UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUSTAVO HERMÍNIO DE ARAÚJO

HELPFUL: Flexible Architecture to Control Heterogeneous Low Power Wide Area Networks

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Juergen Rochol
Coadvisor: Prof. Dr. Cristiano Bonato Both

Porto Alegre
January 2020

CIP – CATALOGING-IN-PUBLICATION

Araújo, Gustavo Hermínio de

HELPFUL: Flexible Architecture to Control Heterogeneous Low Power Wide Area Networks / Gustavo Hermínio de Araújo. – Porto Alegre: PPGC da UFRGS, 2020.

88 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Advisor: Juergen Rochol; Coadvisor: Cristiano Bonato Both.

1. Software-defined networking. 2. Low power wide area network. 3. Network programmability. I. Rochol, Juergen. II. Both, Cristiano Bonato. III. Título.

ABSTRACT

Low Power Wide Area Networks (LPWAN) are candidates to coexist with traditional cellular networks by coping with different types of requirements such as density, reliability, and latency. However, there is no one-size-fits-all technology that can address all the needs of IoT applications. For this reason, the integration of heterogeneous LPWAN becomes necessary. SDN provides a powerful approach by creating a programmable, dynamic, and flexible architecture. Some studies investigate the SDN paradigm to provide a programmable network to IoT applications. Nevertheless, these studies do not take into account the limited capacity of SDN-based networking devices to store the forwarding rules in its architectures. This thesis presents the HELPFUL, an SDN-based architecture that creates a common control abstraction among LPWAN technologies (*e.g.*, LoRa, NB-IoT) running on top of virtualized base stations. We also discuss four rule management strategies for use with HELPFUL, providing support for single and multiple tables. We evaluate our proposal with a series of experiments with a prototype developed using the P4 language. Results show which HELPFUL is flexible enough to change the management strategy to the best fit with the network demands. Consequently, it can reduce the number of messages on the control channel exchanged between the Controller and Gateways. Finally, HELPFUL adds minimal overhead to network performance, regardless of the rule management strategy chosen.

Keywords: Software-defined networking. Low power wide area network. Network programmability.

HELPFUL: Uma arquitetura para controle de *Low Power Wide Area Network* heterogenias

RESUMO

Low Power Wide Area Networks (LPWAN) são candidatas a coexistir com as redes celulares tradicionais por lidar com diferentes tipos de requisitos como densidade, confiabilidade e latência. Entretanto, não existe uma tecnologia LPWAN que consiga atender todas as necessidades das aplicações de Internet das Coisas (IoT). Por essa razão, a integração de diferentes tecnologias LPWAN se faz necessária, criando assim, uma rede LPWAN heterogenia. SDN fornece uma abordagem poderosa, criando uma arquitetura programável, dinâmica e flexível. Alguns estudos já investigam a aplicação do paradigma SDN para fornecer uma rede programável para aplicações IoT. No entanto, esses estudos não levam em consideração a capacidade limitada dos dispositivos de rede baseados em SDN para armazenar as regras de encaminhamento. Nesta dissertação é proposto HELPFUL, uma arquitetura baseada nos conceitos definidos pelas Redes Definidas por Software (SDN) que cria uma abstração comum entre diferentes tecnologias LPWAN que rodam sobre estações base virtualizadas. Nós também discutimos quatro estratégias de gerenciamento de regras para serem utilizadas com HELPFUL, provendo suporte para uma única ou múltiplas tabelas. Nós avaliamos nossa proposta com uma serie de experimentos com protótipo desenvolvimento utilizando a linguagem P4. Os resultados mostram que HELPFUL é flexível o suficiente para que a estratégia de gerenciamento de regras na tabela de fluxos seja alterada para melhor se adequar as necessidades da rede. Consequentemente, HELPFUL reduz a quantidade de mensagens no canal de controle trocadas pelos controlador e *gateway*. Finalmente, HELPFUL adiciona uma sobrecarga mínima ao desempenho da rede independente da estratégia escolhida.

Palavras-chave: Redes Definidas por Software; Low Power Wide Area; Programabilidade de rede.

LIST OF FIGURES

2.1 Transmission Distances Between Wireless Technologies 16
2.2 LoRaWAN Architecture 20
2.3 LoRaWAN Packet Header 21
2.4 SDN Architecture 25
2.5 Flow table structure 27
2.6 PISA architecture 28

3.1 Query result: number of indexed documents by year 31

4.1 HELPFUL architecture overview. Gray boxes represent the architecture's planes, each one comprised of multiple components, depicted as white boxes. Arrows illustrate the data flows that may exist among the architecture's components. 39
4.2 Example of the Translator Packets 42
4.3 Table Miss Sequence Diagram. 46
4.4 Control table update process. 47

5.1 HELPFUL implementation tools. 50
5.2 Experimentation scenario. 52
5.3 Number of rules installed in the flow table. 55
5.4 Number of controller interventions. 56
5.5 Results for table hit ratio. 57
5.6 Table lookup time. 58
5.7 Transmission latency. 58
5.8 Lost packets. 59

LIST OF TABLES

3.1 SDN Architectures with focus on IoT. 32
3.2 Flow Table Management Strategies. 35

5.1 Application Profile Specifications. 51
5.2 Application Profile Specifications. 53
5.3 Experimentation parameters. 54

LIST OF ABBREVIATIONS AND ACRONYMS

16-QAM	*16-Quadrature Amplitude Modulation*
3GPP	*3rd Generation Partnership Project*
5G	*Fifth-Generation*
6LoWPAN	*IPv6 Over Low Power Wireless Personal Area Networks*
8PSK	*8-Phase-shift keying*
ADR	*Adaptive Data Rate*
AES	*Advanced Encryption Standard*
API	*Application Programming Interface*
ASIC	*Application Specific Integrated Circuits*
bps	*Bits per Second*
BS	*Base Station*
CAPEX	*Capital Expenditure*
DPSK	*Differential Phase-Shift Keying*
dst	*Destination*
EC-GSM-IoT	*Extended Coverage Global System Mobile for IoT*
eDRX	*Extended Discontinuous Reception*
FDMA	*Frequency Division Multiple Access*
FEC	*Forward Error Correction*
GMSK	*Gaussian Minimum Shift Keying*
GSM	*Global System for Mobile*
HARQ	*Hybrid Automatic Repeat Request*
Hz	*Hertz*

IoT	*Internet of Things*
IPv4	*Internet Protocol version 4*
ISM	*Industrial, Scientific and Medical*
LPWAN	*Low Power Wide Area Network*
LRU	*Least Recently Used*
LTE	*Long Term Evolution*
LTE-M	*Long Term Evolution Machine Type Communication*
m	*Meters*
MAC	*Media Access Control*
MBF	*Multiple Bloom Filter*
MPLS	*Multi-Protocol Label Switching*
NB-IoT	*Narrowband Internet of Things*
OFDMA	*Orthogonal Frequency Division Multiple Access*
ONF	*Open Network Foundation*
OPEX	*Operational Expenditure*
P4	*Programming Protocol independent Packet Processors*
PRB	*Physical Resource Block*
PSM	*Power Saving Mode*
QoS	*Quality of Service*
QPSK	*Quadrature Phase Shift Keying*
RAM	*Random Access Memory*
REST	*Representational State Transfer*
RPL	*Low-Power and Lossy Networks*
SDN	*Software-Defined Networking*
SRANS	*Static Random-Access Memory*
src	*Source*
TCAM	*Ternary Content Addressable Memory*
TCP	*Transmission Control Protocol*
TDMA	*Time Division Multiple Access*

UDP	*User Datagram Protocol*
UNB	*Ultra-Narrowband*
VLAN	*Virtual Local Area Network*
WLAN	*Wireless Local Area Network*
WPAN	*Wireless Personal Area Network*
WSN	*Wireless Sensor Network*

CONTENTS

1 INTRODUCTION 12

2 BACKGROUND 14

2.1 Contextualization 14

2.2 Low Power Wide Area Network 16

2.3 Review of the LPWAN technologies 18

2.3.1 Unlicensed bands LPWAN technologies 19

2.3.2 License bands LPWAN technologies 23

2.4 Software-Defined Network 24

2.4.1 Architecture Overview 24

2.4.2 Enabling Platforms 26

2.5 Chapter Summary 29

3 RELATED WORK 30

3.1 Systematic Literature Review 30

3.2 SDN Architectures for IoT 32

3.3 SDN Table Management 34

3.4 Discussion about Related Work 36

3.5 Chapter Summary 37

4 HELPFUL: CONCEPTUAL SOLUTION 38

4.1 Architecture Overview 38

4.2 HELPFUL Forwarding 40

4.3 HELPFUL Controller 43

4.4 HELPFUL Application 44

4.5 HELPFUL Management 44

4.6 HELPFUL Interaction 45

4.7 Chapter Summary 48

5 HELPFUL EVALUATION 49
5.1 Framework Implementation 49
5.2 Methodology 51
5.3 Impact of management strategies in flow tables 54
5.4 Impact on the network infrastructure 56
5.5 Chapter Summary 60

6 CONCLUSION AND FUTURE WORK 61
6.1 Summary of Contributions 61
6.2 Final Remarks and Future Work 62

REFERENCES 63

APPENDIXA RESUMO 66

APPENDIXB PUBLISHED PAPER – SBRC 2017 68

APPENDIXC PUBLISHED PAPER – LANCOMM STUDENT WORKSHOP 2019 84

APPENDIXD JORNAL SUBMITED – JNCA 2019 88

1 INTRODUCTION

The next generation of wireless networks is being designed to provide connectivity to multiple Internet of Things (IoT) applications, such as smart homes, industry 4.0, and wearable devices (MINOLI; SOHRABY; OCCHIOGROSSO, 2017). Such new IoT applications will push the boundaries of current communication architectures, demanding coexistent IoT networks, *e.g.*, using Low Power Wide Area Networks (LPWAN) technologies (ALI et al., 2017). The coexistence of heterogeneous LPWAN infrastructures enables the usage of the best features that each technology provides, addressing multiple requirements, for example, massive machine-type communication and low-latency communication (BRUNS et al., 2015).

The deployment and operation of several heterogeneous networks, each one supporting a limited number of IoT devices, evidences research challenges that still require proper investigation(RAZA; KULKARNI; SOORIYABANDARA, 2017). Firstly, current LPWAN technologies are not designed to coexist with each other, as each LPWAN implements individual control protocols, leading to uncoordinated wireless access (POORTER et al., 2017). Secondly, the assumption of multiple wireless access technologies in the traditional architecture of cellular networks would lead to prohibitive deployment costs of transmitter cells (GALLO et al., 2016).

Aiming to cope with these challenges, Base Stations (BS) virtualization was proposed as a solution to reduce CAPital EXpenditure and Operational Expenditure (CAPEX/OPEX) of cellular networks through infrastructure sharing (KIST et al., 2018). However, virtualization adds the need for new access control to the infrastructure to manage the uncontrolled access to the spectrum used by the LPWAN. Software-Defined Networking (SDN) provides a powerful approach by creating a programmable, dynamic, and flexible architecture that allows the abstraction from traditional hardware-based protocol implementations into a software-based network controller (KREUTZ et al., 2015). In this sense, some studies have applied the SDN paradigm to provide a programmable network to IoT applications (LUO; TAN; QUEK, 2012; GALLO et al., 2016; BADDELEY et al., 2018; BERA et al., 2018). Nevertheless, these studies do not take into account the limited capacity of SDN-based networking devices to store the forwarding rules in its architectures. For example, the tables have a limited rule storage capacity, which is several orders of magnitude smaller than that required for the operation of certain types of network, including LPWAN (NGUYEN et al., 2016). This problem becomes worse when ap-

plied in LPWAN environments where a massive amount of end-nodes in a network (*e.g.*, $\approx 10^6$ devices per km^2 (RAZA; KULKARNI; SOORIYABANDARA, 2017)), making it necessary to manage the rules of the flow tables.

In this thesis, we propose HELPFUL: *a flexible architecture to control heterogeneous low power wide area networks* to enable various LPWAN technologies to coexist whit scalability guarantees. HELPFUL is an SDN-based architecture that creates a common control abstraction among LPWAN technologies running on top of virtualized BSs. The main contributions of our work are two-fold: (*i*) the proposal of an architecture which adds an abstraction layer that translates the heterogeneous control protocols to a homogeneous set of messages, and (*ii*) a strategy based on multiple table rule management to address the scalability issues present in the SDN-based devices. Our proposal enables the unification of technology-specific control into a single cross-technology controller, allowing infrastructure and spectrum access to be harmonized.

We developed a prototype based on the P4 language that allows programmability of both control and forwarding planes to validate HELPFUL. In our first evaluation, we enumerate a series of strategies to manage flow table rules when using the HELPFUL architecture. We define the metrics to analyze HELPFUL with each strategy, such as the number of rules installed in the flow table, the number of controller interventions, and the number of table hits. These results indicate the best strategies regarding the reduction of the number of rules installed in flow tables. Our second evaluation considers the overhead of HELPFUL on the network architecture, including table lookup time, latency, and lost frames. The results show that HELPFUL produces a minimum impact on the network. For example, HELPFUL presented $\approx 0.92ms$ of lookup time in a scenario analyzed. Moreover, the HELPFUL architecture added minimal network latency of $\approx 2ms$ and the number of lost frames averaging 1.5%.

The remaining of this thesis is organized as follows. In Chapter 2, we present the background with the main concepts related to the new IoT applications, the Low Power Wide Area Network and Software-Defined Networking. In Chapter 3, we present a review of the most relevant research and related works for this thesis. In Chapter 4, we introduce HELPFUL and show a description and overview of its architectural components. In Chapter 5, we present outline the implementation of HELPFUL showing how each component communicates with others and internal details of its architecture. After, we present our evaluation and associated results, including a performance analysis of the framework. Finally, in Chapter 6, we conclude this dissertation presenting final remarks and future work.

2 BACKGROUND

In this Chapter, we present the basic concepts for understanding this work. First, on Section 2.1 we position our work on the Internet of Things (IoT) landscape. Next, in Section 2.2 we describe the Low Power Wide Area Network technology focus on the main design goals and techniques. Finally, on Section 2.4 we present the Software-Defined Networking (SDN) and describe how this paradigm can be used in the wireless environment.

2.1 Contextualization

The fifth-generation (5G) of cellular networks is being designed to provide connectivity to the Internet of Things (IoT) applications such as smart city, healthcare, smart grid, smart metering, industrial assets monitoring, agriculture, and, wildlife monitoring and tracking. These applications require a new set of communication requirements:

- **Long-Range Communication:** It is desirable for agriculture or wildlife tracking applications where sensors can be deployed over a large geographical area with just a single base station reducing the capital expense (CAPEX).
- **Low Energy Consumption:** It is desirable to reduce the maintenance cost of the end-devices by extended their battery live. This feature is desirable for applications that need to deploy end-devices in remote areas where it is difficult to provide any maintenance.
- **Support for a Massive Number of Devices per Cell:** It is desirable for applications that need a large deploy of end-devices over a geographical area like smart cities and industrial applications. As the number of final nodes increases, the accuracy of these applications also tends to increase.
- **Low Deployment Cost:** It is desirable to reduce the CAPEX of the sensors of his applications and allow the deployment of more sensors.

However, these requirements are not met by the current wireless personal area network (WPAN), wireless local area networks (WLAN) and, cellular networks communication technologies (RAZA; KULKARNI; SOORIYABANDARA, 2017).

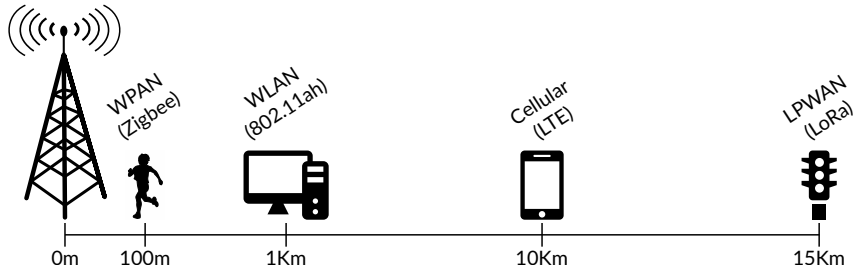
The WPAN and WLAN are not recommended for the new IoT applications because the range of these technologies is limited to a few hundred meters. For example, the maximum distance that Zigbee can provide communication is only 100 meters, and Wi-Fi only transmits over 30 meters. The range of WPAN and WLAN technology can be extended by the usage of multi-hop mesh networking. Where end-devices need to listen to the environment to relay the traffic of the other end-devices and forward to the gateway. Also, the end-devices cannot be arbitrarily deployed having specific positions to provide the mesh network, and cannot be moved anywhere (XIONG et al., 2015). This approach is not desirable due these large deployments are prohibitively expensive, and the end-devices waste energy forwarding traffic of the other end-devices.

The cellular network technologies can provide wide area coverage. However, these technologies bring two disadvantages that make it unfeasible for the new IoT applications. First, the cellular technologies were design to deal with complex waveforms, optimized for voice, high-speed data services, and text. Due to these design decisions, the cellular networks do not achieve energy-efficient high enough for the new IoT applications. Secondly, for these technologies to achieve low power consumption, there is a clear need to strip complexity and reduce cost. It is important to notice that some current cellular networks are being adapted to provide communication to low power devices. Efforts in this direction are underway for cellular networks by the Third Generation Partnership Project (ALI et al., 2017).

The Low Power Wide Area Networks (LPWAN) represent a novel paradigm in providing communication to the IoT applications. LPWAN takes different tradeoffs than the current short wireless and cellular technologies. These technologies offer long-range communication (tens of kilometers), low energy consumption (tens of years of battery live time), high scalability (tens of thousands of devices per cell), and low deployment cost. However, to achieve these requirements, the LPWAN operates at the expense of low data rates (tens of kilobytes per second) and higher latency (order of seconds). These technologies are not indicated to operate with applications that need high throughput and ultra-low latency, such as wireless industrial control.

Nevertheless, LPWAN are indicated to applications that exchange a small amount of data, infrequently, over long distances and, with end-devices distributed over a large geographical area such as smart cities, smart metering, home automation (RAZA; KULKARNI; SOORIYABANDARA, 2017). Figure 2.1 shows an example of the range of different wireless technologies that are present in this section. In the next section, we will describe the techniques used by the LPWAN in order to achieve these requirements. Also, we present an overview of the leading LPWAN technologies.

Figure 2.1: Transmission Distances Between Wireless Technologies



Source: the author (2018)

2.2 Low Power Wide Area Network

Research efforts for future 5G networks have to provide communication to IoT applications meeting the requirements of low power consumption, long-range coverage, and support for a massive number of end-devices. LPWAN will play an essential role by complementing traditional cellular networks to address the communications requirements and provide spectrum efficiency for IoT applications (ALI et al., 2017). In this Section, we present the primary design goals of the LPWAN technologies and the techniques used to achieve these goals. We organize each one of the goals in bullet points that describe the techniques used. The goals are:

1. **Long Distance Communication:** The main innovation that the LPWAN technologies proposed is the long-distance communications with an excellent signal propagation to reach indoor places (*e.g.*, basements, industrial complex). With this wide area coverage, the end-devices can transmit data over tens of kilometers of their base stations in an urban or rural environment. To achieve this goal, the LPWAN used sub-1GHz bands and some modulation techniques (*e.g.*, narrowband, ultra-narrowband, and spread spectrum). Sub-1GHz offers robust and reliable communication at low power budgets and is less sensitive to physical obstacles (*e.g.*, walls, buildings) (ALI et al., 2017). It results in higher reliability and enables long-range and low power communication. The side effect to use these bands is the limited amount of available spectrum combined with the large propagation ranges of the LPWAN will cause an inter-technology interference. The LPWAN technologies use three kinds of modulation techniques to enable a range of tens of kilometers: narrowband, ultra-narrowband, and spread spectrum. Narrowband modulation provides a high link budget by encoding the signal in low bandwidth. Also, this technique shares the overall spectrum efficiently if multiple links. The noise level experienced inside a single narrowband is also minimal. Therefore, no processing gain is required to decode the signal at the receiver, resulting in a simple and inexpensive transceiver design. Some LPWAN technologies carrier a signal in an ultra-narrowband (UNB). This technique reduces the experienced noise and increasing the number of supported end-devices per unit

bandwidth, even more than a narrowband modulation. However, the effective data rate for individual end devices decreases as well, thus increasing the amount of time the radio needs to be kept ON. A spread spectrum technique is used for some LPWAN technologies. This technique consists of spreading a narrowband signal over a wider frequency band but with the same power density. Transmission using the spread spectrum is more resilient to interference and robust to jamming attacks. However, more processing gain is required on the receiver to decode the message. Another outcome that this technique brings is that spreading a narrowband signal over a wide band results in less efficient use of the spectrum (RAZA; KULKARNI; SOORIYABANDARA, 2017).

2. **Low Energy Consumption:** One of the central promises of the LPWAN technologies is the low energy consumption of the end-devices to transmit their traffic over long distances. This is an essential requirement for various business opportunities. To reduce the energy consumption and achieve maximum energy efficiency, the LPWAN applies a sequence of techniques: (*i*) topology specifications, (*ii*) duty cycle and, (*iii*) lightweight medium access control. The LPWAN technologies specify a topology that connects the end-devices directly to the base stations. It results in a start topology that brings a considerable energy-saving advantage. Also, the end-devices do not need to waste energy listen to other devices that want to relay their traffic (*e.g.*, multi-hope mesh networks). Thus, the end-devices never exchange messages directly with each other. The base stations will intermediate all the possible communications. For example, if some end-device needs to transmit data to other end-device, first, it is necessary to transmit to the base station. Then the base station will transmit the data to do the end-device (RAZA; KULKARNI; SOORIYABANDARA, 2017). Radio duty cycling allows end devices to turn off their transceiver and just turned then on when it is required. The duty cycling mechanism depends on the application and traffic pattern. If an application needs to transmit some data, then the transceivers will wake up and transmit. If a downlink transmission is required, then the end-device needs to schedule with the base station. In the most LPWAN technologies, the end-devices may listen for a short duration after their uplink transmissions to receive a replay back. Another option is to listen at a scheduled time agreed with the base station. This technique can be used in hardware components in order to reduce energy consumption. The LPWAN adopts simple random access schemes (*e.g.*, ALOHA). The end-devices are the most-constrained elements transferring the complex task to the base station to simplify the design of the end-devices. The LPWAN base station is capable of transmitting and listening from multiple end-devices using different channels and orthogonal signals simultaneously. Also, the base station is responsible for mechanisms to adapt data rate, support mobility, and suppress network duplicates (RAZA; KULKARNI; SOORIYABANDARA, 2017).

3. **Support of massive number of devices:** Scalability is an essential requirement for LP-

WAN technologies. The LPWAN network will send low volumes of data of a large number of end-devices. To cope up with the increased density of the end devices in certain areas, LPWA networks, like traditional cellular networks, will resort to dense deployments of base stations. Not only the LPWA systems should scale to several connected devices, but each link should be optimized for reliable and energy-efficient communication. Adapting the modulation schemes to reach distances while guaranteeing reliable communication at the same time requires efficient monitoring of link qualities and coordination between end devices and network. To accommodate as many connected devices as possible, efficient exploitation of diversity in channel, time, space, and hardware is vital. Due to the low-power and inexpensive nature of the end devices, much of this is achieved by cooperation from more powerful components in LPWA networks such as base stations and backend systems. LPWA technologies employ multi-channel and multi-antenna communication to parallelize transmissions to and from the connected devices. Further, communication is made resilient to interference by using multiple channels and doing redundant transmissions (XIONG et al., 2015).

4. **Low deployment cost:** The LPWAN technologies adopt some techniques to reduce the capital expenses (CAPEX) and operating expenses (OPEX) for the end-users and network operations. The connective module of the end-devices will eventually cost less than a few dollars. Thus, these reductions on the CAPEX and OPEX will result in the deployment of the end-devices in a large geographical area. To achieve this goal, the LPWAN applies a sequence of techniques. First, the LPWAN transceivers need to deal with simple waveforms compared with the short-range wireless technologies and cellular networks. Also, the LPWAN does not have complex MAC schemes that reduce transceiver footprint, peak data rates, and memory sizes, minimizing the hardware complexity and thus the cost. Thunder, the LPWAN technologies apply a minimal infrastructure. A single LPWAN base station can connect tens of thousands of end-devices distribute over a large geographical area and do not need to implement a pico or microcell (RAZA; KULKARNI; SOORIYABANDARA, 2017).

2.3 Review of the LPWAN technologies

In this Section, we highlight the leading LPWAN technologies. To better understanding the technologies present in this section, we divide into two groups: the LPWAN that uses the unlicensed band and the LPWAN that use license bands. The usage of license bands conflicts with the low deployment cost goal. Due to this conflict, most of the LPWAN technologies use industrial, scientific, and medical (ISM) bands or TV-white spaces. However, the usage of the ISM bands will leave the LPWAN technologies to develop mechanisms of coexistence. Because they are not designed to coexist with the existent wireless technologies. Second, since the amount

of available spectrum is much smaller and the propagation ranges much more extensive, these technologies will cause interference at a much larger scale, leading to severe inter-technology and inter-operator interference. Some LPWAN technologies that are deployed in the licensed band may share the existing cellular bands to avoid the additional licensing cost. However, to get better performance, a stand-alone licensed band can be acquired as well, a trend proprietary LPWAN technologies may eventually follow to avoid performance degradation due to an increase in several connected devices (RAZA; KULKARNI; SOORIYABANDARA, 2017).

This Section is organized as follows. In Subsection 2.3.1, we present the leading LPWAN technologies that use unlicensed bands. To better understanding this work, we focus on LoRa/LoRaWAN technology. In Subsection 2.3.2, we review the leading LPWAN technologies that use license bands. We will focus on Narrowband-IoT (NB-IoT) technology for the best future understanding of this work.

2.3.1 Unlicensed bands LPWAN technologies

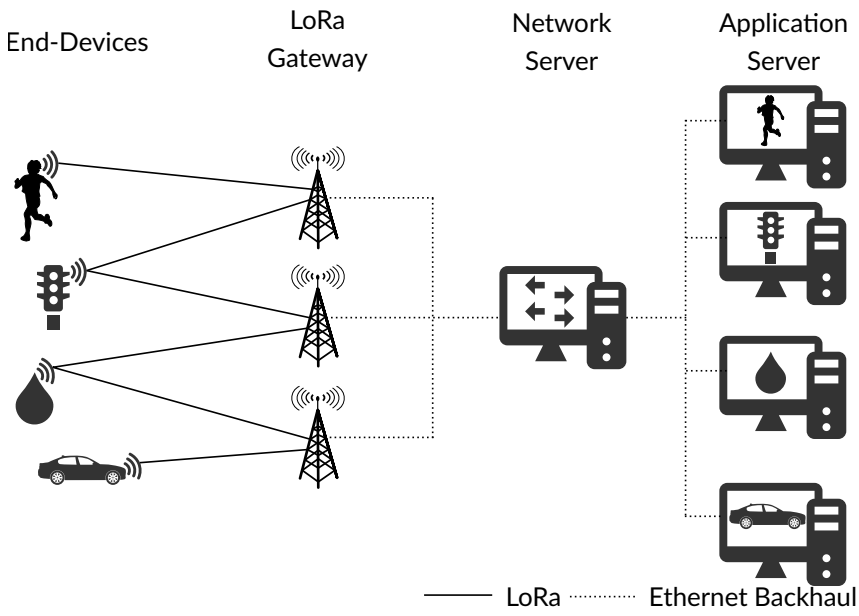
In this Subsection, we review the unlicensed bands LPWAN technologies. First, we present LoRa/LoRaWAN technology. We decide to focus on LoRa/LoRaWAN because, further on this work, we reuse some concepts that are applied to technology. Also, we review Sigfox and Weightless, presenting the technical specifications of these technologies.

LoRa/LoRaWAN

LoRa/LoRaWAN is a technology developed by a consortium called LoRa Alliance with the proposal to provide connectivity over long distances and low energy consumption. LoRa is a physical layer technology that uses the Sub-1GHz band using a spread spectrum technique developed and commercialized by Semtech Corporation. The end-devices communicate with the base stations using a chirp spread spectrum technique that spreads a narrow band input signal over a wider channel bandwidth. LoRa supports multiple spreading factors (7 to 12) to decide the best tradeoff between rage and data rate. Higher spreading factors increase the distance that an end-device can transmit to the base station at the expense of lower data rates. The data rate ranges from 300 bps to 37.5 kbps depending on the spreading factor, and the distances may vary between 15 Km to 20 Km. LoRa combines Forward Error Correction (FEC) with the spread spectrum technique to increase the reliability of the transmission. The communication between end-devices and base stations is a half-duplex mode. To receive data from the base stations the end-devices open a received window 1 second after the transmission, and a second received window will be open 1 second after the first one. This mechanism improves energy efficiency once the end-device can turn off their transceivers while the window is not open. Also, this strategy allows a channel to receive a network control between the end-device and the base station.

While LoRa defines the physical layer of the technology, LoRaWAN defines the network architecture and up layer protocols. In Figure 2.2, it is possible to observe the network architecture of the LoRaWAN network. The end-devices are the sensors where that has the function to sense the environment and send their data to feed the IoT applications. In order to increase the reliability of the communication, the end-devices transmit the data to all LoRa Gateways in the communication range. This approach generates duplicate messages that have to be tread later by network control. The LoRa Gateway receives the messages from the end-devices and will forward it to the Network Control through an IP standard communication technology (*e.g.*, Ethernet, Wi-Fi, GSM). The Network Server has the function to manage the network (*e.g.*, eliminate duplicate packets, schedules transmissions, send acknowledgments, and adapts data rates). Also, the Network Server forwards the data to the destination Application Server. Finally, the Application Server will consume the data send by the end-devices.

Figure 2.2: LoRaWAN Architecture

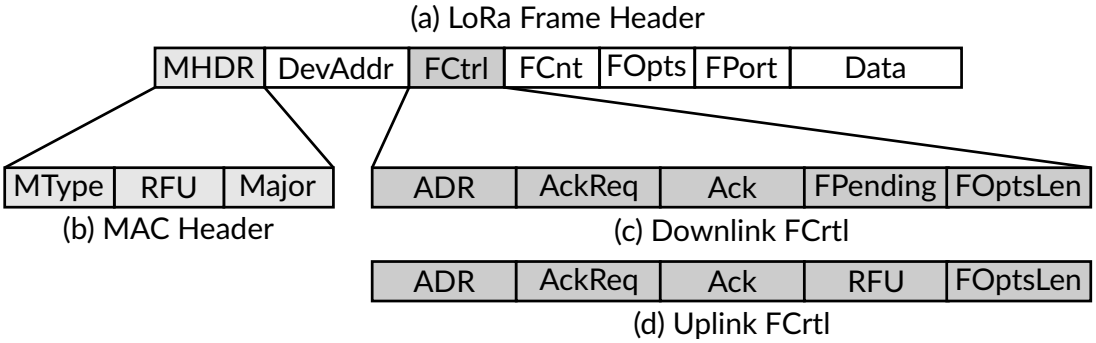


Source: the author (2018)

LoRa/LoRaWAN does not have a control channel. The control information and the user data are transmitted using the same channel. To allow this, LoRaWAN defines a packet header that allows the network server to transmit MAC commands for the end-devices. In Figure 2.3 it is possible to observe the LoRaWAN packet header. The first field of the header is the MAC header. It is used to specify the message type (*MType*) and the major version (*Major*) of the LoRWAN protocol. The *MType* defines indicating, among other things, whether it is an uplink or a downlink message and whether or not it is a confirmed message. The Major defines the LoRaWAN version. The second field is the Device Address, a 2-bit identifier of the end-device. Seven bits are used as the network identifier, and 25 bits are used as the network

address of the end-device. The Frame Control (*FCtrl*) is to manage the network parameters (*e.g.*, request confirmation, send/response MAC commands). The Frame Counter (*FCount*) is a frame counter used to the Network Server to identify if a frame if uplink or downlink. The Frame Options (*FOpts*) is used to piggyback MAC commands on a data message. The Frame Port (FPort) indicates if the frame has just MAC commands or some data do delivery to the destination.

Figure 2.3: LoRaWAN Packet Header



Source: the author (2018)

The *FCtrl* is used to control the network parameters of the end-devices. The Adaptive Data Rate (*ADR*) field is used to inform to the end-device that the network server will set the data rate of the end-devices thought appropriated MAC commands. If these fields are not set, then the end-device will use the default data rate. If the data rate set by the network server is higher than the default data rate of the end-devices, the Adaptive Data Rate Acknowledgement Request (*ADRAckReq*) is used to confirm if the data transmitted is received. The Acknowledgement (*Ack*) field is used to confirm the previous data received. The Frame Pending (*FPending*) is used only in uplink transmissions is only used in downlink communication, indicating that the gateway has more data pending to be sent and therefore asking the end-device to open another receive window as soon as possible by sending another uplink message. The frame-options length field (*FOptsLen*) in *FCtrl* byte denotes the actual length of the frame options field (*FOpts*) included in the frame.

LoRaWAN defines MAC commands that allow customizing end-device parameters. These commands can control the data rate and output power used by the device, as well as the number of times each unconfirmed packet should be sent (*LinkADRReq*), the global duty cycle of the device (*DutyCycleReq*), changing parameters of the receive windows (*RXTimingSetupReq*, *RXParamSetupReq*) and changing the channels used by the device (*NewChannelReq*).

Others Unlicesend LPWAN

Sigfox technology is developed by the SigFox Network Operators (SNOs). This partnership offers an end-to-end LPWAN connectivity by deploying proprietary base stations and equipped with cognitive software-defined radios and connect them to the backend servers using an IP-based network. The end-devices communicate with the base stations using a Binary Phase Shift Keying (BPSK) modulation using an ultra-narrowband (100Hz, resulting in 8000 channels) in a Sub-GHz ISM band (868 MHz in Europe, 915 MHz in the USA). It allows for long-range communication (30–50 km in rural areas and 3–10 km in urban areas) at low bitrate (100 bps). Sigfox was initially supported only uplink transmissions but later evolved to a bidirectional transmission. The downlink transmissions can only be made after an uplink communication. After an uplink transmission, the end device has to wait and listen for a response from the base stations. Also, Sigfox does not implement any collision avoidance mechanism. In summary, the SigFox technology is suitable for very specific, very low data rate uplink IoT applications. Because of its closed nature, it is difficult to innovate within SigFox for external researchers and companies. However, due to its popularity, it must be taken into account as a potentially harmful interfere for the other LPWAN technologies (RAZA; KULKARNI; SOORIYABANDARA, 2017).

Weightless Special Interest Group (WeightLess-SIG) proposed an open LPWAN standard. Weightless technology has three different standards: Weightless N, Weightless W and, Weightless P. Which one of these protocols provide a different set of feature (all of them offering long-range and low energy consumption). Weightless W achieves an excellent signal propagation by using the TV white-spaces. This protocol can support several modulation schemes 16-Quadrature Amplitude Modulation (16-QAM) and Differential-BPSK (DBPESK) using a narrow band technique (resulting in 24 channels). Uplink transmission is made by using Time Division Multiple Access (TDMA), and downlink transmissions are made using the Frequency Division Multiple Access (FDMA). Also, Weightless has an encryption layer using AES 128. Weightless W can transmit packets of 10 bytes at a rate of 1 kbps to 10Mbps. The end-devices can transmit to a base station over a 5 Km. The used of TV white spaces is allowed only in a few regions. For this motivation, Weightless-SIG defines the other two standards that use ISM bands. Weightless P differs from the W version by using an ultra-narrowband modulation and is a one-way communication from the end-device to a base station. Because of that, this version uses less energy than the other Weightless versions. Also, this version use Sub-GHz ISM bands (868 MHz in Europe and 915 MHz in the USA) using a DBPSK scheme. Finally, the Weightless-P can be summarized as a Weightless-N version with two-way communication. It modulates the signals using GMSK and QPSK. Full support for acknowledgments and bidirectional communication capabilities enable over-the-air upgrades of firmware (RAZA; KULKARNI; SOORIYABANDARA, 2017).

2.3.2 License bands LPWAN technologies

To address the new features of the IoT applications, the Third Generation Partnership Project (3GPP) is evolving the current cellular networks. In this Section, we highlight the main LP-WAN technologies that use licensed bands. In subsection 2.3.2 we highlight Narrowband-IoT (NB-IoT). In Section 2.3.2, we review LTE-M and EC-GSM-IoT, presenting the technical specifications of these technologies.

NB-IoT

The Narrowband-IoT is a technology design to enhance the existent LTE network to provide better support for IoT applications. NB-IoT reuses the design of the LTE network but with optimization to provide a low power consumption. The full carrier bandwidth is 180 kHz, the subcarrier spacing can be 3.75 kHz (only for the uplink) or 15 kHz, the highest modulation scheme is QPSK, there is only support for Frequency Division Duplex (FDD) and half-duplex operation. The downlink of NB-IoT is based on OFDMA, and the transmission scheme uses only one physical resource block (PRB) for the LTE network. The uplink is based on single-carrier FDMA. For uplink transmission, there are two possible operation modes, single-tone transmission, and multi-tone transmission. In a single tone, either 3.75 kHz or 15 kHz subcarrier spacing is allowed. In multi-tone, only 15 kHz subcarrier spacing can be used. Multi-tone transmission enables grouping sets of 3, 6, or 12 subcarriers. Additionally, the minimum duration for the resource units used in scheduling depends on the number of assigned subcarriers and the operation mode, ranging from 1 ms in 12 sub-carriers multi-tone transmission to 32ms in 3.75 kHz single tone transmission (ZAYAS; MERINO, 2017).

Regarding the MAC layer, NB-IoT introduces several changes to reduce power consumption and make scheduling more flexible and straightforward. Because of the lower throughput requirements, a single hybrid automatic repeat request (HARQ) process is used. It allows removing the HARQ identifier from the scheduling assignments and thus uses a lower number of control bits for higher efficiency and robustness. Additionally, uplink retransmissions are no longer synchronous but always adaptive and asynchronous both in uplink and downlink. It provides the network with tighter control on the UL scheduling while preventing undesired periodic retransmissions as they will be now generated only when explicitly requested. Also, NB-IoT uses a Power Saving Mode (PSM) that allows the end-device to inform the network that they will sleep indefinitely (WANG et al., 2017).

Others Licesend LPWAN

LTE enhancements for machine type communications is a fork of the traditional LTE network optimize for communication in long distances and high energy efficiency. This technol-

ogy was initially standardized in the 3GPP Release 13 specification, specifically refer to LTE CatM1, suitable for IoT applications. To adapt the LTE technology to operate with low energy consumption, a set of features have been cut. LTE-M only supports half-duplex communications to reduces the complexity of modem and antenna design. Also, a drop in the receive bandwidth from 20 MHz to 1.4 MHz in combination with a reduced transmission power will result in a more cost-efficient and low-power design. Like NB-IoT, the features of PSM and eDRX are added to this standard in order to better energy efficiency. LTE-M can use Quadrature Phase Shift Keying (QPSK) or 16-QAM in both uplink and downlink transmissions. Transmission can be made at a peak rate of 1Mbps. The end-devices can transmit to the base station at a distance of 2.5Km in urban areas and 5km in rural or remote areas (ALI et al., 2017).

Extended Coverage Global System Mobile for IoT (EC-GSM-IoT) is the proposal of the 3GPP of extending the traditional GSM to better server IoT applications. This standard reuses the GSM infrastructure and applies a set of features in order to improve energy consumption. Like NB-IoT and LTE-M, this standard use eDRX and PSM to improve energy consumption. EC-GSM-IoT uses 900 MHz in a coverage area of the 15 km in downlink and uplink transmission. Two modulation techniques, namely Gaussian Minimum Shift Keying (GMSK) and Eight Phase Shift Keying (8PSK), provide variable data rates with the peak rate of 240 kbps with the latter technique. The end-devices can transmit to the base station at a range of 15km (ALI et al., 2017).

2.4 Software-Defined Network

In this Section, we review the Software-Defined Networking paradigm and present the main concepts need to understand this work. In subsection 2.4.1, we present the basic concept of SDN architecture. Finally, in subsection 2.4.2, we describe the main platforms that allow the implementation of the SDN concepts.

2.4.1 Architecture Overview

Software-Defined Networking (SDN) is a network architecture that decouples the control logic functions (control plane) from the forwarding devices (forward plane). The control logic is moved to an external entity and is responsible for determinate the behavior of the network. The forwarding devices do not take any decision autonomously and forward the data accordingly with the control logic determinate. SDN was designed to simplify the deployment of new control plane functions when compared to traditional networks in which the control and data planes are more tightly coupled. SDN architecture can be seen in Figure 2.4. In our work we are assuming the SDN architecture present in (WICKBOLDT et al., 2015) with four conceptual planes:

- **Forwarding Plane:** Includes the devices (*e.g.*, switches) that are responsible for forward-

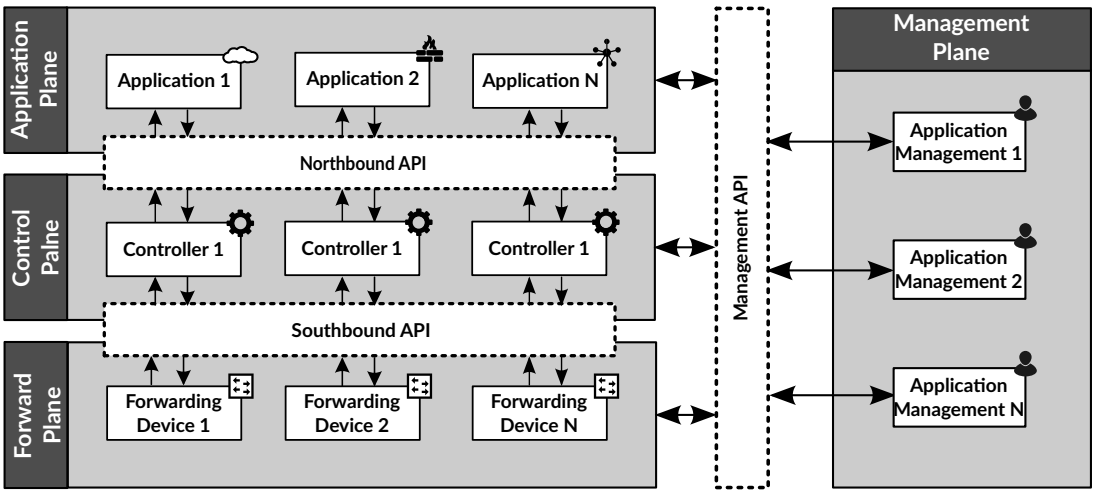
ing the packet data. The devices on this plane store network rules that are defined by the control plane. These rules can be (*i*) forward packet through a determinate port, (*ii*) modify the packet header, (*iii*) send the packet to the network controller, and (*iv*) drop the packet (KREUTZ et al., 2015).

- **Control Plane:** Responsible for all the control logic of the network (*e.g.*, routing protocols, access list). Additionally, this plane manages the information of the forwarding devices. The control plane has the global view of the entry network being able to offer mechanisms for fault diagnosis, make decisions over current traffic distributions, and enforce quality of services (QoS) policies (KREUTZ et al., 2015).

- **Application plane:** Responsible for executing applications that run over the network infrastructure. Generally, these applications perform modifications regarding network aspects, such as network policies and routing behavior, with some degree of human intervention. Examples of network applications deployed in this plane are network visualization, path reservation, and network provisioning (KREUTZ et al., 2015).

- **Management plane:** Responsible for monitoring, configuring, and maintaining the behavior of network elements in each plane. The management focuses on the configuration of these network elements. Consequently, some human intervention may be necessary for the managed applications in this plane (KREUTZ et al., 2015).

Figure 2.4: SDN Architecture



Source: the author (2018)

The SDN architecture also defines tree interfaces that are responsible for the communication between the planes: (*i*) northbound API, (*ii*) southbound API and, (*iii*) management API. The northbound API provides the communication between the application plane and the

control plane. This API enables the programmability of the network controller by exposing network data abstractions to the application plane. Currently, the most used protocol for this communication is the REST (Representational State Transfer). The southbound API defines the communication between the control plane and the forwarding plane. Through this interface, the control plane can configure switches with forwarding actions according to received notifications of incoming packets from the data plane. This is typically standardized and implemented by the OpenFlow protocol. More details about this API can be found in Subsection 2.4.2. The management API is responsible for the communication of the application, control, and forwarding plane with the management plane. The management API is responsible for providing information exchange between network management solutions and the elements in all other planes (KREUTZ et al., 2015).

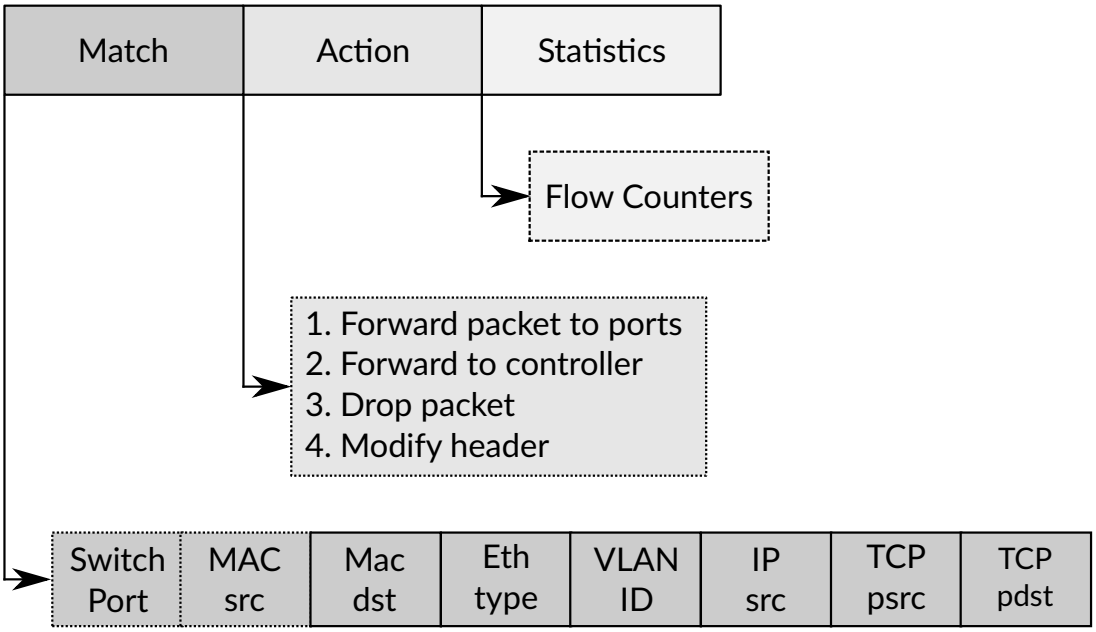
2.4.2 Enabling Platforms

In this Section, we present the two leading technologies that allow us to implement SDN concepts. First, we present the OpenFlow protocol specified by the Open Network Foundation (ONF) and the most adopted SDN enabling technology. Secondly, we present the Programming protocol-independent Packet Processors (P4) technology that allows the network operators to have a programmable control and data plane.

OpenFlow Protocol

The OpenFlow protocol provides a communication interface between the control and forwarding planes. In an OpenFlow-based network, the two main components are the controller and the forwarding devices. The controller is located in the control plane and has the function to define the forwarding rules and install them on the devices in the forwarding plane. The forwarding devices are located in the forwarding plane and have the function to store the rules and forward the data packets. A forwarding rule is defined by a tuple: a match field and an action. The match field is headers from different network protocols (*e.g.*, Ethernet, Ipv4, MPLS) and is used to identify the packets and apply the associated actions. The actions define what the forwarding devices shall do with the matching packets. The possible actions are: forward to some port, drop, forward to the controller and modify header fields. The forwarding devices store the rules defined by the controller in a data structure called flow table, which can be seen in Figure 2.5. Each entry in the flow tables is composed of a rule and statistics counters. When an incoming packet arrives in the forwarding device, a lookup process begins in the flow table. In this process, the match fields are compared against the incoming packets' header fields. When a correspondence occurs, then the action is applied (KREUTZ et al., 2015).

Figure 2.5: Flow table structure



Source: the author (2018)

The flow tables are implemented using Ternary Content Addressable Memory (TCAM). This memory can perform parallel comparisons across all entries that provide a better performance in the data forwarding. However, TCAMs have a limited storage capacity supporting just hundreds of forwarding rules what are not enough for some kinds of networks (*e.g.*, datacenters). To increase the amount of rule storage, forwarding devices use other memory types (*e.g.*, SRAMs, RAM) to aid in the implementation of flow tables. The problem of this approach is that these memories can degrade the performance of the forwarding devices. The match fields available are defined in the OpenFlow version. The first version of the OpenFlow protocol allows only one flow table with just 12 match fields (ethernet, TCP, IPv4). As the protocol was evolving, new specifications were made fields of other protocols were added. However, each time a new protocol was released, the OpenFlow specification had to be extended. In this sense, the Programming Protocol Independent Packet Processors (P4) comes up to add flexibility to the network programmable that SDN cannot provide.

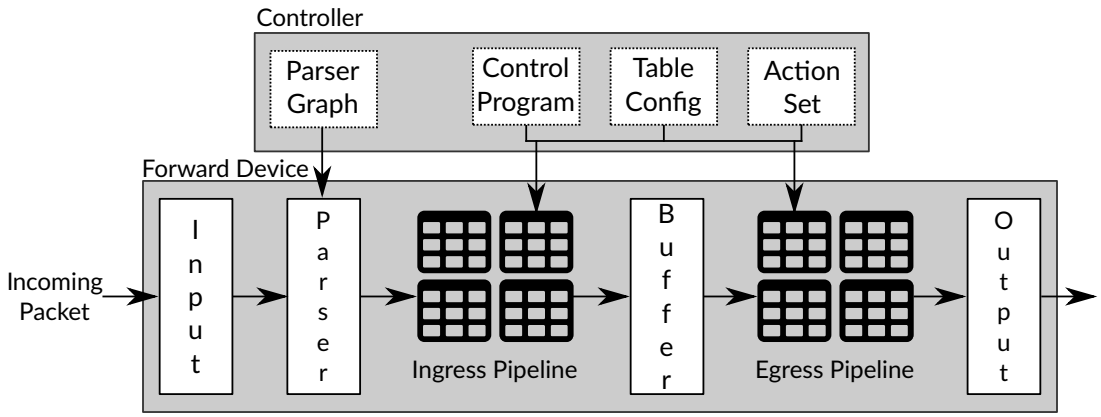
Programming Protocol independent Packet Processors

The Programming Protocol independent Packet Processors is a language for describing forwarding devices that allow the programmability of the data plane. The P4 main objectives are: (*i*) reconfigurability, (*ii*) protocol independence and, (*iii*) platform independence. Reconfigurability concerns the controller getting the forwarding device as it should operate and process the packets. Protocol independence means that the routing devices are not tied to a protocol

in specific (*e.g.*, OpenFlow version 1.3). Instead, the controller must be able to specify how the packet information should be stripped and to define what the match and associated actions are. Platform independence allows a program in P4 to be written, and through a compilation process, this same program can operate either in a virtual switch or in specific hardware (ASIC).

To achieve their goals, P4 uses the PISA (Protocol Independent Switch Architecture) architecture. This model can be seen in Figure 2.6. The parser must first handle packets arriving at the forwarding device. The parser recognizes and extracts header fields from packages. After that, the headers are submitted to the match tables and actions. Tables can be divided into ingress and egress. Ingress tables determine by which port the packets should be forwarded, replicated, or discarded. Ingress table allows controls by instances to be taken as access control, to bug statistical counters. A program written in P4 has two forms of operation: configuration and population. Configuration refers to the behavior of the components (parser, the order of the tables, and the association of counters). The population is the operations done to add and remove rules in the flow tables (BOSSHART et al., 2014).

Figure 2.6: PISA architecture



Source: the author (2018)

P4 allows network operators to have greater control over both the data plane and the control plane. This technology allows innovations in computer networks to be placed in the particle in a faster and more flexible way than if we OpenFlow is used. P4 allows the operator to define the protocols that he wants to implement as well as the interfaces between the control plane and the data plane. Another modulus is that the OpenFlow protocol only implements the header fields of the already widespread protocols, and can not make adaptations for tests with new technologies.

2.5 Chapter Summary

This Chapter presents a contextualization of the requirements of the new IoT applications. In this context, we have verified that LPWAN technologies can complement current wireless technologies providing long-distance communication with low power consumption and high scalability. We also introduce the techniques used by the LPWAN technologies used to achieve your goals. Also presented was a review of the leading LPWAN technologies present in the market and literature. Finally, we present the SDN network paradigm and the main technologies that enable its implementation. In the next Chapter, we will present the main works that use the SDN paradigm to implement networks for IoT. Also, we will show the advantages of applying the SDN paradigm in LPWAN networks.

3 RELATED WORK

This Chapter presents a literature review on the subjects related to our work. The study adopts the principles of a systematic review (PETERSEN; VAKKALANKA; KUZNIARZ, 2015) to mitigate biases in the selection of articles. First, Section 3.1 we describe our systematic literature review process used to obtain the works related to this one. After that, Section 3.2 explores the state-of-the-art SDN-based architectures that provide communication for IoT systems. Our analysis shows that these architectures require rule management strategies that cope with the demand of LPWAN. We focus on this aspect in section 3.3, which explores the literature about methods to manage rules in the flow table of SDN-enabled devices. Finally, Section 3.5 discusses the insights obtained from the related work and their impact on our work.

3.1 Systematic Literature Review

In this Section, we present the systematic research methodology used to review the literature. Firstly, we focus on verifying state of the art about the SDN-based architectures that providing communication to IoT nodes. Through this search, we identify a gap of works that attack the problem of the rule capacity of the flow table in SDN-based architectures for IoT. The LPWAN proposes communication to a massive number of end-devices (*e.g.*, 22000 end-devices for Lora). To address these number of end-devices is necessary, a large capacity of storage rules various levels of magnitude greater than is available in a flow table. Therefore, we performed second research to identify the rule management strategies in the flow table that better meet the requirements of an LPWAN network. In both cases, the goals of the research show the relevance of these two topics and identify the main works related to this one. To perform our research, we use the Scopus research tool. Being one of the largest available databases for academic research with the feature of eliminating irrelevant documents.

To perform the research for the SDN-based architectures for IoT, the following query was made:

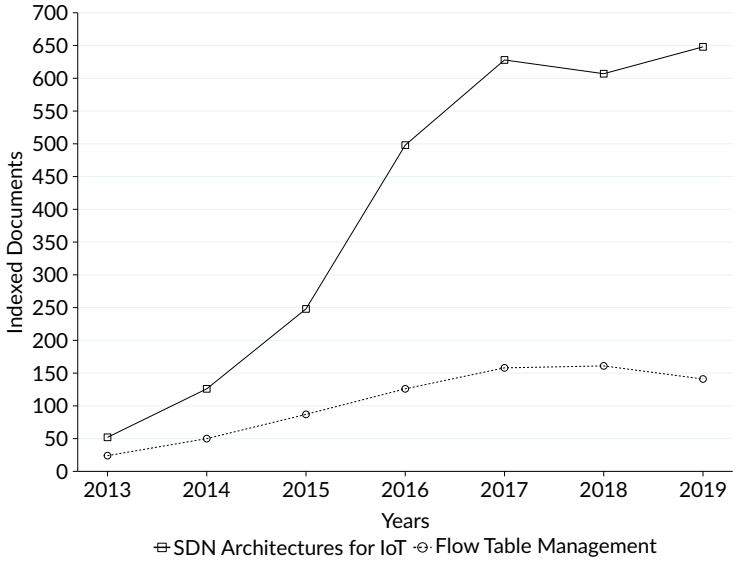
TITLE-ABS-KEY (((sdn AND wireless) OR (sdn AND iot) OR (sdwn))) AND PUBYEAR > 2013

On this query, we explicitly filter academic documents that apply the concepts of the SDN in IoT, wireless sensors, and LPWAN networks. We also apply a time window of five years to get more recent works. To perform the search for the SDN flow table strategies, we apply the flowing query:

TITLE-ABS-KEY ((sdn OR openflow OR p4) AND ("flow table" OR "flow rule" OR "rule") AND (management OR strategy OR mechanism)) AND PUBYEAR > 2013

On this query, we filter works that use the SDN paradigm and the technologies that enable their implementation (*e.g.*Openflow, P4) that proposes strategies or mechanisms to manage the rules in the flow table. We also apply a time window of seven years. On December 20, 2019, these two queries resulted in 2761 documents for the first one and 747 for the second one. The resulting documents were cataloged in a sheet and systematically categorized. The number of indexed documents by year is shown in Figure 3.1 to exhibit the evolution of interest in the area. We use three filters used in sequence to achieve the categorization of all documents:

Figure 3.1: Query result: number of indexed documents by year



Source: the author (2018)

In the first filter, the title and abstract of all results were read; results found to be clearly out of scope in this filter are then marked as such, and the remainder results pass on to the second filter. In the first query, from 2761, a total of 72 studies passed the first filter. In the second query, from 747, a total of 54 studies passed the first filter. In the second filter, the sections for Introduction and Conclusion of the remaining documents were considered; once

again, documents considered in the overall scope were passed on to the third filter, and the remaining results were discarded from consideration. In the first query, a total of 27 studies passed the second filter. In the second query, a total of 29 studies passed the second filter. Finally, the third filter induces the complete read of the article. Documents that go through the third filter are grouped by relevance to our research problem, categorized according to aspects, and, when deemed necessary, summarized for easier recapitulation afterward. In the first query, four studies passed all three filters. In the second query, a total of 3 studies passed all three filters. In the next sections, we discuss the works related to our research.

3.2 SDN Architectures for IoT

This Section shows the studies found in our literature review that focus on SDN-based architectures for IoT networks. It also presents the main insights that led to the second scope of our research, related to strategies to manage rules in flow tables of forwarding devices. Table 3.1 shows the comparison of the studies found in our review process. It identifies each article by the first author and publication year. We compare the studies using four criteria related to the features included in their proposals.

- **Control Plane Programmability:** The ability to modify the forwarding plane device's behavior through a controller.
- **Forwarding Plane Programmability:** The ability to change how forwarding devices process packages.
- **Network Heterogeneity**: The ability to support different wireless technologies.
- **Table Management:** The ability to apply algorithms to optimize the installation, removal, or replacement of rules in the flow table.

Reference	Year	Programable Control Plane	Programable Data Plane	Heterogeneity	Table Management
T. Luo *et. al.*	2012	✓			
P. Gallo *et. al.*	2016	✓		✓	
Baddeley *et. al.*	2018	✓			✓
S. Bera *et. al.*	2018	✓		✓	

Table 3.1: SDN Architectures with focus on IoT.

We argue that these criteria are the most relevant for this study because an SDN-based architecture for LPWAN must ensure control and forwarding plane programmability. Moreover, the architecture must support several LPWAN technologies for better coordination and coexistence between different applications. Therefore, it is necessary to use rule management strategies

in the flow table because LPWAN supports a massive number of end-devices, and the traffic generated by these nodes would result in a rule explosion in flow tables.

T. Luo *et. al.* (LUO; TAN; QUEK, 2012) propose the first study that applies SDN concepts to the IoT context. Their proposal is a modification of the OpenFlow protocol for use with Wireless Sensor Networks (WSN). The forwarding plane comprises sensors that act like forwarding devices, while the control plane centralizes the network intelligence. Authors also propose a modification in the OpenFlow protocol and flow table to enable support for WSN protocols (*e.g.*, ZigBee, 6LoWPAN). Since end-devices also integrate the network's forwarding plane, they must also implement a flow table. However, this approach wastes devices' energy because sensors must forward traffic and also communicate with the network controller. This approach is not ideal for LPWAN networks due to the energy restriction of end-devices. Among the evaluation criteria, the study only implements control plane programmability.

P. Gallo *et. al.* (GALLO et al., 2016) propose using SDN concepts to manage Medium Access Control (MAC) protocols to coordinate different WLAN technologies. Authors propose an architecture that treats each WLAN technology as an independent forwarding plane, with a single controller to manage all of them. The control plane has a MAC protocol database that can modify gateways to change active protocols. The architecture manages protocol changes according to network metrics obtained from the coordinating between involved technologies (*e.g.*, frame loss rate). The architecture employs SDN concepts on to forward flows and to control wireless environment parameters. According to our classification criteria, the described study considers control plane programmability and heterogeneity regarding the use of several WLAN technologies.

Baddeley *et. al.* (BADDELEY et al., 2018) evolve the work from T. Luo *et. al.* (LUO; TAN; QUEK, 2012) to enable application on networks with energy restrictions. They propose optimizations to the OpenFlow protocol to align it with the energy requirements of an IoT network. Among these optimizations, the most relevant are: (*i*) increase the expiration time of rules in the flow table, consequently avoiding the expiration of frequently used rules; (*ii*) a strategy to aggregate flows to reduce the number of entries in the flow table; (*iii*) use of Routing Protocol for Low-Power and Lossy Networks (RPL) to reduce the number of control messages generated by forwarding devices. The study focuses on flow table optimizations to reduce energy consumption with a aggregation strategy. However, aggregation imposes an overhead in the control channel due to the communication required to verify and apply the needed algorithms. According to our criteria, the work focuses on control plane programmability and flow table management.

Finally, S. Bera *et. al.* (BERA et al., 2018) propose an SDN-based architecture for IoT networks that optimizes the control plane to enable real-time reprogramming of the network. The control plane has two main functionalities: device management and topology management. Moreover, it proposes a new rule management policy that enables adaptive device functionality and network programmability. However, this proposal makes several modifications to the rout-

ing devices. From our defined criteria, this study focuses on control plane programmability and heterogeneity.

SDN application in IoT networks has been extensively explored in the literature of this first study that has been registered in 2012. All of the presented works have their proposals of architectures based on SDN considering programmable control plane. The proposal architectures deploy the OpenFlow protocol that is not flexible and updates take a long time to implement. Two proposal architectures present heterogeneity, providing some coordination mechanism among technologies. Finally, only one of the studies showed a design which is concerned with the management of rules in the flow table. Such a criterion is essential for LPWAN since these technologies support a massive amount of end-nodes. Performing the traffic forwarding of this amount of devices will cause an explosion of rules in the flow tables (KOBAYASHI et al., 2014). Since rule management becomes a relevant aspect to an IoT architecture based on SDN, we explore this aspect in more detail next. More specifically, we conduct an additional literature review focused on management strategies for flow table rules on forwarding devices.

3.3 SDN Table Management

We discuss in this Subsection, the second part of our systematic research that consists in analyze the main strategies of rule management in the flow table. The limited capacity for rule storage in forwarding devices motivated studies related to the management of rules in flow tables. The strategies investigated by these studies focus on two major domains: spatial and temporal. Strategies from the temporal domain remove/replace rules from a flow table after a given time interval while spatial strategies consider the forwarding and semantic of flows intra or inter network devices. We describe the four main strategies in the literature:

- Eviction (KIM et al., 2014) (RIFAI et al., 2015).
- Aggregation (CURTIS et al., 2011) (CHALLA; LEE; CHOO, 2016).
- Caching (KATTA et al., 2016) (MARSICO; DORIGUZZI-CORIN; SIRACUSA, 2017).
- Multiple Tables (NAKAGAWA et al., 2013) (ONF, 2015).

We conceptualize each of them and exemplify with the works found in our research. Table 3.2 summarizes the studies discussed in this Section.

A Eviction strategy is classically based on the temporal domain. The removal algorithm for a substitution strategy is a design choice that directly impacts network performance (LEE; KANAGAVELU; AUNG, 2013). For example, the use of a FIFO algorithm removes rules installed for the longest time in the flow table. R. Challa *et. al.* (CHALLA; LEE; CHOO, 2016) propose a probabilistic algorithm to identify rules with a high chance of not being used, which can be replaced by new ones. Autonomous substitution mechanisms manage flow table rules

Reference	Year	Eviction	Aggregation	Caching	Multiple Tables
M. Rifai *et. al.*	2015		✓		
R. Challla *et. al.*	2016	✓			
A. Marsico *et. al.*	2017			✓	
OpenFlow 1.1	2011				✓

Table 3.2: Flow Table Management Strategies.

without the intervention of a controller. This strategy does not rely only on the expiration time of a flow input. Instead, it applies a Multiple Bloom Filter (MBF) data structure to determine candidate rules for elimination. MBF is a space-efficient probabilistic data structure used to register the history of flow. A rule with activity receives a higher importance value through MBF and has a smaller chance of being removed from the flow table. When it is necessary to remove a rule, the algorithm employs MBF to identify a rule with a lesser chance of being used for a given time interval.

The main rule management strategies from the spatial domain are (*i*) Aggregation, (*ii*) Caching, and (*iii*) Multiple Tables. The first one operates by identifying rules with the same action, *e.g.*, forward packet to a given port. The second one, it identifies the rules' matching fields, *e.g.*, source IPs that belong to the same subnet. The third one, it processes them to generate a single rule, *e.g.*, forward all the source IPs of a subnet to the same port. Therefore, a rule can represent several flows, reducing the storage demand in forwarding devices. Aggregation is a traditional strategy to reduce the number of rules in routing tables of current IP networks. For example, OpenFlow-based networking also applies this strategy (NGUYEN et al., 2016), given its known performance.

M. Rifai *et. al.* (RIFAI et al., 2015) propose an aggregation strategy to reduce rules in flow tables. It considers rules as triples (s, t, p) where: s is the source address of a flow; t is the destination address; p is the output port. The algorithm applies a compression heuristic that results in three subtables. Each table aggregates rules according to one of the elements that constitute a rule. First, for each source IP s, the most recurrent output port p is identified, so all rules are replaced by $(s, *, p)$. The remaining rules do not change and have more priority than the aggregate rules. Once the algorithm verifies all source addresses, it applies the same compression considering the destination address $(*, t, p)$ and the output port of the forwarding devices $(s, t, *)$. Finally, the algorithm selects the resulting table with the least number of rules and uses it to replace the original set of rules.

A. Marsico *et. al.* (MARSICO; DORIGUZZI-CORIN; SIRACUSA, 2017) propose a caching strategy for managing flow tables. Caching is a rule management strategy that resembles that applied to operating system memory hierarchies. In the context of SDN, we can consider the Ternary Content Addressable Memory (TCAM) as the main memory with fast access and the controller as secondary memory. The system can transfer a predetermined number

of rules into the secondary memory when TCAM is at its maximum capacity. In this case, it is defined as two types of operations: *swap-in* and *swap-out*. The *swap-in* occurs when the flow table is operating at maximum capacity, and the flow requires a new rule. Therefore, the controller swaps a rule from the flow table to its memory based on periodically collected statistics, *e.g.*, rules with few correspondences. Rules with infinite idle or hard timeout are not eligible for swapping operations. The *swap-out* operation changes part of the rules in the flow table with the ones available in the swap memory. The switch queries the controller when the correspondent rule for a particular packet does not find. The controller first performs a lookup in the swap database and, if it finds a rule, re-installs it in the flow table.

Finally, Multiple Tables management operates by identifying the semantics of flow and dividing it into two or more rules. For example, the forwarding port of flows is received in a given virtual interface. From this division, a new rule is created with simplified semantics and stored in a table that represents it. Therefore, rules already installed and with simples semantics can represent a new rule with a complex semantic, resulting in lower memory usage in forwarding devices. OpenFlow 1.1 provides multiple flow table management, with its current form encouraged by Open Network Foundation (ONF) (ONF, 2015). Its use adds more complexity to the lookup process for OpenFlow devices that implement hardware flow tables. Multiple flow tables management is a subject with scarce exploration in the literature, although having great potential to reduce the number of installed rules.

3.4 Discussion about Related Work

Our literature review indicates that SDN applied to IoT networks is an active research topic with several published articles. The evaluated studies propose adaptions to the OpenFlow protocol to optimize it in terms of energy consumption and compatibility with other protocols such as Zigbee and 6LoWPAN. However, these adaptations are not sufficient to enable the direct employment of SDN concepts to LPWANs because of three main aspects. First, many of these studies focus on multi-hop mesh networks, which present a severe restriction due to the energy constraints of end-nodes. Second, the OpenFlow protocol can limit new solutions due to its rigidity and low versatility for adaption to other network protocols. Third, most of these studies lack proposals to manage flow table rules, which becomes a relevant aspect given the number of rules generated in IoT scenarios.

The third aspect is particularly crucial because of LPWANs promise support $\approx 10^6$ end-nodes per km^2 (RAZA; KULKARNI; SOORIYABANDARA, 2017), resulting in several rules that exceed the capacities of current forwarding devices. Consequently, an architecture that integrates SDN in the IoT context must consider a robust strategy to manage rules installed in forwarding devices. To better explore this aspect, we conducted an additional systematic review to identify the best solutions to manage flow table rules.

Rule management strategies present two main groups related to the temporal and spatial do-

mains. Temporal strategies are to replace rules of the table of flows based on some substitution algorithm. However, this method generates an overhead in the control channel because of the large amount of removal and insertion of new rules. In turn, strategies from the spatial domain provide better utilization of the storage capacity available in flow tables. In this context, three strategies stand out: aggregation, caching, and multiple tables. Aggregation strategies create a set of generic rules capable of addressing multiple streams. Their drawback is that the aggregation process does not allow fine-tuning of individual flows. Caching is a strategy that uses an auxiliary memory to store for faster queries than to process a new rule. Finally, multiple tables are a promising strategy not widely explored in the literature.

3.5 Chapter Summary

In this Chapter, we present the methodology of our systematic research and the results obtained through it. We have verified that the research topic on the application of SDN in IoT networks is still a hot topic and several research has been produced in this context. The works found make adaptations in the OpenFlow protocol so that it is optimized in terms of energy consumption and compatibility with protocols such as Zigbee and 6LoWPAN. However, these adaptations are not sufficient for the application of SDN concepts with LPWAN networks. Firstly, because many of these works consider mesh networks multi-hop, which is not interesting due to the energy restriction of the end-nodes. In addition, the use of the OpenFlow protocol can be a limiter for new solutions due to its rigidity and little versatility for adding new protocols. Another problem of the application of these architectures is the lack of proposals for the management of rules in the tables of flows. LPWAN networks promise support for up to 22000 end nodes which should generate a number of routing rules that are not supported by the routing devices. This factor led us to carry out a second systematic search to identify the best ways of managing rules in the flow table.

The next Chapter presents the HELPFUL architecture. It addresses the weak points of the related study identified in our literature review. HELPFUL provides both forward and control plane programmability, heterogeneity, and scalability by using a multiple flow table strategy. Furthermore, given the relevance of rule management in forwarding devices, HELPFUL incorporates support for different rule management strategies.

4 HELPFUL: CONCEPTUAL SOLUTION

In this Chapter, we introduce a flexible architecture to control heterogeneous LPWANs, called HELPFUL. In Section 4.1, we present an overview of the HELPFUL conceptual architecture. Next, in Section 4.2, we describe in details the HELPFUL Forward plane. After that, in Section 4.3, we describe the HELPFUL Control plane. Next, in Section 4.5, we present the HELPFUL Management Plane. Finally, in Section 4.6 we describe the main interactions between HELPFUL planes.

4.1 Architecture Overview

HELPFUL is an SDN-based architecture divided into four planes: (*i*) forward, (*ii*) controller, (*iii*) application, and (*iv*) management. Figure 4.1 illustrates an overview of the architecture. Next, we describe each of the depicted components.

The HELPFUL Forward plane comprises the network devices that use LPWAN technologies, (*e.g.*, LoRa, Sigfox, and NB-IoT) for interconnection. Two components handle these network devices: (*i*) the Translator and (*ii*) the Dynamic Control Tables. The Translator component creates a common control abstraction among different LPWAN technologies. Its main goal is to hide specific aspects from different LPWAN technologies, in particular from specific fields from packets. It comprises two components: the Packet Dissector and the Packet Factory. The Dissector receives packets from specific LPWAN technologies and abstracts them into common packets handled by the HELPFUL architecture. In turn, the Factory takes abstract packets and convert them into technology-specific packets for the LPWAN infrastructure. Next, the Dynamic Control Tables component stores the rules that define network functions. Each network function has a corresponding table on the Control Tables (*e.g.*, forwarding table, or channel selection table). Furthermore, each table has its rules with its respective match fields and actions (*e.g.*, forward to specific ports, drop, forward to the controller, and rewrite some header). The Pipeline Control manages the packet flow between Translator components and available Control Tables, including the processing order for packets.

The HELPFUL Controller plane configures the network devices connected on the HELPFUL Forward. The configuration of devices comprises the installation of network rules on

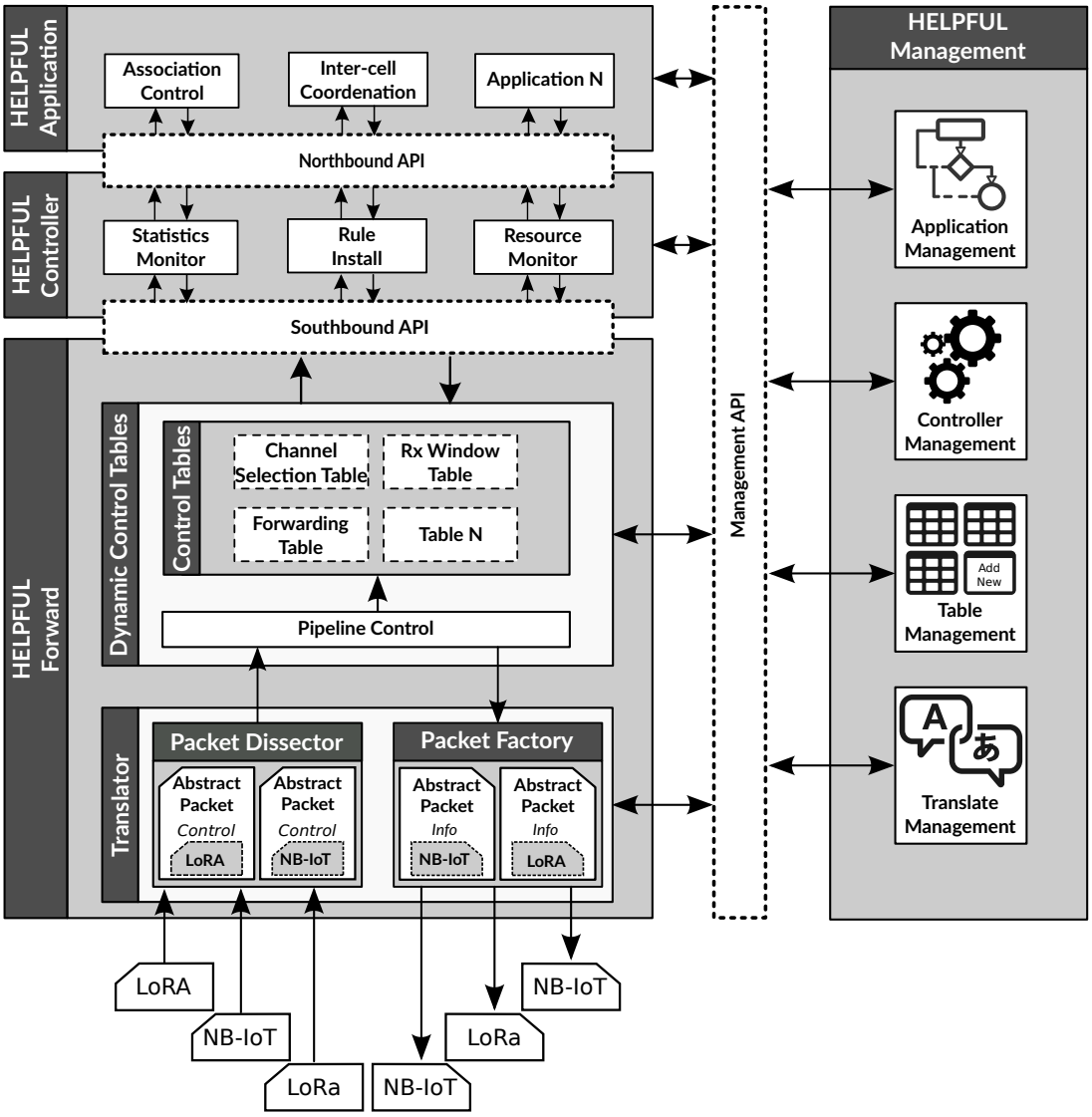


Figure 4.1: HELPFUL architecture overview. Gray boxes represent the architecture's planes, each one comprised of multiple components, depicted as white boxes. Arrows illustrate the data flows that may exist among the architecture's components.

Control Tables. In turn, each table represents one LPWAN technology or a network functionality and holds the match fields and actions for the specific network. The Southbound API is the interface between HELPFUL Forward and the HELPFUL Controller that enables the installation of rules. This API also sends statistical about each rule installed in Control Tables (*e.g.*, packet counts, byte counts, and idle time) to the Statistics Monitor component. Finally, the HELPFUL Controller also collects information about the physical resources available (*e.g.*, TCAM and CPU) on network devices connected on the HELPFUL Forward Plane.

The HELPFUL Applications plane contains orchestration functions, network logic, and high-level network services (*e.g.*, transmission scheduling and vertical handover). This plane executes applications that control the functions of LPWAN technologies. It allows service providers to redefine an entire heterogeneous LPWAN deployment dynamically. In this case,

it is possible to have an application optimized for a specific network demand, *e.g.*, design a scheduling algorithm to mitigate the interference for a massive number of sensors. The Northbound API provided by the HELPFUL Controller enables the interaction from the applications with the services provided by the architecture.

The HELPFUL Management plane independently interacts with all other planes. It comprises five components:

- **Application Management:** Adds, removes, and collects information about applications running in the HELPFUL application plane.
- **Controller Management:** Sends rule definitions (*e.g.*, the matches used on specific tables) and statistical data to the HELPFUL Controller plane.
- **Table Management:** Installs and removes tables from the Dynamic Control Tables.
- **Translate Management:** Configures the translation process that converts a technology-specific packet into an abstracted one for use with the Dynamic Control Tables.
- **Managment API:** Provides and standardized interface for the communication between the Management plane components and the other HELPFUL planes.

The HELPFUL Forward plane contains the core functionality of the architecture, thus its design directly impacts on the flexibility and performance of our proposal. The next Section explores in more detail the HELPFUL architecture components and the decisions taken in its design.

4.2 HELPFUL Forwarding

In this Section, we present the HELPFUL Forwarding Plane and describe its main components. On this plane, it is possible to locate the ((*i*)) sensors or IoT devices that collect data from the environment and periodically transmit the data collected to applications servers or other sensors through an LPWAN technology; and ((*ii*)) forwarding devices that relay the traffic of the sensors. The HELPFUL architecture uses LPWAN topology concepts; in other words, the sensors do not send data directly to their destiny. First of all, the data are sent to the forwarding devices (that apply network control configurations) and just then, forward the sensors data to their destiny. Also, like an SDN architecture, the HELPFUL forwarding devices do not make autonomous decisions, that is, just applying the configurations defined by a network controller. These configurations are called network rules and are stored in the control tables present in the forwarding devices. Each control table represents a particular network function and stores rules related to those functions.

Next we will describe the two components that together make up the forwarding devices on the HELPFUL architecture. First, we will describe the Translator that has the function of

converting the packages of a specific LPWAN technology to a standard packet format that can be matched against the network rules stored on the dynamic control tables. Moreover we describe how dynamic tables control are organized and dived by network functions.

Translator

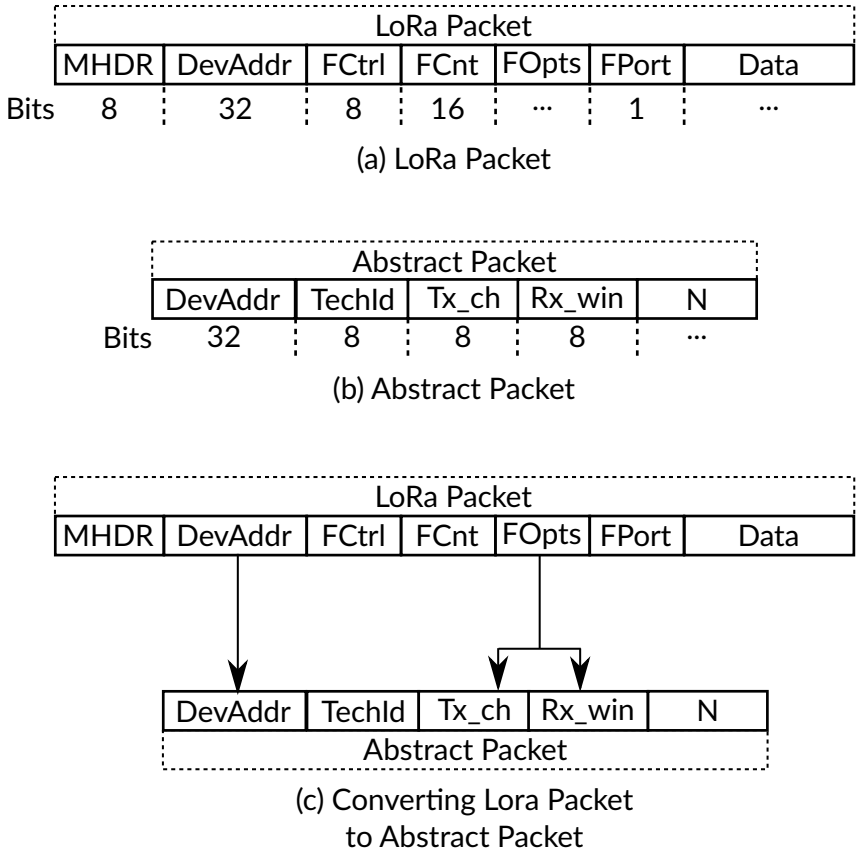
In this Subsection, we will introduce the Translator of the HELPFUL architecture. Its primary purpose is to create an abstraction layer that can convert packages from a specific technology to an abstract packet that can be controlled and forwarded by a HELPFUL forwarding device. In this sense, it is intended to homogenize the control of a heterogeneous LPWAN network. So that a single forwarding device can forward several technologies LPWAN technologies and a single controller could control those technologies. It is possible because of unique features of LPWAN technologies, which control information are sent together with data packets, there are no control channel or packets just for network control(*e.g.*, LoRaWAN and Sigfox). Besides, many of the control information have analogs messages between technologies. As an example, it is possible to cite de control messages present in Lora and NB-IoT technologies, where both technologies have messages to control the size of reception windows, the definition of the channel of the next data transmission and, inform if some packet needs confirmation process. In this sense, the Translator extracts the control information from a package of a specific LPWAN technology and transforms it into a standard package that will be matched against the rules stored on the dynamic control tables.

The Translator has two components: ((*i*)) the wrapper and ((*ii*)) the unwrapper. The former has the function of extracts the control and forwarding information present in a specific LPWAN technology packet and generates an abstract packet that can group the control information of various technologies. The standard packet is only used within the forwarding device to enforce control rules (*e.g.*, set the port to forward, modify packet headers, or discard the packet). After that, the standard packet is sent to the dynamic control tables. The unwarpper is responsible for getting the information of the standard packet and remounting the LPWAN technology-specific packet with the new information added by the dynamic control tables. After that, the packet is forwarding as the network rules definition.

To illustrate the translation process, we present an example that can be seen in Figure 4.2. In this example, a LoRaWAN packet (Figure 4.2(a)) has just arrived at a forwarding device. The standard packet (Figure 4.2(b)) has to be filled with the device and technology identification (fields that are mandatory) and, with the channel and the delay for the second window (fields are optional). First, the translator will search for the required fields of the generic package. It looks in the LoRaWAN header for the DevAddr field that represents the address of the sensor to which the packet will be sent. Then the wrapper fills in the TechId field indicating the technology that originated this package, in this case LoRaWAN. Finally, the wrapper will fetch the information in the FOpts field to collect information regarding the transmission window and

channel selection and transmission window.

Figure 4.2: Example of the Translator Packets



Source: the author (2018)

Dynamic Control Tables

In this Section, we describe the Dynamic Control Tables of the HELPFUL architecture. This component is present in the forwarding devices and has, as the main propose to improve the scalability of an LPWAN network. We divide the Dynamic Control Tables into two subcomponents: (*i*) Control Tables, set of tables that store the network rules defined by a controller; (*ii*) Pipeline Control, which defines which tables a given packet will be matched. Both components will be described below.

The Control Tables have the function of storing the network rules defined by a controller. On the HELPFUL architecture, we propose a model of management rules on the flow tables and a match/action mechanisms that apply the concepts of multiple control tables to improve the scalability of an LPWAN sensor network. In this model, each table represents a network function and has its own set of header fields for identifying a package, actions to be applied and, rules counters. Further, each table implements just the necessary to make his function and

avoiding consuming resources unnecessary (*e.g.*, memory). We can make an analogy with the OpenFlow protocol version 1.4 that has 41 header fields used for the match process. First, all tables present in the forwarding devices will have those 41 headers, even those headers will not be used. Secondly, add new fields will result in maintenance difficult once a new protocol version has to be updated on the forwarding devices.

The Pipeline Control has the function to select the sequence of the tables that a determinate abstract packet will be processed. The pipeline control receives the abstract packet generated by the translator. End then apply selection criteria to define the tables that will process the packet. This selection could be made by the technology that generates the abstract packet of some technology. For example, a LoRaWAN packet could be summited to tables 'Rx Window Table' and 'Rx Channel Table', while an abstract packet generated by an NB-IoT needs to be processed by the 'Rx Channel Tables'. Both packets will be submitted to the 'Forwarding Table' that will define by what port the packet will be forwarded. In this way, it is possible to implement functionalities that are present in various LPWAN technologies or functionalities that are exclusive in one technology. It makes the architecture more flexible and allows a forward device to control more the one LPWAN technology.

4.3 HELPFUL Controller

In this Section, we introduce the HELPFUL Control Plane. This plane has the function to define the network rules and installed it into the forwarding devices by a well defined Southbound API. The HELPFUL control plane is similar to the control plane present on the classic SDN architecture. The main difference is our control plane is more adaptable and flexible than the classic SDN control plane, once the HELPFUL Management Plane can modify the behavior of the controller by adding new rules and action formats, new counters on tables and, new resources to be monitored. The control plane of the HELPFUL architecture is divided into three functions:

- **Rules/Actions Definitions:** It has the function to format and install rules into the forwarding devices by a southbound API. The definition of a rule is similar to the definition of a rule in the SDN classic architecture. The difference is that each table implements its own set of match fields and actions.

- **Rules Statistics:** It has the function to collect statistics present on the control tables and provide those pieces of information to the applications present in the higher plane. This informations will be available to the applications of the HELPFUL application plane present on the top of the architecture. This module continually communicates with the forwarding devices and updates the information for the applications. Each rule in the table has its statistics that are: (*i*) Packet counter, that represents the packets that matched with the rule, (*ii*) Idle time, that represents the time that some rule do not have a match,

(*iii*) Byte counts of each rule.

- **Resource Monitor:** It has the function to monitor the physical resources on the forwarding devices (*e.g.*, memory, and CPU load). This component allows the controller to perform a load balance based on the resources that are available on the forwarding devices. It also can change how the controller manages the rules on the control tables. In a scenario, with a limited amount of memory on the network device, it is possible to use some rule replacement strategy to optimize the memory used to store rules.

4.4 HELPFUL Application

In this Section, we will briefly describe the HELPFUL Application plane. On this plane is located the network applications that can be seen as the control logic of the architecture. This logic is translated into commands to be installed in the forwarding devices to dictate the behavior of the equipment. As an example of applications, we can say transmission scheduling or vertical handover. These applications communicate with the forwarding devices through a Northbound API. This interface allows the operator to dynamically reconfigures the LPWAN network, dictating the behavior of the HELPFUL controller.

The applications can manage the limited spectrum, allocating radio resources, implementing handover mechanisms, managing interface, and performing efficient load-balancing between cells. All these applications can be seen as a set of software, each one controlling one aspect of the HELPFUL architecture. The Northbound API provides a set of messages to the applications that can allow entry reconfiguration of the forwarding devices presents on the LPWAN network. The messages can be decided in three groups: (*i*) Monitoring statistics, (*ii*) Monitoring physical resources, and (*iii*) Install/Uninstall rules. The first group of messages represents the messages used to get the rules statistics to provide the applications the necessary input to naming the aspects. The second proud is responsible for providing to some applications the consumption of physical resources of forwarding devices that allows the applications to instantiated new tables with more capacity or an entry new table. The last group is defined by the messages that inform instructions that the controller will translate into rules and installed on the forwarding devices.

4.5 HELPFUL Management

In this subsection, we will describe the HELPFUL Management plane. This plane is vertical to the other planes present on the HELPFUL architecture. The primary objective of the Management plane is to adapt the components to the other planes accordantly with the needs of the network operator. With this plane, it is possible to instantiate new tables on the forwarding devices, add new translate methods, add new rules into the controller, or add new applications

on the top of the architecture. For each plane present on the preview subsections, the HELPFUL management plane has a component communicated and reconfigure the other planes through a Management API. Those components are:

- **Translator Management:** It has the function to adapt the translator present in the forwarding devices. The network operator had to inform how the process of translation will be done. The translator management has a strict relationship with table management and control management. Once, this component will define the format of the abstract packet. Further, the format of the packet has to be replicated on the new tables that will be added into the forward devices, and the rules add to the controller.

- **Table Management:** It has the function to add new control tables accordantly to the needs of the network operator. To add a new, the network operator has to inform a description of the new table. In this description, it is essential to inform the header fields that will be used in the match process, the actions to apply in the packets, statistics counters to each table entry, and the number of rules the table will contain. Further, that table manager has to update the pipeline control adding the new logic to select the tables that are being installed. In the case of the tables needed more number of rules, the table manager can increase the number of rules that the table could support or add a new table identical. It is important to notice that the format of the abstract packet has to be reflected in the table.

- **Control Management:** It has the function to create the rules based on the tables installed on the forwarding devices. This component works together with the table management because the rules present in the controller has to be reflected in the header fields present on the tables and the abstract packet. Further, the control management can update the controller and modify the statistic monitor and the resource monitor to modify the statistics that have to be collector the frequency that they are collected. Also, new actions and parameters can be added to the controller by the control management.

- **Application Management:** It has the function to instantiate new applications to run in the top of the HELPFUL architecture. The network operator can run the applications that the network needs. For example, with the network operator needs to improve the quality of the handover of the network, it can easily instantiate a new handover application in the application plane through HELPFUL management.

In the next section we describe how the HELPFUL planes interact with each other.

4.6 HELPFUL Interaction

The Control Tables store the network rules defined by a controller. The HELPFUL architecture proposes a management model and a match/action mechanism for flow table rules. This

model applies the concept of multiple control tables to improve the scalability of an LPWAN sensor network. More specifically, each table represents a network function and maintains a set of header fields to identify packages, actions to be applied, and rule counters. Each table maintains only the information required for a specific network function. This way, tables consumes fewer resources, such as memory space. We can make an analogy with the OpenFlow protocol version 1.4. OpenFlow has 41 header fields used for the match process. First, all tables present in the forwarding devices must have those 41 headers, even those not used. Second, the addition of new fields results in difficult maintenance because it requires the installation of an updated firmware on forwarding devices, containing a new OpenFlow version.

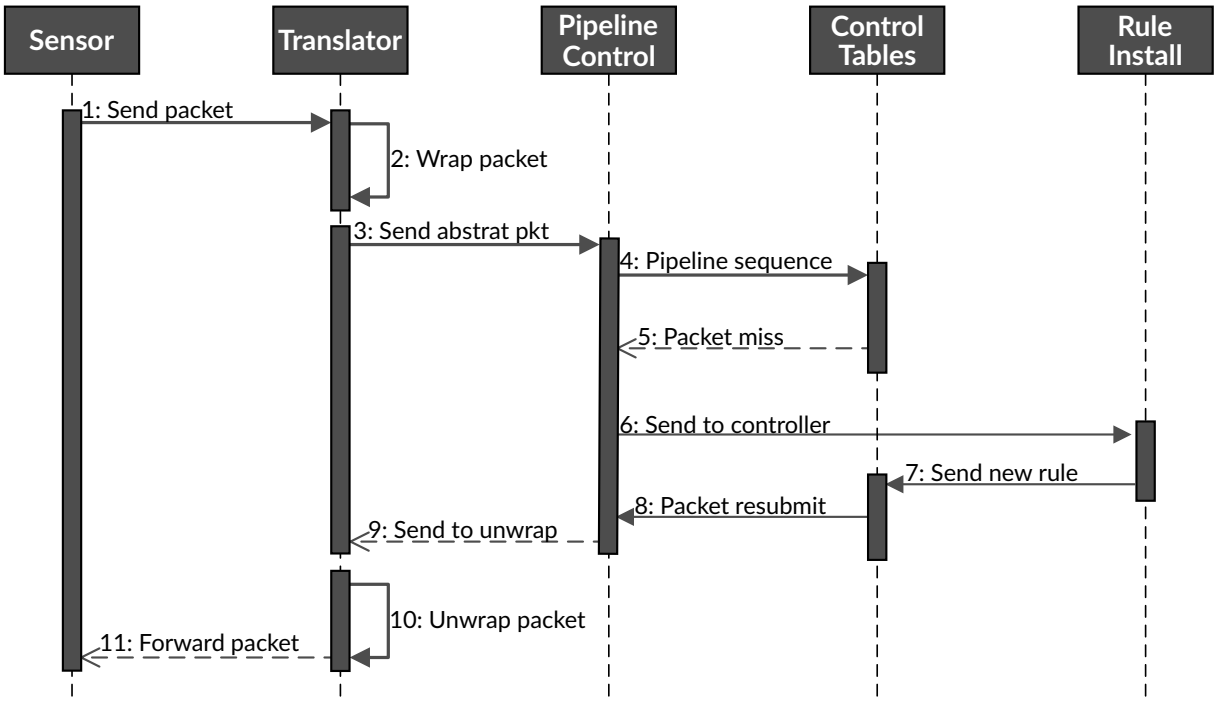


Figure 4.3: Table Miss Sequence Diagram.

Figure 4.3 exemplifies the actions taken by HELPFUL when a table miss event occurs. For example, when a packet arrives on a forwarding device and there is no rule in the flow table that identifies it. The entities that interact with each other in the sequence diagram are:

- **Sensors:** Send user data to application servers.
- **Translate:** Transforms technology-specific packets into abstract ones.
- **Pipeline Control:** Defines the sequence of tables to process an individual packet.
- **Control Tables:** Configure tables and manages the matching process.
- **Rule Install:** Adds new rules into Control Tables.

If a given abstract packet does not match any of the installed rules, it returns to the Pipeline Control and generates a table miss event (steps 1, 2, 3, 4, and 5). The Pipeline Control encapsulates the abstract packet and sends it to the Controller (step 6). The Controller defines a new rule and installs it in the flow table (step 7). When necessary, the Controller applies a substitution algorithm (*e.g.*, random, least recently used) to replace a rule when the table is full. After the installation of the new rules, the Control Tables reprocess the packet that generated the table miss event (steps 8 and 9). Finally, the unwrap transforms the abstract packet in a technology-specific one, for example, LoRa or NB-IoT (steps 10 and 11).

The HELPFUL architecture also enables the addition of new tables in the Dynamic Control Table and new translation definitions. The Management plane acts to change the behavior of control tables and the translator. Its primary objective is to adapt components to other planes according to the needs of the network operator. The functionality of this plane enables the addition of new tables on forwarding devices, new translation methods, new rules into the controller, and new applications on the architecture's top. The HELPFUL Management plane has a component that communicates and reconfigures each of the other architecture planes via the Management API.

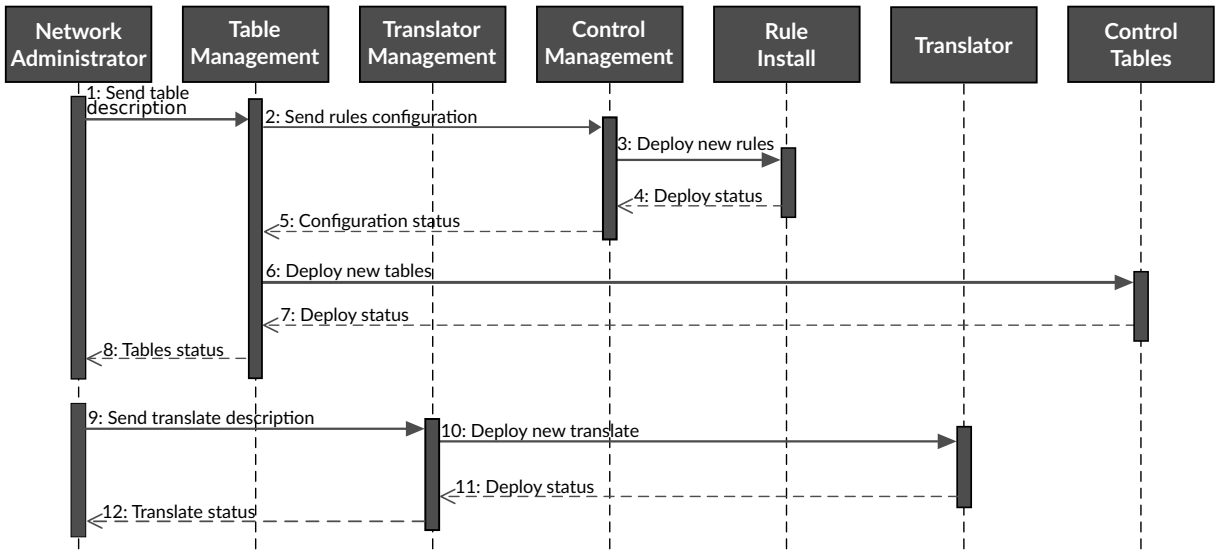


Figure 4.4: Control table update process.

Figure 4.4 illustrates the sequence of messages exchanged in the process of adding a new table to a forwarding device. The entities that interact with each other in the diagram are:

- **Network Administrator:** Inputs the new tables and translator definitions.
- **Table Management:** Interprets the description and forwards it to the Control Management, and also deploys the new tables.
- **Translate Management:** Receives the new translate definitions and deploys the new meanings in the translator present in the forwarding devices.

- **Control Management:** Implements the new rules in the Rule Install component into the controller.
- **Controller:** Responsible for define and install rules in the HELPFUL Gateway.
- **Translator:** Receives the instructions to translate the packets.
- **Control Tables:** Receives the new tables.

The process description is divided into two independent parts: (*a*) table update (step 1) and (*b*) translation update (step 9). These two parts are independent and do not require interactions with each other. In a table update, the network operator informs a new table description (step 1). Table Management interprets the table description and sends the new configuration to Control Management (step 2). After that, Control Management deploys the new rules on Control Tables of the forwarding devices (step 3). After deployment, the Rule Install returns the status to Control Management (step 4) and the status to Table Management (step 5). To finish the update, Table Management deploys the new tables in the Control Tables of the forwarding devices (step 6). In turn, a translate update is very similar to a table update. Therefore, we describe the process omitting the steps that are equal to those of a table update. First, the network operator notifies the new format used by the Translator entity to convert packets between technology-specific and abstract (step 9). Next, the Translator Management deploys the new format into the Translators in forwarding devices (step 10). Finally, the Translator returns the status of the deployment (step 11), and the Translator Management returns the status to the Network Administrator (step 12).

4.7 Chapter Summary

This Chapter presented the HELPFUL architecture, proposed in this master thesis to allow an environmental with dynamic, adaptable and flexible to the emergence LPWAN technologies. The Chapter began with an overview description of the architecture. In the sequence, we describe in details each one of the HELPFUL planes and components. The next Chapter describes the HELPFUL architecture prototype. It discusses the technologies employed in different software components. It also presents the correspondence between the architecture's entities and different software tools. Based on this development, we analyze the HELPFUL performance in the next Chapter.

5 HELPFUL EVALUATION

This Chapter presents the methodology used to evaluate the HELPFUL architecture and the different rule management strategies. Section 5.2 presents the experimental scenario, workload, and the evaluated metrics. Next, Section 5.3 shows the performance analysis of the flow table in HELPFUL. Finally, Section 5.4 presents the impact of the rule table strategies in LPWAN.

5.1 Framework Implementation

This Section describes the HELPFUL architecture prototype, developed to enable the experimental evaluation of the proposal. First, we present an overview of the developed components and the tools used to realize the performance evaluation. Next, we introduce a detailed description of the architecture's main components implementation. It is important to highlight which HELPFUL is an open-source project and the code can be access in a repository public[1].

Figure 5.1 illustrates the tools used to implement the HELPFUL architecture. We use the Mininet[2] SDN emulator to prototype HELPFUL. The bottom part of the picture presents the components implemented as a proof-of-concept:

- **End-nodes:** are the sensors that periodically send messages to Application servers, *i.e.*, LoRa and NB-IoT devices. Sensors prototyping employs the Python programming language.
- **HELPFUL Gateway:** Represents the architecture's forwarding plane, developed using the P4 programming language.
- **Controller:** Implementation uses the Runtime CLI tool[3] available in the P4 language framework.
- **Application Servers:** Uses Python as the programming language.

The upper part of the picture illustrates the software modules and tools used in the implementation. Sensors and Applications share many implementation traits. For example, both

[1] https://github.com/gustavo978/lpwan-ns
[2] http://mininet.org/
[3] https://p4.org/

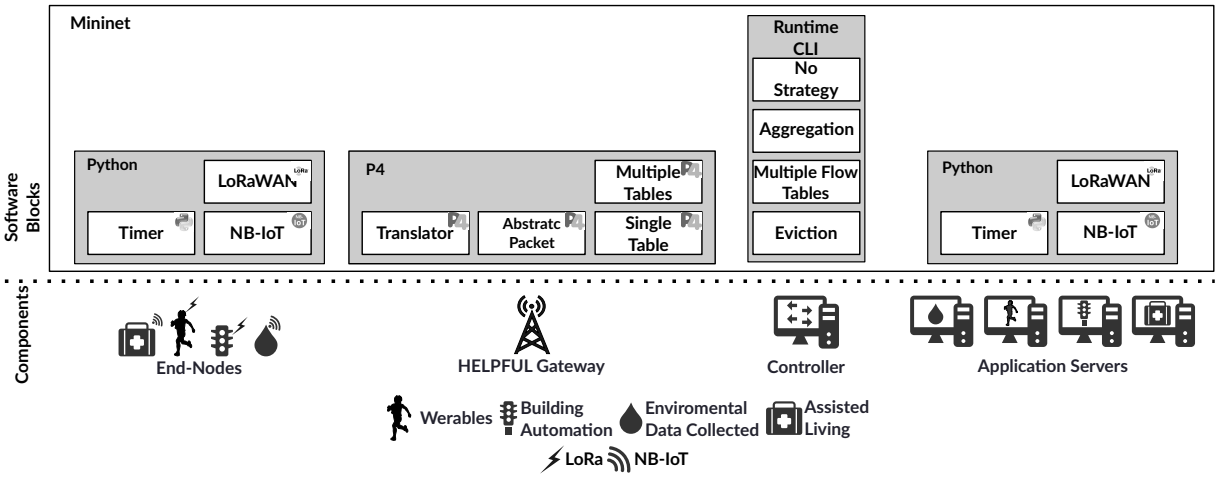


Figure 5.1: HELPFUL implementation tools.

components have a Timer module with a function that counts time and periodically sends messages. They also embed LoRa or NB-IoT communication technologies. The creation and transmission of messages by both components use a raw socket library. Sensors send messages with a periodicity of 5 seconds, but this time can be configurable as a simulation parameter. In turn, Application servers receive these messages, count them, and send an acknowledgment back to sensors. After receiving the acknowledgment, sensors enter an idle mode according to application profiles (described in the next chapter) until the time to send the next message arrives.

The Translator module identifies the packets that arrive in the HELPFUL Gateway. Therefore, it determines which type of packet arrived and then extracted the packet header. Next, the abstract packet is created using the metadata functions present in the P4 language with information of the packet headers extracted. Metadata allows that a group of properties can be assigned to each packet that arrives on the HELPFUL Gateway, independently. After that, the abstract packet is submitted through the Dynamic Control Tables. In our implementation, the HELPFUL Gateway can work with single or multiple tables depending on the selected management strategy.

The HELPFUL Controller employs the reference implementation from the P4 *Runtime CLI*. It also implements the following rule management strategies: (*i*) Unmanaged, (*ii*) Aggregation, (*iii*) Eviction, and (*iv*) Multiple Tables. The Unmanaged strategy does not control rules in the Flow Table. It only performs random substitutions when tables are at maximum capacity. The Aggregation strategy implements the algorithm from Rifai et al. (RIFAI et al., 2015). It creates three tables: the first aggregates rules by source, destination, and output port; the second aggregates rules by destination and output port, and the third aggregates rules only by output port. In the HELPFUL prototype, after table creation, the Controller selects and installs the table with the least amount rules from those three. The Eviction strategy implements a Least Recently Used (LRU) algorithm, which replaces rules not used for the longest time first (KANNAN; BANERJEE, 2014). Finally, the Multiple Tables strategy (ONF, 2015) subdivides one table

(*e.g.*, supporting 2500 rules) into two smaller sub-tables (*e.g.*, supporting 1250 rules). Each of the sub-tables groups a rule set, so HELPFUL has a sub-table for storing rules with the frame's source address and a sub-table for storing rules with the frame's destination address. The first sub-table that the frame access is a source-destination when a frame arrives at the HELPFUL Gateway. This sub-table contains the information that indicates if a packet should be dropped or forwarded to the next sub-table. The second sub-table has information to identify if a particular frame has to be forwarded to the network or dropped. The Multiple Tables strategy requires additional implementations in the HELPFUL Gateway to include the necessary sub-tables. We summarize four strategies and the replacement method used in Table 3.

Strategy	Replacement	Reference
Unmanaged	Random	-
Aggregation	Random	M. Rifai *et. al.*
Eviction	LRU	K. Kannan *et. al.*
Multiple Tables	Random	(ONF, 2015)

Table 5.1: Application Profile Specifications.

5.2 Methodology

The experimental scenario aims to represent an environment with multiple IoT applications, such as smart homes, industry 4.0, and wearable devices. Figure 5.2 illustrates the scenario and its main elements. The network topology encompasses a single cell with one BS to compare the different forms of rule management strategies, similar to the work from A. Marsico *et. al.* (MARSICO; DORIGUZZI-CORIN; SIRACUSA, 2017), R. Challa *et. al.* (CHALLA; LEE; CHOO, 2016), and S. Bera *et. al.* (BERA et al., 2018). In the simulation, BS receives user data sent by end-nodes using the LoRa and NB-IoT technologies. After receiving user data, BS forwards it to application servers over a wired network. The experiments used a total of 1500 end-nodes, starting at 300 and increasing the number of end-nodes by 150 for each iteration. BS has a maximum capacity of 2500 rules, as observed in devices commonly found in the market (STEPHENS et al., 2012), such as Pica8 3290[4]. In this case, BS presents a flow table composed of two header fields to perform the lookup process: source and destination addresses. The actions that can be associated with each rule are: discard a frame or forward a frame to their destination. Implementing forwarding with a single table requires several rules to represent all possible combinations for end-nodes and application servers. Therefore, the required number of rules is $O(n^2)$, where n is the total number of network elements (end-nodes and application servers). In this case, it is possible to overload the maximum capacity of storage rules in the flow table. Moreover, on the evaluation of the Multiple Tables strategy, BS employs a total of

[4]https://www.netsphere.com.hk/Downloads/Library/Pica8/Pica8_P-3290_P-3295_datasheet.pdf

two sub-tables with a maximum capacity of 1250 rules. Therefore, the comparison with the other strategies becomes fair since the maximum capacity is 2500 rules.

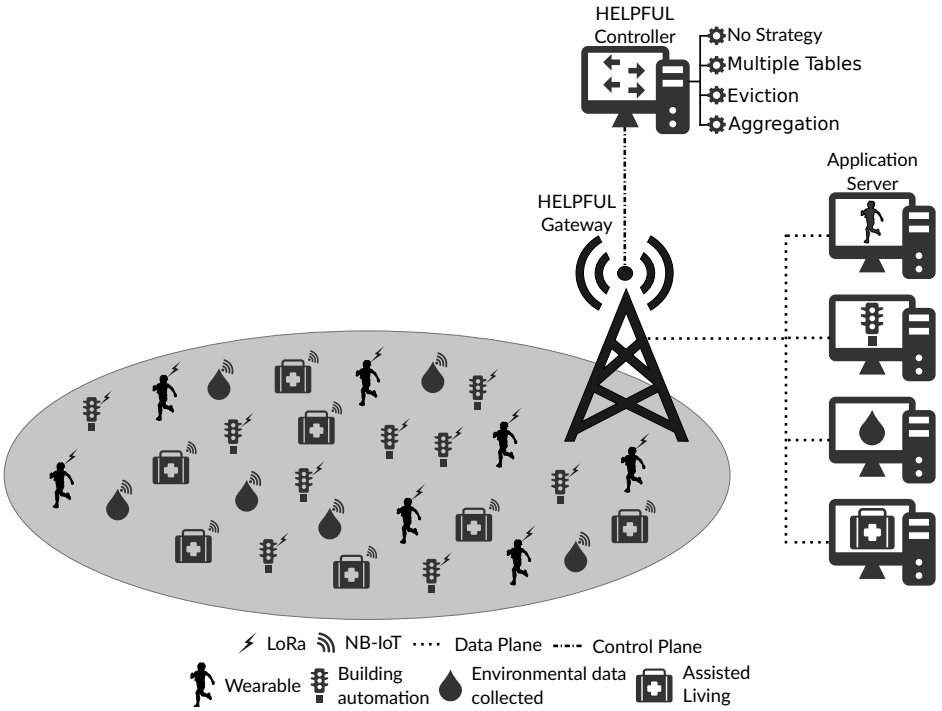


Figure 5.2: Experimentation scenario.

The first goal of the performance analysis is to compare the different rule management strategies. To that end, it is necessary to define the workload of traffic flows in the experimental scenario. Flows are generated based on four application profiles: (*i*) wearable, (*ii*) assisted living, (*iii*) environmental data collected, and (*iv*) building automation. Each application profile defines the total number of messages sent in 24 hours, their size (in bytes), and the technology used by each application, as presented in Table 4. The wearable profile sends ten messages per day with a total size of 20 bytes using LoRa technology. The workload for the assisted living profile encompasses eight messages per day with an overall size of 100 bytes using NB-IoT. The environmental data collection profile sends 24 messages per day of 200 bytes operating on the NB-IoT technology. Finally, the building automation profile sends five messages per day with 50 bytes working on the LoRa technology. These workloads represent actual applications, according to (GSMA, 2016). Each device sends its first message at a random time between 0 and 60 seconds from the beginning of the simulation. It then enters idle mode until the time for the next transmission. The simulation ends once all devices have sent their daily message count. The message transmission interval from each end-node considers a simulation execution time of 1 hour. Finally, the management strategies for flow table rules used in the experiments are Unmanaged, Aggregation, Eviction, and Multiples Tables.

The design for the performance evaluation of each rule management strategy employs six metrics:

Application Profile	Messages per day	Size of message (bytes)	Technology
Wearable	10	20	LoRa
Assisted living	8	100	NB-IoT
Environmental data collected	24	200	NB-IoT
Building automation	5	50	LoRa

Table 5.2: Application Profile Specifications.

- **Number of Rules Installed:** Is the sum of all rules installed on devices by the controller. This metric measures the effectiveness of rule management strategies concerning the storage capacity of flow tables.

- **Number of Controller Interventions:** Considers the sum of messages that insert a new rule in the flow table, the direct packet forwarding messages, and the rule removal messages. This metric defines the overhead that a particular strategy adds to the control channel.

- **Percentage of Table Hits:**

- **Lookup Time:** Considers the time required for the flow table to identify if a matching rule exists for a given packet.

- **Latency:** Is the time a packet needs to reach its destination and return to its source. This metric evaluates the impact of the strategies in the quality of service of LPWAN experimental scenarios.

- **Number of Frames Lost:** Considers the number of frames lost in the communication between the end-nodes and application servers. For each packet sent from an end-node, the application servers are expected to reply with an acknowledgment packet.

Through this metric, we verify how management strategies impact in our LPWAN scenario. All values presented in the next subsection represent central tendencies calculated from a total of 60 runs of each experimental scenario and a confidence level of 95%. All figures show the interval confidence bars (*i.e.*, error bars), but as the variability is very low, these bars are insignificant in some analyzes. This procedure assures the correctness of measurements for the defined metrics. Table 5 summarizes the parameters used in the experimental scenario.

The next Section discusses the results obtained from experiments with HELPFUL architecture. It divides results in two groups. The first is the impact of rule management strategies on the flow table (using the rule quantity installed, controller intervention quantity, and table hit ratio metrics). The second is the performance metrics of the architecture (using lookup time, latency, and packet loss rate). This grouping helps to visualize how rule management strategies impact the performance of the LPWAN scenario as a whole.

Parameter	Values
Rule Management Strategies	Unmanaged, Eviction, Multiple Tables, Aggregation
Number of end-nodes	[300..1500]
Number of application servers	4
Number of switches	1
Number of application profiles	4
Maximum number of rules in the flow tables (1 table)	2500
Maximum number of rules in the flow tables (2 tables)	1250 on each table
Confidence level	95%

Table 5.3: Experimentation parameters.

5.3 Impact of management strategies in flow tables

This Section presents the results from the performance evaluation of the rule management strategies. First, it discusses the number of rules installed in the flow tables, which is the primary metric to measure the performance of management strategies. Next, it presents the results for the Controller intervention, which indicates the overhead on the architecture control channel. Finally, it discusses the results for the table hit ratio, which measures the impact of the lookup process in the flow tables.

Figure 5.3 analyzes the number of rules installed on a BS for each flow table management strategy. The horizontal axis represents the number of end-nodes present in the experiment, which varied from 300 to 1500. In turn, the vertical axis depicts the efficiency of each management strategy based on the number of installed rules. A lower number of installed rules indicates that a strategy has better performance because it will incur lower overhead on devices. Results indicate that there are two distinct groups on the evaluated strategies. The group with a lower number of installed rules include the Multiple Tables and Aggregation strategies. In turn, the group with a higher value of installed rules include Unmanaged and Eviction. The Aggregation strategy presents a 2 times better performance in comparison to the Multiple Tables strategy. Moreover, comparing the Aggregation strategy with both the Unmanaged and Eviction ones indicate a significant performance difference of $\approx 10^2$, as depicted in Figure 5.3.

The observed difference in performance occurs because the Aggregation and Multiple Tables strategies significantly reduce the number of rules installed in the HELPFUL Gateway flow table. These two strategies create generic rules that can represent a higher number of flows in the network, thus reducing the total number of rules. Unmanaged and Eviction strategies cannot represent multiple flows with a single rule and heavily depend on eviction mechanisms to replace rules when tables become full. Therefore, new rules need to be installed in the flow table more frequently, which results in a large number of rules in the flow table.

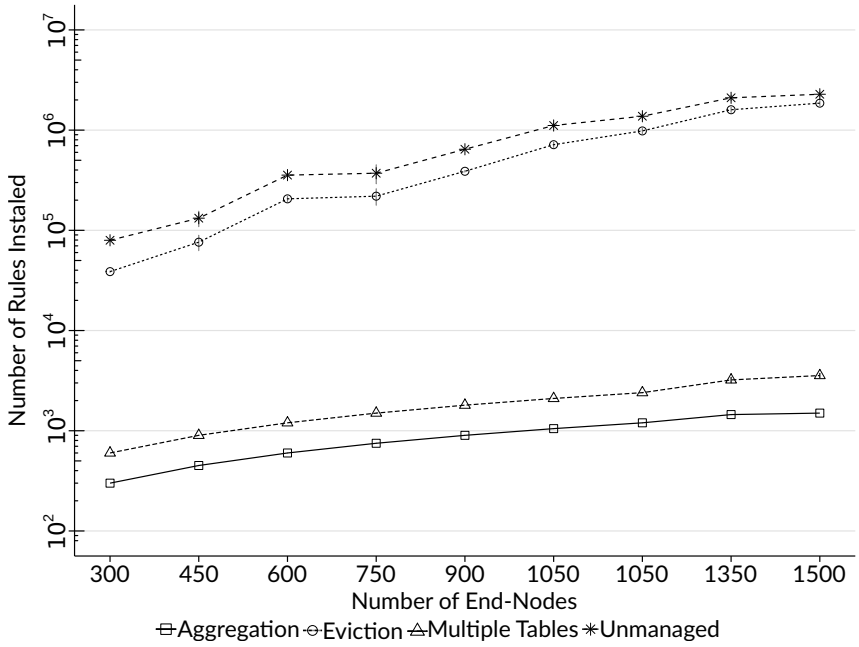


Figure 5.3: Number of rules installed in the flow table.

Figure 5.4 illustrates the results for the Controller intervention metric. The horizontal axis represents the number of end-nodes present in the experiment. In turn, the vertical axis represents the number of Controller interventions used to install or remove rules, or to collect statistics from flow tables. A lower number of Controller interventions represents a better performance of the evaluated strategy. The Multiple Tables strategy, depicted in the lower part of the graph, stands out from the remaining ones. More specifically, the Multiple Tables strategy has a performance $\approx 10^2$ better than the others.

The Multiple Tables strategy presents a better performance because it reduces the overhead in the control channel, differently from other strategies. It enables a rule to represent multiple flows, increasing table hits, and decreasing the exchange of messages between the HELPFUL Gateway and the Controller. For example, the Aggregation strategy regularly needs to query the status of the flow table. After that, the Controller aggregates the rules and replace the flow table in the HELPFUL Gateway. In this step, the Controller needs to remove the rules in the flow table individually and then install the aggregated rules. This rule replacement method increases the exchanged messages between the HELPFUL Gateway and the Controller, considerably. The Eviction strategy has a similar problem. It requires the Controller to check periodically how long a rule is inactive and, if necessary, remove the rule and install a new one. Unmanaged has a behavior where each new packet generates a rule in the flow table since there is no optimization regarding the representation of rules as aggregates or multiple tables. Therefore, the random replacement mechanism generates control messages between the Controller and the HELPFUL Gateway.

The final analysis regarding the impact of strategies for table rule management is the number of table hits, illustrated in Figure 5.5. The horizontal axis depicts the number of end-nodes,

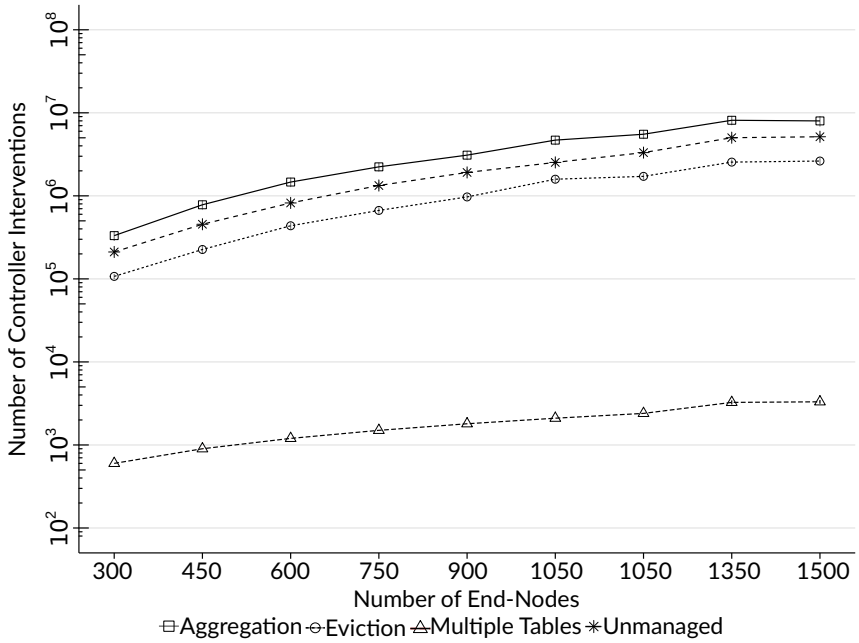


Figure 5.4: Number of controller interventions.

while the vertical axis, the percentage of table hits attained by each strategy. A higher table hit ratio indicates that the strategy has better performance since it will require fewer interventions from the Controller. The results show that the Unmanaged strategy has the worst performance, with a hit ratio varying from 40% to 20%. The Multiple Tables strategy has a constant behavior of up to 1200 end-nodes, with a hit ratio of 70%. Between 1200 and 1500 end-nodes, the strategy presents a performance degradation, with the worse result being a hit ratio of 52%. The Aggregation and Eviction strategies show a behavior similar to that of Multiple Tables, but with a higher table hit-ratio between 300 to 900 end-nodes.

The Aggregation strategy presents the observed behavior because it creates generic rules that represent a higher number of flows, increasing the table hit-ratio. Multiple Tables also results in a higher hit-ratio, but because it transforms a specific rule in two generic ones, maintained in different tables. The Eviction strategy also has a high hit-ratio because it employs a better strategy to manage rules into the available storage space. Finally, the Unmanaged strategy has the worst result because it does not employ any method to optimize the utilization of rule storage space.

5.4 Impact on the network infrastructure

In this section, we present how HELPFUL architecture impacts minimally the network when using Aggregation and Multiple Tables strategies. We decided to focus on these two strategies because they present the best performance in reducing table flow rules, as shown in Subsection 5.2. First, we show the results for the lookup time metric, which defines the search time for a rule that corresponds to a given packet that arrived at the HELPFUL Gateway. Next, we present

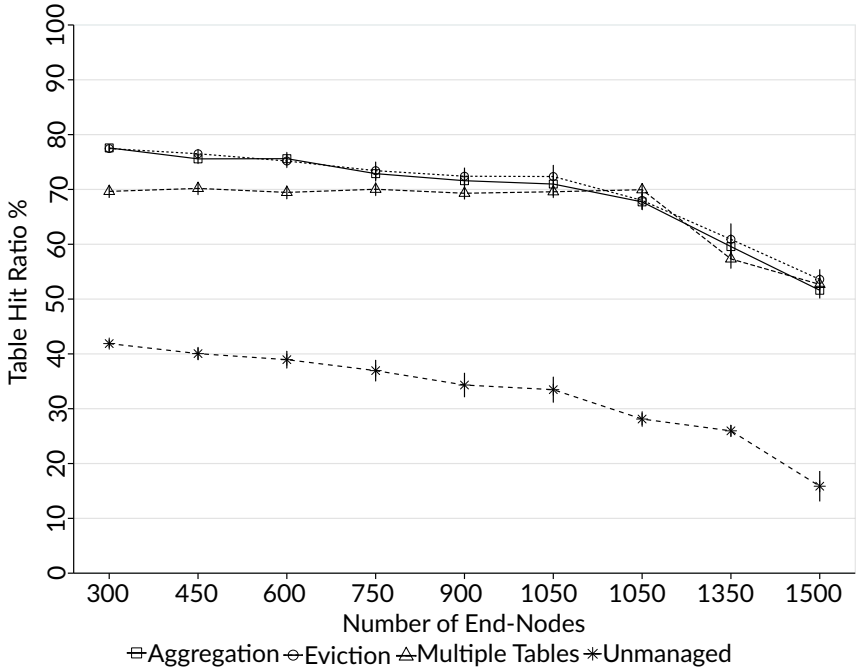


Figure 5.5: Results for table hit ratio.

the results for the latency metric, which indicates how the HELPFUL architecture affects the performance of the network as a whole. Finally, we show the results for the lost packet metric, which demonstrates how the HELPFUL architecture degrades the network minimally.

Figure 5.6 presents the results obtained for the lookup time metric using the HELPFUL architecture. The horizontal axis represents the number of end-nodes in the experiment. In turn, the vertical axis depicts time (in milliseconds) for the HELPFUL Gateway to find a rule which matches a particular packet. In both strategies, it is possible to observe an insignificant degradation in the lookup time performance of the HELPFUL architecture. For the Multiple Tables strategy, when 300 end-nodes access the HELPFUL Gateway, it presents a lookup time of $\approx 0.37ms$. The performance gives a lookup time of $\approx 1.77ms$ when the number of end-nodes increases to 1500. The Aggregation strategy shows less degradation than Multiple Tables and performs better. Even in the worst case (when using 1500 end-nodes), Aggregation reaches $\approx 0.92ms$ of lookup time.

The HELPFUL Gateway implements a sequential lookup to identify which rule corresponds to a given flow. Therefore, the more entries in the flow table, the longer it takes to find a rule. The search process should be repeated for each sub-table when HELPFUL Gateway uses Multiple Tables. In this case, the architecture performance degrades more than $1ms$ approximately. Aggregation performed better since this strategy reduces the total amount of rules in the flow table, and the match process is limited to only one table.

Figure 5.7 illustrates how the HELPFUL architecture impacts the latency of the network. The horizontal axis shows the number of end-nodes used in each iteration. In turn, the vertical axis represents the time (in milliseconds) for the round-trip of a packet between the source and

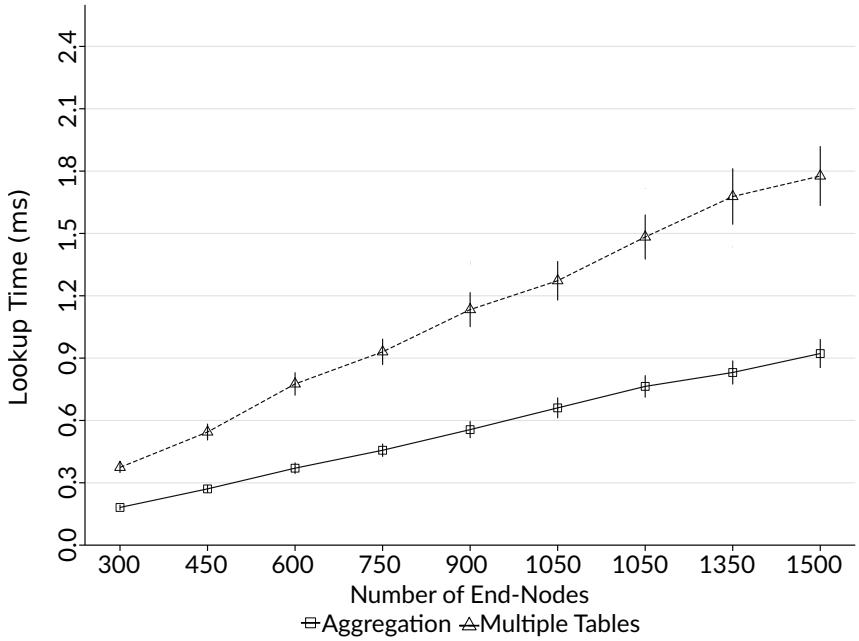


Figure 5.6: Table lookup time.

destination. In this metric, we can observe a behavior similar to what occurs in the lookup time metric. For both strategies used, the HELPFUL architecture tends to increase network latency minimally as the number of nodes increases. For example, the Multiple Tables strategy has a degradation of $\approx 1.73ms$ for the number of end-nodes above 1050, and the worst $\approx 2.43ms$ for 1500 end-nodes. The Aggregation strategy shows the best performance adding only a latency of $\approx 1.63ms$ for 1500 end-nodes.

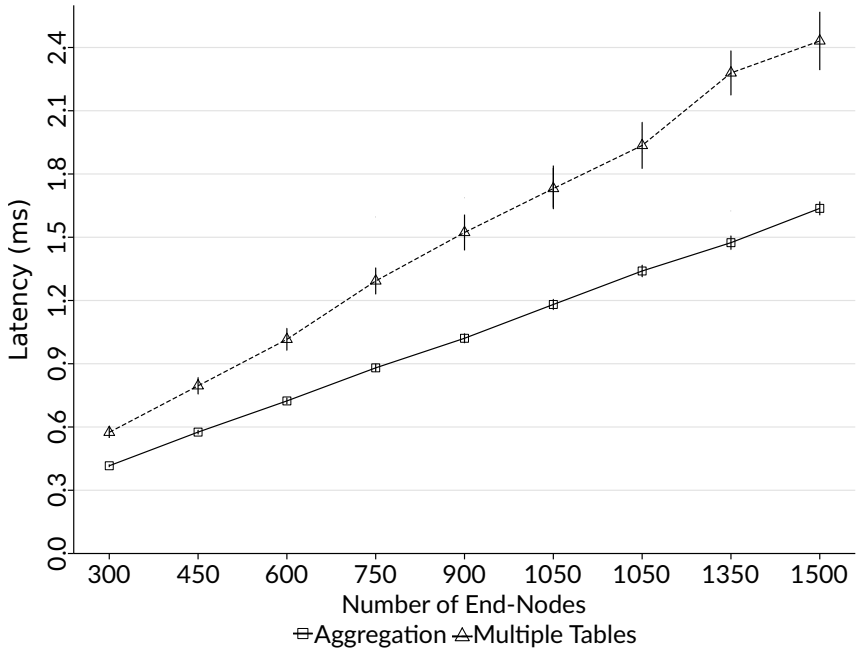


Figure 5.7: Transmission latency.

The Multiple Tables strategy adds a small overhead ($\approx 1ms$) in the HELPFUL Gateway performance when the number of end-nodes increases. This behavior occurs because, at 1050 end-nodes, the usage of the sub-tables reaches their maximum capacity, and replacements must occur. Moreover, the Controller processing time to define and install a new rule generates overhead that increases network latency. In turn, the Aggregation strategy shows the best performance because it can significantly reduce the number of rules in the flow table. Consequently, the control channel overhead due to the rules update does not degrade performance for the latency metric.

Figure 5.8 presents results regarding frames lost during transmission. The horizontal axis depicts the rule management strategies in the flow table. In turn, the vertical axis represents the percentage of lost frames. In this metric, the lower the percentage of dropped frames, the better the network performance. When the HELPFUL architecture uses the Aggregation strategy, it presents an acceptable variation in the number of lost frames averaging 3% for the LPWAN scenario (PETÄJÄJÄRVI et al., 2017). However, the Multiple Tables strategy used in the HELPFUL architecture presents an excellent performance, as a variance averaging less than 0.5%.

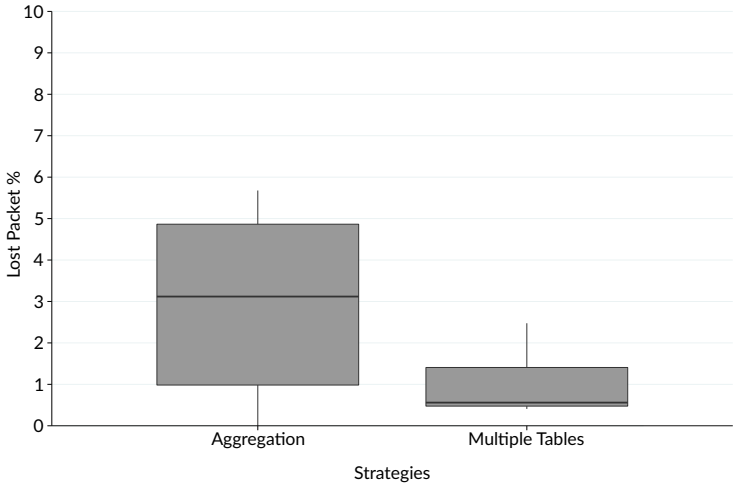


Figure 5.8: Lost packets.

The behavior observed when the HELPFUL architecture uses the Aggregation strategy is due to Controller updates to flow tables. During such an update, the Controller collects the current status and adds or aggregate new rules. During the update process, the Controller needs to remove the unaggregated rules. In this case, the HELPFUL Gateway cannot give out all traffic and can lose some packets. The Multiple Tables strategy has an even lower frame loss because the Controller has to replace rules only when a higher number of end-devices (*e.g.*, more than 1050) communicate. In other scenarios, the Controller interactions are minimal and interfere less in device transmissions.

5.5 Chapter Summary

The presented results demonstrate that HELPFUL architecture enables the usage of multiple LPWAN technologies in heterogeneous networks. Furthermore, the architecture adds minimal overhead to network performance, regardless of the rule management strategy chosen. Moreover, HELPFUL is flexible enough to change the management strategy to the best fit with the network demands. For example, the architecture can switch the strategy in a scenario where a sudden high demand for flow table rules occurs, *i.e.*, multiple application profiles. In such a case, HELPFUL can adapt the Controller and Gateway to use Aggregation as the management strategy, decreasing the number of installed rules. However, in a scenario with control channel degradation, a reduction in the number of messages exchanged between the Controller and Gateway is necessary. HELPFUL can adapt the Controller and Gateway to operate using Multiple Tables strategies, consequently reducing the number of messages on the control channel.

6 CONCLUSION AND FUTURE WORK

The coexistence of heterogeneous LPWAN technologies is desired to exploit the best features each technology has to offer, addressing multiple communication requirements independently. On the other hand, deploying and operating several independent heterogeneous networks, each supporting a limited number of IoT devices, becomes challenging. Firstly, current LPWAN technologies have not been designed to coexist with each other, as each LPWAN implements individual control protocols leading to uncoordinated wireless access. Secondly, the assumption of multiple wireless access technologies in the traditional architecture of the cellular network would lead to prohibitive deployment costs of transmitters cells.

This thesis proposes HELPFUL, *a flexible architecture to control heterogeneous low power wide-area networks*. It enables various LPWAN technologies to coexist with scalability guarantees. More specifically, HELPFUL is an SDN-based architecture that creates a common abstraction in the control plane to allow a heterogeneous LPWAN. It also employs four rule management strategies to work with single or multiple tables to address the scalability issues present in the SDN-based devices. We implemented a prototype of our architecture using the P4 language and the Mininet environment. This prototype enabled the evaluation of the management strategies considering the total number of rules, controller interventions, and table hit ratio. We also conduct experiments to analyze the impact of the overhead that HELPFUL adds to the network using the lookup time, latency, and lost frames.

6.1 Summary of Contributions

The main contributions of our research are the following:

- Results show that the HELPFUL architecture can unify the technology-specific control into a single cross-technology controller. The architecture is flexible and allows the network administrator to add new tables in the Gateway. It only requires the administrator to describe the table using metadata functions of the P4 language.

- The architecture adds insignificant overhead to the network. The Multiple Tables management strategy is effective in reducing the number of installed rules. It also reduces the

number of messages exchanged between the Controller and Gateway and has a satisfactory table hit ratio.

- The architecture also allows the use of various management strategies or a combination of any of then. For example, it is possible to choose which management strategy HELPFUL must use according to the current status of the network.

6.2 Final Remarks and Future Work

As part of future work, we intend to expand our work to use other technologies (*e.g.*, flow tables implemented in hardware) and explore new forms of management. For that, our future research efforts are the following:

- Include more technologies LPWAN in the experimentation scenario.
- Create a method to translate new protocols in a dynamic way.
- Include experiments on real equipment that uses flow tables implemented with TCAM, *e.g.*, using a white box switch.
- Evaluate additional rule management strategies (*i.e.*, rule positioning) or a combination of then (*i.e.*, Multiple Table plus Aggregation).
- Investigate mechanisms that can improve the performance of the studied strategies by offering performance guarantees to the network.

REFERENCES

ALI, A. et al. Technologies and challenges in developing machine-to-machine applications: a survey. **Journal of Network and Computer Applications**, v. 83, n. C, p. 124–139, 2017.

BADDELEY, M. et al. Evolving SDN for Low-Power IoT Networks. In: **4th IEEE Conference on Network Softwarization and Workshops (NetSoft)**. [S.l.: s.n.], 2018. p. 71–79.

BERA, S. et al. Soft-wsn: Software-defined wsn management system for iot applications. **IEEE Systems Journal**, v. 12, n. 3, p. 2074–2081, 2018.

BOSSHART, P. et al. P4: Programming Protocol-Independent Packet Processors. **ACM SIGCOMM Computer Communication Review**, v. 44, n. 3, jul 2014.

BRUNS, R. et al. Intelligent M2M: Complex event processing for machine-to-machine communication. **Expert Systems with Applications**, v. 42, p. 1235–1246, 2015.

CHALLA, R.; LEE, Y.; CHOO, H. Intelligent eviction strategy for efficient flow table management in OpenFlow Switches. In: **IEEE NetSoft Conference and Workshops**. [S.l.: s.n.], 2016. p. 312–318.

CURTIS, A. R. et al. Devoflow: Scaling flow management for high-performance networks. In: **Proceedings of the ACM SIGCOMM 2011 Conference**. [S.l.: s.n.], 2011. p. 254–265.

GALLO, P. et al. Sdn@home: A method for controlling future wireless home networks. **IEEE Communications Magazine**, v. 54, n. 5, p. 123–131, 2016.

GSMA. **Low Power Wide Area Technologies**. [S.l.], 2016.

KANNAN, K.; BANERJEE, S. Flowmaster: Early eviction of dead flow on sdn switches. In: **Distributed Computing and Networking**. [S.l.: s.n.], 2014. p. 484–498.

KATTA, N. et al. Cacheflow: Dependency-aware rule-caching for software-defined networks. In: **Proceedings of the Symposium on SDN Research**. [S.l.: s.n.], 2016. p. 1–12.

KIM, E. et al. A flow entry management scheme for reducing controller overhead. In: **International Conference on Advanced Communication Technology**. [S.l.: s.n.], 2014. p. 754–757.

KIST, M. et al. SDR Virtualization in Future Mobile Networks: Enabling Multi-Programmable Air-Interfaces. In: **2018 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2018. p. 1–6.

KOBAYASHI, M. et al. Maturing of openflow and software-defined networking through deployments. **Computer Networks**, v. 61, p. 151–175, 2014.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, p. 14–76, 2015.

LEE, B.; KANAGAVELU, R.; AUNG, K. An efficient flow cache algorithm with improved fairness in software-defined data center networks. In: **IEEE 2nd International Conference on Cloud Networking (CloudNet)**. [S.l.: s.n.], 2013. p. 18–24.

LUO, T.; TAN, H.-P.; QUEK, T. Sensor openflow: Enabling software-defined wireless sensor networks. **IEEE Communications Letters**, v. 16, n. 11, p. 1896–1899, 2012.

MARSICO, A.; DORIGUZZI-CORIN, R.; SIRACUSA, D. An effective swapping mechanism to overcome the memory limitation of sdn devices. In: **IFIP/IEEE Symposium on Integrated Network and Service Management (IM)**. [S.l.: s.n.], 2017. p. 247–254.

MINOLI, D.; SOHRABY, K.; OCCHIOGROSSO, B. Iot considerations, requirements, and architectures for smart buildings-energy optimization and next-generation building management systems. **IEEE Internet of Things Journal**, v. 4, p. 269–283, 2017.

NAKAGAWA, Y. et al. Domainflow: Practical flow management method using multiple flow tables in commodity switches. In: **Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies**. [S.l.: s.n.], 2013. p. 399–404.

NGUYEN, X. et al. Rules placement problem in openflow networks: A survey. **IEEE Communications Surveys & Tutorials**, v. 18, n. 2, p. 1273–1286, 2016.

ONF. **The BenefFlow Tables and TTPs**. 2015. Open Networking Foundation. Available in: $https://www.opennetworking.org$, Accessed 8 jul. 2019.

PETÄJÄJÄRVI, J. et al. Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage. **International Journal of Distributed Sensor Networks**, v. 13, n. 3, p. 49–60, 2017.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Elsevier Information and Software Technology**, v. 64, p. 1–18, 2015.

POORTER, E. D. et al. Sub-GHz LPWAN Network Coexistence, Management and Virtualization: An Overview and Open Research Challenges. **Wireless Personal Communications**, v. 95, n. 1, p. 187–213, 2017.

RAZA, U.; KULKARNI, P.; SOORIYABANDARA, M. Low power wide area networks: A survey. **IEEE Communications Surveys Tutorials**, v. 19, n. 2, p. 855–873, 2017.

RIFAI, M. et al. Too many SDN rules? compress them with MINNIE. In: **IEEE Global Communications Conference**. [S.l.: s.n.], 2015. p. 0–6.

STEPHENS, B. et al. Past: Scalable ethernet for data centers. In: **International Conference on Emerging Networking Experiments and Technologies**. [S.l.: s.n.], 2012. p. 49–60.

WANG, Y. E. et al. A Primer on 3GPP Narrowband Internet of Things. **IEEE Communications Magazine**, v. 55, n. 3, p. 117–123, 2017.

WICKBOLDT, J. et al. Software-define networking: Management requirements and challenges. **IEEE Communications Magazine**, v. 53, n. 1, jan 2015.

XIONG, X. et al. Low power wide area machine-to-machine networks: Key techniques and prototype. **IEEE Communications Magazine**, v. 53, n. 9, p. 64–71, 2015.

ZAYAS, A.; MERINO, P. The 3gpp nb-iot system architecture for the internet of things. In: **2017 IEEE International Conference on Communications Workshops**. [S.l.: s.n.], 2017. p. 277–282.

AppendixA RESUMO

A próxima geração de redes sem fio está sendo desenvolvida para prover conectividade para múltiplas aplicações de Internet das coisas (IoT), como casas inteligentes, industria 4.0 e vestíveis. Tais aplicações de IoT ultrapassarão os limites das arquiteturas de comunicação atuais, demandando coexistência entre diferentes tecnologias, *e.g.*, utilizando tecnologias Low Power Wide Area Networks (LPWAN). A coexistência de infraestruturas LPWAN heterogenias habilitam a utilização das melhores funcionalidades de cada tecnologias, atendendo a múltiplos requisitos de comunicação, como por exemplo, comunicação massiva de máquina para máquina e comunicação de baixa latência. A implantação e operação de várias redes heterogenias, cada uma suportando uma quantidade limitada de dispositivos IoT, evidencia desafios de pesquisa que ainda precisam de investigação(RAZA; KULKARNI; SOORIYABANDARA, 2017). Primeiramente, as atuais tecnologias LPWAN não foram desenvolvidas para coexistirem entre si, uma vez que cada tecnologia LPWAN implementa seus próprios protocolos de controle, levando a acesso sem fio descoordenado (POORTER et al., 2017). Além disso, múltiplas tecnologias de acesso sem fio na arquitetura tradicional de redes celulares levaria a custos proibitivos de implantação de células transmissoras (POORTER et al., 2017).

Com o objetivo de lidar com esses desafios, a virtualização das estações base (BS) foi proposta como uma solução para reduzir as despesas de capital e as despesas operacionais (CAPEX / OPEX) das redes celulares por meio de compartilhamento de infraestrutura (KIST et al., 2018). Entretanto, virtualização adiciona a necessidade de novos protocolos de controle para a infraestrutura gerenciar o acesso descoordenado de múltiplas tecnologias LPWAN. As Rede Definidas por Software (SDN) fornecem uma abordagem poderosa, criando uma arquitetura programável, dinâmica e flexível que permite a abstração das implementações tradicionais de protocolo baseado em hardware em um controlador de rede baseado em software (KREUTZ et al., 2015). Nesse sentido, alguns estudos aplicaram o paradigma SDN para fornecer uma rede programável para aplicações de IoT (LUO; TAN; QUEK, 2012; GALLO et al., 2016; BADDELEY et al., 2018; BERA et al., 2018). No entanto, tais estudos não levam em consideração a capacidade limitada dos dispositivos de rede baseados em SDN para armazenar as regras de encaminhamento. Por exemplo, as tabelas têm uma capacidade limitada de armazenamento de regras, que é várias ordens de grandesa menores que a necessária para a operação de certos tipos

de rede, incluindo LPWAN (NGUYEN et al., 2016). Esse problema se agrava quando aplicado em ambientes LPWAN em que uma grande quantidade de nós finais em uma rede (*eg*, $\approx 10^6$ dispositivos por $k2$ (RAZA; KULKARNI; SOORIYABANDARA, 2017)) , tornando necessário gerenciar as regras das tabelas de fluxo.

Nesta tese, propomos HELPFUL: *a flexible architecture to control heterogeneous low power wide area networks* para permitir que várias tecnologias LPWAN coexistam com garantias de escalabilidade. HELPFUL é uma arquitetura baseada em SDN que cria uma abstração de controle comum entre as tecnologias LPWAN em execução nas BSs virtualizadas. As principais contribuições de nosso trabalho são duas: (*i*) a proposta de uma arquitetura que adiciona uma camada de abstração que traduz os protocolos de controle heterogêneos em um conjunto homogêneo de mensagens e (*ii*) a estratégia baseada no gerenciamento múltiplas tabelas para endereçar os problemas de escalabilidade presentes nos dispositivos baseados em SDN. Nossa proposta permite a unificação do controle específico da tecnologia em um único controlador entre tecnologias, permitindo a harmonização do acesso à infraestrutura e ao espectro.

Desenvolvemos um protótipo baseado na linguagem P4 que permite a programação dos planos de controle e encaminhamento para validar HELPFUL. Em nossa primeira avaliação, enumeramos uma série de estratégias para gerenciar regras da tabela de fluxo ao usar a arquitetura HELPFUL. Definimos as métricas para analisar o HELPFUL com cada estratégia, como o número de regras instaladas na tabela de fluxo, o número de intervenções do controlador e o número de ocorrências da tabela. Esses resultados indicam as melhores estratégias em relação à redução do número de regras instaladas nas tabelas de fluxo. Nossa segunda avaliação considera a sobrecarga da arquitetura HELPFUL na rede, incluindo tempo de pesquisa de tabela, latência e quadros perdidos. Os resultados mostram que HELPFUL produz um impacto mínimo na rede. Por exemplo, HELPFUL apresentou $\approx 0.92ms$ de tempo de pesquisa em um cenário analisado. Além disso, a arquitetura HELPFUL adicionou latência mínima de rede de $\approx 2ms$ e o número de quadros perdidos em média 1,5%. Além disso, o HELPFUL é flexível o suficiente para alterar a estratégia de gerenciamento da melhor forma possível para atender às demandas da rede. A arquitetura pode mudar a estratégia em um cenário em que ocorre uma alta demanda repentina de regras da tabela de fluxo, *i.e.*, vários perfis de aplicação. Nesse caso, HELPFUL pode adaptar o Controlador e os dispositivos de encaminhamento para usar a Agregação como estratégia de gerenciamento, diminuindo o número de regras instaladas. No entanto, em um cenário com degradação do canal de controle, é necessária uma redução no número de mensagens trocadas entre o Controlador e o dispositivo de encaminhamento. HELPFUL pode se adaptar para operar usando estratégias de Tabelas Múltiplas, consequentemente reduzindo o número de mensagens no canal de controle.

AppendixB PUBLISHED PAPER – SBRC 2017

Gustavo de Araújo, Marcelo Marotta, Juliano Wickboldt, Cristiano Both, Luciano Gaspary, Juergen Rochol, Lisandro Granville. **Caracterizando Estratégias de Domínio Espacial para Gerenciamento de Regras em Redes Definidas por Software**. 2017 SBRC 2017 Brazilian Symposium on Computer Networks and Distributed Systems, Brazil, Belem-Pará, 2017.

- **Title:** *Caracterizando Estratégias de Domínio Espacial para Gerenciamento de Regras em Redes Definidas por Software*.
- **Contribution:** An evaluate of the effectiveness and overhead of the spatial domain strategies for management rules in the flow table in the SDN forwarding devices.
- **Abstract:** Software-Defined Networks (SDN) based on Openflow protocol perform data forwarding through flow tables using match/action mechanisms. These tables have limited rules storage capacity, restricting network scalability and performance. Considering these restrictions, the main strategies for managing spatial domain rules, i.e., flow agregation and multiple flow tables, are promising to use the available storage space of the tables. These strategies impact packet processing in different ways, influencing the performance of the forwarding devices and the network. In spite of the impact, the comparison between flow aggregation and multiple flow tables is poorly exploited, leaving open the definition of which strategy is more appropriate for a network considering its topology and workload. In this article a quantitative characterization is proposed, defining the gains brought by these spatial domain strategies for forwarding devices and in different topologies.
- **Status:** Published.
- **Qualis:** B2.
- **Conference:** Brazilian Symposium on Computer Networks and Distributed Systems (SBRC).
- **Date:** May 15 - May 19, 2017.

- **Local:** Belém, Pará.
- **URL:** <http://sbrc2017.ufpa.br/en/>.

Caracterizando Estratégias de Domínio Espacial para Gerenciamento de Regras em Redes Definidas por Software

Gustavo de Araújo[1], Marcelo Marotta[1], Juliano Wickboldt[1], Cristiano Both[2], Luciano Gaspary[1], Juergen Rochol[1], Lisandro Granville[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul

[2]Departamento de Matemática Aplicada e Ciências Sociais
Universidade Federal das Ciências da Saúde de Porto Alegre

{gustavo.araujo, mamarotta, jwickboldt}@inf.ufrgs.br

cbboth@ufcspa.edu.br, {paschoal, juergen, granville}@inf.ufrgs.br

Abstract. *Software-Defined Networks (SDN) based on Openflow protocol perform data forwarding through flow tables using match/action mechanisms. These tables have limited rules storage capacity, restricting network scalability and performance. Considering these restrictions, the main strategies for managing spatial domain rules, i.e., flow agregation and multiple flow tables, are promising to use the available storage space of the tables. These strategies impact packet processing in different ways, influencing the performance of the forwarding devices and the network. In spite of the impact, the comparison between flow aggregation and multiple flow tables is poorly exploited, leaving open the definition of which strategy is more appropriate for a network considering its topology and workload. In this article a quantitative characterization is proposed, defining the gains brought by these spatial domain strategies for forwarding devices and in different topologies.*

Resumo. *As Redes Definidas por Software (SDN) baseadas no protocolo Openflow realizam o encaminhamento de dados por meio de tabelas de fluxos utilizando mecanismos de match/action. Tais tabelas possuem uma capacidade de armazenamento de regras limitada, restringindo a escalabilidade e o desempenho da rede. Considerando essas restrições, as principais estratégias de gerenciamento de regras de domínio espacial, i.e., agregação de regras e múltiplas tabelas, mostram-se promissoras na utilização do espaço de armazenamento disponível das tabelas. Essas estratégias impactam de diferentes maneiras no processamento dos pacotes influenciando o desempenho dos dispositivos de encaminhamento e, de forma mais geral, da rede. Apesar do impacto gerado, a comparação entre as estratégias de agregação de fluxos e múltiplas tabelas é fracamente explorada, deixando em aberto a definição de qual estratégia é mais adequada para uma rede, considerando sua topologia e carga de trabalho. Neste artigo realiza-se uma caracterização quantitativa definindo os ganhos trazidos por essas estratégias de domínio espacial para dispositivos de encaminhamento e em diferentes topologias.*

1. Introdução

Os conceitos inerentes às Redes Definidas por Software (SDN) mostram-se fundamentais na evolução das redes de computadores e continuam sendo investigados tanto pela industria quanto pela academia [Wickboldt et al. 2015]. Entre os principais conceitos destacam-se: o encaminhamento de tráfego baseado em fluxos e a abstração da lógica de controle para uma entidade em software, chamada controlador. Através desses conceitos, um dispositivo de encaminhamento torna-se capaz de desempenhar diferentes funções, por exemplo, controlar o acesso a um servidor de aplicação ou balancear carga entre diferentes enlaces [Kreutz et al. 2015]. As funções de rede desempenhadas pelos dispositivos são definidas a partir de regras de encaminhamento que são armazenadas em tabelas de fluxos. Tais tabelas possuem uma capacidade de armazenamento de regras limitada, a qual é várias ordens de grandeza menor que o necessário para a operação de determinados tipos de rede como, por exemplo, *backbones* e redes de *datacenters* [Nguyen et al. 2016]. Portanto, o adequado gerenciamento de regras se torna um requisito para a escalabilidade das redes SDN.

Levando em consideração a escalabilidade das redes SDN, estratégias para gerenciamento de regras de encaminhamento foram propostas na literatura. Tais estratégias podem ser classificadas em dois grupos: (*i*) domínio temporal e (*ii*) domínio espacial. As estratégias de domínio temporal objetivam a deleção de regras da tabela de fluxos a partir de uma determinada política (*e.g., Least Recently Used* e *timeouts* dinâmicos e estáticos) [Neves et al. 2016]. Para compor o estado dos fluxos e realizar a deleção, o controlador precisa consultar os dispositivos de encaminhamento de tempos em tempos (*polling*), a fim de obter em tempo real, as informações de utilização de cada fluxo (*e.g.,* duração e número de pacotes recebidos). Esse *polling* insere uma sobrecarga significativa no tráfego do canal de controle, no processamento do controlador e nos dispositivos de encaminhamento. A sobrecarga gerada torna as estratégias de domínio temporal não escaláveis, mesmo para pequenas redes de *datacenters* [Vishnoi et al. 2014].

As estratégias de domínio espacial, por sua vez, objetivam a representação da maior quantidade de fluxos com o menor número de regras de encaminhamento na rede. Esta representação é processada pelo controlador considerando diferentes abordagens de gerenciamento, das quais destacam-se, (*a*) agregação de fluxos e (*b*) múltiplas tabelas. O gerenciamento por agregação de fluxos permite a representação de dois ou mais fluxo utilizando uma única regra, sem alterar a semântica de encaminhamento [Kamiyama et al. 2014]. Já, o gerenciamento por múltiplas tabelas permite que um grupo de fluxos tenha sua semântica dividida, sendo representada em poucas regras pertencentes a duas ou mais tabelas de acordo com as políticas empregadas (*e.g.,* uma tabela para encaminhamento de portas e outra para controle de acesso) [ONF 2015b]. As estrategias de domínio espacial, mostram-se promissoras na utilização do espaço de armazenamento disponível nas tabelas de fluxos dos dispositivos de encaminhamento, pois necessitam de um monitoramento menor ou, em alguns casos, nenhum, quando comparadas as estratégias de domínio temporal.

O gerenciamento por agregação e múltiplas tabelas possuem seus próprios *modus operandi*, impactando no desempenho de cada dispositivo de encaminhamento e, de forma geral, da rede. Esse impacto pode ser mensurado a partir das métricas de desempenho, como o número de regras de encaminhamento, a quantidade de intervenções do controla-

dor, a latência e o *jitter* da rede. Entretanto, existem questionamentos sobre o impacto no emprego de cada uma dessas formas de gerenciamento, o que agrava-se para redes com diferentes topologias, cargas de trabalho e políticas empregadas. Por exemplo, gerenciamento por agregação de fluxos precisa adicionar e remover regras com maior frequência comparada ao gerenciamento por múltiplas tabelas, aumentando latência e *jitter*. Já, o gerenciamento por múltiplas tabelas precisa percorrer um número maior de tabelas na definição da ação a ser tomada comparada ao gerenciamento por agregação, também incorrendo latência ou *jitter*. Como pode ser visto, ambas as formas de gerenciamento apresentam impactos diferentes para as mesmas métricas. Logo, uma comparação entre gerenciamento por agregação e por múltiplas tabelas torna-se fundamental na definição de qual delas é a mais adequada para ser empregada em uma rede. No melhor conhecimento, não existe um trabalho comparativo que investigue ambas as formas de gerenciamento espacial, permanecendo uma questão de pesquisa em aberto.

Neste artigo é proposto uma comparação quantitativa entre os gerenciamentos por agregação e múltiplas tabelas em três diferentes topologias de rede. Primeiramente, busca-se analisar cada dispositivo individualmente, verificando a redução na quantidade de regras que cada forma de gerenciamento de domínio espacial obtêm. Em seguida, é analisada a quantidade de intervenções que o controlador realiza para cada forma de gerenciamento. Para uma comparação completa das formas de gerenciamento, utiliza-se como linha base a operação padrão para encaminhamento de camada 2 em SDN baseada em OpenFlow. Além disso, para avaliar a influência que o gerenciamento traz para diferentes redes, utilizou-se um ambiente de redes de topologia variadas: único salto, anel e árvore. Os resultados mostram que um gerenciamento por agregação de fluxos pode reduzir drasticamente (aproximadamente 95%) a quantidade regras de encaminhamento independente da topologia de rede. Entretanto, a quantidade de intervenções com o controlador mantém-se extremamente alta (acima de 370%), bem como o *jitter*, onde o mesmo apresenta um valor duas vezes superior a linha base, para uma topologia de anel. O gerenciamento por múltiplas tabelas possui o melhor custo-benefício, levando-se em consideração a quantidade de regras instaladas e a quantidade de intervenções necessárias do controlador, apresentando baixa latência e *jitter*, em qualquer topologia de rede.

O restante do artigo está organizado da seguinte maneira. A Seção 2 discute os trabalhos relacionados. A Seção 3 explora e exemplifica as estratégias de gerência de domínio espacial. A Seção 4, apresenta os resultados obtidos. Finalmente, a Seção 5 expõe as principais conclusões do estudo e perspectivas de trabalhos futuros.

2. *Background* e Trabalhos Relacionados

A limitada capacidade de armazenamento dos dispositivos de encaminhamento das SDNs motivou o avanço na área de gerenciamento de regras das tabelas de fluxos. Nessa área, destacam-se as diferentes formas de gerenciamento sendo classificadas em dois grupos: aquelas pertencentes ao domínio temporal e espacial. Desta forma, na Subseção 2.1, descreve-se o gerenciamento de domínio temporal e suas limitações. Já, na Subseção 2.2, as diferentes formas de se utilizar o gerenciamento de domínio espacial são apresentadas.

2.1. Gerenciamento de Domínio Temporal

O gerenciamento de domínio temporal consiste em remover uma regra da tabela de fluxos, após um determinado período de tempo. Essa remoção de regras pode ser realizada

por algoritmos como *Least Recently Used* (LRU), *First In First Out* (FIFO) ou remoção randômica. A escolha do algoritmo de remoção é determinante para o desempenho da rede [Lee et al. 2013]. Por exemplo, a utilização de um algoritmo FIFO removerá as regras que foram instaladas a mais tempo na tabela de fluxos. Esta remoção pode ser realizada de maneira pro-ativa, com o próprio dispositivo de encaminhamento removendo as regras, depois de um período determinado de tempo (*hard timeout*), ou de um período de inatividade do fluxo (*idle timeout*). Entretanto, os fluxos possuem durações variadas que podem tanto ser instantâneas ou maiores que o *hard timeout*, levando a remoção tardia ou indevida das regras. Essas remoções implicam na utilização ineficiente da capacidade disponível de armazenamento [Benson et al. 2010].

Um estudo caracteriza e compara diferentes propostas de gerenciamento de regras de domínio temporal [Neves et al. 2016]. Nessa comparação, as formas de gerenciamento temporal, *i.e., idle timeouts* incrementais adaptativos, remoção probabilística de regras, *hard timeouts* adaptativos, apresentaram, no melhor dos casos, uma utilização da capacidade de armazenamento de regras 15% maior que o caso ótimo e degradando-se (aumentando) gradualmente para os demais casos. Adicionalmente, foi constatado que pequenas mudanças nas características do tráfego (*e.g.,* tempo de duração dos fluxos) afetam consideravelmente o desempenho do gerenciamento de domínio temporal. Em síntese, o gerenciamento de domínio temporal mostra-se não escalável às diferentes redes e resiliente às mudanças nas características das mesmas. Nas redes onde o gerenciamento de domínio temporal mostra-se inadequado, o gerenciamento de domínio espacial pode ser uma potencial solução, sendo o tema deste trabalho e da próxima subseção.

2.2. Gerenciamento de Domínio Espacial

As formas de gerenciamento de domínio espacial podem ser classificadas em dois grupos: (*i*) *inter-switch* e (*ii*) *intra-switch*. No primeiro grupo, o gerenciamento é considerado *inter-switch*, pois as regras são gerenciadas considerando o encaminhamento e a semântica dos fluxos dentro de um conjunto de dispositivos da rede. No segundo grupo, o gerenciamento é considerado *intra-switch*, pois as regras são gerenciadas considerando apenas um dispositivo de encaminhamento e os fluxos que passam por ele.

O gerenciamento *inter-switch*, também denominado como posicionamento de regras, consiste em dividir um conjunto de regras de encaminhamento e distribuí-los entre os dispositivos da rede. Esse tipo de gerenciamento espacial é normalmente modelado como um problema de otimização que deve decidir quais regras devem ser instaladas em cada dispositivo. A função objetivo de otimização depende da aplicação que pretende-se implementar, por exemplo, minimizar a quantidade total de regras instaladas na rede [Kanizo et al. 2013] ou minimizar a energia consumida [Giroire et al. 2014]. Esse gerenciamento pode ocasionar duplicação de regras e sobrecarga no processamento para execução do algoritmo de otimização [Nguyen et al. 2016]. Além disso, este gerenciamento precisa de um monitoramento de fluxos frequente, levando aos mesmos contrapontos do domínio temporal. Assim, o gerenciamento *inter-switch* não é foco deste trabalho.

As principais formas de gerenciamento de domínio espacial *intra-switch* são: (*i*) agregação de fluxos e (*ii*) múltiplas tabelas. Na primeira, o gerenciamento por agregação de fluxos opera a partir da identificação das regras que possuem a mesma ação (*e.g.,* encaminhar pacote para uma determinada porta). Em seguida, identifica-se os campos de correspondência (*match*) das regras que possuem similaridades (*e.g.,* IPs de origem que

pertencem a mesma subrede), para serem agregadas sob uma nova regra única (*e.g.,* todos os IPs origem de uma subrede serão encaminhados para uma mesma porta). Desta forma, uma regra pode representar diversos fluxos, reduzindo, assim, a quantidade de regras da tabela. Agregação de fluxos é uma estratégia tradicional para reduzir a quantidade de regras em tabelas de roteamento das atuais redes IP e motivado pelo bom desempenho obtido, veem sendo aplicada em redes SDN/Openflow [Nguyen et al. 2016].

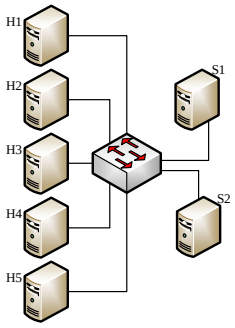
Na segunda forma, o gerenciamento por múltiplas tabelas opera a partir da identificação da semântica de um fluxo (*e.g.,* fluxo com destino a uma interface virtual a ser encaminhado por uma determinada porta), processando sua divisão para duas ou mais regras. A partir dessa divisão, uma nova regra é criada com uma semântica simplificada e armazenada em uma tabela que a represente. Dessa forma, novos fluxos com semânticas compostas são representados por regras já instaladas com semânticas mais simples, utilizando um número menor de regras armazenadas no dispositivo. Gerenciamento por múltiplas tabelas é previsto desde a versão 1.1 do protocolo OpenFlow, sendo a atual forma de gerenciamento incentivada pela *Open Network Foundation* (ONF) [ONF 2015b]. Sua utilização adiciona mais complexidade ao processo de busca na tabela de fluxos (*lookup*) para dispositivos OpenFlow que implementam tabelas de fluxos em hardware. Gerenciamento por múltiplas tabelas é pouco explorada pela literatura, embora apresente grande potencial para reduzir a quantidade de regras utilizadas.

As diferentes formas de gerenciamento espacial *intra-switch*, mostram-se promissoras, por não possuírem a necessidade de um frequente monitoramento do estado atual das tabelas de fluxos. Entretanto, a comparação entre essas formas de gerenciamento é fracamente explorada na literatura, impossibilitando a definição de qual forma de gerenciamento *intra-switch* é a mais adequada para uma determinada rede. Na próxima seção, explora-se como cada uma das formas de gerenciamento de domínio espacial *intra-switch* podem ser aplicadas, para posteriormente serem comparadas em uma rede OpenFlow.

3. Explorando as estratégias de domínio espacial

As formas de gerenciamento de domínio espacial *intra-switch* podem ser comparadas quando aplicadas a uma rede baseada em OpenFlow, onde políticas definem a semântica do encaminhamento de fluxos para a correta operação de uma aplicação de rede. Dessa forma, utiliza-se como exemplo, a implementação de políticas de controle de acesso a servidores de aplicação.

Regras de controle de acesso definem quais fluxos são autorizados a acessar determinados serviços ou nodos da rede. Essas regras são implementadas associando endereços IP aos serviços disponíveis. Idealmente, todas as regras que implementam controle de acesso devem estar presentes no último salto antes do serviço que pretende-se acessar. Desta maneira, evita-se o processamento de regras desnecessárias em dispositivos que possuem a função exclusiva de encaminhamento de pacotes. Entretanto, com a limitação de memória existente em dispositivos de encaminhamento baseado em OpenFlow, posicionar todas as regras em um único dispositivo é impraticável para redes com muitos usuários [Nguyen et al. 2016]. Além disso, conforme a quantidade de regras na tabela de fluxos aumenta, mais processamento é exigido por parte do dispositivo de encaminhamento para realizar o *lookup*. Esse processamento afeta negativamente o desempenho da rede fazendo com que a latência e o *jitter* aumentem. Portanto, o gerenciamento de regras, principalmente, de domínio espacial *intra-switch* torna-se uma exigência.



(a) LAN com cinco clientes e dois servidores de aplicação conectados por um *switch*

Tabela única		
#	Regra	Ação
R1	Se ip.src=H1 e ip.dst=S1	Encaminha na porta P6
R2	Se ip.src=H1 e ip.dst=S2	Descartar pacote
R3	Se ip.src=H2 e ip.dst=S1	Encaminhar na porta P6
R4	Se ip.src=H2 e ip.dst=S2	Descartar pacote
R5	Se ip.src=H3 e ip.dst=S1	Descartar pacote
R6	Se ip.src=H1 e IP.dst=H2	Encaminha na porta P2
R7	Se ip.src=H3 e ip.dst=H2	Encaminhar na porta P2
R8	Se Ip.src=S1 e ip.dst=H2	Encaminha na porta P2
R...	...	...
Rn	Regra padrão	Encaminhar para Controlador

(b) Representação das regras de encaminhamento com uma única tabela

Tabela agregando fluxos		
#	Regras	Ação
R1	Se ip.src=H1 e ip.dst=S1	Encaminha na porta P6
R2	Se ip.src=H1 e ip.dst=S2	Descartar pacote
R3	Se ip.src=H2 e ip.dst=S1	Encaminhar na porta P6
R4	Se ip.src=H2 e ip.dst=S2	Descartar pacote
R678	Se IP.dst=H2	Encaminhar pela porta 2
R...	...	...
Rn	Regra padrão	Encaminhar para Controlador

(c) Representação de uma regra de encaminhamento agregada

Tabela 1: Controle de acesso	
Regra	Ação
Se ip.src=H1 e ip.dst=S2	Descartar pacotes
Se ip.src=H2 e ip.dst=S2	Descartar pacotes
Se ip.src=H3 e ip.dst=S1	Descartar pacotes
Se ip.src=H4 e ip.dst=S1	Descartar pacotes
Se ip.src=H5 e ip.dst=S1	Descartar pacotes
Regra padrão	Encaminhar para tabela 2

Tabela 2: Encaminhamento	
Regras	Ação
Se ip.dst=S1	Encaminha na porta P6
Se ip.dst=S2	Encaminha na porta P7
Se ip.dst=H1	Encaminha na porta P1
Se ip.dst=H2	Encaminha na porta 2
Se ip.dst=H3	Encaminha na porta P3
Se ip.dst=H4	Encaminha na porta P4
Se ip.dst=H5	Encaminha na porta P5
Regra padrão	Enviar para controlador

(d) Representação das regras de encaminhamento com múltiplas tabelas de fluxos

Figura 1. Exemplo de regras de encaminhamento

Para exemplificação, apresenta-se o cenário na Figura 1(a), uma pequena LAN com um único dispositivo de encaminhamento conectando cinco clientes (H1, H2, H3, H4 e H5) a dois servidores de aplicação (S1 e S2). O dispositivo de encaminhamento possui as funções de: (*i*) realizar o controle de acesso aos servidores e (*ii*) encaminhar os pacotes entre os nodos da rede. Em termos de controle de acesso, o dispositivo de encaminhamento autoriza ou rejeita os fluxos dentro da rede de acordo com as seguintes políticas: (*a*) os clientes H1 e H2 têm acesso apenas ao servidor S1, (*b*) os clientes H3 e H4 têm acesso apenas ao servidor S2, (*c*) o cliente H5 possui acesso a ambos os servidores e, finalmente, (*d*) os clientes possuem acesso um ao outro. Em termos de encaminhamento, cada um dos fluxos identificados pode ser redirecionado para uma das portas, onde um determinado nodo final se encontra (*i.e.,* H1:P1, H2:P2, H3:P3 ... S1:P6, S2:P7).

A Figura 1(b) apresenta uma tabela de fluxos com regras de encaminhamento compostas por dois campos de cabeçalho para realizar o processo de *lookup*: IP origem e IP destino. As ações que podem ser associadas a cada regra são: descartar pacotes caso um fluxo não pertença a lista de clientes autorizados ou encaminhar os pacotes dos fluxos para o destino, caso seja permitido. Para implementar o controle de acesso em apenas uma única tabela de fluxos, é necessária uma quantidade de regras que representem todas as combinações possíveis para a comunicação entre clientes e clientes e servidores de aplicação. Portanto, a quantidade de regras necessárias é de $O(n^2)$, sendo n a quantidade total de nodos na rede. Essa quantidade de regras pode ser maior que a capacidade de boa parte dos dispositivos de encaminhamento disponíveis no mercado [Costa et al. 2016].

A Figura 1(c) ilustra um exemplo de como pode ser realizada o gerenciamento por

agregação de fluxos. Nessa forma de gerenciamento, pretende-se mesclar duas ou mais regras de encaminhamento, a fim de representar vários fluxos em uma única regra. Por exemplo: H1, H3 e S1 desejam comunicar-se com H2. Nesse gerenciamento, as regras R6, R7 e R8 que possuem o mesmo destino e a mesma ação associada são mescladas em uma única regra. Assim, com uma única regra é possível representar três fluxos agregados. O gerenciamento por agregação pode reduzir drasticamente a quantidade de regras necessárias para representar fluxos em uma tabela OpenFlow. Por outro lado, esse gerenciamento mescla os fluxos impossibilitando que os mesmos sejam monitorados individualmente de uma forma precisa [Nguyen et al. 2015]. Outro aspecto a ser considerado é a carga de trabalho adicional que um algoritmo de agregação insere no processamento do controlador. Para agregar as regras de encaminhamento, o controlador deve ler o estado atual da tabela de fluxos do dispositivo, realizar a agregação e substituir o conjunto de regras originais pelo conjunto de regras agregadas. Esse processamento pode ocasionar perdas de pacotes, *loops* de encaminhamento e atrasos na rede [Luo et al. 2014].

Na Figura 1(d) pode-se observar o gerenciamento das regras por múltiplas tabelas de fluxos. Nessa forma de gerenciamento, subdivide-se uma tabela de fluxos em duas ou mais tabelas. Cada uma dessas sub-tabelas agrupa um conjunto de regras pertencentes a uma semântica, normalmente, determinada por uma política. A distribuição de regras e sequência pela qual os pacotes são analisados depende da aplicação que se pretende implementar. Seguindo o exemplo de controle de acesso do cenário apresentado, a primeira tabela armazena as regras referentes ao controle de acesso aos servidores. Essas regras são implementadas na forma de uma lista negra, *i.e.,* caso o fluxo pertença a lista, seus pacotes são descartados. Caso contrário, a tabela possui uma regra padrão que é aplicada aos pacotes que não encontrarem nenhuma entrada na lista correspondente. Nesse exemplo, a regra padrão é encaminhar os pacotes para a próxima tabela que armazena as regras de encaminhamento. Nessa segunda tabela, os pacotes pertencentes a cada um dos fluxos são encaminhados para portas de destino do dispositivo de encaminhamento.

As formas de gerenciamento de domínio espacial *intra-switch* propõem diferentes maneiras de se melhorar a utilização da capacidade de armazenamento limitado de entradas nas tabelas de fluxos, através da redução do conjunto de regras utilizadas pelos dispositivos de encaminhamento. Uma comparação entre estas formas de gerenciamento permite a identificação de qual forma de gerenciamento espacial é a mais indicada para uma rede com diferentes características, por exemplo, sua topologia, sendo o enfoque da próxima seção.

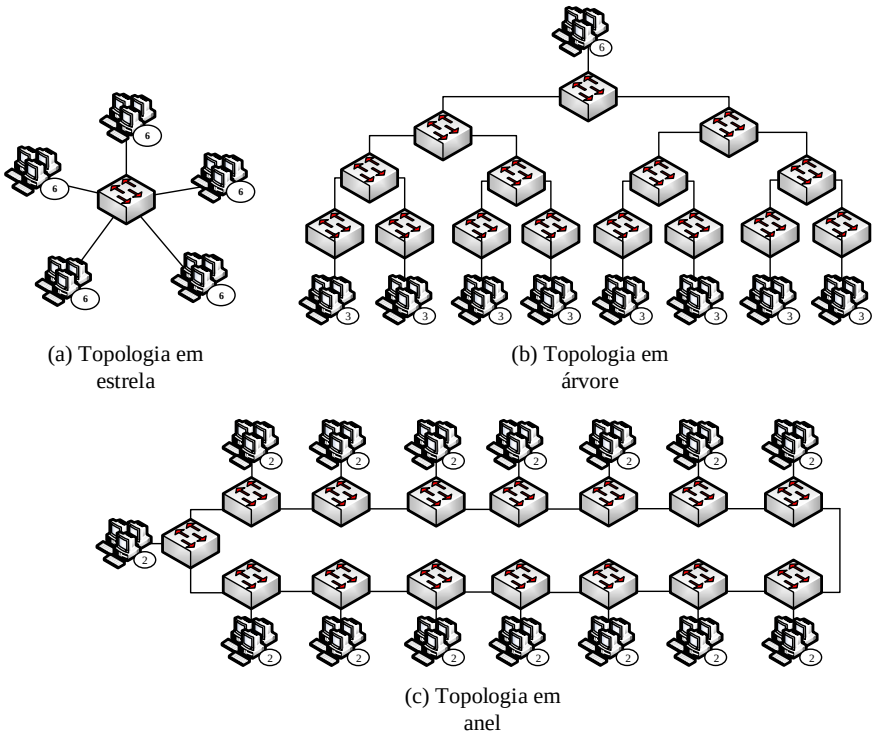
4. Comparação entre formas de gerenciamento de domínio espacial

Nessa seção apresenta-se a metodologia necessária para a realização da comparação entre as diferentes formas de gerenciamento (Subseção 4.1). Baseado nessa metodologia, na Subseção 4.2, discute-se os resultados obtidos.

4.1. Metodologia

Cenário. Para se comparar as diferentes formas de gerenciamento, três topologias são propostas: (*i*) estrela, (*ii*) árvore e (*iii*) anel. Na topologia de estrela, um único dispositivo de encaminhamento é utilizado para comunicar 30 nodos finais com o intuito de avaliar o impacto das formas de gerenciamento nesse dispositivo individualmente, como apresentado na Figura 2(a). Já, para realizar uma análise do impacto da utilização das diferentes

formas de gerenciamento de domínio espacial em um escopo mais amplo de rede, as duas topologias de árvore e anel são utilizadas, contendo 30 nodos finais conectados em 15 dispositivos de encaminhamento, como apresentadas nas Figuras 2(b) e 2(c). Sobre cada uma dessas redes com topologias diferenciadas, um controlador instala todas as regras de encaminhamento de maneira reativa, ou seja, quando um novo fluxo que não possui uma regra correspondente é identificado pelo dispositivo de encaminhamento, o controlador deve gerar uma nova regra que é instalada na tabela de fluxos.



(a) Topologia em estrela

(b) Topologia em árvore

(c) Topologia em anel

Figura 2. Topologias utilizadas nos experimentos

Carga de trabalho. Para realizar a comparação, é necessário a criação dos fluxos que são trafegados dentro das diferentes topologias de rede propostas. Primeiramente, um par de nodos finais é escolhido aleatoriamente (um cliente e um servidor). Os fluxos são gerados com base em dois parâmetros: duração do fluxo e intervalo entre rajadas. A duração de cada fluxo segue uma distribuição de log-normal com média $\mu = 4s$ e desvio padrão $\lambda = 1s$. O intervalo entre as rajadas de dados é um processo Poisson com média $\lambda = 1s$ [Neves et al. 2016] [Benson et al. 2010]. Fixa-se uma quantidade de 500 fluxos ativos na rede. Esta carga de trabalho representa um cenário realista, onde a maioria dos fluxos possui um ciclo de vida curto, enquanto apenas uma pequena parcela é constituída por fluxos com ciclos de vida longos. Dessa forma, cada experimento realizado, precisa ser executada por no mínimo 12 minutos, pois é o período de tempo necessário para que 500 fluxos sejam iniciados e concluídos.

Métricas. O desempenho de cada forma de gerenciamento espacial é comparado de acordo com quatro métricas: (*i*) quantidade de regras instaladas, (*ii*) quantidade de

intervenções do controlador, (*iii*) latência e (*iv*) *jitter*. Por quantidade de regras instaladas, considera-se a soma das regras presentes nos dispositivos de encaminhamento ao final de cada experimento, sendo a principal métrica que define a eficiência de uma forma de gerenciamento em relação a utilização da capacidade de armazenamento dos dispositivos de encaminhamento. Por quantidade de intervenções do controlador, considera-se a soma das mensagens para inserção de uma nova regra na tabela de fluxos (*flowmods*) e das mensagens de encaminhamento direto de pacotes (*packet-out*), sendo a principal métrica que define a sobrecarga inserida por uma forma de gerenciamento. Por latência, considera-se o tempo que um pacote precisa para alcançar o seu destino e retornar para a origem, também conhecido como *Round Trip Time*. Cada forma de gerenciamento irá influenciar a latência da rede de uma maneira diferente, principalmente, se o tempo de processamento da mesma é alto. Por *jitter*, considera-se a variação do atraso entre os pacotes de dados, sendo a métrica que consegue capturar o impacto das formas de gerenciamento na remoção e instalação de novas regras, gerando momentos instáveis na rede. Para a correta mensuração das métricas de desempenho em termos estatísticos, as coletas foram replicadas no mínimo 20 vezes, alcançando um nível de confiança de 95%.

4.2. Resultados

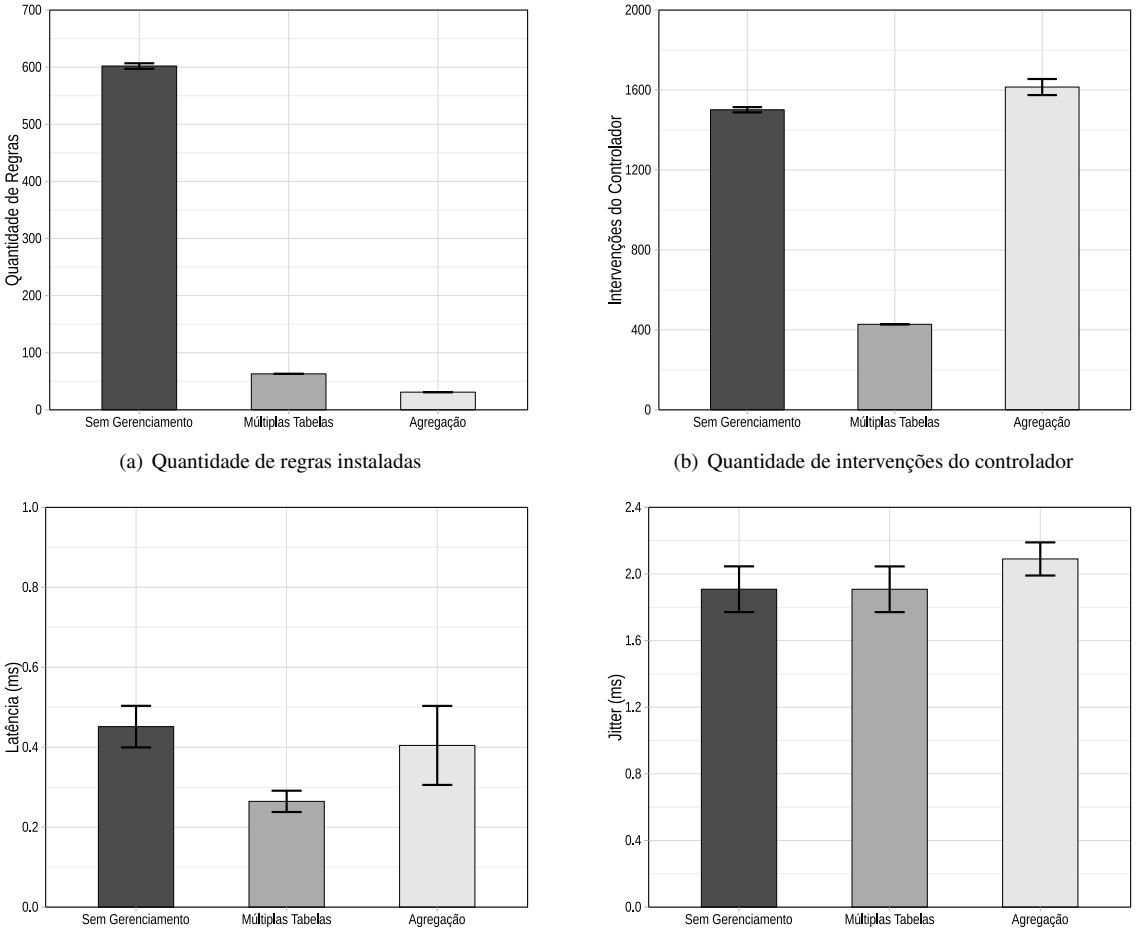
Essa seção apresenta os resultados experimentais obtidos a partir da metodologia proposta, organizados da seguinte forma. Primeiro, analisa-se o comportamento do gerenciamento por agregação e múltiplas tabelas comparando-os para um único dispositivo de encaminhamento na topologia de estrela. Em seguida, realiza-se a comparação entre essas formas de gerenciamento considerando as topologias de árvore e anel. É importante frisar que os experimentos foram realizados em um computador equipado com processador Intel i7-4770S com 4 núcleos de 3.1 GHz e 8GB de RAM. As formas de gerenciamento de domínio espacial foram implementados como aplicações do controlador Ryu (versão 4.2.2). Para mensurar a métrica de *jitter* foi utilizado a ferramenta iPerf versão 2.0.5. Além disso, foram utilizados o Mininet (versão 2.2.1) e Open vSwitch (versão 2.0.2 com suporte a OpenFlow 1.3) para emular uma rede real.

Comparação em uma Topologia Estrela

A topologia estrela permite a comparação entre as formas de gerenciamento de regras de domínio espacial para um único dispositivo de encaminhamento. Resultados coletados a partir do monitoramento desse dispositivo, podem ser observados a partir das Figuras 3(a), 3(b), 3(c) e 3(d). Essas figuras apresentam os gerenciamentos por agregação e múltiplas tabelas, bem como a linha base sem gerenciamento através das colunas e o eixo x.

Na Figura 3(a), o eixo y representa a eficiência de cada uma das formas de gerenciamento a partir da quantidade de regras instaladas no dispositivo de encaminhamento. Nota-se que o gerenciamento por agregação de fluxos reduz significativamente a quantidade total de regras utilizadas, alcançando uma melhora de aproximadamente 95%, quando comparado com a linha base sem gerenciamento. O gerenciamento por múltiplas tabelas, por sua vez, obteve uma redução de aproximadamente 89%, comparado a linha base, ou uma eficiência 6% menor que o gerenciamento por agregação.

Na Figura 3(b), o eixo y representa a quantidade de intervenções do controlador utilizadas para instalar regras e manter o estado da tabela de fluxos ao decorrer do experimentos. O gerenciamento por agregação de fluxos apresentou uma quantidade de



(a) Quantidade de regras instaladas

(b) Quantidade de intervenções do controlador

(c) Latência da rede

(d) Jitter da rede

Figura 3. Resultados obtidos para uma topologia de rede estrela

intervenções do controlador aproximadamente 8% maior do que a linha base e por volta de 370% maior em relação ao gerenciamento por múltiplas tabelas. A grande quantidade de interrupções necessária para a operação do gerenciamento por agregação decorre do processo de atualização da tabela de fluxos com as regras agregadas. Já, o gerenciamento por múltiplas tabelas, obteve uma redução na quantidade de intervenções do controlador de aproximadamente 71% em relação a linha base.

Na Figura 3(c), o eixo y representa a latência média mensurada minuto-a-minuto durante o experimento. Nota-se que o gerenciamento por múltiplas tabelas alcança a melhor latência dentro da rede estrela com 0,26 ms, enquanto o gerenciamento por agregação apresentou 0,40 ms e a linha base 0,45 ms. Os intervalos de confiança permitem identificar que o gerenciamento de múltiplas tabelas é comparativamente inferior aos demais. Entretanto, o mesmo não pode ser afirmado em relação ao gerenciamento por agregação e a linha base, permanecendo como não diferenciáveis para um nível de confiança de 95%.

Na Figura 3(d), o eixo y representa o *jitter* médio coletado durante a duração de cada fluxo. Pode-se observar, que o gerenciamento por múltiplas tabelas e a linha base atingiram um *jitter* médio de 1,9 ms e o gerenciamento por agregação apresenta 2,19 ms. Considerando o intervalo de confiança utilizado, pode-se concluir que as formas de gerenciamento são semelhantes e não diferenciáveis.

Baseado na comparação em uma topologia estrela, pode-se concluir que o gerenciamento por agregação possui a melhor eficiência para a redução das regras de encaminhamento na tabela de fluxos. Entretanto, para atingir essa eficiência são necessárias uma quantidade significativa de intervenções do controlador, inserindo sobrecarga na rede com tráfego de sinalização. Por sua vez, o gerenciamento por múltiplas tabelas, atinge uma eficiência semelhante ao gerenciamento por agregação, mas sem a necessidade de uma grande quantidade de intervenções do controlador, apresentando o melhor custo benefício entre as estratégias de domínio espacial *intra-switch* neste cenário.

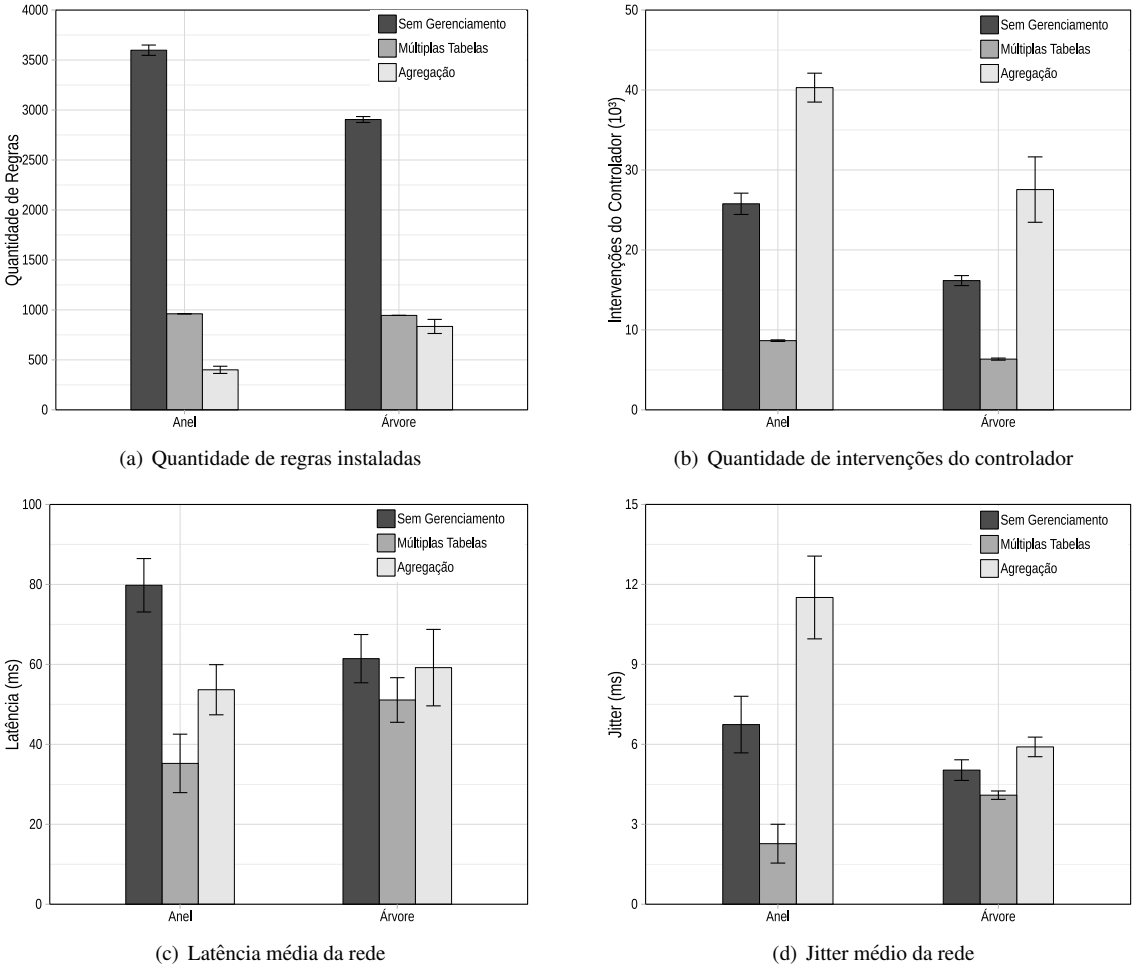
Comparação em Topologias de Árvore e Anel

As redes de topologias em anel e árvore permitem extrapolar a comparação entre as diferentes formas de gerenciamento espacial *intra-switch* para redes com números maiores de dispositivos. Para realizar essa comparação, os resultados coletados a partir do monitoramento dos dispositivo podem ser observados a partir das Figuras 4(a), 4(b), 4(c) e 4(d). Essas figuras apresentam as formas de gerenciamento espacial por agregação e múltiplas tabelas, bem como a linha base sem gerenciamento através das colunas com diferentes cores. Já, no eixo x, as colunas são agrupadas de acordo com as duas topologias.

A Figura 4(a) representa no eixo y a quantidade média das regras instaladas em cada dispositivo de encaminhamento. Para as topologias em anel e árvore, a eficiência do gerenciamento por agregação de fluxos é superior que as demais formas, alcançando uma melhora de 88% para anel e 71% para árvore, quando comparado com a linha base sem gerenciamento. Ainda que seja o mais eficiente, o gerenciamento por agregação apresentou baixa resiliência em relação a troca de topologias, duplicando sua quantidade de regras instaladas entre as topologias de anel para árvore. Já, o gerenciamento por múltiplas tabelas, mostrou-se resiliente a alteração das topologias, mantendo sua eficiência praticamente intacta, com 961 regras instaladas para anel (*i.e.*, uma redução de 73% comparado a linha base) e 945 regras para topologia em árvore (*i.e.*, uma redução de 67% comparado a linha base). É importante salientar que a resiliência é fundamental para a estabilidade e previsão do número de regras a serem utilizadas em redes reprogramáveis.

Na Figura 4(b), o eixo y representa a quantidade de intervenções do controlador ao decorrer do experimentos necessárias para instalar e manter o estado das tabelas de fluxos. Com um intervalo de confiança de 95%, pode-se constatar que o gerenciamento por múltiplas tabelas apresenta a menor quantidade de intervenções do controlador para as topologias de anel (66% abaixo da linha base) e para topologias em árvore (66% abaixo da linha base). Já, o gerenciamento por agregação de fluxo obteve a maior quantidade de intervenções do controlador, 155% acima da linha base para topologia em anel e 170% para topologia em árvore. Um fato interessante é o impacto significante no crescimento da quantidade de intervenções do controlador entre as topologia de árvore para anel, onde, no melhor caso, 2310 intervenções foram acrescidas utilizando o gerenciamento por múltiplas tabelas e, no pior caso, 12460 intervenções extras foram requisitadas pelo gerenciamento por agregação. Assim, a topologia da rede tem grande influência na quantidade de intervenções do controlador sem alteração na quantidade de fluxos ou número de encaminhamento presentes em relação a forma de gerenciamento utilizada.

Na Figura 4(c), o eixo y representa a métrica de latência média da rede. A forma de gerenciamento que propiciou a menor latência média para uma topologia em anel é por



(a) Quantidade de regras instaladas

(b) Quantidade de intervenções do controlador

(c) Latência média da rede

(d) Jitter médio da rede

Figura 4. Resultados obtidos para as topologias anel e árvore

múltiplas tabelas, com uma latência média de 35,22 ms. Já, para a topologia em árvore, todas as formas de gerenciamento, incluindo-se a linha base, são semelhantes dado a intersecção entre as barras de erro calculadas com um nível de confiança de 95%. É importante frisar que o gerenciamento por múltiplas tabelas apresentou uma latência 57% maior entre as topologias de anel para árvore e a mesma análise não pode ser realizada para o gerenciamento por agregação e linha base.

Na Figura 4(d), o eixo y representa o *jitter* médio mensurado nas redes com topologias em anel e árvore. O gerenciamento por múltiplas tabelas possui o menor impacto no *jitter* para ambas as topologias de redes, alcançando o valor médio de 2.27 ms para anel e 4.09 ms para árvore. Já, o gerenciamento por agregação obteve o maior impacto no *jitter* para ambas as redes, apresentando um valor médio de 11,50 ms para anel e 5.9 ms para árvore. É importante observar que as formas de gerenciamento possuem valores semelhantes para ambas as topologias, exceto pelo gerenciamento por agregação com um valor significantemente alto de *jitter*, próximo aos 12 ms. Isto significa, que o atraso da rede gerenciada por agregação de regras torna-se instável e compromete a utilização de aplicações sensíveis ao *jitter*, como *Voice Over IP* e vídeo chats seguros.

Baseado na comparação entre topologias de árvore e anel, percebe-se que o gerenciamento por agregação de fluxos mostrou-se mais eficiente entre as formas de geren-

ciamento espacial *intra-switch*. Entretanto, essa forma de gerenciamento apresentou uma elevada quantidade de intervenções do controlador e degradação da qualidade de serviço da rede. O gerenciamento por múltiplas tabelas demonstrou-se resiliente as topologias mantendo tanto a quantidade de regras instaladas, quanto a número de intervenções do controlador inalterados. Essa forma de gerenciamento não impactou significativamente na qualidade de serviço da rede. Assim, múltiplas tabelas apresenta o melhor custo benefício, para as topologias estrela, anel e árvore.

5. Conclusão e Trabalhos Futuros

Neste artigo apresentou-se uma comparação de duas formas de gerenciamento de regras de domínio espacial *intra-switch*, por agregação e múltiplas tabelas, para redes baseada em OpenFlow. Essas formas de gerenciamento foram comparadas em redes com diferentes topologias, *i.e.,* estrela, anel e árvore. A partir dos resultados obtidos, pode-se afirmar que o gerenciamento por agregação de fluxos reduz significativamente a quantidade de regras de encaminhamento independente da topologia utilizada. Entretanto, essa forma de gerenciamento requer uma quantidade elevada de intervenções do controlador, além de impactar negativamente na latência e no *jitter* das redes. Além disso, o gerenciamento por agregação de fluxos não é resiliente a troca de topologias, tendo sua eficiência alterada. Por outro lado, o gerenciamento por múltiplas tabelas reduz a quantidade de regras de encaminhamento, necessitando uma baixa quantidade de intervenções do controlador e com baixo impacto na latência e *jitter* das redes. Além disso, demonstrou-se resiliente a mudança de topologia, mantendo sua eficiência praticamente constante, tanto para as redes em anel, quanto em árvore. Assim, o gerenciamento por múltiplas tabelas possui o melhor custo benefício, considerando regras de domínio espacial *intra-switch*.

Como trabalhos futuros, pretende-se realizar experimentação em equipamentos reais que utilizem tabela de fluxos implementadas com memórias *Ternary Content-Addressable Memory* (TCAM). Além disso, pretende-se extrapolar a quantidade de estratégias avaliadas realizando um estudo mais abrangente utilizando outras formas de gerenciamento espacial de regras (*i.e.,* posicionamento de regras). Por fim, pretende-se investigar mecanismos que possam melhorar o desempenho das estratégias estudadas oferecendo garantias de desempenho a rede [ONF 2015b] [ONF 2015a].

Agradecimentos

Este trabalho foi financiado pelo programa Horizon 2020 da União Europeia para pesquisa, desenvolvimento tecnológico e demonstração no âmbito do acordo nº. 688941 (FUTEBOL), bem como pelo Ministério da Ciência, Tecnologia, Inovação e Comunicação (MCTIC) através da RNP/CTIC.

Referências

Benson, T., Akella, A., e Maltz, D. A. (2010). Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 267–280, New York. ACM.

Costa, L., Vieira, A., Silva, E., Macedo, D., Gomes, G., Correia, L., e Vieira, L. (2016). Avaliação de desempenho de planos de dados openflow. *34o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2016*.

Giroire, F., Moulierac, J., e Phan, T. K. (2014). Optimizing rule placement in software-defined networks for energy-aware routing. In *Proc. IEEE Global Communications Conference*, pages 2523–2529.

Kamiyama, N., Takahashi, Y., Ishibashi, K., Shiomoto, K., Otoshi, T., Ohsita, Y., e Murata, M. (2014). Flow aggregation for traffic engineering. In *Proc. IEEE Global Communications Conference*, pages 1936–1941.

Kanizo, Y., Hay, D., e Keslassy, I. (2013). Palette: Distributing tables in software-defined networks. In *Proc. IEEE INFOCOM 2013*, pages 545–549.

Kreutz, D., Ramos, F., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., e Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Lee, B. S., Kanagavelu, R., e Aung, K. M. M. (2013). An efficient flow cache algorithm with improved fairness in software-defined data center networks. In *Proc. IEEE 2nd Int. Conf. Cloud Networking (CloudNet)*, pages 18–24.

Luo, S., Yu, H., e Li, L. M. (2014). Fast incremental flow table aggregation in sdn. In *Proc. 23rd Int. Conf. Computer Communication and Networks (ICCCN)*, pages 1–8.

Neves, M., Oliveira, R., Mazzola, F., Marcon, D., Gaspary, L., e Barcellos, M. (2016). Contando os segundos: Avaliação de estratégias de domínio temporal para a gerência de regras em redes sdn. *34o. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2016*.

Nguyen, X. N., Saucez, D., Barakat, C., e Turletti, T. (2015). Officer: A general optimization framework for openflow rule allocation and endpoint policy enforcement. In *Proc. IEEE Conf. Computer Communications (INFOCOM)*, pages 478–486.

Nguyen, X. N., Saucez, D., Barakat, C., e Turletti, T. (2016). Rules placement problem in openflow networks: A survey. *IEEE Communications Surveys Tutorials*, 18(2):1273–1286.

ONF (2015a). The benefflow tables and TTPs. Open Networking Foundation. Disponível em: $https://www.opennetworking.org$.

ONF (2015b). The benefits of multiple flow tables and TTPs. Open Networking Foundation. Disponível em:

.

Vishnoi, A., Poddar, R., Mann, V., e Bhattacharya, S. (2014). Effective switch memory management in openflow networks. In *Proc. of the 8th ACM International Conference on Distributed Event-Based Systems*, DEBS '14, pages 177–188, New York. ACM.

Wickboldt, J., De Jesus, W., Isolani, P., Both, C., Rochol, J., e Granville, L. (2015). Software-define networking: Management requirements and challenges. *IEEE Communications Magazine*, 53(1):278–285.

AppendixC PUBLISHED EXTENDED ABSTRACT – LANCOMM STUDENT WORKSHOP 2019

Gustavo de Araújo, Cristiano Both, Juergen Rochol **A Flexible Architecture to Control Heterogeneous Low Power Wide Area Networks**. 2019 LANCOMM Student Workshop 2019 Latin American Student Workshop on Data Communication Networks, Brazil, Gramado-Rio Grande do Sul, 2019.

- **Title:** A Flexible Architecture to Control Heterogeneous Low Power Wide Area Networks.
- **Contribution:** an SDN-based architecture to create a common control abstraction among LPWAN technologies (*e.g.*, LoRa and NB-IoT) running on top of virtualized base stations.
- **Abstract:** LPWAN networks are candidates to complement traditional cellular networks by enriching different types of requirements such as density, reliability, and latency. However, no one-size-fits-all technology that can address all the requirements of IoT applications. For this reason, the deployment of heterogeneous LPWAN networks becomes necessary. On this paper, we present the HELPFUL an SDN-based architecture to create a common control abstraction among LPWAN technologies (*e.g.*, LoRa and NB-IoT) running on top of virtualized BSs. Moreover, we present a multiple flow tables rule management deployed on P4 language to address the scalability issues present in the SDN-based devices.
- **Status:** Published.
- **Qualis:** B4.
- **Conference:** Latin American Student Workshop on Data Communication Networks (LANCOMM).
- **Date:** May 7, 2017.
- **Local:** Gramado, Rio Grande do Sul.

- **URL:** <http://sbrc2019.sbc.org.br/en/>.

A Flexible Architecture to Control Heterogeneous Low Power Wide Area Networks

Gustavo Araújo
Federal University of Rio Grande do Sul - UFRGS

Cristiano Bonato Both
Vale Do Rio Sinos University - Unisinos

Juergen Rochol
Federal University of Rio Grande do Sul - UFRGS

ABSTRACT

LPWAN networks are candidates to complement traditional cellular networks by enriching different types of requirements such as density, reliability, and latency. However, no one-size-fits-all technology that can address all the requirements of IoT applications. For this reason, the deployment of heterogeneous LPWAN networks becomes necessary. On this paper, we present the HELPFUL an SDN-based architecture to create a common control abstraction among LPWAN technologies (*e.g.*, LoRa and NB-IoT) running on top of virtualized BSs. Moreover, we present a multiple flow tables rule management deployed on P4 language to address the scalability issues present in the SDN-based devices.

1 INTRODUCTION

The coexistence of heterogeneous LPWAN technologies is desired to exploit the best features each technology has to offer, addressing multiple communication requirements independently. On the other hand, deploying and operating several independent heterogeneous networks, each supporting a limited number of IoT devices, becomes challenging. Firstly, current LPWAN technologies have not been designed to coexist with each other, as each LPWAN implements individual control protocols leading to uncoordinated wireless access. Secondly, the assumption of multiple wireless access technologies in the traditional architecture of the cellular network would lead to prohibitive deployment costs of transmitters cells.

Aiming to cope with the deployment cost challenge, virtualization of Base Stations (BS) have already proposed as a solution to reduce CAPEX of cellular networks through infrastructure sharing [4]. The problem is that virtualization adds the need for new control access to the infrastructure, on uncontrolled access to the spectrum used by the LPWAN. Software-Defined Networking (SDN) provides a powerful approach by creating a programmable, dynamic, and flexible architecture that allows the abstraction from traditional hardware-based protocol implementations into a software-based network controller. In this sense, some studies have applied the SDN paradigm to provide a programmable network to IoT applications [5] [3] [1] [2]. However, these works do not take into account the limited capacity of SDN-based forwarding devices to store the forwarding rules. The tables have a limited rule storage capacity, which is several orders of magnitude smaller than that required for the operation of certain types of network, including LPWAN. This problem becomes worse when applied in LPWAN environments where a massive amount of end-nodes in a network making it necessary to manage the rules of the flow tables.

In this work, we propose HELPFUL (*Flexible Architecture to control Heterogeneous Low Power Wide Area Network*) to enable heterogeneous LPWAN networks to coexist whit scalability guarantees. HELPFUL is an SDN-based architecture to create a common control abstraction among LPWAN technologies running on top of virtualized BSs. Furthermore, we present multiple flow tables rule management to address the scalability issues present in the SDN-based devices. As the main contribution of this paper, we define an architecture that adds an abstraction layer that can translate the heterogeneous control protocols to a homogeneous set of messages. The technology-specific control could be unified into a single cross-technology controller, enabling infrastructure and spectrum access to be harmonized. For validation of HELPFUL, we prototype using the P4 language that allows having programmability of both control and forwarding plane.

2 RELATED WORK

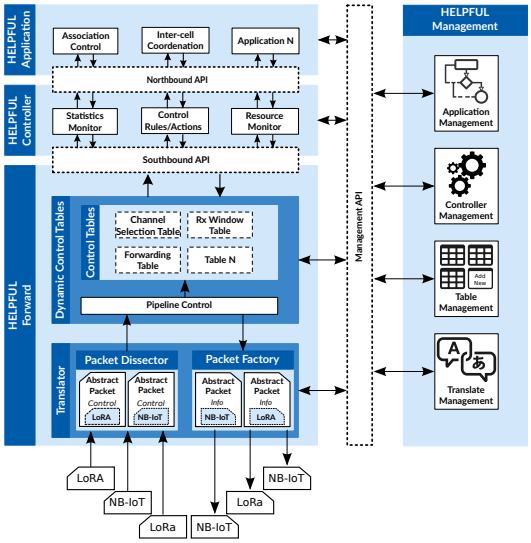
LPWAN technologies are not designed to coexist with each other. In the literature are found some studies that apply SDN concepts for coexistence between different technologies. In this sense, Gallo *et al.*, [3] propose the use of gateways that can be reprogrammed by a controller according to the network overhead. However, this proposal does not allow a gateway simultaneously operate with more than one technologies at the time. Other investigations adopt the OpenFlow protocol for simultaneous operation with more than one network technology [5] [1] [2]. However, these studies do not take into account the limited storage capacity of the flow tables. To better use the storage capacity of the tables it is needed to apply some management strategies is necessary. In the literature, there are four main ways of management: substitution, flow aggregation, caching, and multiple flow tables [6]. In our work, we are using the

multiple flow tables strategies because of the cost-benefit compared to the other strategies.

3 HELPFUL

In this section, we introduce the HELPFUL conceptual architecture and its main components. On its essentially the HELPFUL is an SDN-based architecture divided into four planes: forward, controller, application, and management. We implement our architecture using the P4 language. The light blue boxes represent the planes in Figure

Figure 1: HELPFUL Architecture



The HELPFUL Forward plane is composed of network devices interconnected through an LPWAN technology (*e.g.*, LoRaWAN, NB-IoT) or wired network technology. On our architecture the network devices are divided into two components: (*i*) the Translator and (*ii*) the Dynamic Control Tables. The Translator is responsible for creating a common control abstraction among LPWAN technologies by wrapped the information from a technology-specific packet and generated a new and abstract packet that can be submitted and processed by the Dynamic Control Tables. This Control is responsible for storing the rules that define the network functions. Each network function has a corresponding table on the Control Tables (*e.g.*, forwarding table, channel selection table).

Each table implements its own rules with his respective match fields and actions (*e.g.*, forward to specific ports, drop, forward to the controller, and rewrite some header). On this model, each table represents a network function and has its own set of header fields for identifying a package, actions to be applied and, rules counters. Further, each table implements just the necessary to make his function and avoiding consuming resources unnecessarily.

Vertically to the other planes, we have the HELPFUL Management. This plane has four components: (*i*) Application Management, (*ii*) Controller Management, (*iii*) Table Management and (*iv*) Translate Management. The Application Management is responsible for adding, removing, and get information for the network administrator about the applications that are running on the HELPFUL application. The Controller Management is responsible for informing the HELPFUL Controller the definitions of the rules and statistics of the tables. Table Management has the function of install and removes tables from the Dynamic Control Tables. The Translate Management configure the translator notifying how to wrapped a technology-specific and created an abstract packet that will be processed by the Dynamic Control Tables.

4 FUTURE WORK

As future work, we are concluding the implementation of the HELPFUL to carry out experimentation, and we can measure its benefits correctly. We will evaluate our rule management solution in the multiple flow table comparing to the existing forms of management of flow aggregation, caching, and eviction. In our preliminary results, it is already possible to identify that the type of control by multiple tables is the one that has the best cost-benefit compared to the aggregation of flows.

REFERENCES

[1] Michael Baddeley, Reza Nejabati, George Oikonomou, Mahesh Sooriyabandara, and Dimitra Simeonidou. 2018. Evolving SDN for Low-Power IoT Networks. *2018 4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018* (2018).

[2] Samaresh Bera, Sudip Misra, Sanku Kumar Roy, and Mohammad S. Obaidat. 2018. Soft-WSN: Software-defined WSN management system for IoT applications. *IEEE Systems Journal* (2018).

[3] Pierluigi Gallo, Katarzyna Kosek-Szott, Szymon Szott, and Ilenia Tinnirello. 2016. SDN@home: A method for controlling future wireless home networks. *IEEE Communications Magazine* (2016).

[4] Maicon Kist, Juergen Rochol, Luiz A Dasilva, and Cristiano Bonato Both. 2018. SDR Virtualization in Future Mobile Networks : Enabling Multi-Programmable Air-Interfaces. *2018 IEEE International Conference on Communications (ICC)* (2018).

[5] Tie Luo, Hwee-pink Tan, and Tony Q S Quek. 2012. Sensor OpenFlow : Enabling Software-Defined Wireless Sensor Networks. *IEEE Communications Letters* (2012).

[6] Xuan Nam Nguyen, Damien Saucez, Chadi Barakat, and Thierry Turletti. 2016. Rules Placement Problem in OpenFlow Networks: A Survey. *IEEE Communications Surveys and Tutorials* (2016).

AppendixD JOURNAL OF NETWORK AND COMPUTER APPLICATIONS 2019

Gustavo de Araújo, Rodolfo Antunes, Cristiano Both, Juergen Rochol **HELPFUL: Flexible Architecture to Control Heterogeneous Low Power Wide Area Networks**. 2019 Journal of Network and Computer Applications.

- **Title:** HELPFUL: Flexible Architecture to Control Heterogeneous Low Power Wide Area Networks
- **Contribution:** an SDN-based architecture to create a common control abstraction among LPWAN technologies (*e.g.*, LoRa and NB-IoT) running on top of virtualized base stations.
- **Abstract:** Low Power Wide Area Networks (LPWAN) are candidates to coexist with traditional cellular networks by coping with different types of requirements such as density, reliability, and latency. However, there is no one-size-fits-all technology that can address all the needs of IoT applications. For this reason, the integration of heterogeneous LPWAN becomes necessary. This article presents the HELPFUL, an SDN-based architecture that creates a common control abstraction among LPWAN technologies (*e.g.*, LoRa, NB-IoT) running on top of virtualized base stations. We also discuss four rule management strategies for use with HELPFUL, providing support for single and multiple flow tables. We evaluate our proposal with a series of experiments with a prototype developed using the P4 language. Results show which HELPFUL is flexible enough to change the management strategy to the best fit with the network demands. Consequently, it can reduce the number of messages on the control channel exchanged between the Controller and Gateways. Finally, HELPFUL adds minimal overhead to network performance, regardless of the rule management strategy chosen.
- **Status:** To apper.
- **Qualis:** A2.
- **Conference:** Journal of Network and Computer Applications (JNCA).
- **URL:** <https://www.journals.elsevier.com/journal-of-network-and-computer-applications>.