] for both services (S_1 and S_2), $A_v = 100\%$ and $A_vS = 100\%$. Once these values are obtained, the algorithm stops its execution.

A summary of the complete drone allocation procedure is depicted in Figure 5(b). The final drones sequence is D_1, D_4, and D_2 for S_1 and D_2, D_3, and D_1 for S_2. Moreover, as a relevant result the algorithm allows knowing the number of drones (resources) needed to reach a certain availability level. In the example, during a time window of $T_A^E = 7$ [time slots], with 4 drones and an optimal scheduling, the system can reach an $A_v = 100\%$ in the services execution.

Table 6: Progressive allocation of drones to fulfill the network services.

(a) Initial drone allocation

S_j	*Drone information per time slot*
1	1 1 1 1 0 0 0
2	2 2 2 0 0 0 0
Time Slot	1 2 3 4 5 6 7

(b) Second drone allocation

S_j	*Drone information per time slot*
1	1 1 1 1 4 0 0
2	2 2 2 3 3 3 0
Time Slot	1 2 3 4 5 6 7

(c) Third drone allocation

S_j	*Drone information per time slot*
1	1 1 1 1 4 2 2
2	2 2 2 3 3 3 0
Time Slot	1 2 3 4 5 6 7

(d) Final drone allocation

S_j	*Drone information per time slot*
1	1 1 1 1 4 2 2
2	2 2 2 3 3 3 1
Time Slot	1 2 3 4 5 6 7

4.2. Complexity Analysis. The *NS* and *ND* values have an impact on the growth of complexity of the algorithm. The growth rate of the problem is not linear, because it depends on the product of $NS \cdot ND$, as shown in (4) and in Table 4(b).

The steps 3 and 4 define the growth of the algorithm. This growth rate as a function of *NS* and *ND* can be expressed as

$$f(NS \cdot ND) = NS \cdot ND + C(NS \cdot ND, NS) \quad (11)$$

where the second term is the dominant term within the expression. As described in Section 4 it represents total set of all combinations (*AllCombDronServ*) that the algorithm must analyze in a mandatory manner in order to find the valid combinations *ValCombDronServ* to be processed.

Thus, according the Big-O classification [16], ignoring the low-order terms, i.e., the first term in (11), the order of growth of the drone scheduling algorithm is $\mathcal{O}(C(NS \cdot ND, NS))$. Hence, this complexity reveals the drawbacks that the algorithm has for the selection of *NS* and *ND* values.

5. Performance Evaluation

To validate the resource planning algorithm proposed in the previous section, it is necessary to define reasonable scenarios that can integrate all the different parameters that should be assessed and provide the complex environment where this type of algorithms is normally applied. The evaluation will

then be carried out through extensive simulations using these scenarios. Section 5.1 describes the simulation setup and the two application scenarios and then the evaluation results are presented and discussed in Section 5.2.

5.1. Simulation Setup. The drone scheduling strategy is evaluated in two different application scenarios: a *Generic Scenario* that uses random values for some parameters and a *Realistic Scenario* whose values are based on experiments and measurements, as shown in Table 7. These scenarios are described in detail below, and, for practical reasons, in both scenarios the *NS* has been limited up to $NS = 7$ services. The scheduling algorithm has been implemented using Matlab (Matlab R2017b).

The *Generic Scenario* has been run on a computer equipped with a 3.33 GHz x 12 cores Intel Core i7 Extreme processor and 12 GB RAM. This simulation leverages parallel processing and multiple CPU cores (up to 6 cores) have been used in each simulation. In order to ensure stability of the results, each case (*NS*) was repeated 50 times except for $NS = 7$, which was executed only once due to its excessive running time. Results are shown, in Figures 10 and 11, with a confidence interval of 95%. The total running time for this scenario (all cases) exceeded 300 hours.

Meanwhile, the *Realistic Scenario* has been executed on a machine with a 3GHz x 4 cores Intel Core i5-7400 processor and 64 GB RAM. In this case parallel processing has also been exploited, and all 4 cores have been used during the simulation. Because this scenario is deterministic, only one simulation has been executed for each case ($T_A^E = 5$, $T_A^E = 10$ and $T_A^E = 15$ hours). The execution time for this scenario is around 80 hours, and results are shown in Figure 12.

5.1.1. Generic Scenario. This scenario corresponds to a very general application environment with the intention of performing an initial validation of the proposed solution. To this end, it is assumed that the different drones can execute the services in the air (with a much higher power consumption), a subset can land on the ground after its launch from the ground control station, or even a hybrid situation can also be possible (different cases are discussed in Section 3.1).

The scenario is not particularized for specific applications and it is considered that applications can vary from the provision of video surveillance services to the provision of connectivity services, etc. (this is modeled in the scenario by considering the power demanded by the different services P_d^j to be a random value between 1 and 5 A). In addition, to provide more diversity, batteries capacities C_B^k are considered to be different for each drone, choosing for them random values between 1 and 5 Ah. This assumption (see Table 7) will produce services with different T_A^R values (*service execution state*) varying from $T_A^R = 0.2$ [hours] (minimum value) to $T_A^R = 5$ [hours] (maximum value).

Considering the parameters described above, a $T_B^{r,k} = 10$ [minutes] (*replacement state*) and a time window of $(T_A^E) = 10$ [hours], the algorithm has to perform many transitions among the *service execution state* and the *replacement state* in order to optimally allocate the available resources to the corresponding services, which is exactly the situation that is wanted to be forced in this scenario in order to test the algorithm capabilities. The metrics achieved for the different *NS* and *ND* values are shown later in Section 5.2.

5.1.2. Realistic Scenario

Scenario Definition. The objective of this scenario is to test the algorithm under more real conditions replacing the random values used in the Generic Scenario by some other values that may be closer to some real ones.

In particular the scenario that will be described in this section (Figure 6) shows a set of drones each one with an onboard SBC (they carry an RPi with its own battery) linked through an ad hoc WiFi network and using a certain FANET (Flying Adhoc Network) routing protocol to guarantee connectivity. The drones are including different VNFs depending on the role they are assuming (Access Point (AP), router, or telemetry transmitter (i.e., video or sensor data).

In this scenario the energy consumption for a particular drone may depend on many diverse factors. In first place there are two different types of batteries and also drones that are flying and drones that are landed (so depending on the situation the battery that limits the service maybe either one or the other). For the drones that are not flying (the drone battery is not presenting any limitations for them and only the RPi battery is used) the measurements must consider the different WiFi interfaces, the WiFi communications (different traffic including video, telemetry, routing messages, etc.), CPU load, external hardware, etc.

As it can be appreciated it is not easy in this environment to evaluate how the energy consumption curve will perform for the different drones and how many drones are in fact needed in order to guarantee that the service can be maintained over time (considering a certain replacement time), etc. This is considered to be a suitable scenario so as to validate the combinatory algorithm and the rest of the section will provide more details on the scenario itself and about the validation methodology.

A total of seven drones have been considered in this scenario that may represent a natural disaster use case (e.g., earthquake, fire, flood) where drones can enable communications between emergency services islands; as seen in Figure 6 drones accommodate different VNFs and play different roles within the network to perform the overall service that will be taken into account by the algorithm:

(1) *Optimized Link State Routing (OLSR) Router VNF.* Because of the drone network nature (e.g., node mobile, volatile network) OLSR [17] has been selected as routing protocol. OLSR is a distributed and proactive routing protocol used to establish connections between participant nodes in an ad hoc wireless network proposed for Mobile Ad hoc Networks (MANETs) and extended for Flying Ad hoc Networks (FANETs). The main advantage of this type of routing protocol is its dynamic discovery allowing stateless

Table 7: Summary of simulation parameters.

Scenario	T_A^E	NS	ND	T_{init}^j	C_B^k	P_d^j	$T_B^{r,k}$
Generic	10 hours	1-7	0-11	0, for all j	uniform distributed random value [1- 5] [Ah], for all k	uniform distributed, random value [1- 5] [A], for all j	10 [min], for all k
Realistic	5, 10, 15 hours	7	0-10	0, for all j	3 [Ah], for all k	S1 Router: 292.02 [mA] S2 Router: 292.35 [mA] S3 AP + Router: 371.82 [mA] S4 AP + Router: 373.62 [mA] S5 Telemetry TX: 288.76 [mA] S6 Telemetry TX: 288.23 [mA] S7 Flying: 9000 [mA]	10 [min], for all k

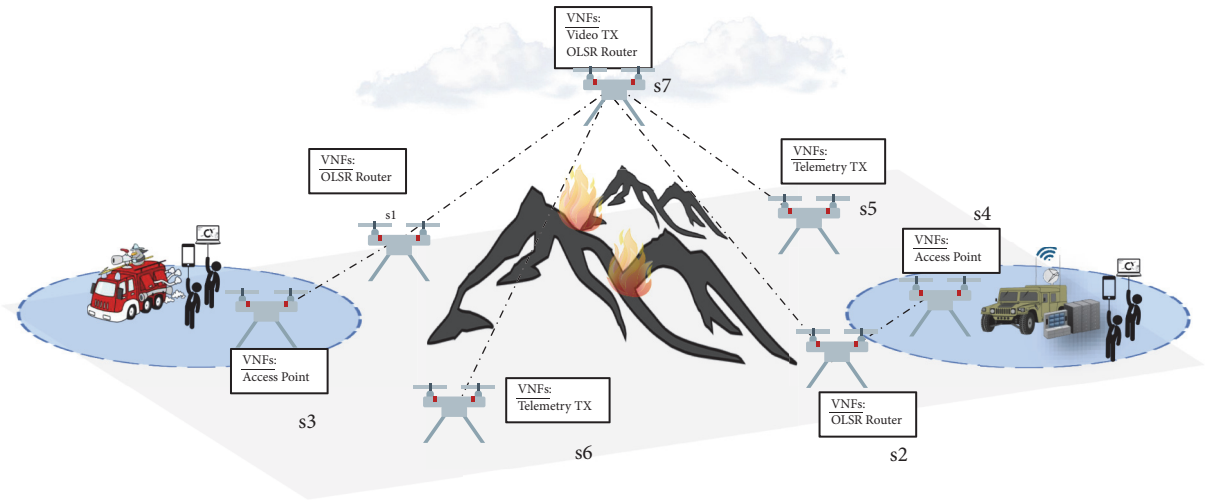


Figure 6: Drone swarm providing network connectivity in a disaster situation.

VNFs that prevent the costly process of VNF migration. Drones s1 and s2 in Figure 6 are OLSR routers.

(2) *Access Point VNF.* Wireless AP VNFs are used to interconnect wireless communication equipment from emergency services (end user terminals). The selected technology has been the normal 2.4 GHz IEEE 802.11. Drones s3 and s4 in Figure 6 are APs.

(3) *Telemetry Transmitter VNF.* Data transmission VNFs have been used in different drones within the network. As it can be appreciated in Figure 6, two types of data transmitters have been specified, (a) telemetry transmitter (32 Kbps flow that can either represent GPS information or sensor data such as temperature or humidity) and (b) video transmitter in standard quality (200 Kbps flow) that is enough to have an overview of the disaster area by the emergency services. Drones s5 and s6 in Figure 6 are telemetry transmitters.

Drones s1 to s6 are landed and the energy demanded is only related to the network services while drone s7 is flying (Figure 6).

Parameters Estimation. In order to validate the algorithm, it is necessary to provide as input a rough estimation of the power demanded by each drone. However, in this realistic environment, besides user traffic (video, telemetry, etc.), numerous factors may affect battery lifetimes. Parameters like the pattern of energy consumption, environmental conditions, or battery status are significant factors to take into account in real applications, but it is extremely complex to model them in simulated environments therefore despised. However, there is unpredictable network traffic which is considered to measure energy consumption, such as packets retransmission, WiFi management packets, routing messages, etc.

The power consumption will be directly measured using a real RPi and a specific power meter. In order to do so, it is required that the RPi resembles the real conditions as stated in the scenario definition in terms of traffic and CPU load and considers the necessary hardware to enable wireless communication since the consumption depends heavily on these parameters.

To calculate all these values, a simulation using ns-3 (ns-3: https://www.nsnam.org/) network simulator has been performed. As it can be seen in Figure 6, the simulation

Table 8: Simulation details.

Parameter	Values
Traffic	CBR
Telemetry Transmission Rate	32 Kbps
Video Transmission Rate	200 Kbps
Network Protocol	UDP
Routing Protocol	OLSR
Simulation Time	3600 seconds
Number of Drones	7
Mobility Model	Static

includes the seven drones (one of them is flying (s7) while the rest of them are landed). The drones that accommodate the telemetry transmitter VNFs generate one flow to each one emergency service involved on the scene. More details can be consulted in Table 8 where the simulation parameters are specified. The trajectory of the flying node (s7) has been precalculated using Matlab and included in the simulation using traces with the ns format.

After this simulation, it will be possible to calculate the traffic that will be processed by each drone, including all the different components that have been previously mentioned.

To be able to properly analyze the traffic at each drone, the following characterization has been done for the traffic depending on the source and the destination as represented in Figure 8:

(1) *Transit traffic:* received traffic to be forwarded because that drone is not the final destination of the packet. This traffic is telemetry data.

(2) *Sink traffic:* traffic that is consumed by that particular drone. This traffic corresponds to OLSR management or WiFi management packets since telemetry data is consumed out of the analyzed network by the emergency services.

(3) *Source traffic:* traffic that is generated at that particular drone. This traffic can be either OLSR management, WiFi management, or telemetry data.

Figure 7 shows the average throughput for each drone obtained by the simulation. These results will be replicated into real RPis in order to perform the power measurements. Note that the flying node s7 transmitting video is not represented in the figure. Flight-engines are demanding all the available energy in this drone and power consumption due to traffic processing is negligible in comparison. No power measurement is performed here since the battery that limits the service execution is the one of the drone itself (in Table 7 this power consumption is modeled as 9000 [mA] since together with the battery capacity that is used, a 20 minutes flight estimation is obtained which is quite normal for regular drones). In addition, as expected the traffic consumed by drone (only OLSR management) is insignificant compared with transit and generated traffic.

Power Consumption Measurements. As it has been mentioned, the main purpose of the simulations described in the previous

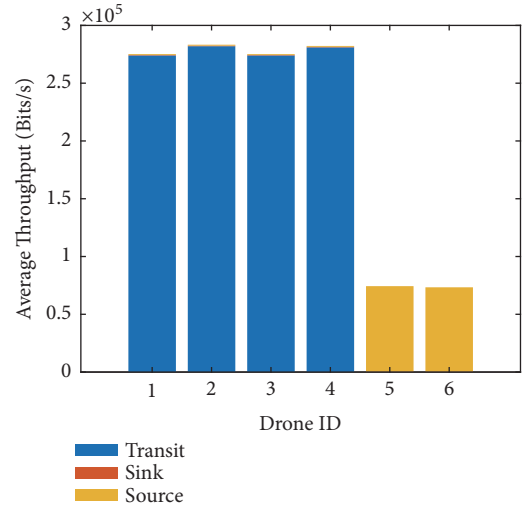


Figure 7: Average throughput for Realistic Scenario.

section is to estimate the different data flows (Figure 7) that have to be injected into the RPi in order to emulate the power consumption that it would have in a real scenario.

In order to perform these measurements, the testbed depicted in Figure 8 has been built. It includes three different RPis 3B and the Monsoon FTA22D Power Meter (Monsoon FTA22D Power Meter: https://www.msoon.com/) (which provides a robust power measurement solution for mobile devices with high accuracy (±1%)).

The RPi source generates two flows, one of them consumed by the RPi hosting the VNF and the second one consumed by the RPi destination. The RPi that accommodates the VNF is also generating another flow that is consumed by the RPi destination. In this way, we can emulate the traffic involved with each device on the network, to generate this traffic, the Iperf (Iperf: https://iperf.fr/) tool.

The RPi VNF is then powered by the Monsoon (Vout voltage of 4.2 V) main channel and then the average power is derived from instantaneous current (Figure 9) and voltage and divided by the duration of the sampling run (200 seconds). After conducting the measurements, it has been verified that the power consumption is not significantly increased unless the network interface is close to the maximum bandwidth. The authors in [8] are finding similar results. The most relevant power increase is due to the use of an external hardware (extra WiFi card to create the AP) carried by drones s3 and s4.

5.2. Results. In both scenarios, *Generic* and *Realistic*, the optimal drone scheduling is performed. As a result, the algorithm allows knowing the level of services availability achieved when using a given number of available drones. Seen in another way, the proposed strategy can be used to know how many drones need to be deployed to reach a certain services availability level (in the simulations from $A_v = 0\%$ to $A_v = 100\%$). In this context, the results provided by the drone scheduling algorithm can be used in the design, planning, and deployment stages of network services

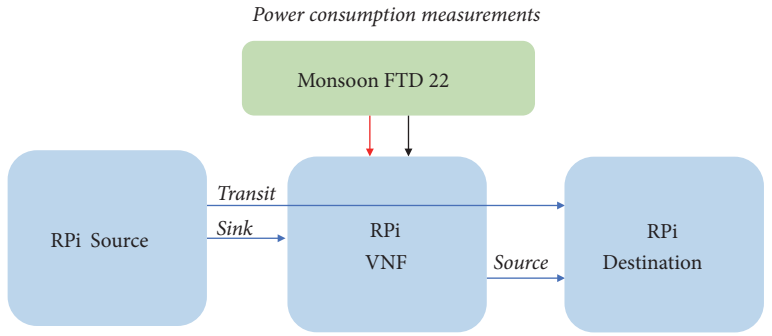


Figure 8: Power consumption measurements methodology.

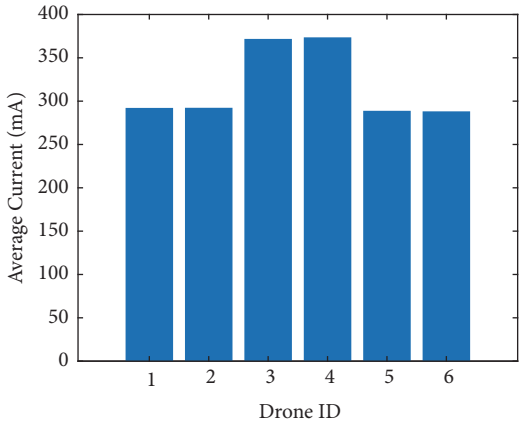


Figure 9: Average Current for Realistic Scenario.

executed by drones. Therefore, the information provided by the scheduling strategy can be used for both performance evaluation and system dimensioning.

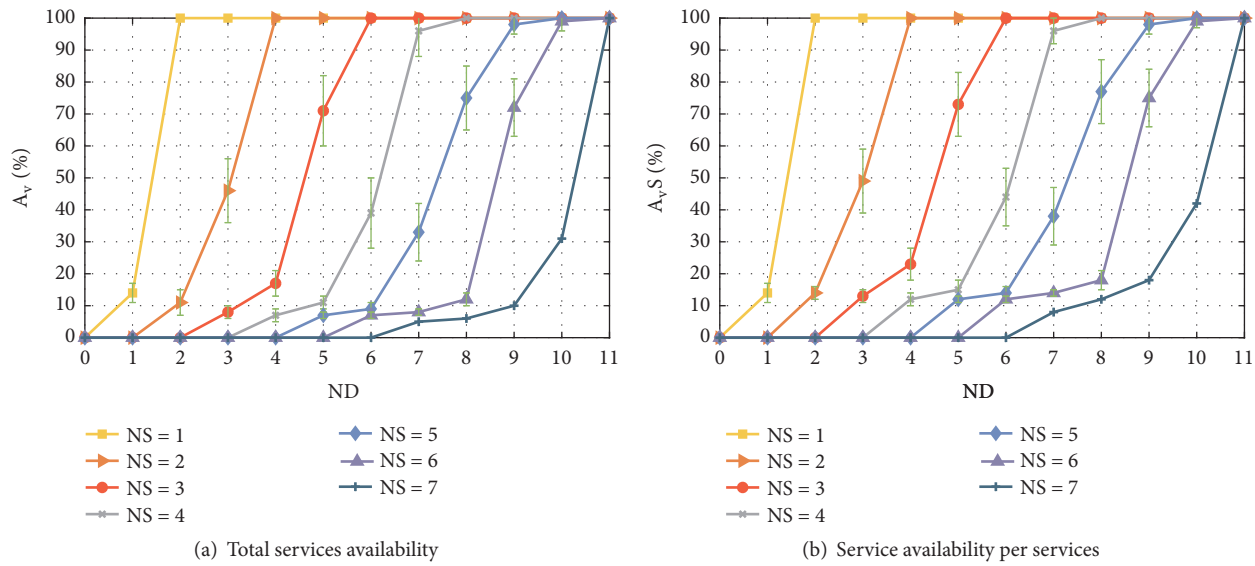
Regarding the *Generic Scenario*, whose results are shown in Figure 10(a) (A_v), in Figure 10(b) (A_vS), and in Figure 11 ($A_v = 100\%$), the algorithm provides the different availability levels (metrics) for each of the services from $NS = 1$ up to $NS = 7$. As discussed above, this information allows knowing the number of drones needed to reach a certain A_v value, or the A_v obtained with a given amount of available resources (drones). For example, as shown in Figure 10(a), with $NS = 3$ services are needed $ND = 6$ drones to reach an $A_v = 100\%$ (also $A_vS = 100\%$ in this case). Similarly, if as system consists of $NS = 5$ services and demand $A_v = 90\%$, the number of drones required is $ND = 9$ drones. In the latter example, as shown in Figure 10(a), for $NS = 5$ there is no an exact ND value that meets $A_v = 90\%$; in this case the selection should round the upper bound value ($ND = 9$), due to the fact that the ND is an integer number. This rounding operation also ensures that the value obtained is greater than or equal to the requested value.

Another relevant result that can be extracted from the results provided by the algorithm (Figure 10(a)) is ND as a function of NS to reach $A_v = 100\%$, as shown in Figure 11 (also $A_vS = 100\%$). This summarized information represents a practical and useful tool that can be used in the design and planning phases of services that have as a constraint the 100% of availability for their operation (e.g., provision of communications services in emergency or search scenarios). For example, in the *Generic Scenario*, it is appreciated that for $NS = 6$ are needed at least $ND = 10$ drones to reach and $A_v = 100\%$.

In addition, the results obtained help corroborate the criteria that were considered in the design of the algorithm. For example, as discussed in Section 3.4, all A_vS values must be equal or greater than A_v values. This condition is verified by establishing a comparison between Figure 10(a) (A_v) and Figure 10(b) (A_vS) for the *Generic Scenario*, and between Figure 12(a) (A_v) and Figure 12(b) (A_vS) for the *Realistic Scenario*. For example, for the (*Generic Scenario*), with $NS = 5$ services and $ND = 7$ services, $A_v = 33\%$ instead of $A_vS = 38\%$.

In the *Generic Scenario* in specific, the A_v and A_vS metrics obtained are very similar to each other (always $A_v \geq A_vS$). This situation obeys one or more of the following considerations: (i) T_A^E and T_A^R are large compared to the size of time slots (minimum amount of time in the system, 10 [min] in the simulations) and (ii) in the final allocations there is not much difference between T_A^R of all services (i.e., C_B^k) and (P_d^j) have similar values for all services). In the *Realistic Scenario* instead, A_v (Figure 12(a)) and A_vS (Figure 12(b)) values are different from each other, mainly for $T_A^E = 5$ [hours] and $T_A^E = 10$ [hours]; this due to the considerable difference of T_A^R for all services, in particular referred to S_7, whose demanded consumption (9 [A]) is much greater than the rest of services. In this case, the algorithm needs to allocate a greater number of drones to execute the service, or in other words to keep the service in active mode a high replacement rate is necessary (i.e., high transition between the *service execution state* and the *replacement state*).

On the other hand, in Figure 12(a) (A_v) and Figure 10(b) (A_vS) are shown the performance metrics achieved for the *Realistic Scenario*. In this scenario, all cases ($T_A^E = 5$, 10 and 15 [hours]) consider $NS = 7$ services, and as a result of the evaluation, the algorithm provides that the required ND to reach $A_v = 100\%$, in all cases, is equal to $ND = 10$ drones. This value represents the minimum amount of resources (drones and batteries) that the system must use to face a reliable ($A_v = 100\%$) network services deployment.



(a) Total services availability (b) Service availability per services

FIGURE 10: Performance evaluation of the drone scheduling strategy, Generic Scenario.

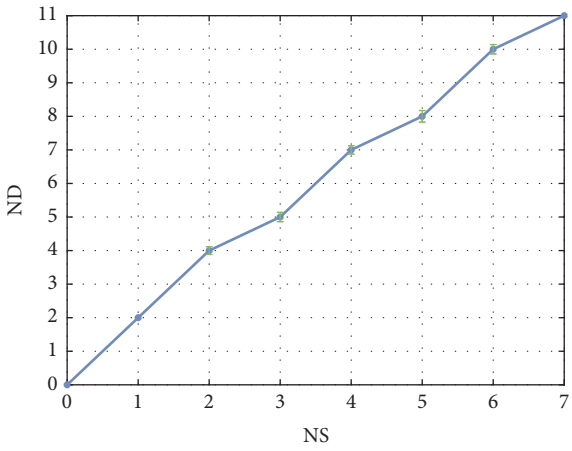


FIGURE 11: *ND* to reach an $A_v = 100\%$ - Generic Scenario.

In this scenario, different T_A^E values have been considered, to evaluate the behavior of the strategy in both short-term and long-term real applications. In this regard, the scheduling algorithm performs a few numbers of allocation procedures when $T_A^E = 5$ [hours] and $T_A^E = 10$ [hours] (only S_5 needs to used different resources); instead, a high transition between *service execution* and *state* is experienced when $T_A^E = 15$ [hours].

6. Conclusions

In this paper, an optimal drone scheduling algorithm is developed, which by leveraging 5G and NFV capabilities is able to perform an efficient energy-aware management of resources for network services provisioning. Through this strategy, it is possible to calculate the required number of drones for a certain degree of service, to be used in real scenarios. The scheduling strategy, based on two states, *service execution* and *replacement*, provides the information about the number of drones and their sequence of replacement to run services and reach a certain availability level, during a finite time interval. Thus, the proposed scheduling algorithm can be used as a useful tool in system dimensioning and missions planning tasks, in order to provide reliable and safe drone-based network services deployments.

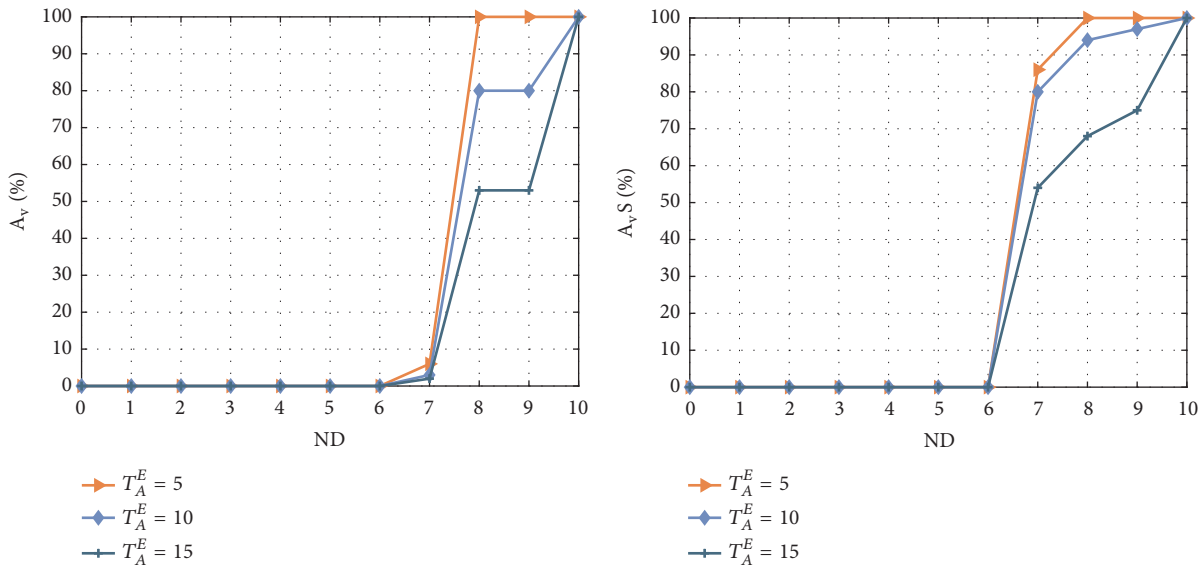
The algorithm can perform the optimal scheduling in both short- and long-term applications, and it can be used as a resource/availability planner in a wide variety of real scenarios, such as emergency scenarios, relief disaster services, and search and rescue tasks.

Simulations results validate the performance of the proposal and provide the metrics achieved, as well as the amount of resources needed for the execution of services in different scenarios.

The results provided by the simulations can be used to know the level of availability for a certain number of services and available drones. Likewise, these results allow knowing the number of drones needed to run services to guarantee 100% of availability level.

Finally, the paper presents the evaluation of the proposal for scenarios up to $NS = 7$ services and $ND = 11$ drones, limited by to the complexity of the algorithm. Although these limits can be very useful and quite adequate values in many practical scenarios and applications, it is necessary to develop additional strategies in order to be able to handle larger scenarios and in a faster running time. In this regard, the brute-force search solution developed is also useful as a baseline method to design heuristics or metaheuristics approaches. Currently, authors are working on developing these solutions as part of the future works.

Based on information provided by the implemented strategy, the future works also include the modeling of the services availability as a mathematical function, in terms of



(a) Total services availability

(b) Service availability per services

FIGURE 12: Performance evaluation of the drone scheduling strategy, Realistic Scenario.

the number of services, their power consumed, the capacity of the batteries, and the number of drones.

Data Availability

The data and results used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been supported by the Ministerio de Economía y Competitividad of the Spanish Government under projects TEC2016-76795-C6-1-R and TEC2016-76795-C6-3-R and also AEI/FEDER, UE. Christian Tipantuña acknowledges the support from Escuela Politécnica Nacional (EPN) and from Secretaría de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT) for his doctoral studies at Universitat Politécnica de Catalunya (UPC).

References

[1] H. Shakhatreh, A. Sawalmeh, A. Al-Fuqaha et al., "Unmanned aerial vehicles: A survey on civil applications and key research challenges," 2018, https://arxiv.org/abs/1805.00881.

[2] B. Nogales, V. Sanchez-Aguero, I. Vidal, F. Valera, and J. Garcia-Reinoso, "A nfv system to support configurable and automated multi-uav service deployments," in *Proceedings of the Proceedings of the 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, pp. 39–44, ACM, 2018.

[3] K. J. S. White, E. Denney, M. D. Knudson, A. K. Mamerides, and D. P. Pezaros, "A programmable sdn+ nfv-based architecture for uav telemetry monitoring," in *Proceedings of the Consumer Communications & Networking Conference (CCNC), 2017 14th IEEE Annual*, pp. 522–527, IEEE, 2017.

[4] B. Nogales, V. Sanchez-Aguero, I. Vidal, and F. Valera, "Adaptable and automated small uav deployments via virtualization," *Sensors*, vol. 18, no. 12, p. 4116, 2018.

[5] M. Sara, I. Jawhar, and M. Nader, "A softwarization architecture for UAVs and WSNs as Part of the cloud environment," in *Proceedings of the Cloud Engineering Workshop (IC2EW), 2016 IEEE International Conference on*, pp. 13–18, IEEE, 2016.

[6] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2016.

[7] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[8] F. Kaup, P. Gottschling, and D. Hausheer, "PowerPi: measuring and modeling the power consumption of the raspberry Pi," in *Proceedings of the Local Computer Networks (LCN), IEEE 39th Conference on*, pp. 236–243, 2014.

[9] A Carroll, G. Heiser et al., "An analysis of power consumption in a smartphone," in *Proceedings of the In USENIX annual technical conference*, vol. 14, p. 21, Boston, Mass, USA, 2010.

[10] P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs, and A. Azcorra, "Per-frame energy consumption in 802.11 devices and its implication on modeling and design," *IEEE/ACM Transactions on Networking (ToN)*, vol. 23, no. 4, pp. 1243–1256, 2015.

[11] S. Dunbar, F. Wenzl, C. Hack et al., "Wireless far-field charging of a micro-UAV," in *Proceedings of the Wireless Power Transfer Conference (WPTC)*, pp. 1–4, IEEE, 2015.

[12] M. Wang, "Systems and methods for uav battery exchange," 2016, US Patent 9,346,560.

[13] K. A. O. Suzuki, P. Kemper Filho, and J. R. Morrison, "Automatic battery replacement system for UAVs: Analysis and design," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 563–586, 2012.

[14] C. Tipantuña and X. Hesselbach, "Demand-Response power management strategy using time shifting capabilities," in *Proceedings of the Ninth International Conference on Future Energy Systems*, pp. 480–485, ACM, 2018.

[15] ETSI Industry Specification Group (ISG) NFV, "ETSI GS NFV 002 V1.1.1: Network Functions Virtualisation (NFV); Architectural Framework," Tech. Rep., 2013.

[16] S. Johnson Michael, R. Garey, and David, *Computers and Intractability: A Guide to the Theory of NP-completeness*, WH Free. Co., San Francisco, Calif, USA, 1st edition, 1979.

[17] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," Technical report, 2003.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 6202854, 12 pages
https://doi.org/10.1155/2018/6202854

Research Article

QoE Evaluation: The TRIANGLE Testbed Approach

Almudena Díaz Zayas,[1] **Laura Panizo**,[1] **Janie Baños**,[2] **Carlos Cárdenas**,[2] **and Michael Dieudonne**[3]

[1]*University of Malaga, Spain*
[2]*DEKRA, Spain*
[3]*Keysight Technologies, Belgium*

Correspondence should be addressed to Almudena Díaz Zayas; adz@uma.es

Received 11 October 2018; Accepted 25 November 2018; Published 18 December 2018

Academic Editor: Giovanni Stea

This paper presents the TRIANGLE testbed approach to score the Quality of Experience (QoE) of mobile applications, based on measurements extracted from tests performed on an end-to-end network testbed. The TRIANGLE project approach is a methodology flexible enough to generalize the computation of the QoE for any mobile application. The process produces a final TRIANGLE mark, a quality score, which could eventually be used to certify applications.

1. Introduction

The success of 5G (the fifth generation of mobile communications), and to some extent that of 4G, depends on its ability to seamlessly deliver applications and services with good Quality of Experience (QoE). Along with the user, QoE is important to network operators, product manufacturers (both hardware and software), and service providers. However, there is still no consensus on the definition of QoE, and a number of acronyms and related concepts (e.g., see [1]) add confusion to the subject: QoE (Quality of Experience), QoS (Quality of Service), QoSD (Quality of Service Delivered/achieved by service provider), QoSE (Quality of Service Experience/Perceived by customer/user), and so forth. This is a field in continuous evolution, where methodologies and algorithms are the subject of study of many organisations and standardization bodies such as the ITU-T.

TRIANGLE project has adopted the definition of QoE provided by the ITU-T in Recommendation P.10/G.100 (2006) Amendment 1 "Definition of Quality of Experience (QoE)" [2].

> *"The overall acceptability of an application or service, as perceived subjectively by the end-user"*

In [2], the ITU-T emphasizes that the Quality of Experience includes the complete end-to-end system effects: client (app), device, network, services infrastructure, and so on. Therefore, TRIANGLE brings in a complete end-to-end network testbed and a methodology for the evaluation of the QoE.

Consistent with the definition, the majority of the work in this area has been concerned with subjective measurements of experience. Typically, users rate the perceived quality on a scale, resulting in the typical MOS (Mean Opinion Score). Even in this field, the methodology for subjective assessment is the subject of many studies [3].

However, there is a clear need to relate QoE scores to technical parameters that can be monitored and whose improvement or worsening can be altered through changes in the configurations of the different elements of the end-to-end communication channel. The E-model [4], which is based on modelling the results from a large number of subjective tests done in the past on a wide range of transmission parameters, is the best-known example of parametric technique for the computation of QoE. Also, one of the conclusions of the Project P-SERQU, conducted by the NGMN (Next Generation Mobile Networks) [5] and focused on the QoE analysis of HTTP Adaptive Streaming (HAS), is that it is less complex and more accurate to measure and predict QoE based on traffic properties than making a one-to-one mapping between generic radio and core network QoS to QoE. The TRIANGLE

project follows also a parametric approach to compute the QoE.

Conclusions in [5] point out that a large number of parameters in the model could be cumbersome due to the difficulty of obtaining the required measurements and because it would require significantly more data points and radio scenarios to tune the model. The TRIANGLE approach has overcome this limitation through the large variety of measurements collected, the variety of end-to-end network scenarios designed, and mostly the degree of automation reached, which enables the execution of intensive test campaigns covering all scenarios.

Although there are many proposals to calculate the quality of experience, in general, they are very much oriented to specific services, for example, voice [6] or video streaming [7, 8]. This paper introduces a methodology to compute the QoE of any application, even if the application supports more than one service.

The QoE, as perceived by the user, depends on many factors: the network conditions, both at the core (CN) and at the radio access (RAN), the terminal, the service servers, and human factors difficult to control. Due to the complexity and the time needed to run experiments or make measurements, most of the studies limit the evaluation of the QoE to a limited set of, or even noncontrolled, network conditions, especially those that affect the radio interface (fading, interference, etc.). TRIANGLE presents a methodology and a framework to compute the QoE, out of technical parameters, weighting the impact of the network conditions based on the actual uses cases for the specific application. As in ITU recommendation G1030 [9] and G1031 [10], the user's influence factors are outside of the scope of the methodology developed in TRIANGLE.

TRIANGLE has developed an end-to-end cellular network testbed and a set of test cases to automatically test applications under multiple changing network conditions and/or terminals and provide a single quality score. The score is computed weighting the results obtained testing the different uses cases applicable to the application, for the different aspects relevant to the user (the domains in TRIANGLE), and under the network scenarios relevant for the application. The framework allows specific QoS-to-QoE translations to be incorporated into the framework based on the outcome of subjective experiments on new services.

Note that although the TRIANGLE project also provides means to test devices and services, only the process to test applications is presented here.

The rest of the paper is organized as follows. Section 2 provides an overview of related work. Section 3 presents an overview of the TRIANGLE testbed. Section 4 introduces the TRIANGLE approach. Section 5 describes in detail how the quality score is obtained in the TRIANGLE framework. Section 6 provides an example and the outcome of this approach applied to the evaluation of a simple App, the Exoplayer. Finally, Section 7 summarizes the conclusions.

2. State of the Art

Modelling and evaluating QoE in current and next generation of mobile networks is an important and active research area [8]. Different types of testbeds can be found in the literature, ranging from simulated to emulated mobile/wireless testbeds, which are used to obtain subjective or objective QoE metrics, to extract a QoE model, or to assess the correctness of a previously generated QoE model. Many of the testbeds reviewed have been developed for a specific research, instead of for a more general purpose, such as the TRIANGLE testbed, which can serve a wide range of users (researchers, app developers, service providers, etc.). In this section, some QoE-related works that rely on testbeds are reviewed.

The QoE Doctor tool [12] is closely related to the TRIANGLE testbed, since its main purpose is the evaluation of mobile apps QoE in an accurate, systematic, and repeatable way. However, QoE Doctor is just an Android tool that can take measurements at different layers, from the app user interface (UI) to the network, and quantify the factors that impact the app QoE. It can be used to identify the causes of a degraded QoE, but it is not able to control or monitor the mobile network. QoE Doctor uses an UI automation tool to reproduce user behaviour in the terminal (app user flows in TRIANGLE nomenclature) and to measure the user-perceived latency by detecting changes on the screen. Other QoE metrics computed by QoE Doctor are the mobile data consumption and the network energy consumption of the app by means of an offline analysis of the TCP flows. The authors have used QoE Doctor to evaluate the QoE of popular apps such as YouTube, Facebook, or mobile web browsers. One of the drawbacks of this approach is that most metrics are based on detecting specific changes on the UI. Thus, the module in charge of detecting UI changes has to be adapted for each specific app under test.

QoE-Lab [13] is a multipurpose testbed that allows the evaluation of QoE in mobile networks. One of its purposes is to evaluate the effect of new network scenarios on services such as VoIP, video streaming, or web applications. To this end, QoE-Lab extends BERLIN [14] testbed framework with support for next generation mobile networks and some new services, such as VoIP and video streaming. The testbed allows the study of the effect of network handovers between wireless technologies, dynamic migrations, and virtualized resources. Similar to TRIANGLE, the experiments are executed in a repeatable and controlled environment. However, in the experiments presented in [13], the user equipment were laptops, which usually have better performance and more resources than smartphones (battery, memory, and CPU). The experiments also evaluated the impact of different scenarios on the multimedia streaming services included in the testbed. The main limitations are that it is not possible to evaluate different mobile apps running in different smartphones or relate the QoE with the CPU, battery usage, and so forth.

De Moor et al. [15] proposed a user-centric methodology for the multidimensional evaluation of QoE in a mobile real-life environment. The methodology relies on a distributed testbed that monitors the network QoS and context information and integrates the subjective user experience based on real-life settings. The main component of the proposed architecture is the *Mobile Agent,* a component to be installed in the user device that monitors contextual data (location,

velocity, on-body sensors, etc.) and QoS parameters (CPU, memory, signal strength, throughput, etc.) and provides an interface to collect user experience feedback. A processing entity receives the (device and network) monitored data and analyzes the incoming data. The objective of this testbed infrastructure is to study the effects of different network parameters in the QoE in order to define new estimation models for QoE.

In [16], the authors evaluated routing protocols BATMAN and OLSR to support VoIP and video traffic from a QoS and QoE perspective. The evaluation took place by running experiments in two different testbeds. First, experiments were run in the Omnet++ simulator using the InetManet framework. Second, the same network topology and network scenarios were deployed in the Emulab test bench, a real (emulated) testbed, and the same experiments were carried out. Finally, the results of both testbeds (simulated and real-emulated) were statistically compared in order to find inconsistencies. The experiments in the simulated and emulated environments showed that BATMAN achieves better than OLSR and determined the relation between different protocol parameters and their performance. These results can be applied to implement network nodes that control in-stack protocol parameters as a function of the observed traffic.

In [17], a testbed to automatically extract a QoE model of encrypted video streaming services was presented. The testbed includes a software agent to be installed in the user device, which is able to reproduce the user interaction and collect the end-user application-level measurements; the network emulator NetEm, which changes the link conditions emulating the radio or core network, and a Probe software, which processes all the traffic at different levels, computes the TCP/IP metrics, and compares the end-user and network level measurements. This testbed has been used to automatically construct the model (and validate the model) of the video performance of encrypted YouTube traffic over a Wi-Fi connection.

More recently, in [18], Solera et al. presented a testbed for evaluating video streaming services in LTE networks. In particular, the QoE of 3D video streaming services over LTE was evaluated. The testbed consists of a streaming server, the NetEm network emulator, and a streaming client. One of the main contributions of the work is the extension of NetEm to better model the characteristics of the packet delay in bursty services, such as video streaming. Previously to running the experiments in the emulation-based testbed, the authors carried out a simulation campaign with an LTE simulator to obtain the configuration parameters of NetEm for four different network scenarios. These scenarios combine different positions of the user in the cell and different network loads. From the review of these works, it becomes clear that the setup of a simulation or emulation framework for wireless or mobile environments requires, in many cases, a deep understanding of the network scenarios. TRIANGLE aims to reduce this effort by providing a set of preconfigured real network scenarios and the computation of the MOS in order to allow both researchers and app developers to focus on the evaluation of new apps, services, and devices.

3. TRIANGLE

The testbed, the test methodology, and the set of test cases have been developed within the European funded TRIANGLE project. Figure 1 shows the main functional blocks that make up the TRIANGLE testbed architecture.

To facilitate the use of the TRIANGLE testbed for different objectives (testing, benchmarking, and certifying), to remotely access the testbed, and to gather and present results, a web portal, which offers an intuitive interface, has been implemented. It provides access to the testbed hiding unnecessary complexity to App developers. For advanced users interested in deeper access to configuration parameters of the testbed elements or the test cases, the testbed offers a direct access to the Keysight TAP (Testing Automation Platform), which is a programmable sequencer of actions with plugins that expose the configuration and control of the instruments and tools integrated into the testbed.

In addition to the testbed itself, TRIANGLE has developed a test methodology and has implemented a set of test cases, which are made available through the portal. To achieve full test case automation, all the testbed components are under the control of the testbed management framework, which coordinates their configuration and execution, processes the measurements made in each test case, and computes QoE scores for the application tested.

In addition, as part of the testbed management framework, each testbed component is controlled through a TAP driver, which serves as bridge between the TAP engine and the actual component interface. The configuration of the different elements of the testbed is determined by the test case to run within the set of test cases provided as part of TRIANGLE or the customized test cases built by users. The testbed translates the test cases specific configurations, settings, and actions into TAP commands that take care of commanding each testbed component.

TRIANGLE test cases specify the measurements that should be collected to compute the KPI (Key Performance Indicators) of the feature under test. Some measurements are obtained directly from measurement instruments but others require specific probes (either software or hardware) to help extract the specific measurements. Software probes, running on the same device (UE, LTE User Equipment) that the application under test, include DEKRA Agents and the TestelDroid [19] tool from UMA. TRIANGLE also provides an instrumentation library so that app developers can deliver measurement outputs, which cannot otherwise be extracted and must be provided by the application itself. Hardware probes include a power analyzer connected to the UE to measure power consumption and the radio access emulator that, among others, provides internal logs about the protocol exchange and radio interface low layers metrics.

The radio access (LTE RAN) emulator plays a key role in the TRIANGLE testbed. The testbed RAN is provided by an off-the-shelf E7515A UXM Wireless Test Set from Keysight, an emulator that provides state-of-the-art test features. Most important, the UXM also provides radio channel emulation for the downlink radio channel.

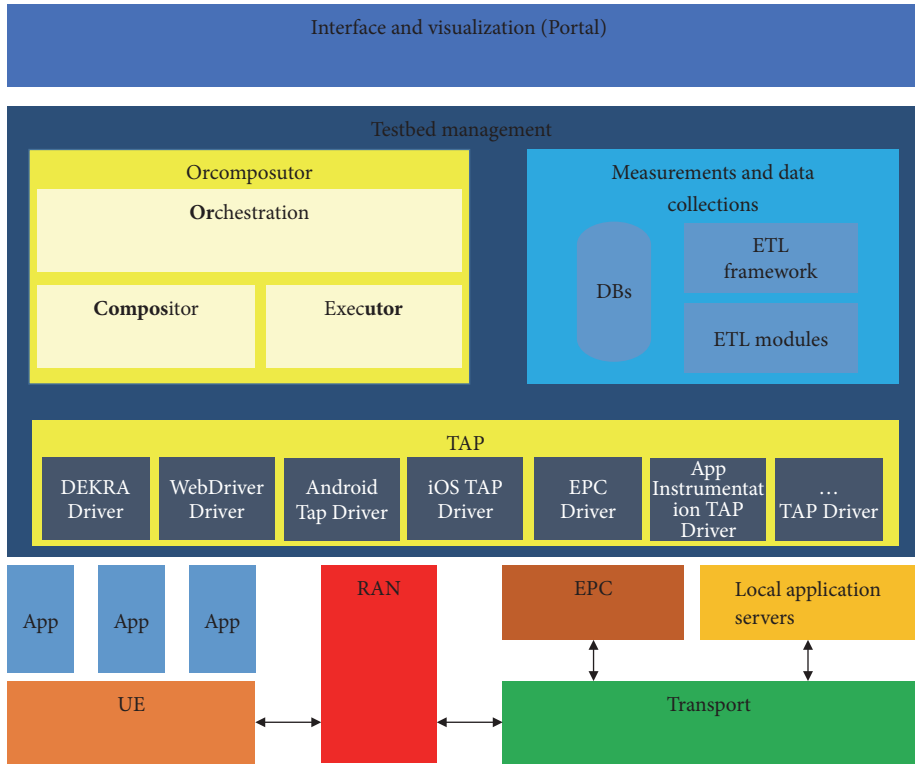


Figure 1: TRIANGLE testbed architecture.

In order to provide an end-to-end system, the testbed integrates a commercial EPC (LTE Evolved Packet Core) from Polaris Networks, which includes the main elements of a standard 3GPP compliant LTE core network, that is, MME (Mobility Management Entity), SGW (Serving Gateway), PGW (Packet Gateway), HSS (Home Subscriber Server), and PCRF (Policy and Charging Rules Function). In addition, this EPC includes the EPDG (Evolved Packet Data Gateway) and ANDSF (Access Network Discovery and Session Function) components for dual connectivity scenarios. The RAN emulator is connected to the EPC through the standard S1 interface. The testbed also offers the possibility of integrating artificial impairments in the interfaces between the core network and the application servers.

The Quamotion WebDriver, another TRIANGLE element, is able to automate user actions on both iOS and Android applications whether they are native, hybrid, of fully web-based. This tool is also used to prerecord the app's user flows, which are needed to automate the otherwise manual user actions in the test cases. This completes the full automation operation.

Finally, the testbed also incorporates commercial mobile devices (UEs). The devices are physically connected to the testbed. In order to preserve the radio conditions configured at the radio access emulator, the RAN emulator is cable conducted to the mobile device antenna connector. To accurately measure the power consumption, the N6705B power analyzer directly powers the device. Other measurement instruments may be added in the future.

4. TRIANGLE Approach

The TRIANGLE testbed is an end-to-end framework devoted to testing and benchmarking mobile applications, services, and devices. The idea behind the testing approach adopted in the TRIANGLE testbed is to generalize QoE computation and provide a programmatic way of computing it. With this approach, the TRIANGLE testbed can accommodate the computation of the QoE for any application.

The basic concept in TRIANGLE's approach to QoE evaluation is that the quality perceived by the user depends on many aspects (herein called domains) and that this perception depends on its targeted use case. For example, battery life is critical for patient monitoring applications but less important in live streaming ones.

To define the different 5G uses cases, TRIANGLE based its work in the Next Generation Mobile Network (NGMN) Alliance foundational White Paper, which specifies the expected services and network performance in future 5G networks [20]. More precisely, the TRIANGLE project has adopted a modular approach, subdividing the so-called "NGMN Use-Cases" into blocks. The name *Use Case* was kept in the TRIANGLE approach for describing the application, service, or vertical using the network services. The diversification of services expected in 5G requires a concrete categorization to have a sharp picture of what the user will be expected to interact with. This is essential for understanding which aspect of the QoE evaluation needs to be addressed. The final use cases categorization was defined in [11] and encompasses both the services normally accessible via mobile

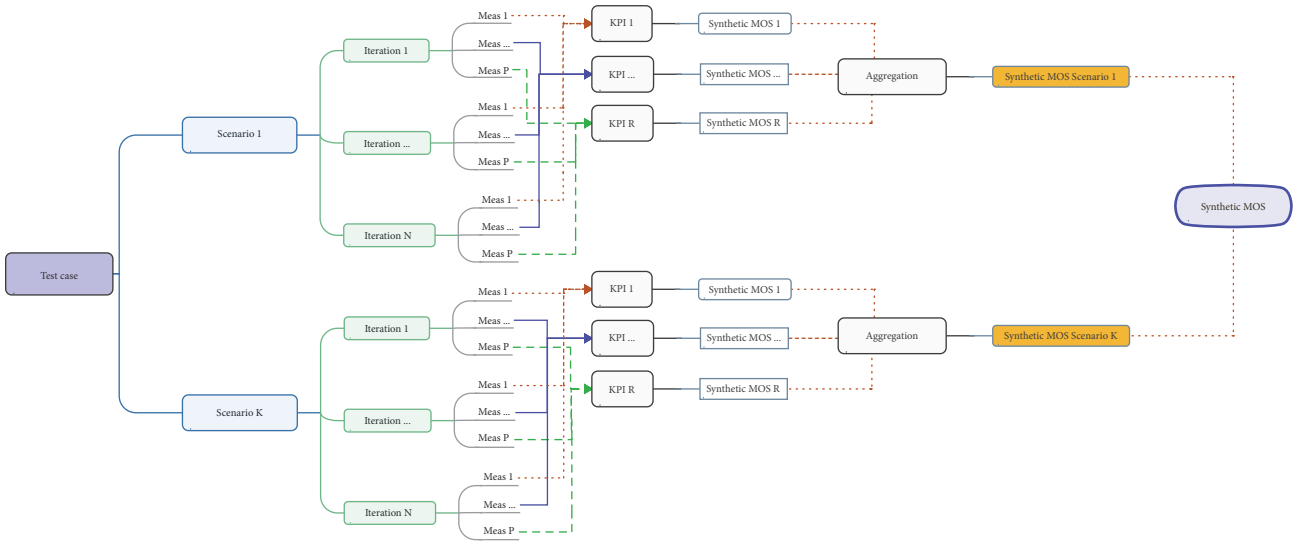


Figure 2: The process to obtain the *synthetic-MOS* score in a TRIANGLE test case.

Table 1: Uses cases defined in the TRIANGLE project.

Identifier	**Use Case**
VR	Virtual Reality
GA	Gaming
AR	Augmented Reality
CS	Content Distribution Streaming Services
LS	Live Streaming Services
SN	Social Networking
HS	High Speed Internet
PM	Patient Monitoring
ES	Emergency Services
SM	Smart Metering
SG	Smart Grids
CV	Connected Vehicles

phones (UEs) and the ones that can be integrated in, for example, gaming consoles, advanced VR gear, car units, or IoT systems.

The TRIANGLE domains group different aspects that can affect the final QoE perceived by the users. The current testbed implementation supports three of the several domains that have been identified: Apps User Experience (AUE), Apps Energy consumption (AEC), and Applications Device Resources Usage (RES).

Table 1 provides the use cases and Table 2 lists the domains initially considered in TRIANGLE.

To produce data to evaluate the QoE, a series of test cases have been designed, developed, and implemented to be run on the TRIANGLE testbed. Obviously, not all test cases are applicable to all applications under test, because not all applications need, or are designed, to support all the functionalities that can be tested in the testbed. In order to automatically determine the test cases that are applicable to an application under test, a questionnaire (identified as features questionnaire in the portal), equivalent to the classical conformance testing ICS (Implementation Conformance Statement), has been developed and is accessible through the portal. After filling the questionnaire, the applicable test plan, that is, the test campaign with the list of applicable test cases, is automatically generated.

The sequence of user actions (type, swipe, tap, etc.) a user needs to perform in the terminal (UE) to complete a task (e.g., play a video) is called the "*app user flow*." In order to be able to automatically run a test case, the actual application user flow, with the user actions a user would need to perform on the phone to complete certain tasks defined in the test case, also has to be provided.

Each test case univocally defines the conditions of execution, the sequence of actions the user would perform (i.e., the app user flow), the sequence of actions that the elements of the testbed must perform, the traffic injected, the collection of measurements to take, and so forth. In order to obtain statistical significance, each test case includes a number of executions (iterations) under certain network conditions (herein called *scenarios*). Out of the various measurements made in the different iterations under any specific network conditions (*scenario*), a number of KPIs (Key Performance Indicators) are computed. The KPIs are normalized into a standard 1-to-5 scale, as typically used in MOS scores, and referred to as *synthetic-MOS*, a terminology that has been adopted from previous works [7, 21]. The synthetic-MOS values are aggregated across network scenarios to produce a number of intermediate synthetic-MOS scores, which finally are aggregated to obtain a synthetic-MOS score in each test case (see Figure 2).

The process to obtain the final TRIANGLE mark is sequential. First, for each domain, a weighted average of the synthetic-MOS scores obtained in each test case in the domain is calculated. Next, a weighted average of the synthetic-MOS values in all the domains of a use case is calculated to provide a single synthetic-MOS value per use

Table 2: TRIANGLE domains.

Category		Identifier	Domain
Applications		AUE	Apps User experience
		AEC	Apps Energy consumption
		RES	Device Resources Usage
		REL	Reliability
		NWR	Network Resources
Devices	Mobile Devices	DEC	Energy Consumption
		DDP	Data Performance
		DRF	Radio Performance
		DRA	User experience with reference apps
	IoT Devices	IDR	Reliability
		IDP	Data Performance
		IEC	Energy consumption

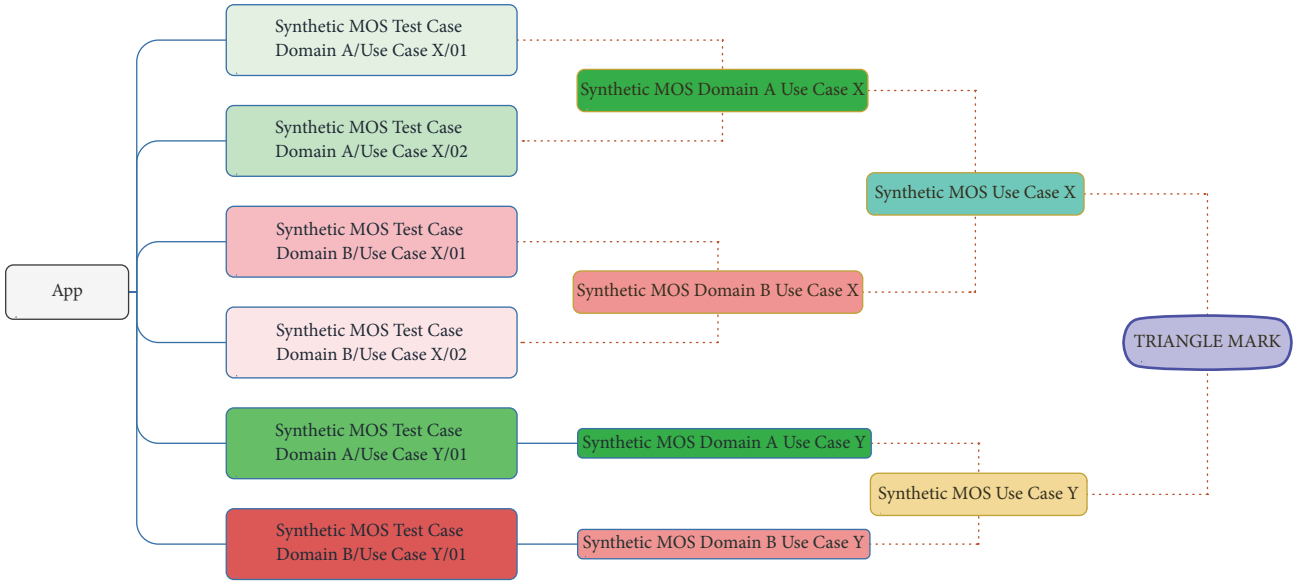


Figure 3: The process to obtain the TRIANGLE mark.

case. An application will usually be developed for one specific use case, as those defined in Table 1, but may be designed for more than one use case. In the latter case, a further weighted average is made with the synthetic-MOS scores obtained in each use case supported by the application. These sequential steps produce a single TRIANGLE mark, an overall quality score, as shown in Figure 3.

This approach provides a common framework for testing applications, for benchmarking applications, or even for certifying disparate applications. The overall process for an app that implements features of different use cases is depicted in Figure 3.

5. Details of the TRIANGLE QoE Computation

For each use case identified (see Table 1) and domain (see Table 2), a number of test cases have been developed within the TRIANGLE project. Each test case intends to test an individual feature, aspect, or behaviour of the application under test, as shown in Figure 4.

Each test case defines a number of measurements, and because the results of the measurements depend on many factors, they are not, in general, deterministic, and, thus, each test case has been designed not to perform just one single measurement but to run a number of iterations (N) of the same measurement. Out of those measurements, KPIs are computed. For example, if the time to load the first media frame is the measurement taken in one specific test case, the average user waiting time KPI can be calculated by computing the mean of the values across all iterations. In general, different use case-domain pairs have a different set of KPIs. The reader is encouraged to read [11] for further details about the terminology used in TRIANGLE.

Recommendation P.10/G.100 Amendment 1 Definition of Quality of Experience [2] notes that the overall acceptability may be influenced by user expectations and context. For the definition of the context, technical specifications ITU-T

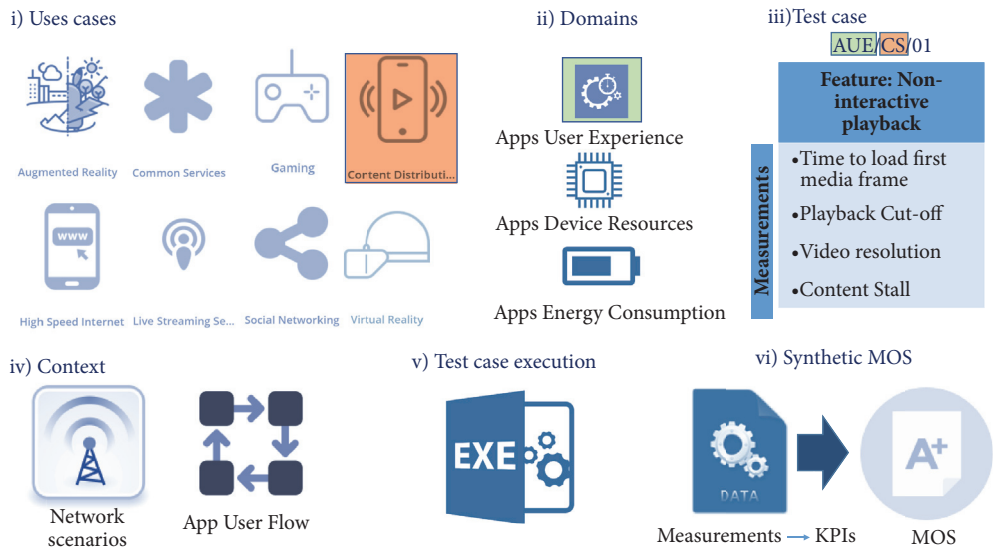


Figure 4: QoE computation steps.

G1030 "Estimating end-to-end performance in IP networks for data applications" [9] and ITU-T G1031 "QoE factors in web-browsing" [10] have been considered in TRIANGLE. In particular, ITU-T G1031 [10] identifies the following context influence factors: location (cafeteria, office, and home), interactivity (high-level interactivity versus low-level interactivity), task type (business, entertainment, etc.), and task urgency (urgent versus casual). User's influence factors are, however, outside of the scope of the ITU recommendation.

In the TRIANGLE project, the context information has been captured in the networks *scenarios* defined (Urban - Internet Cafe Off Peak; Suburban - Shopping Mall Busy Hours; Urban – Pedestrian; Urban – Office; High speed train – Relay; etc.) and in the test cases specified in [11].

The test cases specify the conditions of the test but also a sequence of actions that have to be executed by the application (app user flows) to test its features. For example, the test case that tests the "Play and Pause" functionality defines the app user flow shown in Figure 5.

The transformation of KPIs into QoE scores is the most challenging step in the TRIANGLE framework. The execution of the test cases will generate a significant amount of raw measurements about several aspects of the system. Specific KPIs can then be extracted through statistical analysis: mean, deviation, cumulative distribution function (CDF), or ratio.

The KPIs will be individually interpolated in order to provide a common homogeneous comparison and aggregation space. The interpolation is based on the application of two functions, named Type I and Type II. By using the proposed two types of interpolations, the vast majority of KPIs can be translated into normalized MOS-type of metric (*synthetic-MOS*), easy to be averaged in order to provide a simple, unified evaluation.

Type I. This function performs a linear interpolation on the original data. The variables min_{KPI} and max_{KPI} are the worst and best known values of a KPI from a reference case. The

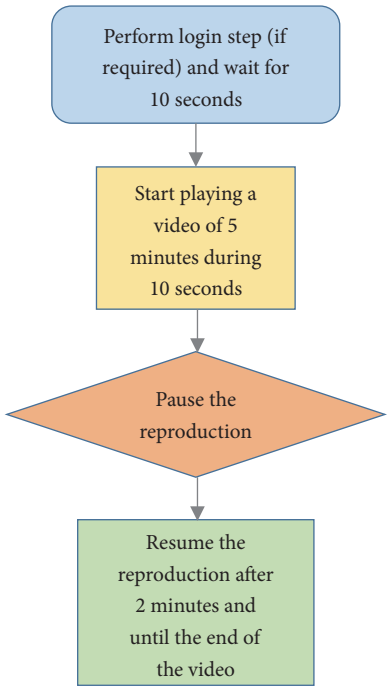


Figure 5: App user flow used in the "AUE/CS/02 Play and Pause" test case.

function maps a value, v, of a KPI, to v' (*synthetic-MOS*) in the range [1-to-5] by computing the following formula:

$$v' = \frac{v - min_{KPI}}{max_{KPI} - min_{KPI}} (5.0 - 1.0) + 1.0 \quad (1)$$

This function transforms a KPI to a synthetic-MOS value by applying a simple linear interpolation between the worst and best expected values from a reference case. If a future input case falls outside the data range of the KPI, the new value will

Table 3: AUE/CS/002 test case description.

Identifier	*AUE/CS/002 (App User Experience/Content Streaming/002)*
Title	Play and pause
Objective	Measure the ability of the AUT to pause and the resume a media file.
Applicability	(ICSG_ProductType = Application) AND (ICSG_UseCases includes CS) AND ICSA_CSPause
Initial Conditions	AUT in in [AUT_STARTED] mode. (Note: Defined in D2.2 [11] Appendix 4)
Steps	(1) The Test System commands the AUT to replay the Application User Flow (Application User Flow that presses first the Play button and later the Pause button). (2) The Test System measures whether pause operation was successful or not.
Postamble	(i) Execute the Postamble sequence (see section 2.6 in D2.2 [11] Appendix 4)
Measurements (Raw)	(i) Playback Cut-off: Probability that successfully started stream reproduction is ended by a cause other than the intentional termination by the user. (ii) Pause Operation: Whether pause operation is successful or not. (iii) Time to load first media frame (s) after resuming: The time elapsed since the user clicks resume button until the media reproduction starts. (*Note: For Exoplayer the RESUME button is the PLAY button*)

be set to the extreme value $\min_{KPI}$ (if it is worse) or $\max_{KPI}$ (if it is better).

Type II. This function performs a logarithmic interpolation and is inspired on the opinion model recommended by the ITU-T in [9] for a simple web search task. This function maps a value, v, of a KPI, to v' (*synthetic-MOS*) in the range [1-to-5] by computing the following formula:

$$v' = \frac{5.0 - 1.0}{\ln\left(\left(a * worst_{KPI} + b\right)/worst_{KPI}\right)} \bullet \left(\ln\left(v\right) - \ln\left(a * worst_{KPI} + a\right)\right) + 5 \quad (2)$$

The default values of a and b correspond to the simple web search task case (a = 0,003 and b = 0,12) [9, 22] and the worst value has been extracted from the ITU-T G1030. If during experimentation a future input case falls outside the data range of the KPI, the parameters a and b will be updated accordingly. Likewise, if through subjective experimentation other values are considered better adjustments for specific services, the function can be easily updated.

Once all KPIs are translated into synthetic-MOS values, they can be averaged with suitable weights. In the averaging process, the first step is to average over the network *scenarios* considered relevant for the use case, as shown in Figure 2. This provides the synthetic-MOS output value for the test case. If there is more than one test case per domain, which is generally the case, a weighted average is calculated in order to provide one synthetic-MOS value per domain, as depicted in Figure 3. The final step is to average the synthetic-MOS scores over all use cases supported by the application (see Figure 3). This provides the final score, that is, the TRIANGLE mark.

6. A Practical Case: Exoplayer under Test

For better understanding, the complete process of obtaining the TRIANGLE mark for a specific application, the Exoplayer, is described in this section. This application only has one use case: content distribution streaming services (CS).

Table 4: Measurement points associated with test case AUE/CS/002.

Measurements	**Measurement points**
Time to load first media frame	Media File Playback - Start Media File Playback - First Picture
Playback cut-off	Media File Playback - Start Media File Playback - End
Pause	Media File Playback - Pause

Exoplayer is an application level media player for Android promoted by Google. It provides an alternative to Android's MediaPlayer API for playing audio and video both locally and over the Internet. Exoplayer supports features not currently supported by Android's MediaPlayer API, including DASH and SmoothStreaming adaptive playbacks.

The TRIANGLE project has concentrated in testing just two of the Exoplayer features: "Noninteractive Playback" and "Play and Pause." These features result in 6 test cases applicable, out of the test cases defined in TRIANGLE. These are test cases AUE/CS/001 and AUE/CS/002, in the App User Experience domain, test cases AEC/CS/001 and AEC/CS/002, in the App Energy Consumption domain, and test cases RES/CS/001 and RES/CS/002, in the Device Resources Usage domain.

The AUE/CS/002 "Play and Pause" test case description, belonging to the AUE domain, is shown in Table 3. The test case description specifies the test conditions, the generic app user flow, and the raw measurements, which shall be collected during the execution of the test.

The TRIANGLE project also offers a library that includes the measurement points that should be inserted in the source code of the app for enabling the collection of the measurements specified. Table 4 shows the measurement points required to compute the measurements specified in test case AUE/CS/002.

Table 5: Reference values for interpolation.

Feature	Domain	KPI	Synthetic MOS Calculation	KPI_min	KPI_max
Non-Interactive Playback	AEC	Average power consumption	Type I	10 W	0.8 W
Non-Interactive Playback	AUE	Time to load first media frame	Type II	KPI_worst=20 ms	
Non-Interactive Playback	AUE	Playback cut-off ratio	Type I	50%	0
Non-Interactive Playback	AUE	Video resolution	Type I	240p	720p
Non-Interactive Playback	RES	Average CPU usage	Type I	100%	16%
Non-Interactive Playback	RES	Average memory usage	Type I	100%	40%
Play and Pause	AEC	Average power consumption	Type I	10 W	0.8 W
Play and Pause	AUE	Pause operation success rate	Type I	50%	100%
Play and Pause	RES	Average CPU usage	Type I	100%	16%
Play and Pause	RES	Average memory usage	Type I	100%	40%

The time to load first media picture measurement is obtained subtracting the timestamp of the measurement point "Media File Playback – Start" from the measurement point "Media File Playback – First Picture."

As specified in [11], all scenarios defined are applicable to the content streaming use case. Therefore, test cases in the three domains currently supported by the testbed are executed in all the *scenarios*.

Once the test campaign has finished, the raw measurement results are processed to obtain the KPIs associated with each test case: average current consumption, average time to load first media frame, average CPU usage, and so forth. The processes applied are detailed in Table 5. Based on previous experiments performed by the authors, the behaviour of the time to load the first media frame KPI resembles the web response time KPI (i.e., the amount of time the user has to wait for the service) and thus, as recommended in the opinion model for web search introduced in [9], a logarithmic interpolation (type II) has been used for this metric.

The results of the initial process, that is, the KPIs computation, are translated into *synthetics-MOS* values. To compute these values, reference benchmarking values for each of the KPIs need to be used according to the normalization and interpolation process described in Section 5. Table 5 shows what has been currently used by TRIANGLE for the App User Experience domain, which is also used by NGMN as reference in their precommercial Trials document [23].

For example, for the "time to load first media frame" KPI shown in Table 5, the type of aggregation applied is averaging and the interpolation formula used is Type II.

To achieve stable results, each test case is executed 10 times (10 iterations) in each network scenario. The synthetic-MOS value in each domain is calculated by averaging the measured synthetic-MOS values in the domain. For example, synthetic-MOS value is the RES domain obtained by averaging the synthetic-MOS value of "average CPU usage" and "average memory usage" from the two test cases.

Although Exoplayer supports several video streaming protocols, in this work only DASH [24] (Dynamic Adaptive Streaming over HTTP) has been tested. DASH clients should seamlessly adapt to changing network conditions by making decisions on which video segment to download (videos are encoded at multiple bitrates). The Exoplayer's default adaptation algorithm is basically throughput-based and some parameters control how often and when switching can occur.

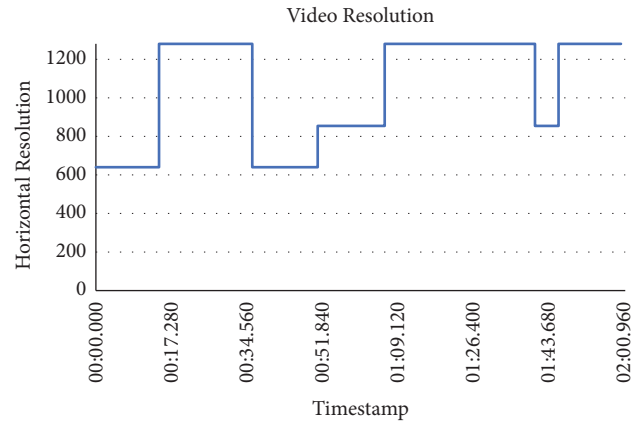


Figure 6: Video Resolution evolution in the Driving Urban Normal scenario.

During the testing, the testbed was configured with the different network scenarios defined in [11]. In these scenarios, the network configuration changes dynamically following a random pattern, resulting in different maximum throughput rates. The expected behaviour of the application under test is that the video streaming client adapts to the available throughput by decreasing or increasing the resolution of the received video. Figure 6 depicts how the client effectively adapts to the channel conditions.

However, the objective of the testing carried out in the TRIANGE testbed is not just to verify that the video streaming client actually adapts to the available maximum throughput but also to check whether this adaptation improves the users' experience quality.

Table 6 shows a summary of the synthetic-MOS values obtained per scenario in one test case of each domain. The scores obtained in the RES and AEC domains are always high. In the AUE domain, the synthetic MOS associated with the Video Resolution shows low scores in some of the scenarios because the resolution decreases, reasonable good scores in the time to load first media, and high scores in the time to playback cut-off ratio. Overall, it can be concluded that the

Table 6: Synthetic MOS values per test case and scenario for the feature "Noninteractive Playback".

	AUE domain			AEC domain	RES domain	
	Test Case AUE/CS/001			Test Case AEC/CS/001	Test Case RES/CS/001	
Scenario	Time to load first media frame	Playback Cut-off ratio	Video Resolution mode	Average Power Consumption	Average CPU Usage	Average RAM Usage
HighSpeed Direct Passenger	2.1	3.1	2.3	4.7	4.3	4.2
Suburban Festival	3.8	4.7	3.1	4.8	4.3	4.1
Suburban shopping mall busy hours	3.7	3.7	1.3	4.8	4.4	4.1
Suburban shopping mall off-peak	3.6	3.1	2.3	4.8	4.3	4.1
Suburban stadium	3.8	2.9	2.1	4.7	4.4	4.1
Urban Driving Normal	2.6	3.9	2.8	4.7	4.4	4
Urban Driving Traffic Jam	3.4	3.7	1.6	4.8	4.4	4
Urban Internet Café Busy Hours	3.8	3.7	1.9	4.8	4.4	4
Urban Internet Café Off Peak	3.8	3.1	2.3	4.8	4.3	4
Urban Office	3.8	4.7	3.3	4.8	4.5	4.3
Urban Pedestrian	3.9	2.6	2	4.7	4.4	4
	3.5	3.6	2.3	4.7	4.4	4.1

DASH implementation of the video streaming client under test is able to adapt to the changing conditions of the network, maintaining an acceptable rate of video cut-off, rebuffering times, and resources usage.

The final score in each domain is obtained by averaging the synthetic-MOS values from all the tested network scenarios. Figure 7 shows the spider diagram for the three domains tested. In the User Experience domain, the score obtained is lower than the other domains, due to the low synthetic-MOS values obtained for the video resolution.

The final synthetic MOS for the use case Content Distribution Streaming is obtained as a weighted average of the three domains, representing the overall QoE as perceived by the user. The final score for the Exoplayer version 1.516 and the features tested (Noninteractive Playback and Play and Pause) is 4.2, which means that the low score obtained in the video resolution is compensated with the high scores in other KPIs.

If an application under test has more than one use case, the next steps in the TRIANGLE mark project approach would be the aggregation per use case and the aggregation over all use cases. The final score, the TRIANGLE mark, is an estimation of the overall QoE as perceived by the user.

In the current TRIANGLE implementation, the weights in all aggregations are the same. Further research is needed to appropriately define the weights of each domain and each use case in the overall score of the applications.

7. Conclusions

The main contribution of the TRIANGLE project is the provision of a framework that generalizes QoE computation and enables the execution of extensive and repeatable test campaigns to obtain meaningful QoE scores. The TRIANGLE project has also defined a methodology, which is based on the transformation and aggregation of KPIs, its transformation into synthetic-MOS values, and its aggregation over the different *domains* and *use cases*.

The TRIANGLE approach is a methodology flexible enough to generalize the computation of QoE for any application/service. The methodology has been validated testing the DASH implementation in the Exoplayer App. To confirm the suitability of the weights used in the averaging process and the interpolation parameters, as well as to verify the correlation of the obtained MOS with that scored by users, the authors have started experiments with real users and initial results are encouraging.

The process described produces a final TRIANGLE mark, a single quality score, which could eventually be used to certify applications after achieving a consensus on the different values of the process (weights, limits, etc.) to use.

Data Availability

The methodology and results used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

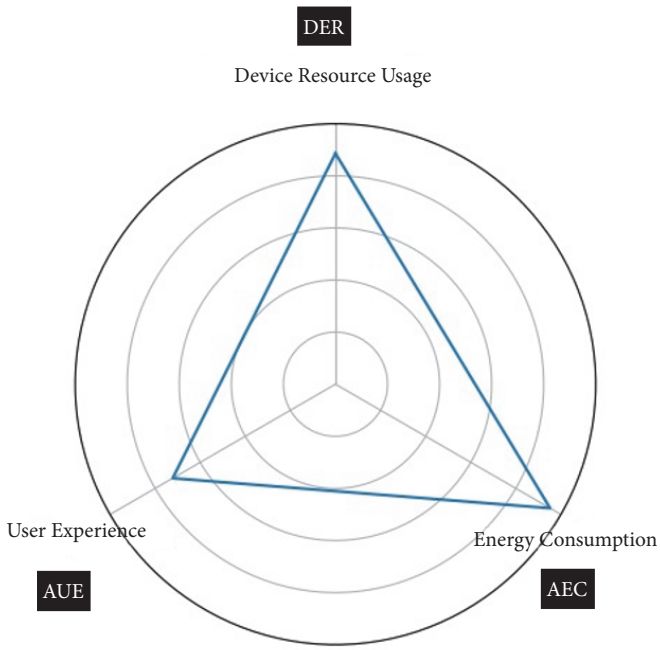


Figure 7: Exoplayer synthetic-MOS values per domain.

Acknowledgments

The TRIANGLE project is funded by the European Union's Horizon 2020 Research and Innovation Programme (Grant Agreement no. 688712).

References

[1] ETSI, "Human factors: quality of experience (QoE) requirements for real-time communication services," Tech. Rep. 102 643, 2010.

[2] ITU-T, "P.10/G.100 (2006) amendment 1 (01/07): new appendix I - definition of quality of experience (QoE)," 2007.

[3] F. Kozamernik, V. Steinmann, P. Sunna, and E. Wyckens, "SAMVIQ - A new EBU methodology for video quality evaluations in multimedia," *SMPTE Motion Imaging Journal*, vol. 114, no. 4, pp. 152–160, 2005.

[4] ITU-T, "G.107 : the E-model: a computational model for use in transmission planning," 2015.

[5] J. De Vriendt, D. De Vleeschauwer, and D. C. Robinson, "QoE model for video delivered over an LTE network using HTTP adaptive streaming," *Bell Labs Technical Journal*, vol. 18, no. 4, pp. 45–62, 2014.

[6] S. Jelassi, G. Rubino, H. Melvin, H. Youssef, and G. Pujolle, "Quality of Experience of VoIP Service: A Survey of Assessment Approaches and Open Issues," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 491–513, 2012.

[7] M. Li, C.-L. Yeh, and S.-Y. Lu, "Real-Time QoE Monitoring System for Video Streaming Services with Adaptive Media Playout," *International Journal of Digital Multimedia Broadcasting*, vol. 2018, Article ID 2619438, 11 pages, 2018.

[8] S. Baraković and L. Skorin-Kapov, "Survey and Challenges of QoE Management Issues in Wireless Networks," *Journal of Computer Networks and Communications*, vol. 2013, Article ID 165146, 28 pages, 2013.

[9] ITU-T, "G.1030: estimating end-to-end performance in IP networks for data applications," 2014.

[10] ITU-T, "G.1031 QoE factors in web-browsing," 2014.

[11] EU H2020 TRIANGLE Project, *Deliverable D2.2 Final report on the formalization of the certification process, requirements and use cases*, 2017, https://www.triangle-project.eu/project-old/deliverables/.

[12] Q. A. Chen, H. Luo, S. Rosen et al., "QoE doctor: diagnosing mobile app QoE with automated UI control and cross-layer analysis," in *Proceedings of the Conference on Internet Measurement Conference (IMC '14)*, pp. 151–164, ACM, Vancouver, Canada, November 2014.

[13] M. A. Mehmood, A. Wundsam, S. Uhlig, D. Levin, N. Sarrar, and A. Feldmann, "QoE-Lab: Towards Evaluating Quality of Experience for Future Internet Conditions," in *Testbeds and Research Infrastructure*, Korakis T., Li H., Tran-Gia P., and H. S. Park, Eds., vol. 90 of *TridentCom 2011, Lnicst*, pp. 286–301, Springer, Development of Networks and Communities, Berlin, Germany, 2012.

[14] D. Levin, A. Wundsam, A. Mehmood, and A. Feldmann, "Berlin: The Berlin Experimental Router Laboratory for Innovative Networking," in *TridentCom 2010. Lnicst*, T. Magedanz, A. Gavras, N. H. Thanh, and J. S. Chase, Eds., vol. 46 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 602–604, Springer, Heidelberg, Germany, 2011.

[15] K. De Moor, I. Ketyko, W. Joseph et al., "Proposed framework for evaluating quality of experience in a mobile, testbed-oriented living lab setting," *Mobile Networks and Applications*, vol. 15, no. 3, pp. 378–391, 2010.

[16] R. Sanchez-Iborra, M.-D. Cano, J. J. P. C. Rodrigues, and J. Garcia-Haro, "An Experimental QoE Performance Study for the Efficient Transmission of High Demanding Traffic over an Ad Hoc Network Using BATMAN," *Mobile Information Systems*, vol. 2015, Article ID 217106, 14 pages, 2015.

[17] P. Oliver-Balsalobre, M. Toril, S. Luna-Ramírez, and R. García Garaluz, "A system testbed for modeling encrypted video-streaming service performance indicators based on TCP/IP metrics," *EURASIP Journal on Wireless Communications and Networking*, vol. 2017, no. 1, 2017.

[18] M. Solera, M. Toril, I. Palomo, G. Gomez, and J. Poncela, "A Testbed for Evaluating Video Streaming Services in LTE," *Wireless Personal Communications*, vol. 98, no. 3, pp. 2753–2773, 2018.

[19] A. Álvarez, A. Díaz, P. Merino, and F. J. Rivas, "Field measurements of mobile services with Android smartphones," in *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC '12)*, pp. 105–109, Las Vegas, Nev, USA, January 2012.

[20] NGMN Alliance, "NGMN 5G white paper," 2015, https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf.

[21] "Infrastructure and Design for Adaptivity and Flexibility," in *Mobile Information Systems*, Springer, 2006.

[22] J. Nielsen, "Response Times: The Three Important Limits," in *Usability Engineering*, 1993.

[23] NGMN Alliance, "Definition of the testing framework for the NGMN 5G pre-commercial networks trials," 2018, https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/180220_NGMN_PreCommTrials_Framework_definition_v1_0.pdf.

[24] 3GPP TS 26.246, "Transparent end-to-end Packet-switched Streaming Services (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH)," 2018.

Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7659085, 11 pages
https://doi.org/10.1155/2018/7659085

Research Article

Packet Scheduling for Multiple-Switch Software-Defined Networking in Edge Computing Environment

Hai Xue, Kyung Tae Kim, and Hee Yong Youn

College of Information and Communication Engineering, Sungkyunkwan University, Suwon, Republic of Korea

Correspondence should be addressed to Hee Yong Youn; youn7147@skku.edu

Received 23 August 2018; Revised 10 October 2018; Accepted 31 October 2018; Published 18 November 2018

Guest Editor: Jorge Navarro-Ortiz

Software-defined networking (SDN) decouples the control plane and data forwarding plane to overcome the limitations of traditional networking infrastructure. Among several communication protocols employed for SDN, OpenFlow is most widely used for the communication between the controller and switch. In this paper two packet scheduling schemes, FCFS-Pushout (FCFS-PO) and FCFS-Pushout-Priority (FCFS-PO-P), are proposed to effectively handle the overload issue of multiple-switch SDN targeting the edge computing environment. Analytical models on their operations are developed, and extensive experiment based on a testbed is carried out to evaluate the schemes. They reveal that both of them are better than the typical FCFS-Block (FCFS-BL) scheduling algorithm in terms of packet wait time. Furthermore, FCFS-PO-P is found to be more effective than FCFS-PO in the edge computing environment.

1. Introduction

Nowadays, the traditional IP network can hardly deal with the huge volume of traffic generated in the rapidly growing Internet of Things (IoT). As an efficient solution to this issue, a new type of networking paradigm called software-defined networking (SDN) had been proposed [1]. In SDN the control plane and data plane of the network are decoupled, unlike the traditional network. The role of the switches in SDN is delivering the data packets, while the controller is installed separately for controlling the whole network. OpenFlow [2] is one of the most popular protocols developed for the communication between the controller and switch in SDN, and it is only an open protocol [3].

The proliferation of IoT and the success of rich cloud service induced the horizon of a new computing paradigm called edge computing [4]. Here how to efficiently deliver the data from the IoT nodes to the relevant switches is a key issue. While there exist numerous nodes in the edge computing environment, some of them are deployed to detect critical events such as fire or earthquakes. For such peculiar edge nodes, the data must be delivered with the highest priority due to its importance. Therefore, effective priority-based scheduling and a robust queueing mechanism are imperative to maximize the performance of the network where the data processing occurs at the edge of the network. A variety of network traffics are also needed to be considered for accurately estimating the performance [5], allowing early identification of a potential traffic hotspot or bottleneck. This is a fundamental issue in the deployment of SDN for edge computing.

Recently, numerous studies have been conducted to maximize the performance of the controller and OpenFlow switch of SDN. However, to the best of our knowledge, little explorations exist for the performance of SDN switches with priority scheduling. This is an undoubtedly important issue for edge computing since the edge nodes deal with the received data based on the priority. Furthermore, priority scheduling is necessary with multiple switches. In this paper, thus, the priority scheduling scheme for multiple-switch SDN is proposed. Here packet-in messages might be sent to the controller from each of the switches. On the basis of the $M/G/1$ queueing model for the OpenFlow switch, the First Come First Served-Pushout (FCFS-PO) and First Come First Served-Pushout-Priority (FCFS-PO-P) mechanism are proposed for efficiently handling the overload issue. With FCFS-PO, the packets are served based on the order of arrival, while the oldest one is pushed out when the buffer is full.

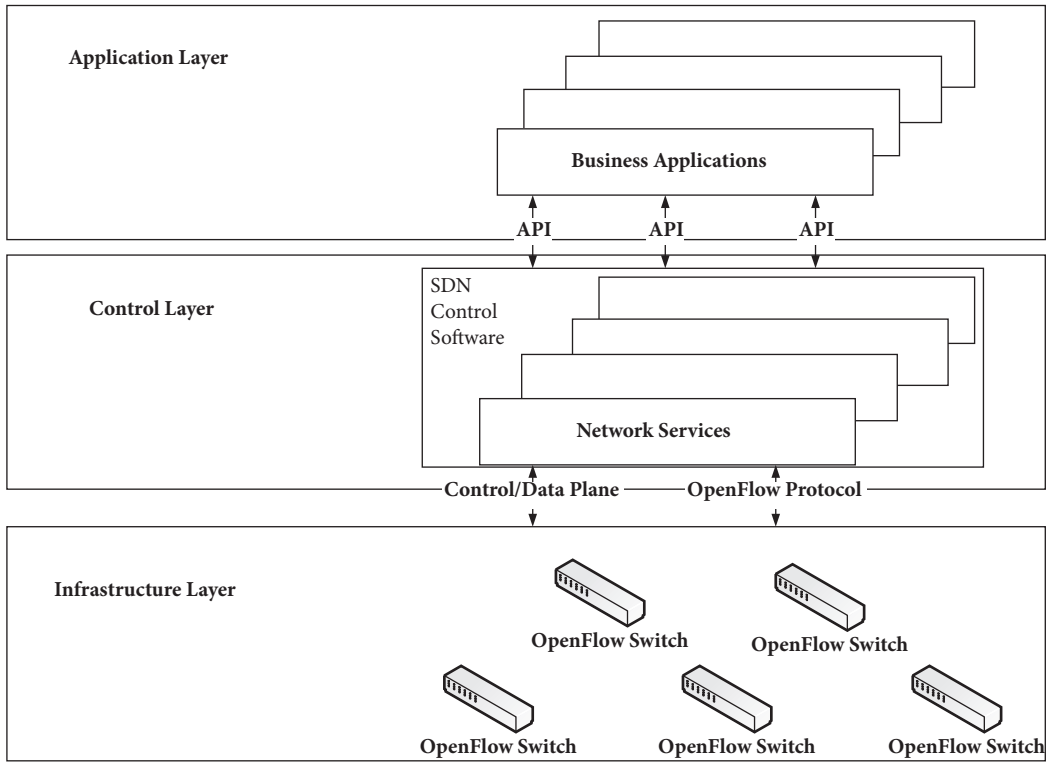


Figure 1: The three-layer structure of SDN.

With FCFS-PO-P, the packets are serviced by the order of the priority on top of the FCFS-PO. Extensive experiment on a testbed reveals that FCFS-PO-P scheduling allows better performance than FCFS-PO in terms of sojourn and wait time for various traffic conditions. Furthermore, both of them display much smaller wait time than the typical FCFS-Block (FCFS-BL) scheduling. The main contributions of the paper are summarized as follows:

(i) The packet scheduling issue is identified with the SDN controller in edge computing environment. The FCFS-PO and FCFS-PO-P scheduling scheme are proposed for coping with this issue

(ii) Analytical model of the proposed scheduling scheme is developed using queueing theory, which can be applied to the evaluation of priority-based scheduling scheme

(iii) The scheduling for solving the packet scheduling problem of SDN controller in edge computing environment is revealed by comparing the proposed method with existing methods

The rest of the paper is structured as follows. Section 2 provides an overview of SDN and edge computing. In Section 3 the priority-based scheduling schemes for multiple-switch SDN are proposed along with their analytical modeling. Section 4 evaluates the performance of the proposed schemes. At last, Section 5 concludes the paper and outlines the future research direction.

2. Related Work

SDN decouples the control logic from the underlying routers and switches. It promotes the logical centralization of network control and introduces the capability of programming the network [6–8]. As a result, flexible, dynamic, and programmable functionality of network operations could be offered. However, the merits are achieved with the sacrifice on essential network performance such as packet processing speed and throughput [9], which is attributed to the involvement of a remote controller for the administration of all forwarding devices.

2.1. SDN. SDN was originated from a project of UC Berkeley and Stanford University [10]. In SDN the network control plane making decisions on traffic routing and load balancing is decoupled from the data plane forwarding the traffic to the destination. The network is directly programmable, and the infrastructure is allowed to be abstracted for flexibly supporting various applications and services. The experts and vendors claim that this greatly simplifies the networking task [11]. There exist three layers in SDN unlike the traditional network of two layers, and a typical structure of SDN based on the OpenFlow protocol is depicted in Figure 1.

The separation of control plane and data plane can be realized by means of a well-defined programming interface between the controller and switches. The controller exercises direct control covering the state of the data plane elements via a well-defined application programming interface (API) as shown in Figure 1. The most notable example of such API is

Table 1: The fields of an entry of a flow table.

Field	Description
Match	Port, packet header, and metadata forwarded from the previous flow table
Priority	Matching precedence of the entry
Counter	Statistics for matching the packets
Instruction	Action or pipeline processing
Timeout	Maximum effective time or free time before the entry is overdue
Cookie	Opaque data sent by the OpenFlow controller

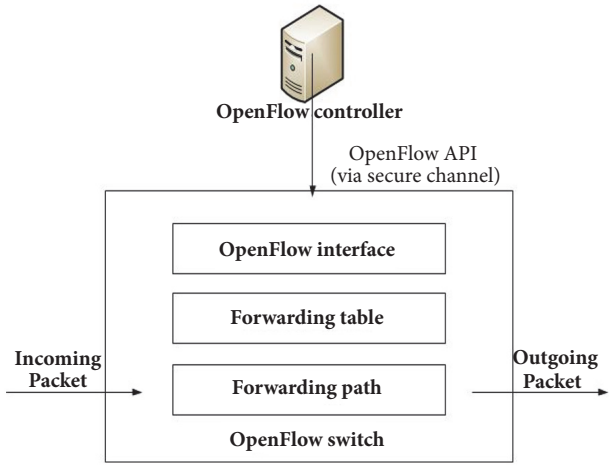


Figure 2: The structure of the OpenFlow protocol.

OpenFlow. There exists one or more tables of packet-handling rules (flow table) in an OpenFlow switch. Each entry of a flow table consists of mainly six fields as listed in Table 1. The rule is matched with a subset of the traffic, and proper actions such as dropping, forwarding, and modifying are applied to the traffic. The flow tables are used to determine how to deal with the incoming packets.

The communication protocol between the controller and switches is particularly important due to the decoupling of the two planes. The OpenFlow protocol is widely used for SDN, which defines the API for the communication between them as Figure 2 illustrates.

In SDN the controller manipulates the flow tables of the switch by adding, updating, and deleting the flow entries. The operation occurs either reactively as the controller receives a packet from the switch or proactively according to the implementation of the OpenFlow controller. A secure channel is supported for the communication between the controller and switch. The OpenFlow switch supports the flow-based forwarding by keeping one or more flow tables [12, 13].

In Xiong et al. [14] a queueing model was proposed for estimating the performance of the SDN controller with the input of a hybrid Poisson stream of packet-in messages. They also modeled the packet forwarding of OpenFlow switches in terms of packet sojourn time [15]. In Sood et al. [16] an analytical model for an SDN switch was developed, which includes the key factors such as flow-table size, packet arrival rate, number of rules, and position of rules. Ma et al. [17] focused on delay estimation with real traffic and end-to-end delay control. A model was proposed in Mahmood et al. [18] which approximates multiple nodes of the data plane as an open Jackson network with the controller. A hybrid routing forwarding scheme as well as a congestion control algorithm was proposed in Shen et al. [19] to solve the traffic scheduling and load balancing problem in software-defined mobile wireless networks. In Miao et al. [20] the performance of SDN was investigated using the Markov Modulated Poisson Process (MMPP) in the presence of bursty and correlated arrivals.

2.2. Edge Computing. With the new era of computing paradigm called edge computing, data are transferred to the network edge for the service including analytics. As a result, computing occurs close to the data sources [21]. The number of sensor nodes deployed on the surroundings is rapidly increasing due to the prevalence of IoT in recent years. Edge computing is an emerging ecosystem which aims at converging telecommunication and IT services, providing a cloud computing platform at the edge of the whole network. It offers storage and computational resources at the edge, reducing the latency and increasing resource utilization. A simplified structure of edge computing is depicted in Figure 3.

As the futuristic vision of IoT, the latest development of SDN could be integrated with edge computing to provide more efficient service. Here how to make the edge nodes more efficient in processing the data is a matter of great concern. Besides, it is essential to accurately estimate the performance of the OpenFlow switch for better usage. While more than thousands of edge nodes may exist in the real environment, some of them are deployed in the critical area whose data must be processed with high priority. Therefore, priority-based scheduling is inevitable to dynamically control the operation of the network considering the property of the traffics.

3. The Proposed Scheme

In this section the proposed scheduling schemes for SDN having multiple switches and the analytical models of the performance are presented. First, the priority scheduling scheme with a single switch and controller is presented. Then, the FCFS-PO and FCFS-PO-P scheme based on multiple switches are introduced which can effectively handle the switch overload issue.

3.1. Priority Scheduling with Single Switch

3.1.1. Basic Structure. SDN allows easy implementation of a new scheme on the existing network infrastructure by decoupling the control plane from the data plane and connecting them via an open interface. The SDN of a single switch and controller is depicted in Figure 4.

Both the controller and switch support the OpenFlow protocol for the communication between them. When a

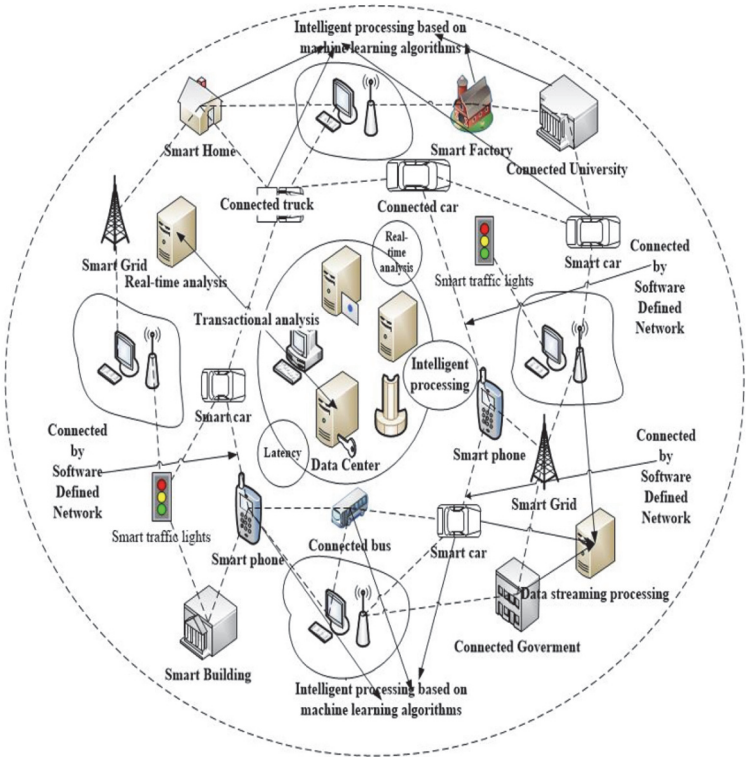


Figure 3: A simplified structure of edge computing.

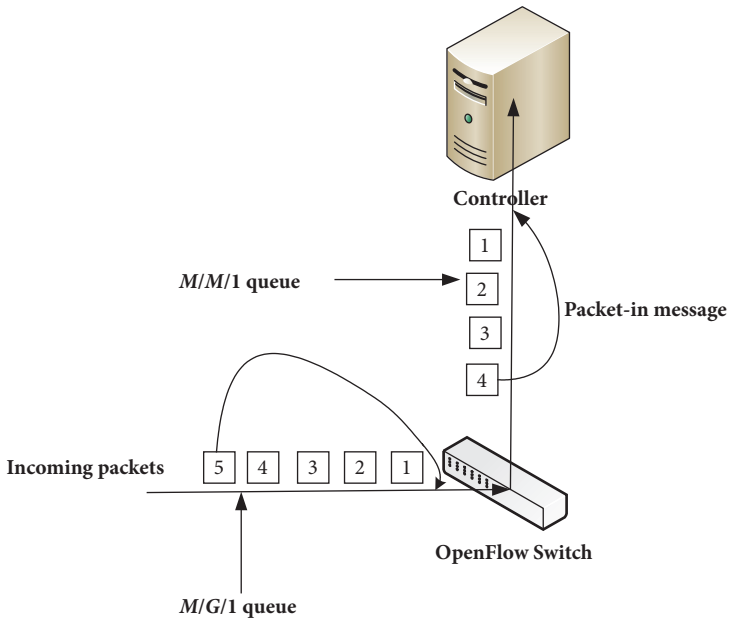


Figure 4: The structure with a single switch and controller.

packet arrives at the OpenFlow switch, the switch performs lookup with its flow tables. If a table entry matches, the switch forwards the packet in the conventional way. Otherwise, the switch requests the controller for the instruction by sending a packet-in message, which encapsulates the packet information. The controller then determines the rule for it, which is installed in other switches. After that, all the packets belonging to the flow are forwarded to the destination without requesting the controller again [15]. The process of packet matching operation is illustrated in Figure 5. Notice from the figure that both the incoming packets and packet-in messages are queued. Here the incoming packets and packet-in

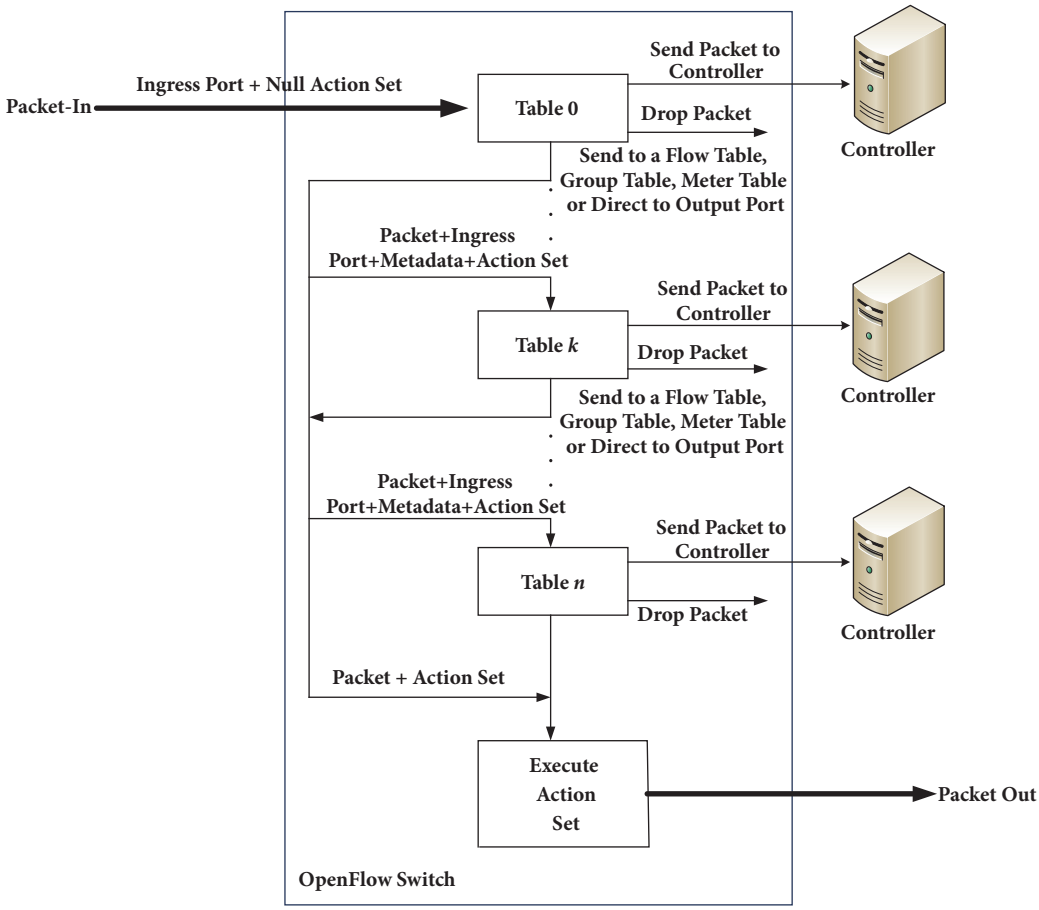


FIGURE 5: The process of packet matching.

messages are handled with the $M/G/1$ model and $M/M/1$ model, respectively, as in the existing researches [14, 16].

3.1.2. Priority-Based Scheduling. Refer to Figure 5. The incoming packets to the OpenFlow switch are queued, and they are scheduled by either the FCFS or FCFS with priority (FCFS-P) scheduling policy. FCFS is applied for regular traffic, while FCFS-P is applied for urgent packets.

In the FCFS $M/G/1$ queueing system, the average sojourn time of a packet consists of two components. The first component is the time required for the processing of the packets which are waiting in the queue at the time of the packet arrival, and the second one is the time due to the packet which is in service. The second component is nonzero if the system is busy, while it is zero if the system is empty. Considering the two components, the average sojourn time of a packet with FCFS scheduling is obtained as follows. With the $M/G/1$ queue, λ is the arrival rate of the Poisson process and $\overline{x}$ is average service time. The models in this subsection can be referred to in [22].

$$S = N_q \overline{x} + \rho \frac{\overline{x}^2}{2\overline{x}}. \tag{1}$$

Here N_q is the number of packets in the queue, ρ is the probability that the queue is busy, and $x^2/(2\overline{x})$ is the average residual lifetime of the service. By adopting Little's equation and the P-K (Pollaczek and Khinchin) equation, (1) can be expressed as follows:

$$S = x + \frac{\lambda \overline{x}^2}{2(1-\rho)}. \tag{2}$$

The model for FCFS-P is different from the FCFS $M/G/1$ queue due to the out-of-order operation. There exist several classes of packets having different Poisson arrival rates and service times. Assuming that there are m classes, $\overline{x}_i$ and λ_i are average service time and arrival rate of class-i, respectively. The time fraction of the system working on class-i is then $\rho_i = \lambda_i \overline{x}_i$. The packets are processed according to the priority with preemptive node, while the class of a smaller number is given higher priority.

In the $M/G/1$ queueing system of FCFS-P, the average sojourn time of an incoming packet consists of three components. The first component is the time taken for the packet currently in service to be finished, S_0. The second one is the time taken for the service of the packets waiting in the queue whose priority is higher than or equal to it. The third component is for the service of the packets arriving after itself but having higher priority. The number of packets of class-m is $\lambda_m S_m$ according to Little's result theorem [23]. Hence,

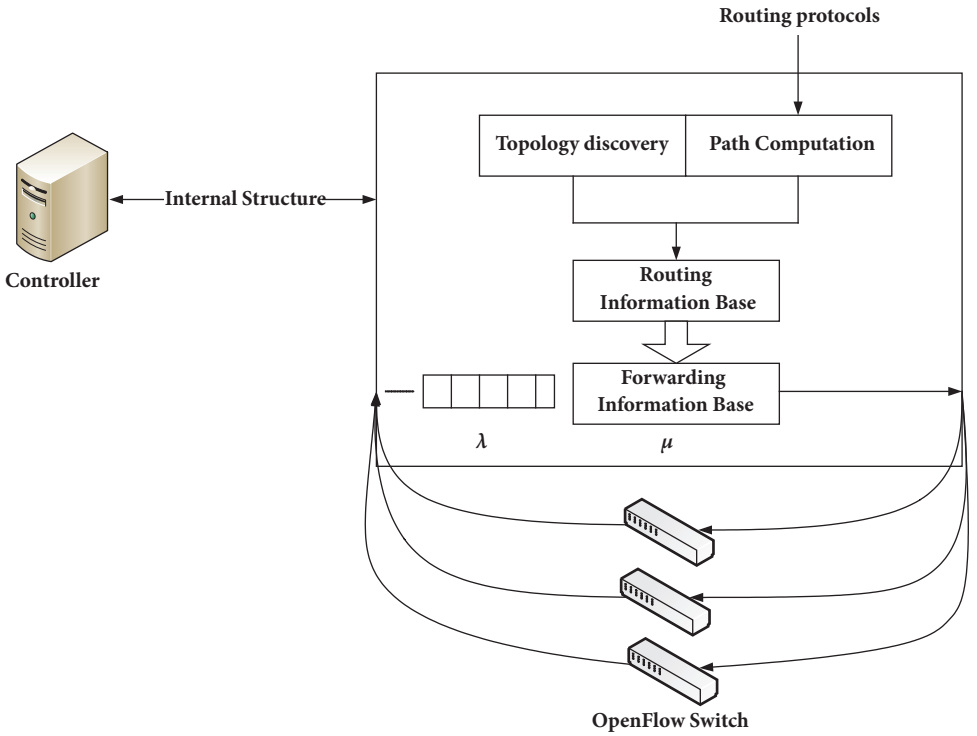


Figure 6: The processing of packet-in messages in the SDN controller.

the sojourn time of the FCFS-P $M/G/1$ queueing system is as follows:

$$S_m = S_0 + \sum_{i=1}^{m} \overline{x}_i \left(\lambda_i S_i\right) + \sum_{i=1}^{m-1} \overline{x}_i \left(\lambda_i S_i\right). \tag{3}$$

Equation (3) can be solved by recursively applying the equation as follows:

$$\sigma_m = \sum_{i=1}^{m} \rho_i. \tag{4}$$

$$S_m = \frac{S_0}{\left(1 - \sigma_m\right)\left(1 - \sigma_{m-1}\right)} \tag{5}$$

With preemptive scheduling, S_0 is due to the packets of higher priority. Therefore, it can be expressed as follows:

$$S_0 = \sum_{i=1}^{m} \rho_i \frac{\overline{x}_i^2}{2\overline{x}_i}. \tag{6}$$

3.1.3. Packet-In Messages. As previously stated, the OpenFlow switch sends a packet-in message to the relevant controller as a flow setup request when a packet fails to match the flow table. The diagram of the processing of the packet-in message is illustrated in Figure 6. Notice that the process of packet-in message has a one-to-one correspondence with the process of flow arrival regardless of the number of switches connected to the controller. Each flow arrival is assumed to follow the Poisson distribution. Therefore, the packet-in messages coming from several switches constitute a Poisson stream by the superposition theorem [15].

3.2. Priority Scheduling with Multiple Switches

3.2.1. Basic Structure. Since SDN decouples the control plane and data forwarding plane, the data transmission consists of two types. One is from the edge nodes to the switches, and the other one is from a switch to the remote controller for the packet-in message. They are illustrated in Figure 7.

There is at least one flow table in an OpenFlow switch, and the incoming packets are matched from the first flow table. Upon a match with a flow entry, the instruction set of the flow entry is executed. On table miss, there are three options: dropping, forwarding to the subsequent flow tables, and transmitting to the controller as a packet-in message. One option is selected by the programmer when the flow tables are sent to the OpenFlow switches.

3.2.2. FCFS-PO Scheduling. With FCFS-PO, if the buffer is full when a packet enters, it is put at the tail of the queue while the packet at the head is pushed out. All the packets move forward one position when a packet is served. Here the position of a packet in the buffer is decided by the number of waiting packets.

The FCFS-PO mechanism is modeled assuming that the packets arriving at the switch follow the Poisson arrival with the rate of $\lambda(\lambda > 0)$. This means that the interarrival times are independent and follow exponential distribution. The packet service times are also independent and follow general

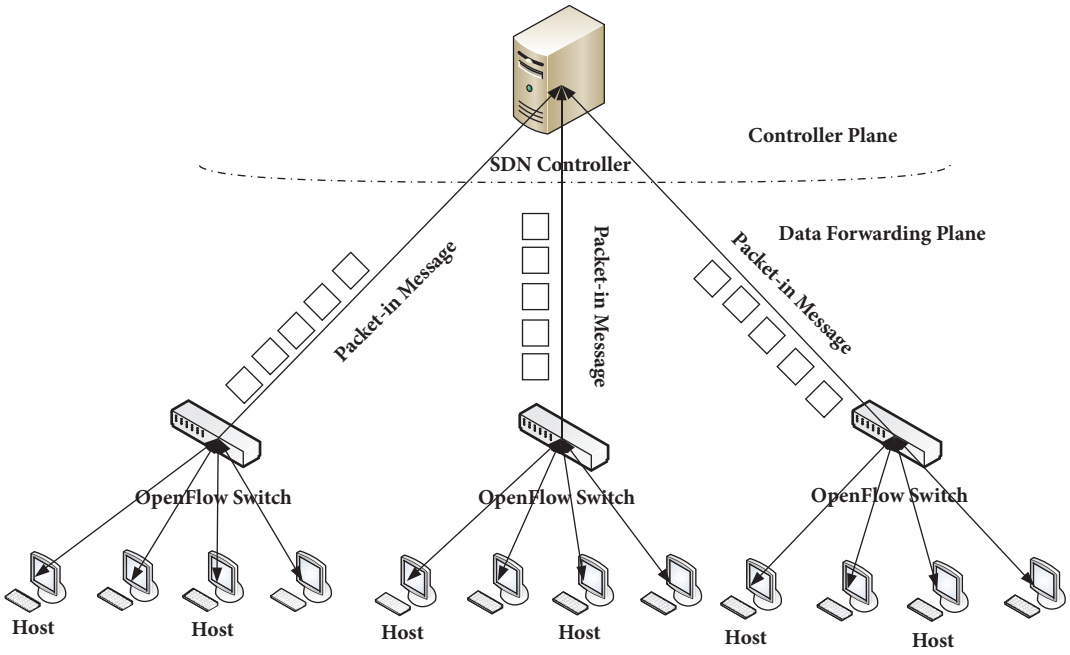


Figure 7: The structure of multiple-switch SDN.

distribution $(1/\mu)$. Also, assume that the system can contain N packets at most and the buffer size is $(N-1)$. At last, the arrival process and service process are assumed to be independent.

Assume that $Q_i(1 \le i \le (N-1))$ is the steady state probability of the $M/G/1/N$ queue, while $P_i(1 \le i \le (N-1))$ represents the steady state probability of the $M/G/1$ queue. Q_i is obtained as follows using the approach employed in [24]:

$$Q_i = \frac{P_i}{\sum_{i=0}^{N-1} P_i}, \quad 1 \le i \le (N-1). \tag{7}$$

$Q_a(1 \le a \le N)$ is the steady state probability of a packets already in the system when a new packet arrives.

Then, according to the Poisson Arrivals See Time Averages (PASTA) character [24], the following equation can be obtained:

$$\begin{aligned} Q_a &= \frac{Q_a}{Q_0 + \rho}, \quad a = 0, 1, 2, \ldots, (N-1) \\ Q_N &= 1 - \frac{1}{Q_0 + \rho}, \quad a = N. \end{aligned} \tag{8}$$

Here $\rho = (\lambda/\mu)$. And two performance indicators of the queue in the steady state are dealt with as follows:

(i) The probability of the packets to get the service, P_x,

$$P_x = \frac{1}{(Q_0 + \rho)} \tag{9}$$

(ii) The packet loss rate of the system, P_l,

$$P_l = 1 - P_x = 1 - \frac{1}{Q_0 + \rho} \tag{10}$$

With the FCFS-PO mechanism, when the buffer is full of packets, the incoming packet will be put at the end, and the buffer management policy will push out the head-of-line (HOL) packet for the newly incoming packet. Let us denote by $P_s(a,b)$ the steady state probability for the packet in state (a,b), and it finally gets the service when a packet leaves the system. When $a = 1$, the packet is in service. When $2 \le a \le N$ and $0 \le b \le (N-a)$, $p_s(a,b)$ needs to wait for the service. Assume that there are m incoming packets while a packet is in service. Then, $m \le (N-b-2)$ for $p_s(a,b)$ to get the service. If $m \le (N-a-b)$, there is no packet to be pushed out. Then, the position of the packet is changed from (a,b) to $(a-1, b+m)$. The buffer is full while $(N-a-b)+1 \le m \le (N-b-2)$, and the incoming packet pushes out the packet at position_2. Then, the position of the packet is changed from (a,b) to $(N-m-b-1, b+m)$. Finally, the state-transition equation of $P_s(a,b)$ is obtained as follows [25]:

$$\begin{aligned} &P_s(a,b) \\ &\quad = \prod \alpha_m \bullet (m \mid (a,b)) \\ &\qquad + \sum_{m=N-a-b+1}^{N-b-2} \alpha_m \bullet P_s(N-m-b-1, b+m). \end{aligned} \tag{11}$$

Here α_m can be obtained from the result of imbedded Markov points in [26]. Then, the average waiting time is analyzed. The following Boolean function is used to show if the packet was served or not:

$$L = \begin{cases} 0 & \textit{Served} \\ 1 & \textit{Not served}. \end{cases} \tag{12}$$

Using (9) and (10), the probability of $P(L = 0)$ and $P(L = 1)$ functions can be easily obtained.

Let B_n $(n \geq 1)$ represent the number of incoming packets while n packets are being served, and $R(t)$ is the remaining time of the current packet in service. The Laplace-Stieltjes Transform (LST) of b_a is obtained as follows:

$$b_a(\theta) = \left[\int_0^\infty \frac{(\lambda c)^a}{a!} e^{-(\lambda+\theta)c} dR(t) \mid \frac{Q_n + \rho - 1}{Q_n + \rho}\right] \cdot \left(\frac{Q_n + \rho - 1}{Q_n + \rho}\right). \tag{13}$$

Denote by $H(a,b)$ the average waiting time for $P_s(a,b)$ until it finally gets the service, and the waiting time is no longer than t. Then, the probability of $H(a,b,\theta)$ is shown as follows:

$$H(a,b,\theta) = \int_0^\infty e^{-\theta t} dH(a,b,\theta). \tag{14}$$

So, $H(a,b)$ can be obtained:

$$H(a,b) = -\lim_{\theta \longrightarrow 0} H(a,b,\theta). \tag{15}$$

The conditional average waiting time for $p_s(a,b)$ is as follows:

$$\begin{aligned} T = \rho \bullet \sum_{a=1}^{N-1} \sum_{b=0}^{N-a-1} Q_a \Big[& b_a \bullet H(a,b) + P_s(a,b) \\ & \bullet \frac{(b+1)}{\lambda} \bullet b_{b+1} \Big] + \rho \bullet \sum_{a=1}^{N-1} \sum_{b=N-a}^{N-2} Q_a \Big[b_a \\ & \bullet H(N-b-1,b) + P_s(N-b-1,b) \\ & \bullet \frac{(b+1)}{\lambda} b_{b+1} \Big] + \rho \bullet \sum_{b=0}^{N-2} Q_N \Big[b_a H(N-b-1,b) \\ & + P_s(N-b-1,b) \bullet \frac{(b+1)}{\lambda} \bullet b_{b+1} \Big]. \end{aligned} \tag{16}$$

Here Q_a and $b_a (1 \leq a \leq N)$ are given by (8) and (13), respectively. $H(a,b)$ is given by (15).

3.2.3. FCFS-PO-P Scheduling. In the modeling of FCFS-PO-P, the arrival process and service process of the incoming packets are the same as the modeling of FCFS-PO. In this paper, assume that the newly incoming packet has the highest priority among the packets already in the queue. Then, the newly incoming packet will be put in the front position to get service first. And the buffer management policy pushes out the packet in position_N which waited longest in the queue. Let $P_s(a)$ denote the steady state probability of packet in position_a and it is finally served. With the FCFS-PO-P policy, $p_s(a) (2 \leq a \leq N)$ waits in the sequence. Assume that m incoming packets arrived while a packet is in service. Then, $m \leq (N - a)$ for the packet to be served, and $p_s(a)$ moves to position_$(a+m-1)$ when the service ends. The state-transition equation of $P_s(a)$ is as follows [25]:

$$\begin{aligned} P_s(a) = & \prod \alpha_m \bullet (m \mid (a,b)) \\ & + \sum_{m=0}^{N-a} \alpha_m \bullet P_s(a+m-1), \quad a = 2,3,\ldots,N. \end{aligned} \tag{17}$$

Similarly, α_m can be obtained from [26]. Now, the average waiting time is analyzed. The probability of $p_s(a)$ to get the service and the service time not being larger than t is $G(a,c)$. Assume that the following equation is the LST of $G(a,c)$:

$$h(a,\theta) = \int_0^\infty e^{-\theta c} dG(a,c). \tag{18}$$

Similar to (17), the recursive equation of $h(a,\theta)$ is obtained:

$$h(a,\theta) = \sum_{m=0}^{N-a} \alpha_m(\theta) \bullet h(a+m-1,\theta), \quad a = 2,3,\ldots,N. \tag{19}$$

Let $H(a)$ denote average waiting time for $p_s(a)$ until it finally gets the service. Similar to (14) and (15), $H(a)$ can be obtained as follows:

$$H(a) = -\lim_{\theta \longrightarrow 0} h(a,\theta). \tag{20}$$

At last, the average waiting time for $p_s(a)$ is as follows:

$$T = \rho \bullet \sum_{a=0}^{N-2} b_a H(a+1) + P_s(a+1) \bullet \frac{(a+1)}{\lambda} \bullet b_{a+1}. \tag{21}$$

4. Performance Evaluation

In this section the performance of the proposed scheduling schemes for the OpenFlow switch is evaluated.

4.1. Experiment Environment. A testbed is implemented with three Raspberry Pi nodes for the experiment. The data with the priority scheduling are collected from the testbed, and they are compared with the analytical models. Open vSwitch is installed in the Raspberry Pi node to implement the real switch with the OpenFlow protocol. Open vSwitch is a multilayer, open source virtual switch targeting all major hypervisor platforms. It was designed for networking in virtual environment, and thus it is quite different from the traditional software switch architecture [27]. The parameters of the Raspberry Pi nodes are listed in Table 2.

For the experiment one remote controller and three OpenFlow switches are implemented. Three Raspberry Pi nodes operate as the OpenFlow switches, and Floodlight was installed in another computer working as a remote controller. Floodlight is a Java-based open source controller, and it is one of the most popular SDN controllers supporting physical and virtual OpenFlow compatible switches. It is based on the Bacon controller from Stanford University [28]. Note that

Table 2: The parameters of Raspberry Pi node.

Version	Raspberry Pi 3 Model B
SoC	BCM2837
CPU	Quad Cortex A53@1.2GHZ
Instruction set	ARMv8-A
GPU	400MHz VideoCore IV
RAM	1GB SDRAM
Storage	32G
Ethernet	10/100

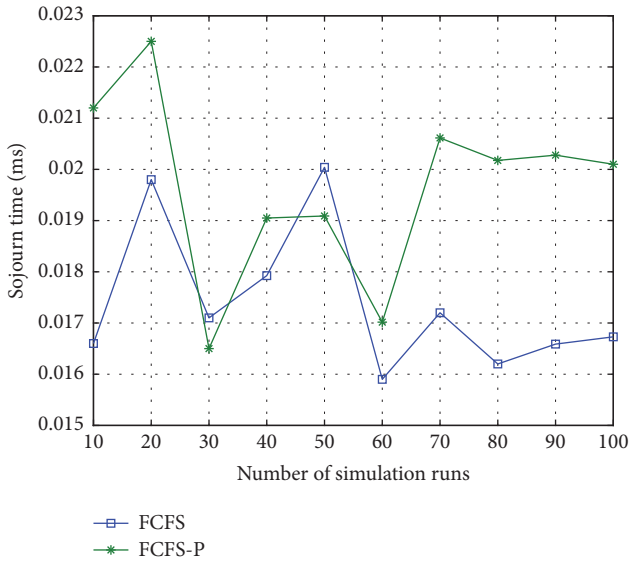


Figure 8: The comparison of sojourn times with FCFS and FCFS-P.

four network interfaces are supplied with the Raspberry Pi node. Therefore, three OpenFlow switches are installed for collecting data, and one interface is used to be connected to the controller network. The network traffics are generated by the software tool "iPerf" [29]. One host is selected as the server and another as the terminal by iPerf. Then, the host sends packets to the server by the UDP protocol in the bandwidth of 200Mbps.

4.2. Simulation Results. First, the simulation results of the proposed scheduling algorithm with a single switch are presented. Figure 8 compares the FCFS and FCFS-P scheduling algorithm. Notice that the FCFS-P algorithm mostly causes longer sojourn time than the FCFS algorithm. The FCFS-P algorithm occasionally displays similar performance to FCFS when the incoming packets have high priority.

Figure 9 compares the data from the experiment and analytical model. As aforementioned, three variables, λ, $\overline{x}$, and ρ, are used in the FCFS scheduling algorithm. Similar to the previous research [13], λ and $\overline{x}$ are empirically set to be 2 and 0.016, respectively. ρ is set to be 0.1, 0.5, and 0.9 to check the performance with various conditions. It reveals that

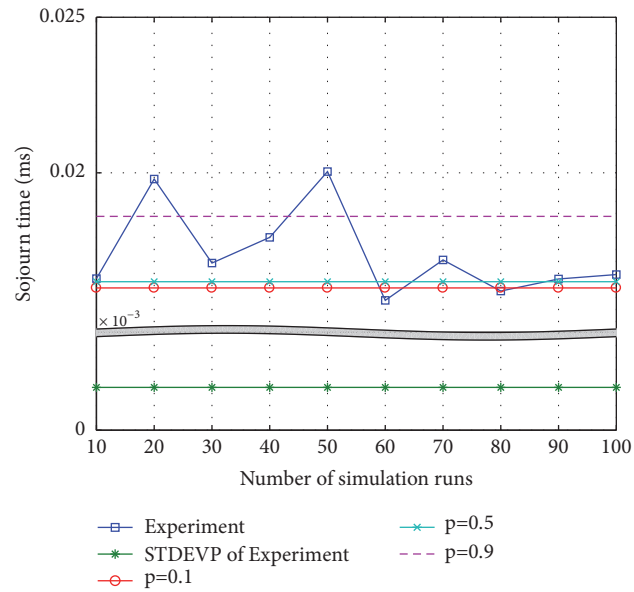


Figure 9: The comparison of the data from the experiment and analytical model with FCFS.

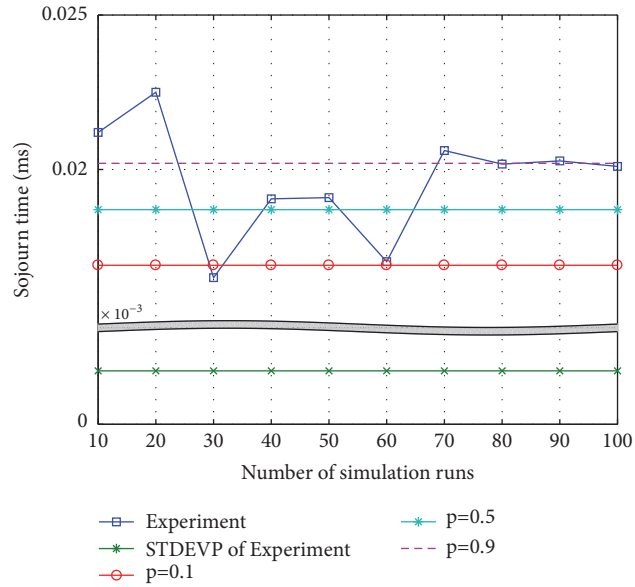


Figure 10: The comparison of the data from the experiment and analytical model with FCFS-P.

the data stabilizes as the simulation time passes, and ρ of 0.5 allows the closest estimation.

Figure 10 is for the case of FCFS-P. Here the values of ω_0, $\overline{x}_i$, λ_i, and S_i are 0.02, 0.018, 0.016, and 2, respectively. In this case, ρ of 0.9 shows the best result.

Assume that $N = 10$ and $\mu = 5$. Figure 11 compares FCFS-PO and FCFS-PO-P as λ varies. Observe from the figure that the average waiting time with FCFS-PO-P is smaller than with the FCFS-PO mechanism.

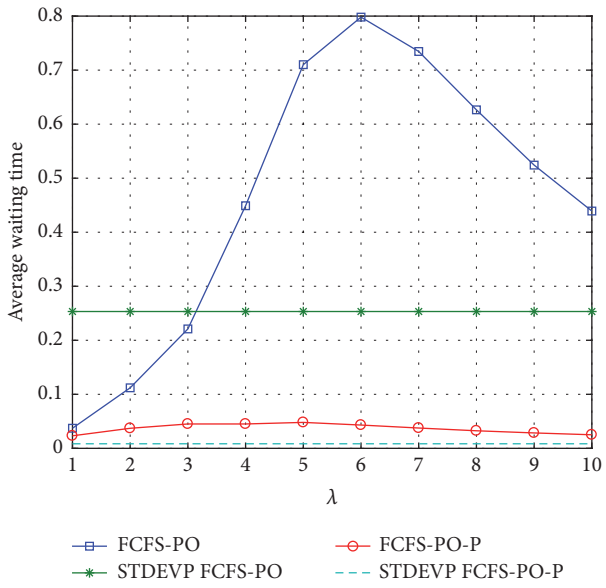


Figure 11: The comparison of waiting times with FCFS-PO and FCFS-PO-P.

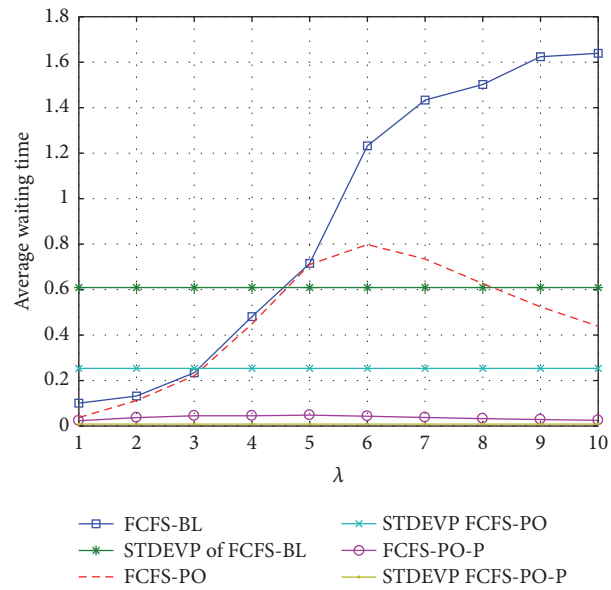


Figure 12: The comparison of the waiting time of the three mechanisms.

The probability density function of wait time with FCFS-PO in the steady state is as follows [30]:

$$W(t) = \sum_{n=1}^{N-1} V_n \left[B^{(n-1)}(t) \bullet R(t) \right] + V_0. \quad (22)$$

Here $B^{(n)}(t)$ is the n-fold convolution of $B(t)$ and $B(t) \cdot R(t)$ is the convolution of $B(t)$ and $R(t)$. The incoming packet will be processed directly if there is no packet waiting in the system, by which means the waiting time of the packet is 0. V_0 is the probability of the system to be idle and V_n is the probability that there are $n(1 \le n \le (N-1))$ packets waiting in the system. Then, the incoming packet will wait at position_$(n+1)$. The total wait time consists of the remaining service time and the service time of the packet in front of it. The average wait time can then be obtained as follows:

$$T(t) = \int_0^{\infty} t dW(t). \quad (23)$$

The FCFS-PO-P mechanism is compared with FCFS-BL and FCFS-PO in Figure 12. With FCFS-BL, the incoming packet is lost if the buffer is full. Notice from the figure that the pushout mechanism is more effective than the BL mechanism. Particularly, the superiority gets higher as λ increases. Generally speaking, the FCFS-PO-P mechanism is the best among the three mechanisms.

5. Conclusion

In this paper the FCFS-PO and FCFS-PO-P scheduling algorithms for multiple-switch SDN have been proposed targeting the edge computing environment. Since there exist some nodes requiring urgent processing in this environment, the priority-based scheduling approach was proposed. Here some switches might be overloaded if the packet arrival rate is high, and the proposed FCFS-PO and FCFS-PO-P mechanism are able to effectively deal with the situation. The SDN environment was implemented with three Raspberry Pi node switches and one independent computer of remote controller, and the sojourn time and wait time were analyzed. The measured data from the testbed were compared with those from the analytical models to validate them. The experiment results show that the FCFS-PO-P scheduling is better than the FCFS-PO scheduling based on the wait time. The comparison with the existing FCFS-BL scheduling algorithm reveals that FCFS-PO-P scheduling is the best among them, while FCFS-BL is the worst.

Several parameter values have been set empirically for the network operation. In the future the analytical models will be developed by which the value of the parameters can be properly decided. Also, for a more comprehensive understanding of the scheduling issues in SDN, the process of packet-in messages based on the $M/M/1$ model will be developed.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partly supported by the Institute for Information & Communications Technology Promotion (IITP)

grant funded by the Korea government (MSIT) (No. 2016-0-00133, research on edge computing via collective intelligence of hyperconnection IoT nodes), Korea, under the National Program for Excellence in SW supervised by the IITP (Institute for Information & Communications Technology Promotion) (2015-0-00914), the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2016R1A6A3A11931385, research of key technologies based on software-defined wireless sensor network for real-time public safety service; 2017R1A2B2009095, research on SDN-based WSN supporting real-time stream data processing and multiconnectivity), the second Brain Korea 21 PLUS project, and Samsung Electronics.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] Open Networking Foundation". Available at http://www.opennetworking.org, access March 2014.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[5] J. Ansell, W. Seah, B. Ng, and S. Marshall, "Making Queueing Theory More Palatable to SDN/OpenFlow-based Network Practitioners," in *Proceedings of the 8th Internatioanl Workshop on Management of the Future Internet (ManFI)*, pp. 1119–1124, 2016.

[6] H. Farhady, H. Lee, and A. Nakao, "Software-Defined Networking: a survey," *Computer Networks*, vol. 81, pp. 79–95, 2015.

[7] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[8] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: state of the art and research challenges," *Computer Networks*, vol. 72, pp. 74–98, 2014.

[9] A. Gelberger, N. Yemini, and R. Giladi, "Performance analysis of Software-Defined Networking (SDN)," in *Proceedings of the 2013 IEEE 21st International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication, MASCOTS 2013*, pp. 389–393, USA, August 2013.

[10] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, 2013.

[11] M.-K. Shin, K.-H. Nam, and H.-J. Kim, "Software-defined networking (SDN): A reference architecture and open APIs," in *Proceedings of the 2012 International Conference on ICT Convergence: "Global Open Innovation Summit for Smart ICT Convergence", ICTC 2012*, pp. 360-361, October 2012.

[12] W. Xia, Y. Wen, C. Heng Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[13] F. R. B. Cruz and T. Van Woensel, "Finite queueing modeling and optimization: A selected review," *Journal of Applied Mathematics*, vol. 2014, 2014.

[14] K. Choi, Y. Suh, and D. Yoo, "Extended collaborative filtering technique for mitigating the sparsity problem," *International Journal of Computers Communications & Control*, vol. 11, no. 5, p. 631, 2016.

[15] B. Xiong, K. Yang, J. Zhao, W. Li, and K. Li, "Performance evaluation of OpenFlow-based software-defined networks based on queueing model," *Computer Networks*, vol. 102, pp. 172–185, 2016.

[16] K. Sood, S. Yu, and Y. Xiang, "Performance Analysis of Software-Defined Network Switch Using M/Geo/1 Model," *IEEE Communications Letters*, pp. 114–119, 2016.

[17] H. Ma, J. Yan, P. Georgopoulos, and B. Plattner, "Towards SDN based queuing delay estimation," *China Communications*, vol. 13, no. 3, pp. 27–36, 2016.

[18] K. Mahmood, A. Chilwan, O. Østerbø, and M. Jarschel, "Modelling of OpenFlow-based software-defined networks: The multiple node case," *IET Networks*, vol. 4, no. 5, pp. 278–284, 2015.

[19] D. Shen, W. Yan, Y. Peng, Y. Fu, and Q. Deng, "Congestion Control and Traffic Scheduling for Collaborative Crowdsourcing in SDN Enabled Mobile Wireless Networks," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[20] W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu, "Performance Modelling and Analysis of Software Defined Networking under Bursty Multimedia Traffic," *ACM Transactions on Multimedia Computing Communication, and Applications*, vol. 12, no. 5s, pp. 77-19, 2016.

[21] M. Patel and A. Pandya, "Edge Computing: Design a Framework for Monitoring Performance between Datacenters and Devices of Edge Networks," *International Journal of Computer & Mathematical Sciences*, vol. 6, no. 6, pp. 73–77, 2017.

[22] D. Finkel, "Book review: Fundamentals of Performance Modeling by M.K. Molloy (Macmillan, 1989)," *ACM SIGMETRICS Performance Evaluation Review*, vol. 18, no. 3, p. 23, 1990.

[23] J. D. C. Little, "Little's Law as viewed on its 50th anniversary," *Operations Research*, vol. 59, no. 3, pp. 536–549, 2011.

[24] J. Shortle, J. Thompson, D. Gross, and C. Harris, *Fundamentals of Queueing Theory*, John Wiley Sons, New York, USA.

[25] Y. Lee, B. Choi, B. Kim, and D. Sung, "Delay Analysis of an $M/G/1/K$ Priority Queueing System with Push-out Scheme," *Mathematical Problems in Engineering*, 2007.

[26] S. Kasahara, H. Takagi, Y. Takahashi, and T. Hasegawa, "M/G/L/K System with Push-Out Scheme Under Vacation Policy," *Journal of Applied Mathematics and Stochastic Analysis*, vol. 9, no. 2, pp. 143–157, 1996.

[27] B. Pfaff, J. Pettit, T. Koponen et al., "The design and implementation of open vSwitch," in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2015*, pp. 117–130, USA, May 2015.

[28] H. Akcay and D. Kaplan, "Web-Based User Interface for the Floodlight SDN Controller," *International Journal of Advanced Networking and Applications*, vol. 08, no. 05, pp. 3175–3180, 2017.

[29] H. Zheng, Y. Zhao, X. Lu, and R. Cao, "A Mobile Fog Computing-Assisted DASH QoE Prediction Scheme," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[30] M. Rawat, H. Rawat, and S. Ansari, "Analysis of GI/GI/1 Queue by Push-Out Simulation Technique," *International Journal of Scientific and Research Publications*, vol. 3, no. 6, pp. 1–4, 2013.