NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Wireless Software Defined Network Deployment and Optimization with Emphasis on Internet of Things Applications

Maroulis E. Nikolaos
Tsiatsios A. Georgios

Supervisor (or supervisors): Athanasia Alonistioti, Epicurus Professor

ATHENS

MARCH 2017

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη και Βελτιστοποίηση Ασύρματης Δικτύωσης Βασισμένη στο Λογισμικό με Έμφαση σε Εφαρμογές στο Διαδίκτυο των Πραγμάτων

Μαρούλης Ε. Νικόλαος
Τσιάτσιος Α. Γεώργιος

Επιβλέπων: **Αθανασία Αλωνιστιώτη, Επίκουρος Καθηγητής**

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2017

BSc THESIS

Wireless Software Defined Network Deployment and Optimization with Emphasis on Internet of Things Applications

Maroulis E. Nikolaos

S.N.: 1115201000212

Tsiatsios A. Georgios

S.N.: 1115201000169

SUPERVISOR: Athanasia Alonistioti, Epicurus Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη και Βελτιστοποίηση Ασύρματης Δικτύωσης Βασισμένη στο Λογισμικό με Έμφαση σε Εφαρμογές στο Διαδίκτυο των Πραγμάτων

Μαρούλης Ε. Νικόλαος

Α.Μ.: 1115201000212

Τσιάτσιος Α. Γεώργιος

Α.Μ.: 1115201000169

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Αθανασία Αλωνιστιώτη, Επίκουρος Καθηγητής**

ABSTRACT

We present the architecture and implementation of EmPOWER, a platform created for further SDN/NFV research and experimentation. The platform is based on the OpenFlow standard, but also extends its capabilities and features on the wireless and the mobile domain. EmPOWER rests on a single platform, which consists of general purpose hardware and operating system. It also provides three types of virtualized network resources such as forwarding nodes (OpenFlow switches), packet processing nodes (Micro Servers) and radio processing nodes (WiFi Access Points or LTE eNodeBs). The EmPOWER Network operating System consists of an OpenFlow Controller (e.g. Floodlight) and the EmPOWER master that runs on top of it and has a global view of the network, this allows the controller to have knowledge over the clients, the flows and the infrastructure of the network. An EmPOWER agent sits on top of each Access Point or WTP (Wireless Termination point) via the Click Modular Router and OpenVSwitch. The clients' abstraction, LVAP (Light Virtual Access Point), helps the agents running on the WTPs to allow multiple clients to be treated as a collection of logically isolated clients connected to different ports of a switch. Our implementation consists of an application on the EmPOWER SDN controller (SDN application layer), whose main functionality is to designate redirection rules, specified by the user, through a custom UI. These rules' goal is to determine the redirection of a client's packet to another client, directly after the WTP (Access Point) that the source client is assigned, receives it. The user identifies each client by its MAC address and a Label assigned to each client by the controller, he can also choose between two categories of rules, group rules and explicit rules. Group rules consist of a name, which is used for identification, many source clients and a target client. That means that packets received from the specified source LVAPs are directly redirected to the destination target LVAPS (client). Explicit rules consist of a source WTP, a source LVAP, a rule type, a destination WTP and a destination LVAP, which means that a packet sent from the source LVAP connected to the source WTP will be redirected to the destination LVAP in the destination WTP. These Rules are sent to the WTP from the controller through the Openflow Protocol. The purpose of all this is the direct communication of two or more clients via the wireless SDN network, without the packet leaving the data layer. As a result, the packet's redirection takes significantly less time compared to traditional networks, all that without the source LVAP having to know the destination LVAP(s). This complies with the basic concepts of Internet of Things as the communication between devices, becomes more manageable and handy. Considering the benefits of the EmPOWER Platform, the SDN in general and Internet of Things, we conclude that it is a promising solution, but leaves room for improvement as at an early stage of development.

SUBJECT AREA: Software Defined Networking

KEYWORDS: EmPOWER, OpenFlow, Light Virtual Access Point, Wireless Termination Point, Internet of Things, Packet Flow Redirection

ΠΕΡΙΛΗΨΗ

Παρουσιάζουμε την αρχιτεκτονική και την εγκατάσταση του EmPOWER, μιας πλατφόρμας που δημιουργήθηκε για περεταίρω έρευνα και πειραματισμό του SDN/NFV. Η πλατφόρμα βασίζεται στο πρότυπο OpenFlow, αλλά επίσης επεκτείνει τις δυνατότητες και τα χαρακτηριστικά του στους τομείς του ασύρματου και του κινητού δικτύου. Το EmPOWER στηρίζεται σε μια ενιαία πλατφόρμα, η οποία αποτελείται από υλικό και λειτουργικό σύστημα γενικού σκοπού. Επίσης παρέχει τρείς τύπους εικονικών δικτυακών πόρων, όπως κόμβους προώθησης (μεταγωγείς OpenFlow), κόμβους επεξεργασίας πακέτων (Micro Servers) και κόμβους επεξεργασίας ασυρμάτου (WiFI σημείων πρόσβασης ή LTE eNodeBs). Το λειτουργικό σύστημα δικτύωσης EmPOWER αποτελείται από έναν διαχειριστή OpenFlow (πχ. Floodlight) και τον EmPOWER Master, ο οποίος τρέχει πάνω από αυτόν και έχει γενική όψη του δικτύου, αυτό επιτρέπει στον διαχειριστή να έχει την επίγνωση των πελατών, των ροών και της υποδομής του δικτύου. Ο πράκτορας EmPOWER βρίσκεται πάνω σε κάθε σημείο πρόσβασης ή WTP(Ασύρματο Τερματικό Σημείο), μέσω του Click(λογισμικό δρομολογητή) και του OpenVswitch. Το LVAP (Ελαφρύ Εικονικό σημείο Πρόσβασης) μια αφαίρεση των πελατών, η οποία επιτρέπει στους πράκτορες πάνω στα ενεργά WTPs, να αντιμετωπίζουν πολλαπλούς πελάτες ως μια συλλογή λογικά απομονωμένων πελατών συνδεδεμένους σε διαφορετικές θύρες του μεταγωγέα. Η υλοποίηση μας αποτελείται από μια εφαρμογή στον EmPOWER SDN διαχειριστή (επίπεδο εφαρμογής SDN), της οποίας η κύρια λειτουργεία είναι ο ορισμός κανόνων ανακατεύθυνσης, ορισμένοι από τον χρήστη μέσω μιας διεπαφής. Στόχος αυτών των κανόνων είναι να καθοριστεί η ανακατεύθυνση των πακέτων του πελάτη σε έναν άλλο πελάτη αφότου το WTP (σημείο πρόσβασης) όπου ο πηγαίος πελάτης είναι συνδεμένος, το λάβει. Ο χρήστης αναγνωρίζει κάθε πελάτη από την διεύθυνση MAC και την ανατεθειμένη από τον διαχειριστή ετικέτα, επίσης ο χρήστης μπορεί να επιλέξει δύο κατηγορίες κανόνων, ομαδικούς και ρητούς κανόνες. Οι ομαδικοί κανόνες αποτελούνται από ένα όνομα, το οποίο είναι για την ταυτοποίηση του, από πολλούς πηγαίους πελάτες και ένα στόχο πελάτη. Αυτό σημαίνει ότι τα πακέτα τα οποία λαμβάνονται από συγκεκριμένα πηγαία LVAP ανακατευθύνονται απευθείας στο LVAP στόχο(πελάτη). Οι ρητοί κανόνες αποτελούνται από ένα πηγαίο WTP, πηγαίο LVAP, τον τύπο του κανόνα, τον στόχο WTP και LVAP, το οποίο σημαίνει ότι το πακέτο το οποίο στέλνεται από το πηγαίο LVAP που είναι συνδεδεμένο στο πηγαίο WTP θα ανακατευθυνθεί στο στόχο LVAP στο στόχο WTP. Αυτοί οι κανόνες αποστέλνονται από τον διαχειριστή στο WTP μέσω του πρωτοκόλλου OpenFlow. Ο σκοπός αυτού, είναι η απευθείας επικοινωνία μεταξύ δύο ή περισσοτέρων πελατών στο ασύρματο δίκτυο SDN, χωρίς να χρειαστεί το πακέτο να φύγει από το επίπεδο δεδομένων. Ως αποτέλεσμα, η ανακατεύθυνση των πακέτων χρειάζεται σημαντικά λιγότερο χρόνο σε σχέση με τα παραδοσιακά δίκτυα και όλα αυτά χωρίς το πηγαίο LVAP να γνωρίζει το/τα LVAP στόχου/ων. Αυτό συμμορφώνεται με τις βασικές έννοιες του Διαδικτύου των Πραγμάτων αφού η επικοινωνία μεταξύ συσκευών, γίνεται πιο διαχειρίσιμη και εύχρηστη. Αποτιμώντας τα πλεονεκτήματα της πλατφόρμας EmPOWER, του SDN και του Διαδικτύου των Πραγμάτων, καταλήγουμε ότι αποτελεί μια υποσχόμενη λύση, αλλά αφήνει περιθώρια εξέλιξης καθώς βρίσκεται σε πρώιμο στάδιο ανάπτυξης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Δικτύωση Βασισμένη στο Λογισμικό

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: EmPOWER, OpenFlow, Ελαφρύ Εικονικό Σημείο Πρόσβασης, Ασύρματο Τερματικό Σημείο,Διαδίκτυο των Πραγμάτων, Ανακατεύθυνση Ροής Πακέτων

AKNOWLEDGMENTS

During our course for the fulfillment of this Thesis, several people helped us during its completion and we would like to thank them. First of all, we would like to thank our supervisor, Epicurus Professor Athanasia Alonistioti for her support, for providing her knowledge and allowing us to use SCAN Lab's Equipment in order to accomplish our thesis requirements. Furthermore, we would like to give special thanks to Alexandros Tsakrilis and Panagiotis Kontopoulos for setting up the EmPOWER platform and providing us with their knowledge and guidance over the challenges we were facing. Another person we would like to thank is Dr. Roberto Riggio from the Future Networks (FuN) at CREATE-NET for giving us an insight on the fundamentals of the platform. Also, we would to thank Dimitris Soukaras for his technical knowhow on the lab's network infrastructure and finally all the SCAN Lab members for their support.

CONTENTS

LIST OF IMAGES

LIST OF TABLES

PREFACE

This thesis was developed in Athens from July 2016 till March 2017 at the Department of Informatics and Telecommunications of the National Kapodistrian University of Athens. The concepts and technologies that have been developed during this thesis constitute an important role in the future of wireless Software Defined Networking. The EmPOWER testbed gave us a solid platform for experimentation and development for our ideas. On top of that we would like to thank our thesis supervisor, Epicurus Professor, Athanasia Alonistioti, for giving us the opportunity to develop our thesis, for supplying us the SCAN Lab's equipment for experimentation and finally for the guidance and knowledge that she provided. Concluding, this constitutes an essential part of acquiring our Bachelor degree.

1. INTRODUCTION

The recent advances of Information Technologies, the diffusion of ultra-broadband (fixed and radio) connectivity, the continuous reduction of hardware costs and the wider and wider availability of open source software solutions, are creating the conditions for introducing a deep innovation in the architectural design and in the operations of future telecommunications networks and services.

We are witnessing a period of rapidly growing interest on the part of industry and academia in Software Defined Networks (SDN) and Network Function Virtualization (NFV). The growing interest in these paradigms is most probably motivated by the novelty of the overall context, specifically their techno-economic sustainability and high level performance. Thanks to these techno economic trends, SDN and NFV principles will soon impact not only current telecommunications fixed and mobile networks, but also service and application platforms. In fact, SDN and NFV can be seen as facets of a broad innovation wave, called Softwarization, which will contribute to automating processes, optimizing costs, reducing time to market, providing better services. At the same time, the Internet of Things (IoT), will generate a plethora of new services and applications, ranging from industrial and mission critical ones to precision agriculture, to Smart Cities, etc.

1.1 From Traditional Networks to Software Defined Networking

The explosion of mobile devices and content, server virtualization, and advent of cloud services are among the trends driving the networking industry to re-examine traditional network architectures. Many conventional networks are hierarchical, built with tiers of Ethernet switches arranged in a tree structure. This design made sense when client-server computing was dominant, but such a static architecture is ill-suited to the dynamic computing and storage needs of today's enterprise data centers, campuses, and carrier environments. Meeting current market requirements is virtually impossible with traditional network architectures. Faced with flat or reduced budgets, enterprise IT departments are trying to squeeze the most from their networks using device-level management tools and manual processes. Carriers face similar challenges as demand for mobility and bandwidth explodes, profits are being eroded by escalating capital equipment costs and flat or declining revenue. Existing network architectures were not designed to meet the requirements of today’s users, enterprises, and carriers, rather network designers are constrained by the limitations of current networks.

Now let’s give a mild introduction of the traditional networking. Network Devices have a control plane that provides information used to build a forwarding table. They also consist of a data plane that consults the forwarding table. The forwarding table is used by the network device to decide where to send frames or packets that are entering. Both planes exist directly on the networking device.

Software Defined Networking on the other hand, abstracts this concept, and places the Control Plane functions on an SDN controller. The SDN controller can be a server running SDN software. The Controller communicates with a physical or virtual switch Data Plane through a protocol called OpenFlow. OpenFlow conveys the instructions to the data plane on how to forward data. The network device must run the OpenFlow

protocol for this to be possible. The following graphic, summarizes the basic concept difference between traditional networking and SDN.

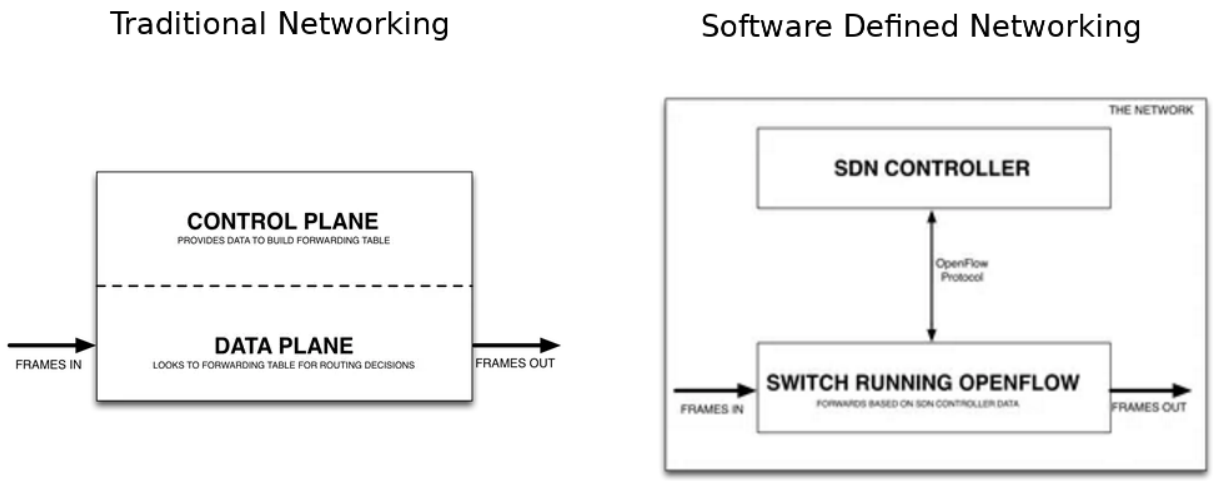

Image 1.1 Traditional Network and SDN Synopsis

With SDN, the applications can be network aware, as opposed to traditional networks where the network is application aware (or rather, application ambivalent). Traditional (i.e. non SDN) applications only implicitly and indirectly describe their network requirements, typically involving several human processing steps, e.g. to negotiate if there are sufficient resources and policy controls to support the application. Traditional networks do not expose information and network state to the applications using them, instead using an SDN approach, SDN Applications can monitor network state and adapt accordingly. The control plane is logically centralized and decoupled from the data plane.

The SDN Controller summarizes the network state for applications and translates application requirements to low level rules. This does not imply that the controller is physically centralized. For performance, scalability, and/or reliability reasons, the logically centralized SDN Controller can be distributed so that several physical controller instances cooperate to control the network and serve the applications. Control decisions are made on an up to date global view of the network state, rather than distributed in isolated behavior at each network hop. With SDN, the control plane acts as a single, logically centralized network operating system in terms of both scheduling and resolving resource conflicts, as well as abstracting away low level device details, e.g. electrical vs. optical transmission. The SDN Controller has complete control of the SDN Datapaths, subject to the limit of their capabilities, and thus does not have to compete/contend with other control plane elements, which simplifies scheduling and resource allocation. This allows networks to run with complex and precise policies with greater network resource utilization and quality of service guarantees. This occurs through a well understood common information model (e.g. as the one defined by OpenFlow).

Table 1 Traditional Networks and SDN Point to Point comparison

Traditional Networking	Software Defined Networking
They are Static and inflexible networks. They are not useful for new business ventures. They possess little agility and flexibility	They are programmable networks during deployment time as well as at later stage based on change in the requirements. They help new business ventures through flexibility, agility and virtualization.
They are Hardware appliances.	They are configured using open software.
They have distributed control plane.	They have logically centralized control plane.
They use custom ASICs and FPGAs.	They use merchant silicon.
They work using protocols.	They use APIs to configure as per need.

1.2 SDN Deployment for the Internet of Things

The Internet of Things (IoT) is a paradigm that is rapidly gaining ground in modern wireless telecommunications. The basic idea is the pervasive presence around us of a variety of smart things or devices such as Radio-Frequency Identification (RFID) tags, sensors, actuators, smart mobile phones through unique addressing schemes which are able to interact with each other and cooperate with their neighbors to reach common goals in an intelligent way. Advancement in wireless networking has let these thousands of smart devices connect to the Internet anywhere and anytime. With the development of IoT, the amount of data produced per day increases exponentially. Nowadays we are also in the Cloud computing and big data era in which most of computing and communication resources are shared and provided to users. The characteristics of diversity, dynamics, and 1 big data explosion bring a big challenge for the design of the IoT architecture in the Cloud and big data era. Networks should now be more intelligent, more powerful, more efficient, more secure, more reliable, and more scalable to meet the requirements of the characteristics of diversity and dynamics

Introduction of both NFV and SDN to the IoT framework could leverage the network efficiency and attain the programmability and flexibility of networks. For example, using the OpenFlow-based SDN (SDN-OF) technologies with NFV implementation, it is possible to implement the IoT networking functions such as prioritizing critical/control traffic for QoS in a centralized programmable controller. It is believed that through such network function virtualization for the SDN-based IoT framework, the efficiency and the network agility of IoT could be leveraged significantly.

1.3 The Wireless SDN Domain and the EmPOWER Solution

Due to the unreliable nature of the wireless medium, the wireless domain poses a challenge for the research community and the enterprises. The complexity of the network protocols makes it difficult to expand them, leading to the reliance of the vendor's tools. This necessity results in less flexibility, less manageability and more difficult expansion in this domain.

Software Defined Networks and OpenFlow were designed with infrastructure networks in mind, and more specifically for wired networks and the adaptation to the wireless context is not straight forward. For example, most of the current wireless SDN implementations only work well when slicing uses different channels. Researchers implemented a wireless mesh network using SDN and their work showed serious issues regarding the time required to set up a new rule from the controller to the access points, the time needed to parse rules for an incoming packet, or the control traffic volume.

Regardless wireless domain is also where SDN bears the highest potential, as it provides functions that could foster a better collaboration between access points to reduce interferences or to enhance security. Implementing SDN requires at least to be able to define slices and to limit interactions between these slices, and to let the network devices measure and report their status to the relevant controllers.

A notable experimental stable solution to the wireless network and mobile domain, which allow us to test experimental protocols is the Empower project. The empower platform was created by Dr. Roberto Riggio and his research team Future Networks (FuN) at CREATE-NET.

5G-EmPOWER is an open Mobile Network Operating System for Wi-Fi and LTE networks. Its flexible architecture and the high-level programming APIs allow for fast prototyping of apps and services.5G-EmPOWER blurs the line between radio and core network introducing the concept of Programmable Data Plane which abstracts the radio and packet processing resources available in a network.

5G-EmPOWER natively support multi-tenancy which allows supporting verticals with orthogonal requirements over the same network while ensuring performance isolation and efficient spectrum utilization. It is important to mention that the EmPOWER platform was initially designed not only for the wired domain but also for the wireless satisfying the limitations of the wired domain. Subsequently it enhanced its capabilities by supporting the mobile domain. In addition, it has expanded its capabilities to establish the experimentation of 5G that is an emerging concept. The platform is continuously updating in order to keep up with the hot trends in software defined networking.

2. THE SOFTWARE DEFINED NETWORKING APPROACH

2.1 SDN Definition

Software Defined Networking (SDN) is changing the way we design and manage networks, it defines a new approach to computer networking that allows network administrators to programmatically initialize, control, change, and manage network behavior dynamically through the abstraction of lower-level functionality. SDN is defined from two characteristics. Firstly, SDN separates the control plane (which carries signaling traffic and is responsible for routing) from the data plane (which forwards traffic according to decisions that the control plane makes). Secondly SDN consolidates the control plane, as a result a single software control program controls multiple data-plane elements, in this context the SDN control plane exercises direct control over the state in the network's data-plane elements. The goal is to leverage this separation, and the associated programmability, in order to reduce complexity and enable faster innovation at both planes. Concluding, the plethora of the capabilities that SDN provide, has contributed to receive great support from the wired community.

2.2 SDN Architecture

A Software Defined Networking architecture defines how a networking and computing system can be built using a combination of open, software-based technologies and commodity networking hardware that separate the control plane and the data layer of the networking stack.

The aim of SDN is to provide open interfaces that enable the software development that controls the flow of network traffic provided by a set of network resources, along with the possible inspection and modification of traffic in the network. In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications. As a result, enterprises and carriers gain unprecedented programmability, automation, and network control, enabling them to build highly scalable, flexible networks that readily adapt to changing business needs.

2.2.1 Overview

The SDN Architecture complies with the following characteristics:

- Directly programmable: Network control is directly programmable because it is decoupled from forwarding functions.
- Agile: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs.

- Centrally managed: Network intelligence is (logically) centralized in software-based SDN controllers that maintain a global view of the network, which appears to applications and policy engines as a single, logical switch.
- Programmatically configured: SDN lets network managers configure, manage, secure, and optimize network resources very quickly via dynamic, automated SDN programs, which they can write themselves because the programs do not depend on proprietary software.
- Open standards-based and vendor-neutral: When implemented through open standards, SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-specific devices and protocols.

The Software Defined Networking method centralizes the control over the network by separating the control logic to off-device computer resources.

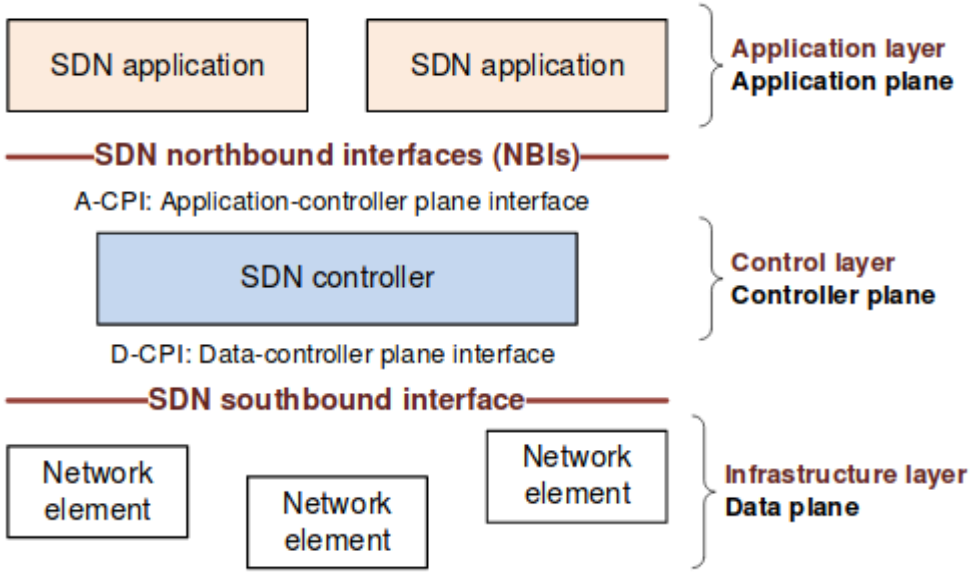

Image 2.1 SDN Layer Overview

All SDN models are based on a SDN Controller and as well to southbound APIs and northbound APIs. In general, the SDN architecture is based on four “planes”:

1. SDN Controller
2. SDN Application Plane
3. Data Plane (SDN Networking Devices/Elements)
4. SDN Management

There are two more planes, which are often included in the management plane, because they complement it. These are:

- SDN Administration
- SDN ONF Protocols

Controller Plane

The “brain” of the network, SDN Controllers offer a centralized view of the overall network, and enable network administrators to dictate how the underlying systems (Data Plane) should handle network traffic. The minimum functionality of the SDN controller is to faithfully execute the requests of the applications it supports, while isolating each application from all others. To perform this function, an SDN controller may communicate with peer SDN controllers, subordinate SDN controllers, or non-SDN environments, as necessary. A common but non-essential function of an SDN controller is to act as the control element in a feedback loop, responding to network events to recover from failure, re-optimize resource allocations, or otherwise.

Northbound APIs (Application Plane)

The northbound application program interfaces (APIs) are used to communicate between the SDN Controller and the services and applications running over the network. The northbound APIs can be used to facilitate innovation and enable efficient orchestration and automation of the network. As a result, Northbound APIs are arguably the most critical APIs in the SDN environment, it can potentially support and enable innovative applications whose value of SDN is important. Because they are so critical, northbound APIs must support a wide variety of applications in a SDN environment.

Data Plane

The data plane comprises a set of one or more network elements, each of which contains a set of traffic forwarding or traffic processing resources. Resources are abstractions of underlying physical capabilities or entities.

Management

Each application, SDN controller and network element has a functional interface to a manager. The minimum functionality of the manager is to allocate resources from a resource pool in the lower plane to a particular client entity in the higher plane, and to establish reachability information that permits the lower and higher plane entities to mutually communicate. Additional management functionality is not precluded, subject to the constraint that the application, SDN controller, or NE have exclusive control over any given resource.

Administration

Each entity in a north-south progression through the planes may belong to a different administrative domain. The manager is understood to reside in the same administrative domain as the entity it manages

ONF protocols

The OF-config protocol is positioned to perform some of the functions that are needed at the management interface. The OF-switch protocol is positioned to perform some of the functions that are needed at the D-CPI (Data-Controller Plane Interface) and possibly at the A-CPI (Application-Controller Plane Interface).

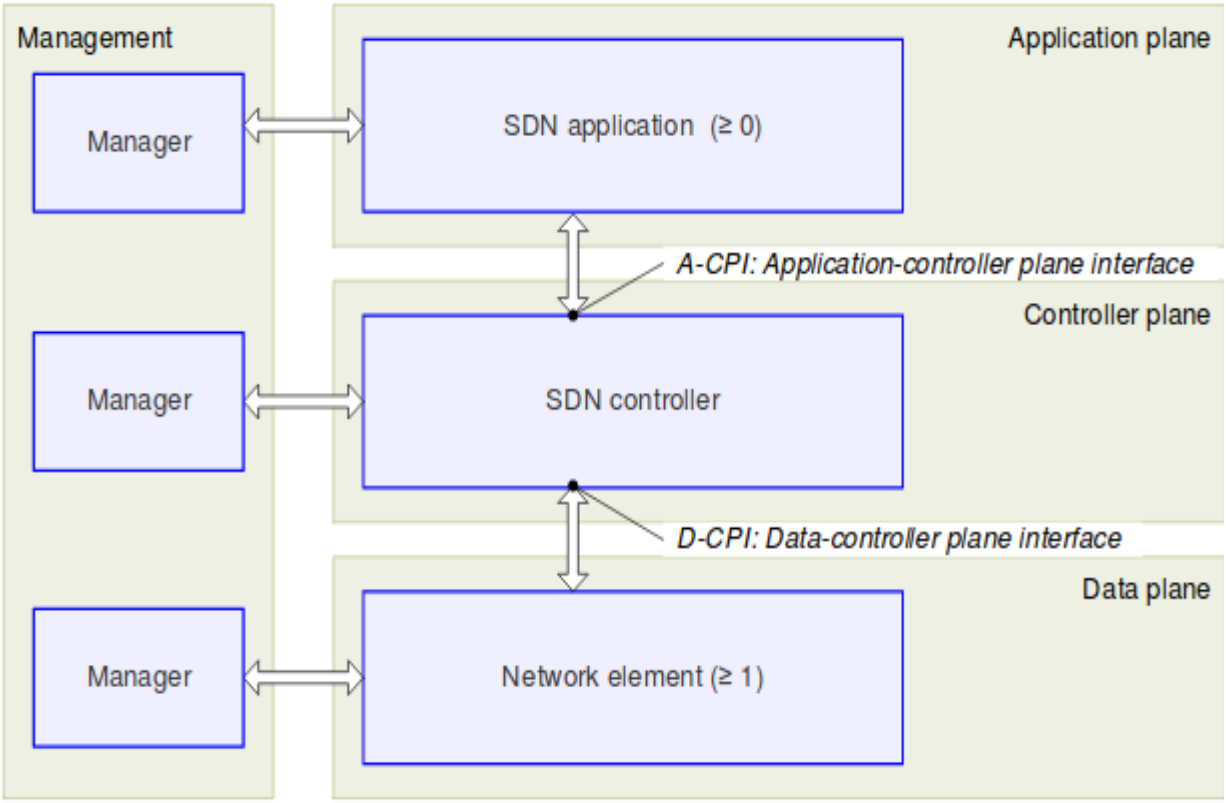

Image 2.2 SDN component structure

2.2.2 Basic Principles

An SDN architecture is characterized by three key attributes:

- **Logically centralized intelligence**

In the SDN architecture, network control is distributed from forwarding using a standardized southbound interface: OpenFlow. In comparison to local control, a centralized controller has a broader perspective of the resources under its control, and can potentially make better decisions about how to deploy them.

- **Decoupling of controller and data planes**

This principle calls for separable controller and data planes. However, it is understood that control must necessarily be exercised within data plane systems. The D-CPI between SDN controller and network element is defined in such a way that the SDN controller can delegate significant functionality to the NE, while remaining aware of NE state.

- **Abstraction**

In an SDN network, the applications that consume SDN services are abstracted from the underlying network technologies. However, it is understood that control must necessarily be exercised within data plane systems. Network devices are also

abstracted from the SDN Control Layer to ensure portability and future-scalability in network services, the network software resident in the Control Layer. The principle of abstracting network resources and state to applications via the A-CPI allows for programmability of the network. The applications are able to specify requirements and request changes to their network services, by providing information about their resources and state, via the SDN controller. Further, the concept of hierarchically recursive application/controller layers and trust domains also allows application programs to be created that may combine a number of component applications, further the architecture decomposes functional entities and the information and operations that need to be exchanged over various interfaces among them into a not necessarily comprehensive set of functional components.

2.2.3 SDN controller functional components

It is important to conceptualize a minimum set of functional components within the SDN controller, namely data plane control function (DPCF), coordinator, virtualizer, and agent. Subject to the logical centralization requirement, an SDN controller may include arbitrary additional functions. A resource data base (RDB) models the current information model instance and the necessary supporting capabilities.

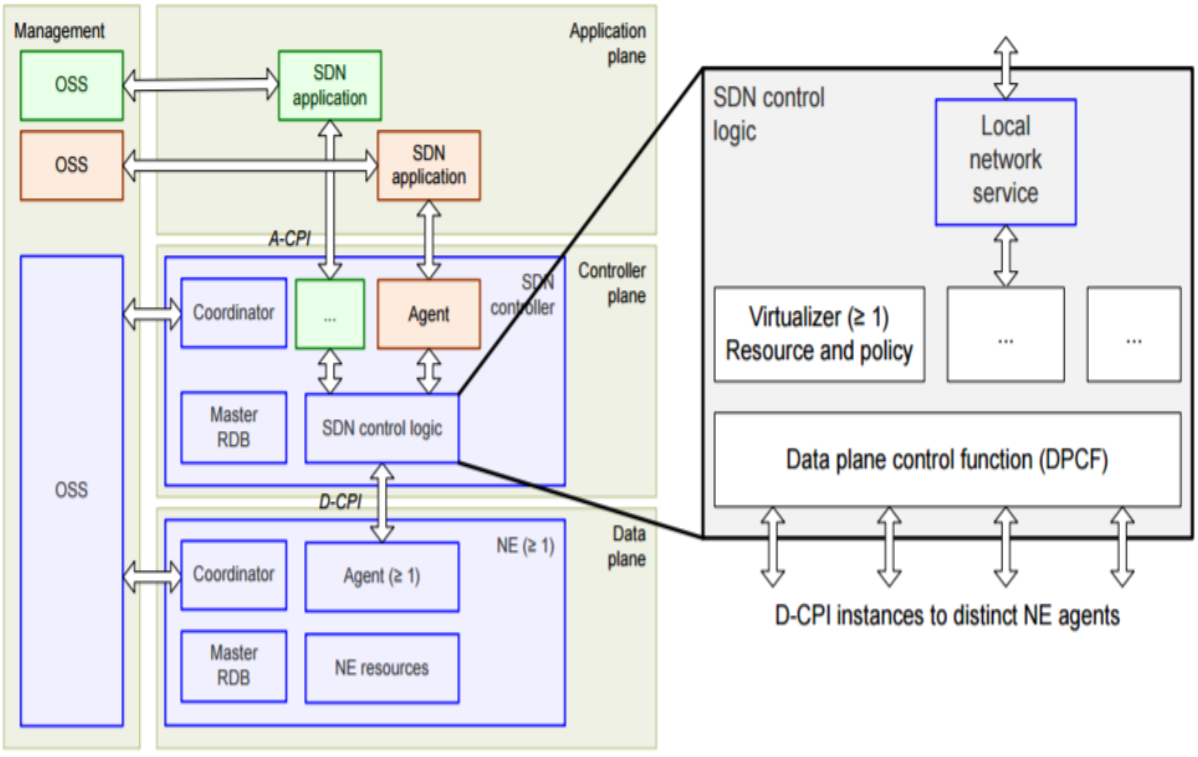

Image 2.3 SDN Control Logic

Data plane control function

The DPCF component effectively owns the subordinate resources available to it and uses them as instructed by the OSS/coordinator or virtualizer(s) that controls them. These resources take the form of an information model instance accessed through the agent in the subordinate level. Because the scope of an SDN controller is expected to span multiple (virtual) NEs or even multiple virtual networks (with a distinct D-CPI instance to each), the DPCF must include a function that operates on the aggregate. This function is commonly called orchestration. This architecture does not specify orchestration as a distinct functional component.

Coordinator

To set up both client and server environments, management functionality is required. The coordinator is the functional component of the SDN controller that acts on behalf of the manager. Clients and servers require management, throughout all perspectives on data, control and application plane models, so coordinator functional blocks are ubiquitous.

Virtualizer

In the SDN architecture, virtualization is the allocation of abstract resources to particular clients or applications; in NFV, the goal is to abstract network functions away from dedicated hardware, for example to allow them to be hosted on server platforms in cloud data centers. A virtualizer is instantiated by the OSS/coordinator for each client application or organization. The OSS/coordinator allocates resources used by the virtualizer for the A-CPI view that it exposes to its application client, and it installs policy to be enforced by the virtualizer. The effect of these operations is the creation of an agent for the given client.

Agent

Any protocol must terminate in some kind of functional entity. A controller-agent model is appropriate for the relation between a controlled and a controlling entity, and applies recursively to the SDN architecture. The controlled entity is designated the agent, a functional component that represents the client’s resources and capabilities in the server’s environment. An agent in a given SDN controller represents the resources and actions available to a client or application of the SDN controller. Even though the agent’s physical location is inside the server’s trust domain (i.e., on a server SDN controller platform), the agent notionally resides in the client’s trust domain.

Additional functions may take the form of applications or features supported by the controller. These features may be exported to some or all of the server’s external applications clients, or used internally by the provider administration for its own purposes. As components of the SDN controller, such applications or features are subject to the same synchronization expectation as other controller components. To facilitate integration with third party software, the interfaces to such applications or features may be the same as those of others at the A-CPI.

2.3 OpenFlow

OpenFlow is the first standard communications interface between the control and the forwarding layers of an SDN architecture. OpenFlow allows direct access and manipulation to the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based). It is the absence of an open interface to the forwarding plane that has led to the characterization of today's networking devices as monolithic, closed and mainframe-like, a protocol like OpenFlow is needed to move network control out of the networking switches to logically centralized control software.

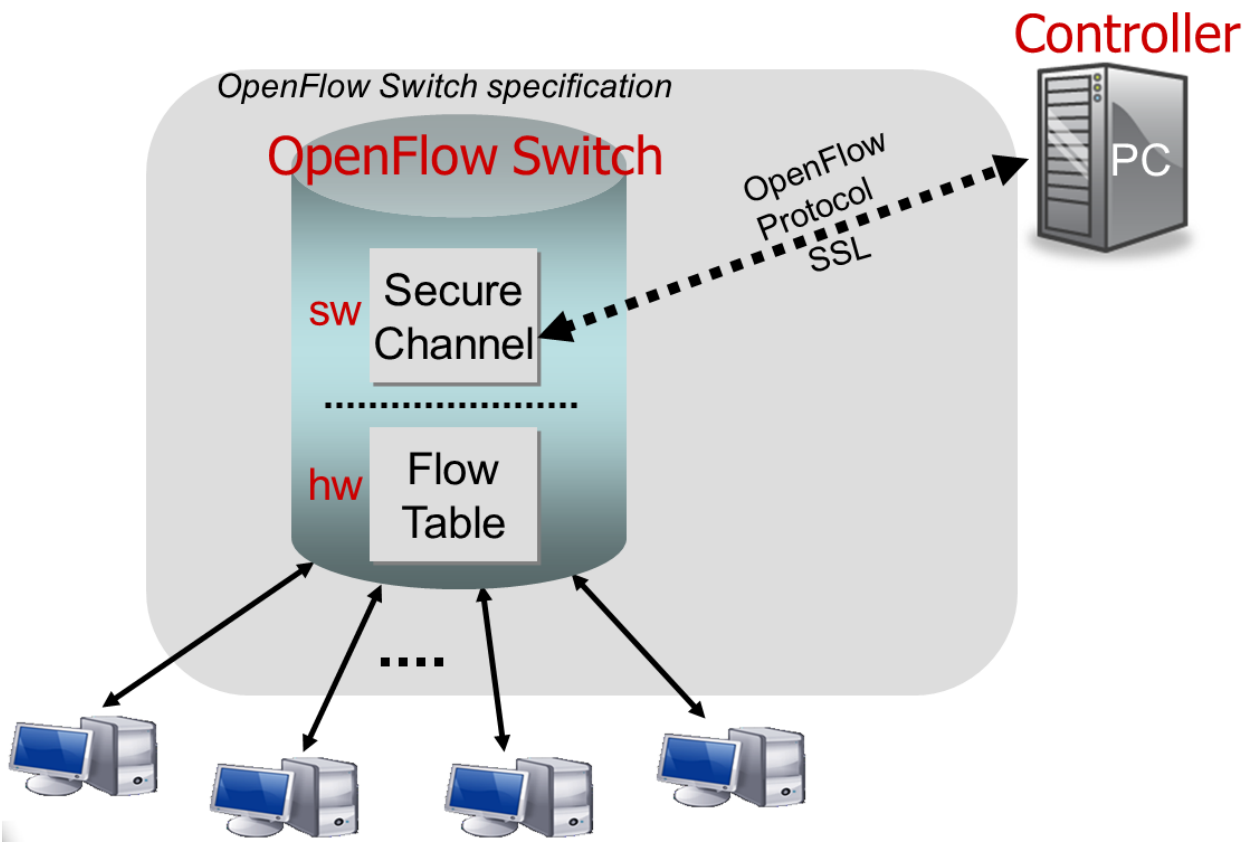

Image 2.4 OpenFlow Switch Datapath

An OpenFlow switch has one or more tables of packet-handling rules. Each rule matches a subset of traffic and performs certain actions on the traffic that matches a rule, actions include dropping, forwarding, or flooding. Depending on the rules installed by a controller application, an OpenFlow switch can behave like a router, switch, firewall, network address translator, or something in between.

The datapath of an OpenFlow Switch consists of a FlowTable, and an action associated with each flow entry. The set of actions supported by an OpenFlow Switch is extensible, but below we describe a minimum requirement for all switches. For high-performance and low-cost the data-path must have a carefully prescribed degree of flexibility. This means forgoing the ability to specify arbitrary handling of each packet and seeking a more limited, but still useful range of actions.

More specifically an OpenFlow switch consists of at least three parts:

1. A Flow Table with an action associated with each flow entry, telling the switch how to process the flow.
2. A Secure Channel that connects the switch to an SDN controller, allowing commands and packets to be sent between the controller and the switch.
3. The OpenFlow Protocol, which provides an open and standard way for a controller to communicate with a switch. By specifying a standard interface, through which entries in the Flow Table can be defined externally, the OpenFlow Switch avoids the need for experimenters to program the switch.

2.4 The contribution of SDN to the Internet of Things

The technology provided by the Software Defined Networking (SDN) offers flexibility and general programmability leading the evolution of the networking domain. The enormous benefits of the network control opens new ways by defining powerful and simple switching elements (forwarders) that can use any single field of a packet or message to determine the outgoing port to which it will be forwarded. Those advantages can be applied to the Internet of Things (IOT) exposing the ability of devices to connect to heterogeneous network and communicate to each other. This concept has imposed new complex requirements to both networking and Internet working schemes in current and future networks, specially the Internet. Therefore, networks must welcome heterogeneity, not just in devices but also in networking behavior and underlying protocols.

The traditional IP networks are often proposed as the solution for IoT,however it faces significant challenges. The main negative impact is that the objects and protocols have specific designs because they have to cover specific requirements and objectives, so forcing them to fit with a common and singular protocol is not a good option for most object designers. On the other hand, the Software Defined Networking approach encompasses the widespread programmability of network elements, both endpoints and intermediate. Contemplating the characteristics of SDN from the IoT perspective has led researchers to consider how SDN can be used to keep heterogeneity in networks and objects while building a bigger cooperation scheme by just integrating into the network.

With better network sharing in place, the number of IoT-enabled devices will increase, making the entire concept more attractive to more vendors as IoT becomes the norm.

3. THE EMPOWER PLATFORM

5G-EmPOWER builds upon a single platform consisting of general purpose hardware (x86) and operating system (Linux) in order to deliver three types of virtualized network resources: forwarding nodes (OpenFlow switches), packet processing nodes (Micro Servers), and radio processing nodes (WiFi Access Points or LTE eNodeBs).It was created by Dr. Roberto Riggio and his research team Future Networks (FuN) at CREATE-NET, who manage to keep it up to date with the latest software defined networking trends.

Although the OpenFlow standard offers a plethora of capabilities in the wired domain, including controllers, the open virtual switch and virtual slicing platforms, in the wireless domain it lacks proper support and documentation. That's where the EmPOWER platform comes in, expanding that domain with its flexible architecture and the high-level programming APIs that allow for fast prototyping of applications and services.

It is crucial to mention that the EmPOWER Platform was initially designed for the wired and wireless domain. Subsequently it expanded its capabilities to establish the experimentation of 5G that is an emerging concept, enhancing its scope by supporting the mobile domain and changing the name from EmPOWER to 5G-EmPOWER. Concluding to the admission that in this thesis, the terms EmPOWER and 5G-EmPOWER refer to the same platform.

3.1 EmPOWER Architecture

The system architecture (as shown in image 3.1), consists of a single Master and multiple Agents running on each Access Point (AP). The Master, implemented on top of an OpenFlow controller, has a global view of the network in terms of clients, flows, and infrastructure. The Agents allow multiple clients to be treated as a set of logically isolated clients connected to different ports of a switch. Network application run on top of the controller and can exploit either the embedded Floodlight REST interface or an intermediate interpreter (e.g. Pyretic). Each network application effectively runs in an isolated slice controlling all or just a subset of the available APs. The EmPOWER testbed is built from open and freely available toolkits like the OpenVSwitch and the Click Modular Router for the datapath and also Floodlight as the controller. Network applications, i.e. slices can either exploit the Floodlight REST interface or can be built on top of other SDN frameworks.

Considering the System's exploitation of a "logically centralized" architecture, developers are provided with a set of powerful programming abstractions to control the behavior of the network. Applications can, for example, register events associated with the actual network conditions and receive updates when such conditions change, e.g. a client moving away from an AP and closer to another. Such primitives can be used to devise and implement novel resource allocation and/or mobility management schemes without having to deal with all the WiFi–dependent implementation details, such as directly handling the IEEE 802.11 state machine or devising workarounds to the limitations of the IEEE 802.11 standard that do not allow the infrastructure to control clients' handovers.

The EmPOWER framework builds on a light virtual AP (LVAP) abstraction which decouples association/authentication from the physical connection between clients and AP. With LVAPs every client that tries to associate to the WLAN receives a unique BSSID, i.e. every client is given the illusion of having a dedicated AP. Similarly, each physical AP hosts an LVAP for each connected client. Therefore, migrating an LVAP between two physical APs, effectively results in client handover without requiring any re-association and re-authentication.

Finally, an Arduino add-on, Energino, is also provided, which allows measuring the energy consumption of an Access Point it is attached to. The measurement circuit is composed of a voltage sensor (based on a voltage divider), and a current sensor (based on the Hall effect). The statistics gathered are exported in a format compatible with the Internet of Things (IoT) platforms. Moreover, it allows the testbed administrator to power on and off any node in the network using an HTTP RESTful interface, acting as a "chassis manager".

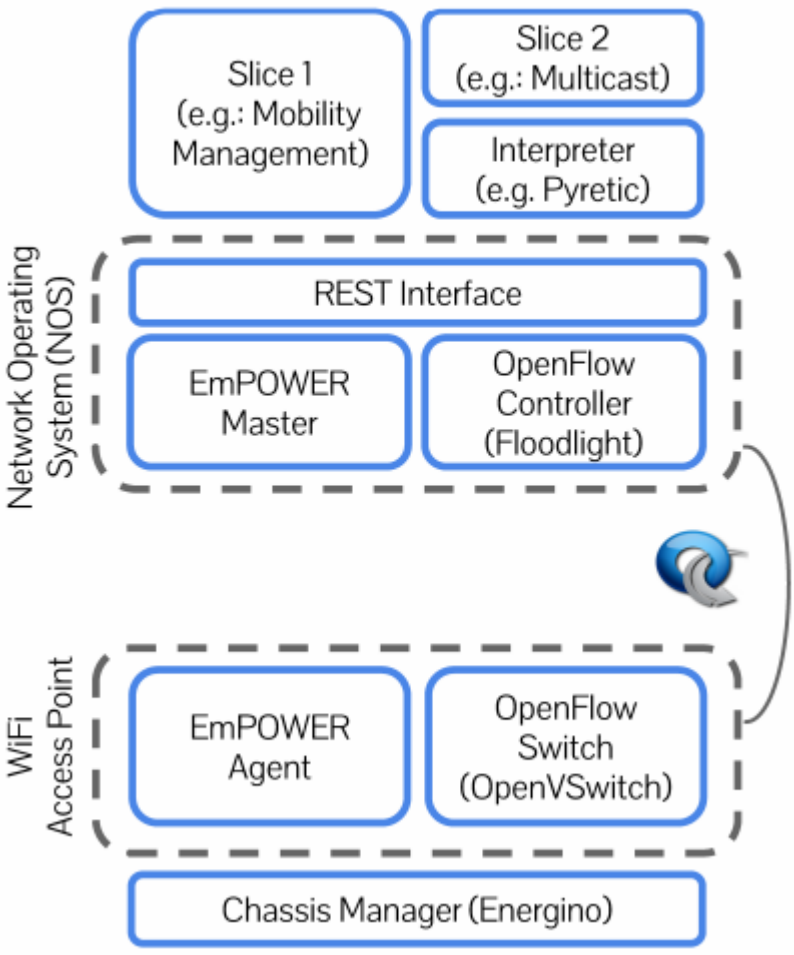

Image 3.1 Overview of the EmPOWER System Architecture

The EmPOWER testbed's network architecture (a brief overview is shown in figure 3.2), consists of programmable Access Point that are equipped with two Ethernet ports. One of them is connected to the control and management network. This allows experimenters collect network statistics and to perform administrative tasks without

affecting the actual user traffic that flows through the second Ethernet interfaces. VLANs are used at the switch in order to keep control and data traffic separated.

Each node is equipped with two Ethernet ports. One of them is connected to the control network allowing the controller to collect statistics without affecting the experiment. The second interface is connected to the OpenFlow switch and is used for running the actual experiment's traffic. Finally, another network collects the energy consumption statistics generated by the Energino devices. It is worth noticing that, unlike other WiFi testbeds, EmPOWER does not allow the experimenter to upload a custom OS on each AP but rather provides a set of APIs through which the experimenter can control the behavior of the AP from a centralized controller. The server runs the latest available software for Floodlight and FlowVisor. It is worth stressing that in the EmPOWER architecture new services and algorithms are deployed in the form of Network Applications on top of the Floodlight controller and exploiting its native REST interface. Each application is logically isolated from the others and has complete control over its slice, however physical level parameters such as the operating frequency for the hot-spot are not. Nevertheless, the application can control parameters such as Modulation and Coding Scheme and Transmission Power on a per–frame basis (if required by the experiment).

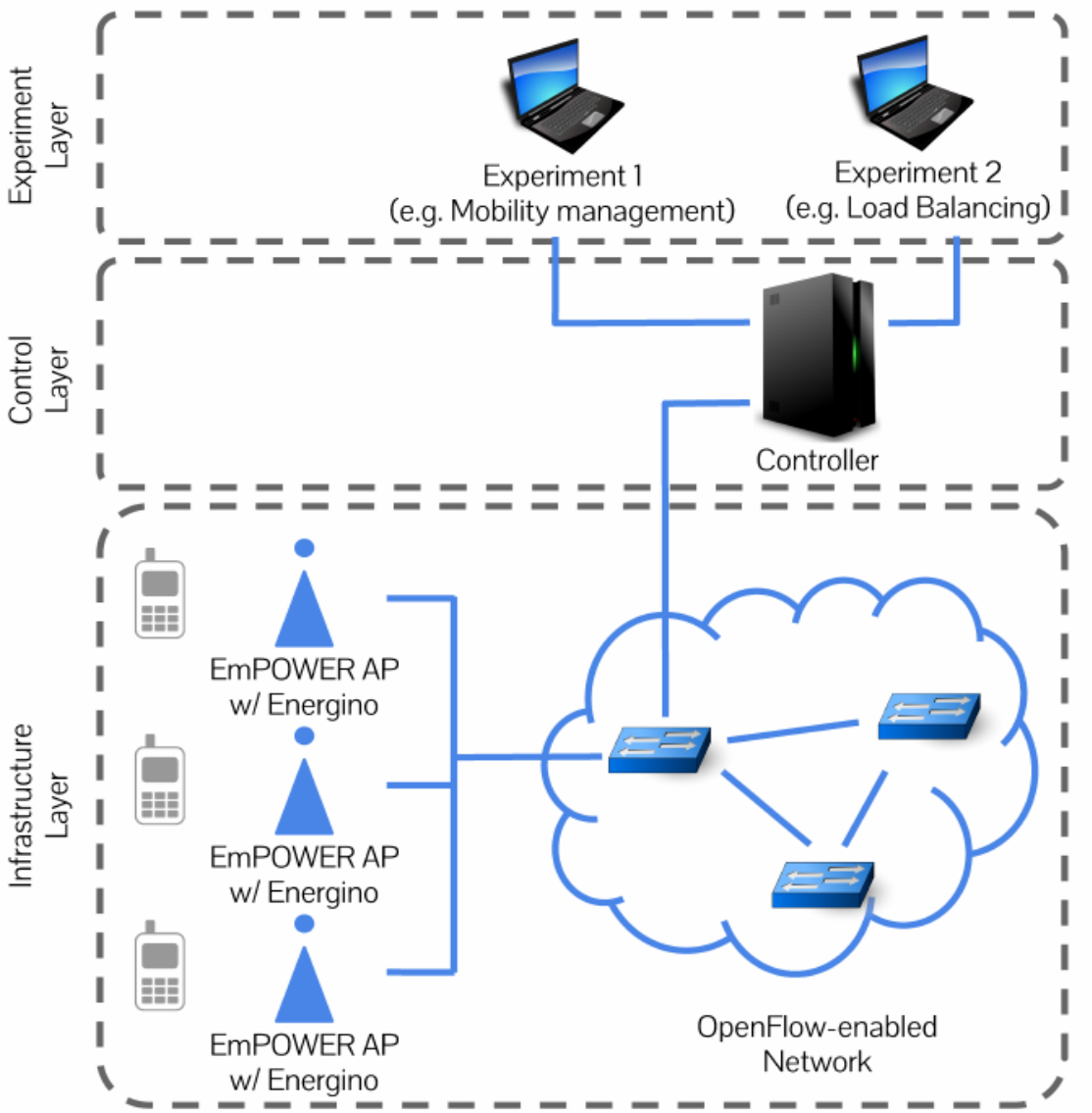

Image 3.2 EmPOWER Network Architecture

3.2 EmPOWER Components and Abstractions

The 5G-EmPOWER provides full visibility of the network state, by building a set of high-level programming abstractions that allow dynamic deployment and orchestration of network services by the experimenters. The platform consists from plenty components, modules and abstractions, but only the basic for this thesis will be described, in order to demonstrate the application that was developed.

Wireless Termination Points (WTP)

The Wireless Termination Points are the physical points of attachment in the Radio Access Network (RAN), providing clients with wireless connectivity. In a traditional network WTPs are similar to Access Points (APs), while in a LTE network with eNodeBs. The WTPs are connected to the Controller through a secure channel and belong in SDN's Data Plane.

Light Virtual Access Point (LVAP)

The Light Virtual Access Point (LVAP) abstraction allows developers to describe the desired state of the network leaving to the controller the task of implementing it. One LVAP is created for every station probing the network. Every time a LVAP connects to the network triggers an event. In more detail the LVAP sends a Probe request to the WTP it forwards the request to the controller which will trigger the creation of LVAP if its allowed. Moreover, every allowed LVAP is specified with a unique BSSID, on the other hand when a client is disconnected a de-association event will be triggered as result the LVAP will be removed from WTP and controller connected devices as well.

The LvapConnection Object

It represents a connection to a WTP using the LVAP Protocol. One LvapConnection object is created for every WTP in the network. Its function is to implement the logic for handling incoming messages. Some of them are the Authentication and Association Requests, "Hello" messages sent from the WTP in a loop used as a keepalive between the AC and the WTP and finally some handlers specifically for the purposes of this thesis.

3.3 EmPOWER Installation

Using the instructions provided from Dr. Roberto Riggio, the platform's wiki as well as the documentation by empower.create-net.org and the paper by the CREATE-NET team and Dr. Roberto Riggio. The installation of the empower to support our Thesis Use case was completed in 4 stages:

1. As the controller, one PC (virtual machine) with Debian 8.4 Jessie (Single Core 2.4 GHz CPU, 2048 MB RAM), in which Python 3.4.2 and all required libraries (python3- tornado (Version 4.2.1), python3-sqlalchemy (Version 1.0.8), python3-construct (Version 2.5.2), protobuf (Version 3.0.0), protobuf3-to-dict (Version 0.1.2)) were installed, in order to run the empower-runtime controller.

2. For the WTPs, two Soekris net5501 [1 net5501-60 (Single Core 433 MHz, 256 MB RAM), 1 net5501-70 (Single Core 500 MHz, 512 MB RAM)] were used in which the configured empower-openwrt-15.05 image was installed. This the network element providing clients with wireless connectivity, i.e. an Access Point in IEEE 802.11 terminology). Also, two 300Mbps Wireless N PCI TL-WN951N Adapters were used, one in each Soekris, as wireless interface.

3. An image is flashed for the WTP, the Empower-openwrt-15.05 image. It uses the OpenWRT embedded linux, as an agent the EmPOWER Lvap Agent which is based on the Click modular router and OpenVswitch. The agent has been configured in order to satisfy the purposes of the Thesis. In order to flash the agent, the hash-code inside the Makefile which points to the Agent's Code, is edited to the configured agent git repository.

4. Two Arduino Yun boards are used as clients based on the ATmega32u4 and the Atheros AR9331.The Atheros processor supports a Linux distribution based on OpenWRT named Linino OS. The board has built-in Ethernet and WiFi support, a USB-A port, micro-SD card slot, 20 digital input/output pins (7 of them can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and 3 reset buttons. The two Arduinos are our network's LVAPs. Each LVAP is connected to a WTP serving as client for exchanging packets over the network and each one has a different role in the thesis' use case. The Arduinos' Configuration is presented in Image 3.3 and 3.4.

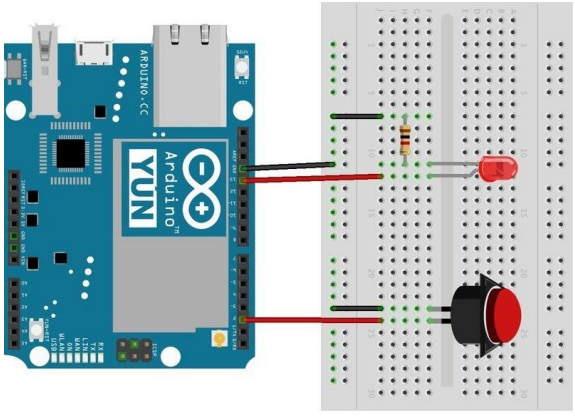

Image 3.3 Sender Arduino Setup

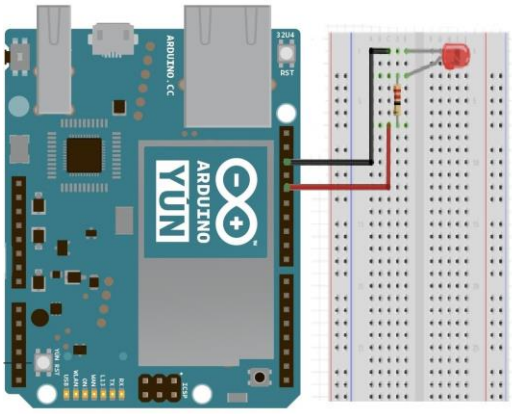

Image 3.4 Receiver Arduino Setup

The controller machine is connected to a LAN network via Ethernet where a router has the role of DHCP Server. On the same network the two WTPs are connected through Ethernet connections and belong to the same subnet of the controller in order to communicate with it. Between the controller and the router or the WTP and the router switch(es) may reside. This topology is presented below on image 3.5.

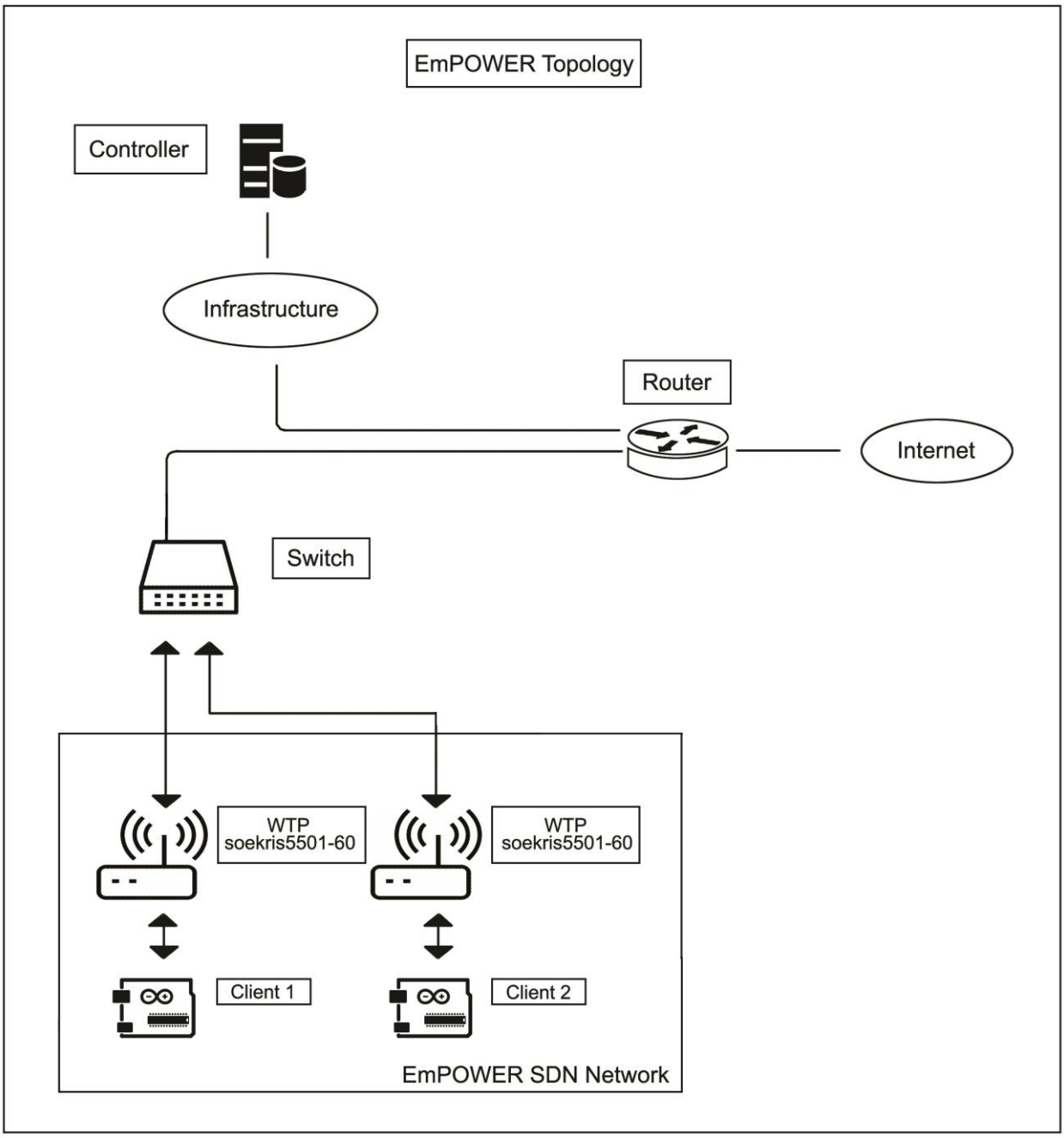

Image 3.5 SCAN Lab's EmPOWER Topology

After the installation and controller initialization, the controller can be accessed at sdn.scanlab.gr:8888

To access the controller in order to add Applications, LVAPs or WTPs you are presented with the form in image 3.6. There are two roles for users, admin and simple user. Given the correct user credentials the controller leads to a page where a tenant can be created or deleted.

Image 3.6 Login Form

Each tenant represents a slice of the virtual networks on top of the same hardware archiving separation of traffic flows into separate slices of the network. These subspaces, or slices, are managed by a controller and share network resources furthermore each slice has the illusion of its own distinct network resources. As a result, network experimentation and network security can be accomplished. For the thesis' use case one tenant has been created and used, named Loki. In Image 3.7 a form for viewing the tenants, add or remove them etc. is presented.

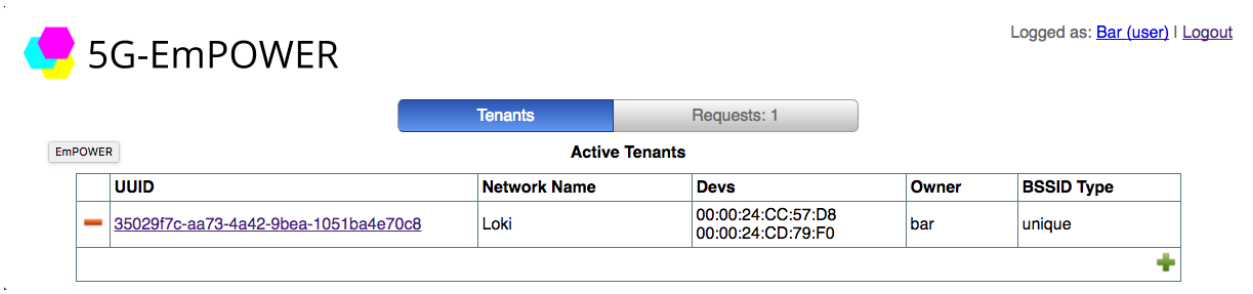

Image 3.7 Tenant Preview

Image 3.8 presents the form for adding LVAPs.

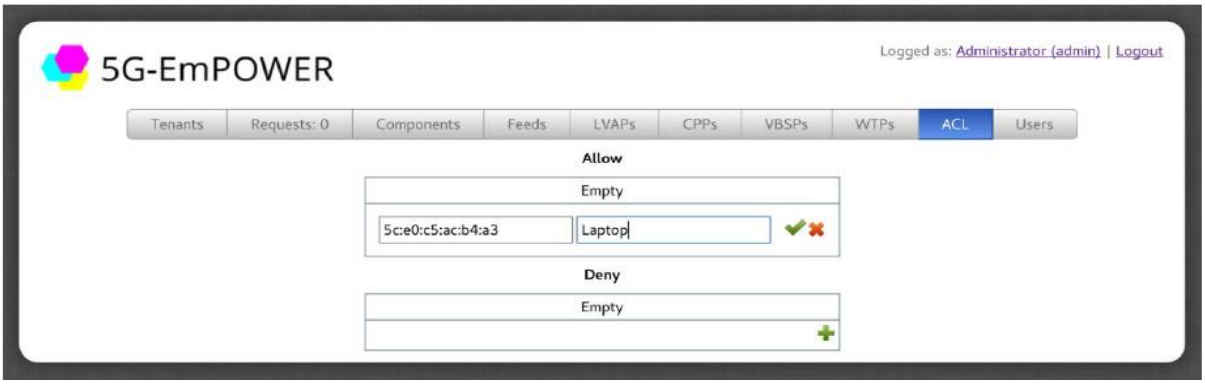

Image 3.8 Form for adding LVAPs

3.4 EmPOWER Agent

EmPOWER Access Points are based on the OpenWRT which is a Linux Distribution for embedded devices. Instead of trying to create a single static firmware, OpenWRT provides a fully writable filesystem with package management. This frees users from application selection and configuration provided by the vendor and allows you to customize the device through the use of packages to suit any application. For developer, OpenWRT is the framework to build an application without having to build a complete firmware around it. For users, this means the ability for full customization in order to use the device in ways never envisioned.

On top of the OpenWRT the Click Modular Router along with OpenVSwitch is built. Click is a software architecture for building flexible and configurable routers. A Click router is assembled from packet processing modules called elements. Individual elements implement simple router functions like packet classification, queueing, scheduling, and interfacing with network devices. A router configuration is a directed graph with elements at the vertices and packets flow along the edges of the graph. Several features make individual elements more powerful and complex configurations easier to write, including pull connections, which model packet flow driven by transmitting hardware devices, and flow-based router context, which helps an element locate other interesting elements.

4. THE LOKI APPLICATION

This network application tries to add a new functionality in the EmPOWER Platform. The application achieves the following goals:

- Easily configure and set Packet redirection rules for the clients' packets.
- Direct communication based on rules without leaving the data plane.
- Implementation of a packet flow manager for a Wireless SDN Network.
- Test new scenarios.
- Expand the capabilities of EmPOWER and generally Wireless Communications.

4.1 Application Overview

Loki constitutes a network application that gives the ability to the user to establish communication between the network's LVAPs and WTPs based on custom rules that configure the packet flow. In order to set those packet flow rules the user interacts with the application's frontend User Interface, which provides all the features and functionalities of the application. Furthermore, the Loki backend is responsible for receiving these rules, manage them accordingly and send them to the WTPs in order to be applied. Finally, the Network Slicing that the EmPOWER Platform provides, makes the application to affect only the slice where it is running rather than the whole network. A brief overview is given in Image 4.1.

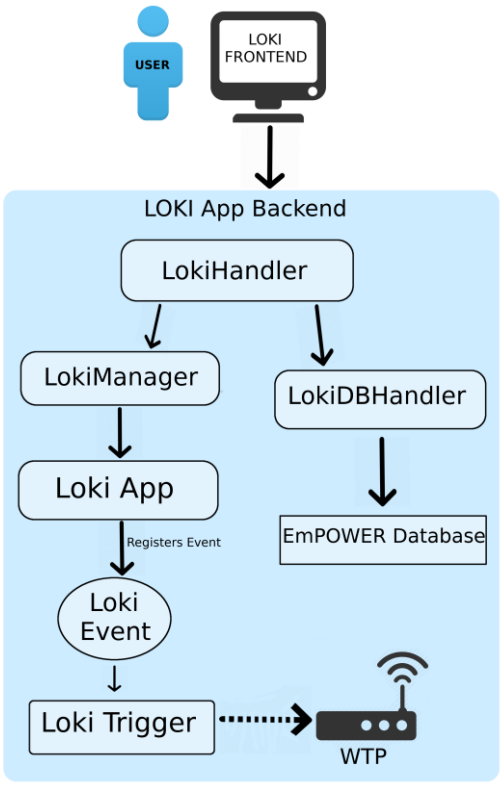

Image 4.1 Loki Application's Backend Overview

As the figure illustrates LOKI app consists from LOKIHandler, LokiManager, LokiDBHandler, LokiApp, LokiEvent, LokiTrigger. Finally, a packet is sent to the WTP (Openflow Protocol is used for the communication between the controller and the WTP), the WTP stores it to a hashtable and its applied accordingly.

1. When the User inserts the rules to the form given, an Ajax Call is sent. Then the LokiHandler gets the rules from the Ajax call in a json format and therefore calls the LokiManager(2) and the LokiDBHandler(3). The handler extends the class EmpowerAPIHandlerUsers which is a handler class provided from the EmPOWER REST API for resolving user request. The handler class implements two basic methods GET and POST. The front-end sends Post Request when a rule is inserted and Get when it wants to show various information to the User.

2. The LokiManager stands between the Loki App and LokiHanlder. It consists from various functions whose main purpose is to modify those rules given so the LokiTrigger can easily access them, signal that a LokiEvent is Ready to be registered while it also keeps various information about LVAPs and WTPs.

3. The LokiDBHandler is a set of functions responsible for storing and handling the rules in the EmPOWER Data Base for further processing. The Loki Rules are stored in their own tables in the EmPOWER database and only LokiDBHandler modifies them. There are actually 3 tables in the Database, one for the explicit Rules, one for the Group Rules and finally one that contains all the LVAPs of the Group Rules Table.

4. The LokiApp is a class that contains the main body of the whole LOKI application. The LokiApp constantly communicates with LokiManager in order to see if it has any new rule, if there is the LokiApp is responsible for registering a LokiEvent. LokiApp also collects various data and gives it to LokiManager in order to be temporarily stored.

5. The LokiTrigger is a class responsible for catching that event, in order to send the rule to the WTP. When LokiTrigger catches that event, it handles the rule accordingly, deciding to which WTP to send each rule based on its parameters. The rule is formatted in a C Struct, with the help of Python Construct and is later sent to the WTP, while also the WTP sends back an answer as a confirmation. The Openflow Protocol implemented in the EmPOWER Platform is Responsible for the communication between the WTP and the Controller.

When the agent receives a packet, it forwards it to through his elements as is the Click Modular Router's architecture. Each element affects the incoming packet in a unique way such as collecting stats, header filtering and editing, sending response association messages back to the sender and many more, for the thesis purposes further changes have been to some of these elements. Before each packet is sent to the WTP, a unique code for its packet type is assigned, so the Agent knows how to handle it. Therefore, in order for the WTP to be able to receive the packets sent from LOKI, custom functions, packet structures and a custom element were made in the EmPOWER LVAP Agent. The element called lokiTrigger (.cc and. hh) is the class of the trigger. Functions were made in an element provided by the EmPOWER Agent, the lvapmanager. The handle_add_rule_trigger handles an add rule packet, it unparses it and stores the rule's source and destination MAC addresses in a flow hashtable (key: source, value: destination). Moreover, the handle_remove_rule_trigger handles a remove rule packet and erases the entry from the flow HashTable. Finally, the packet's structure is a C struct defined in the empowerpacket element.

Now let's assume the WTP has been given LOKI packet flow rules from the controller and has clients connected to it. When a packet arrives from a client (already authenticated and associated), as it is being forwarded through the elements, it passes through the decapsulation process, where the packet's encapsulated data is unpacked. A custom decapsulation element (empowerWifiDecapsulation) is provided from EmPOWER. There, a condition checks if a flow rule for the packet sender (source's ethernet address), exists in the flow Table, if it does the packet's destination MAC Address is replaced with the one that the flow Table suggests, if it doesn't, nothing changes. Finally, either way, the packet is being forwarded normally to each element it is supposed to, based on the Click configuration.

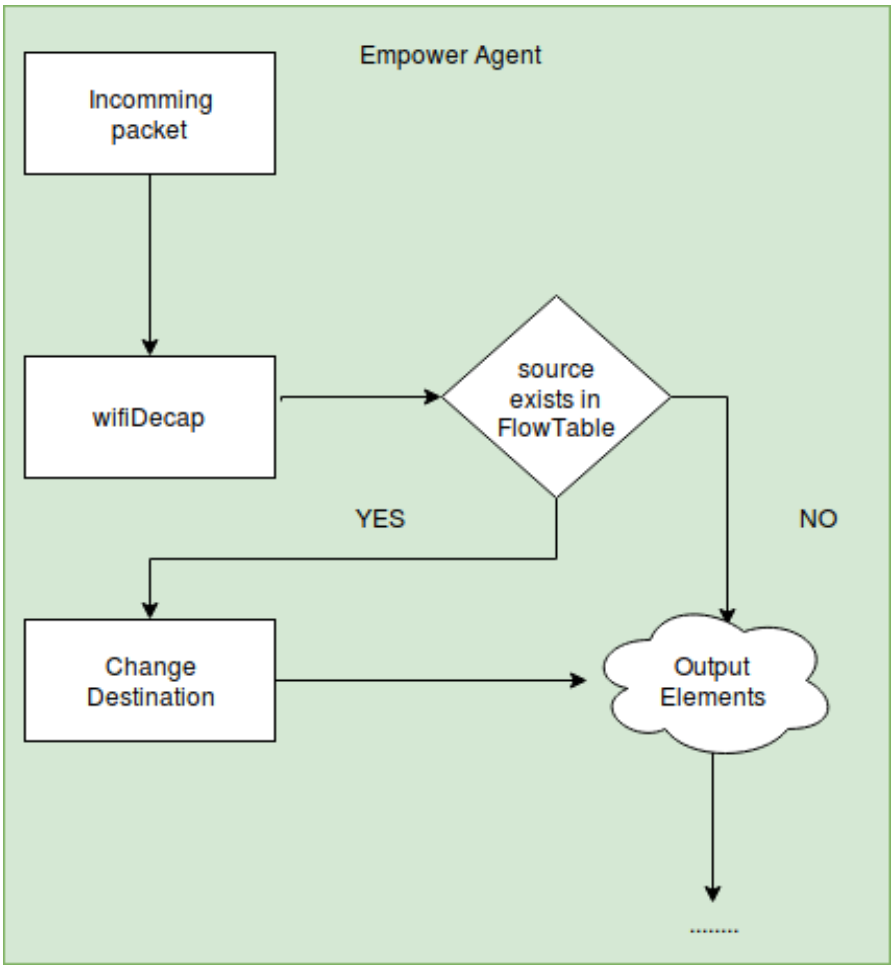

Image 4.2 Configured EmPOWER Agent Synopsis

Finally, the components tab is responsible for the deployment of EmPOWER's modules. This is where the Loki application can be added, as shown in image 4.3. In this point is important to mention that the application is running only on the Loki tenant. Due to the slicing that EmPOWER offers, the experiments will not intervene with the experiments on other tenants.

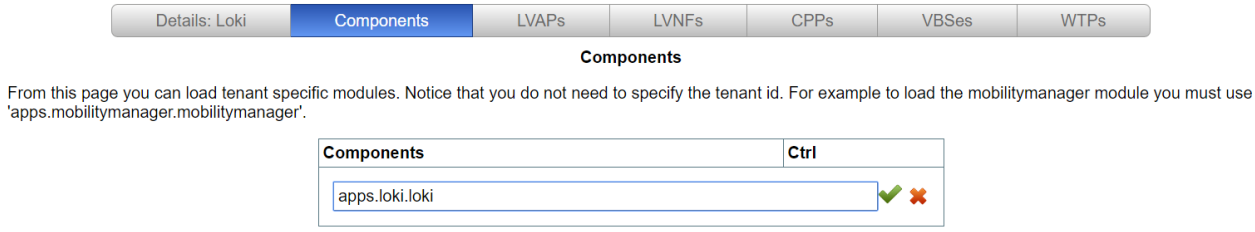

Image 4.3 Adding Loki Application in User's Component Page

After the application has been added, by clicking on the component Loki's front-end is presented, depicted in Image 4.4. There the user can start his interaction with the application. The topbar's tabs provide information about the available LVAPs and WTPs, while the main body's tabs contain the basic functionalities of the application.

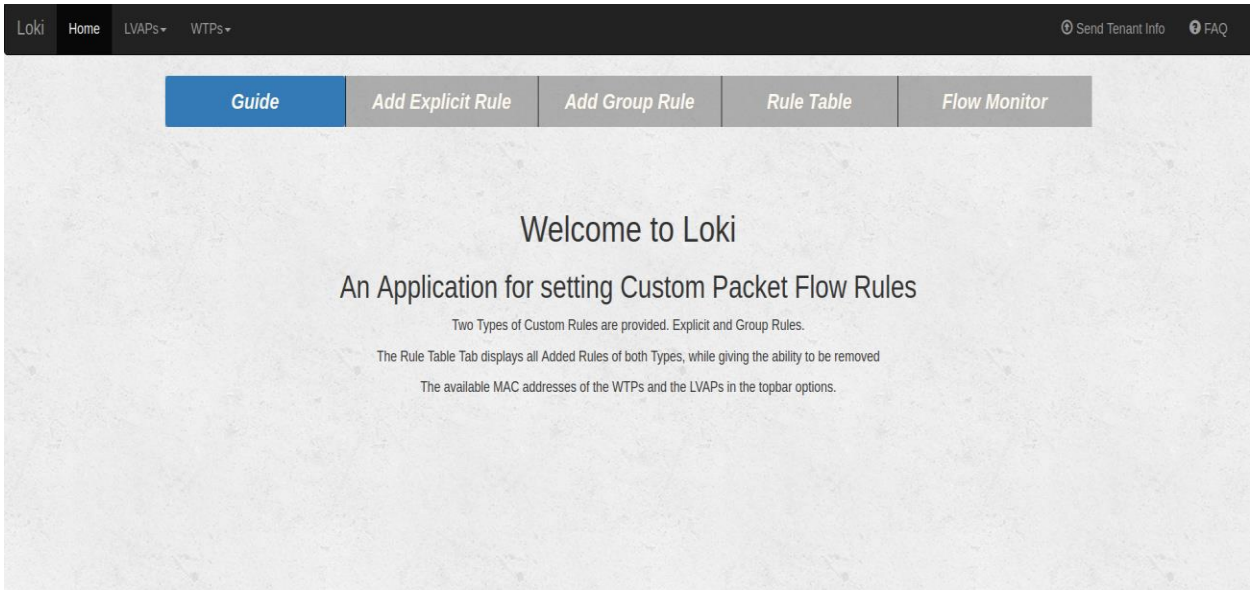

Image 4.4 Loki Front-End Welcome Page

4.2 Scenarios and Use Cases

Besides the back-end functionality, the user's convenience was also a priority while building the application. As mentioned, the front-end is designed to be user-friendly and practical, thus providing the user with an easy to use interface for applying the rules, a better visualization of the existing rules and an asynchronously updated table of the packet flow traffic based on the user's rules.

The basic idea behind the developing of the application was to make client to client communication faster, more direct and more manageable. As mentioned before, the clients we use for our purposes are Arduino Yuns, each connected wirelessly to a different WTP. The Sender client sends a layer2/layer3 packet with a dummy destination IP to the WTP it is connected to. After that, when the packet arrives in the WTP, the process depicted in Image 4.2 takes place, setting the packet flow and therefore redirecting the packet if a Loki flow rule is applied.

Loki Application Main Functionalities Are:

1. Add a Group Rule
2. Set an Explicit Rule
3. View and remove the Rules
4. View an asynchronously updated

Smart City Scenario – Emergency Signaling over SDN: Group Rule Implementation

Considering a city where an SDN Network is set, using the EmPOWER Platform, many WTPs would be installed as Access Points. On top of that, in every apartment an Arduino Yun with an Emergency Sensor (e.g. Fire Detection Sensor) would be built, acting as a client, abstracting each apartment as an LVAP of the city network.

The System Administrator could benefit from the Loki Application by taking advantage of the Group Rule functionality it provides. Considering the scenario that every apartment has an Arduino client (LVAP) with a Fire Detection Sensor, connected to any WTP, the administrator could set a Loki Rule, setting the Packet Flow of those LVAPs and redirect the packets to the City's Fire Station LVAP, thus, saving time by keeping the packet in the Data Pane.

The process is simple and consists of three small steps. To begin with, by clicking on the Group Rule Tab the user is presented with the form that consists of three fields, as shown in Image 4.5.

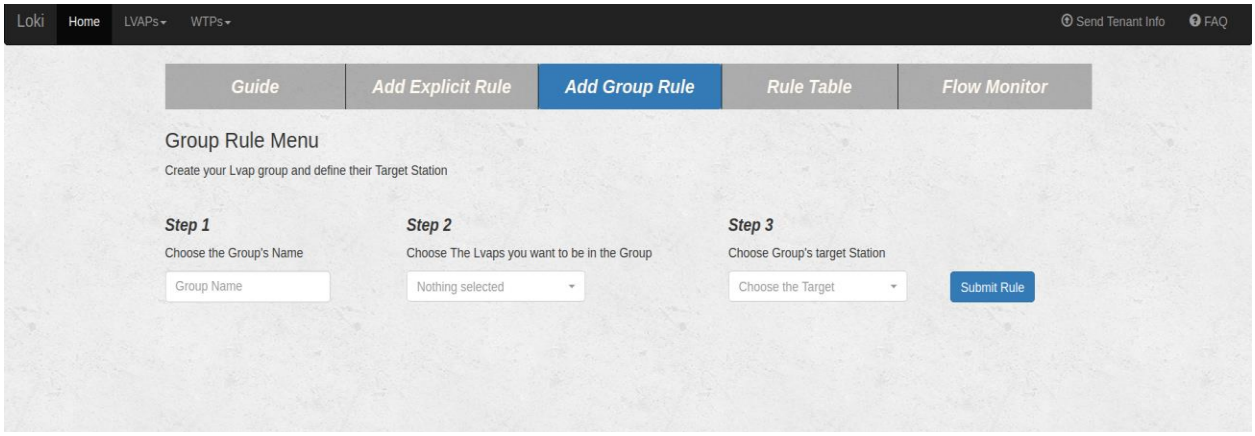

Image 4.5 The Group Rule Tab

The first field is the Group's Name, setting this is necessary, as it is the identification of each rule and helps the user to locate it in the table. The second field is a dropdown form with multiple available choices. In this scenario, the administrator can browse and choose the available LVAPs (i.e. each Apartment) that constitute the Group's LVAPs. Moreover, the dropdown provides an overview of the network's Active LVAPs as well as an input search bar, it is depicted in Image 4.6.

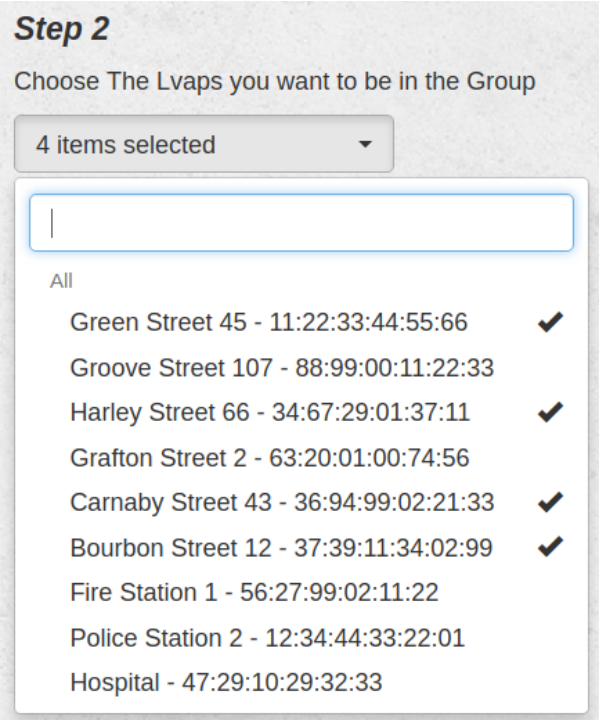

Image 4.6 Group's LVAPs Choice

Finally, the third field requires the Group's Target LVAP. In this scenario, the administrator can locate and choose the destination Service for the Group's LVAPs, thus, the Fire Station Service, as depicted in Image 4.7.

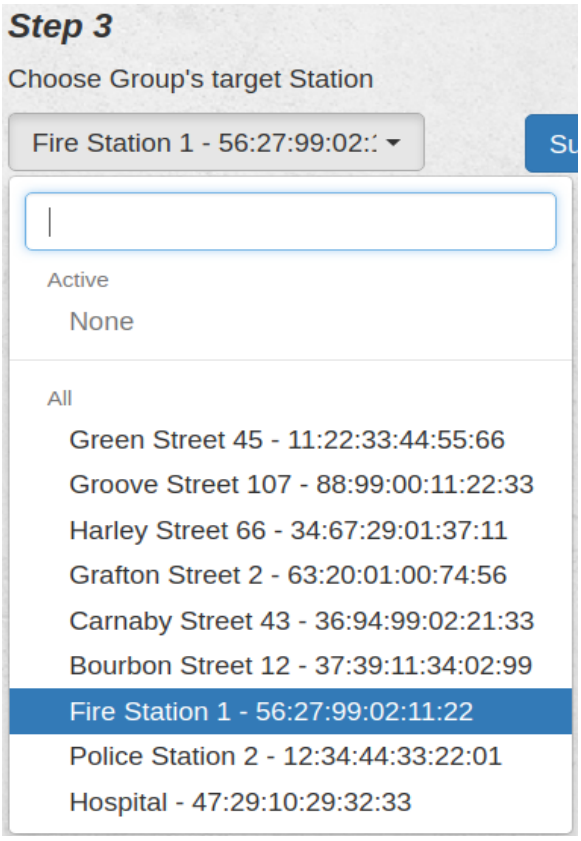

Image 4.7 Group's Target Station Choice

When the Rule is set, the user can locate and remove it in the Rule Table tab as shown in Image 4.8

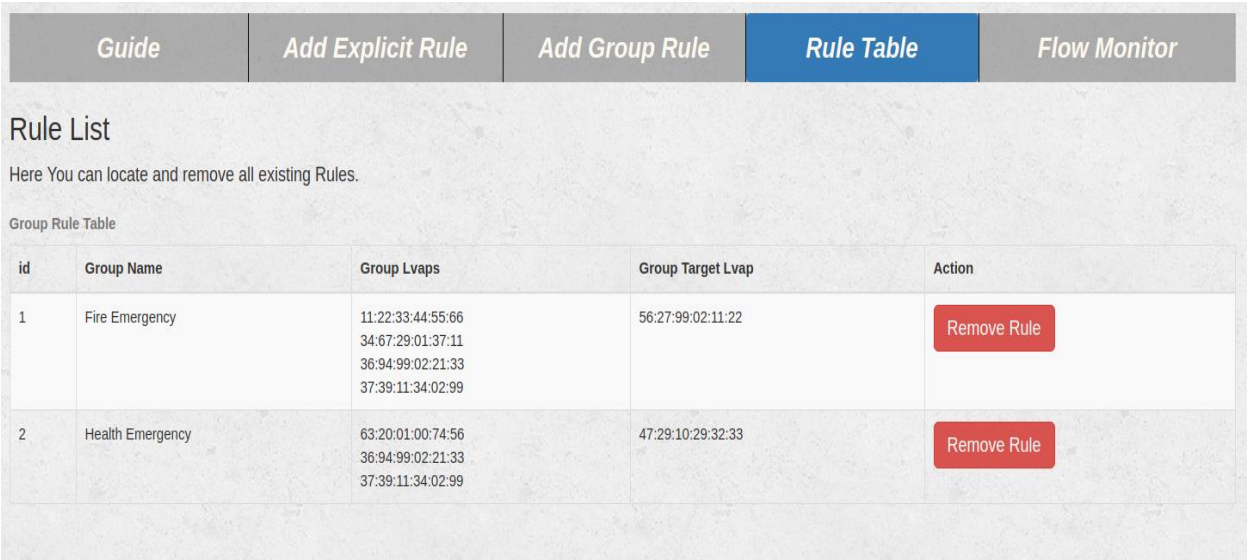

Image 4.8 Application Rules' Overview

There are several benefits the application contributes in this scenario. This is achieved by providing a user-friendly and easy to use interface for applying the rules, giving an overview of the LVAPs and the Group Rules given, the ability to remove them, as well as, a functional back-end for applying those rules to the WTPs.

On SCAN Lab's Equipment, the scenario is simulated. The Sender Arduino Yun (Image 3.3) acts as the apartment's Arduino client, with the button simulating the fire detector, as well as, the receiver Arduino Yun (Image 3.4) simulating the Fire Station.

Smart Home Scenario – Convenience Services over SDN: Explicit Rule

Considering a 2-floor apartment with a WTP on the main entrance and each floor with its own WTP, all connected with a switch. An Arduino Yun client with a fingerprint recognition is installed as an LVAP in the entrance which identifies the home owner's fingerprint. Various smart devices are abstracted as LVAPS connected in the WTPs inside the house as well (e.g. anti-theft alarm, smart heating, Television etc.). This home's owner could benefit from the Loki application by setting an explicit rule to the EmPOWER Network.

Assuming the user wants to open enable the Alarm after leaving home by scanning his finger on the fingerprint sensor in the entrance. The explicit rule gives him the ability to do so, by providing a form for setting the Source LVAP (Arduino client with fingerprint recognition), the source WTP (Entrance WTP), Type (Alarm Set), Destination WTP (1st Floor WTP) and Source LVAP (Smart Alarm System).

First thing the user is presented by opening the explicit rule tab is depicted in image 4.9.

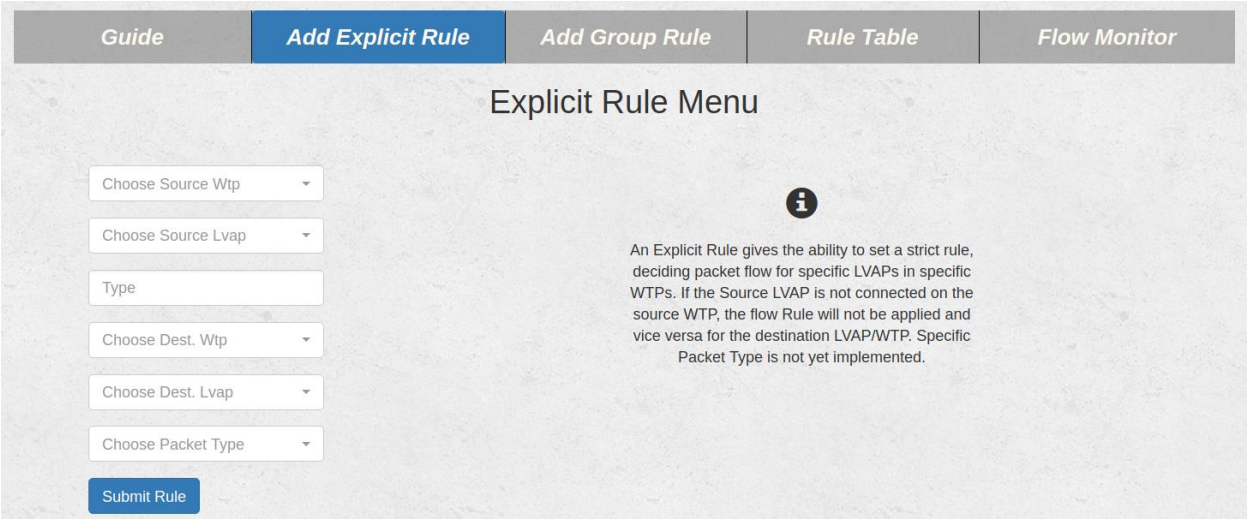

Image 4.9 Explicit Rule Form

The user defined LVAPs can be set in the administrator's page on the WTPs tab, as shown in Image 4.10.

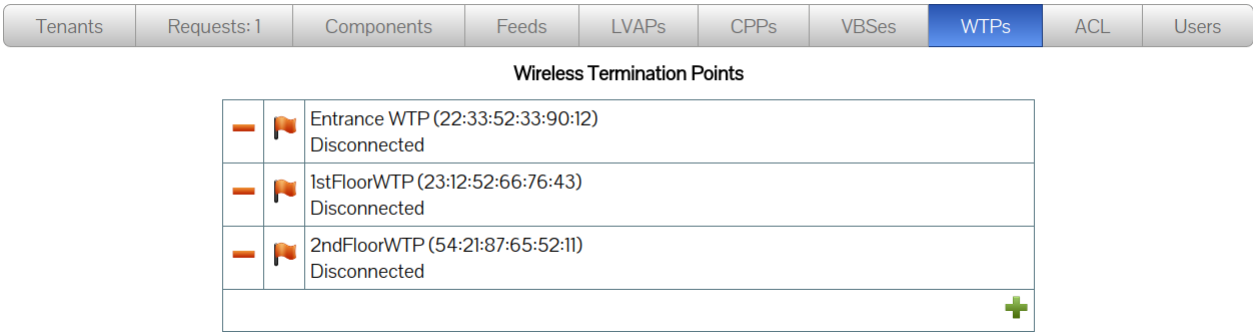

Image 4.10 WTPs Table

In order to set the WTPs and the LVAPs for the rule the dropdown menus depicted in Image 4.11 and 4.12 are provided.

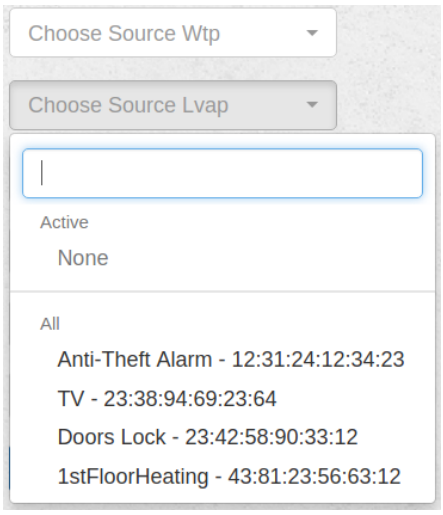

Image 4.11 Choosing the LVAPs

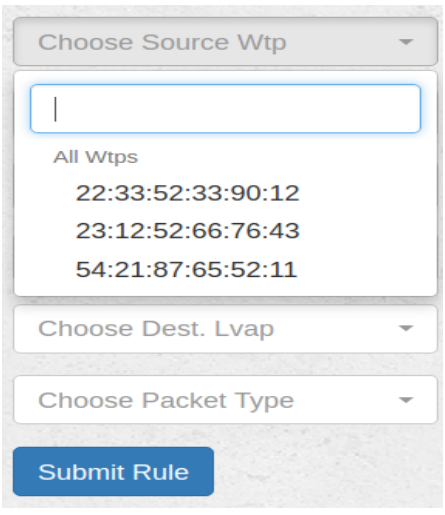

Image 4.12 Choosing the WTPs

Finally, a preview of the rules, as well as, a Remove Rule option, is provided in the Rule Table tab (Image 4.13).

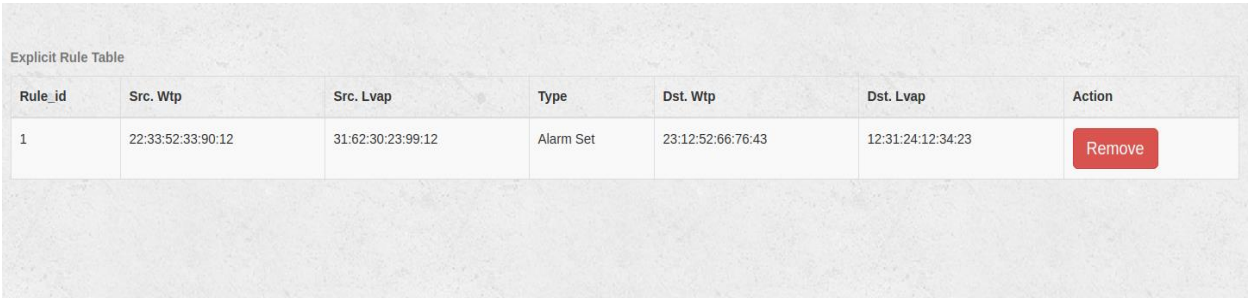

Image 4.13 Explicit Rules Table

It is worth mentioning that the after the Rules are applied, they are stored in the EmPOWER database, therefore, even after a system's shutdown, they still exist.

4.3 Technologies and tools used for the implementation

It is necessary to highlight the tools and technologies used for the controller's and EmPOWER Agent's implementation. Starting from the frontend and ending with the backend the following technologies were used.

Hyper Text Markup Language (HTML)

HTML is considered the standard markup language used to create web pages and web applications. HTML consists of a set of markup symbols inserted in a file in order to be displayed on a World Wide Web browser page.

Cascading Style Sheets (CSS)

CSS is a style sheet language used for describing the presentation of a document written in a markup language, such as HTML. CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colors, and fonts.

Bootstrap 3

Twitter Bootstrap is a free and open-source front-end web framework for designing websites and web applications. It contains a range of HTML and CSS design templates for many interface components such as forms, buttons, navigations, modals and other, as well as optional Javascript extensions.

Javascript

Javascript is a high-level, dynamic interpreted programming language commonly usedin web development. It is a client-side scripting language, which means that the code runs on the client's browser.

JQuery Library

jQuery is a cross-platform JavaScript library designed to simplify the client-side scripting. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library.

Asynchronous Javascript and XML (Ajax)

Ajax is a set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications. With Ajax, Web applications can send data to and retrieve from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

SQLAlchemy

SQLAlchemy is an open source SQL toolkit and object-relational mapper (ORM) for the Python programming language. SQLAlchemy provides a full suite of well-known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language. SQLAlchemy highlights that SQL databases behave less and less like object collections the more size and performance start to matter, while object collections behave less and less like tables and rows the more abstraction starts to matter.

Python

Python is a high-level level, interpreted and dynamic programming language. It enables the users to express concepts in a few lines of code, unlike other programming languages such as C, C++ or Java. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. According to this Empower platform is written in Python offering to the developer multiple capabilities.

Tornado

Tornado is a Python web framework and asynchronous networking library. The usage of non-blocking network I/O makes it ideal for application with long polling periods as it can support many open connections. This framework provides a REST interface for the EmPOWER platform. As a result, the application takes advantage of it in order to handle the Ajax calls generated from the frontend.

For the empower-agent implementation:

C++

C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation. It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights.

Github

GitHub is a web-based Git or version control repository and Internet hosting service. It offers all of the distributed version control and source code management functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

Click Modular Router

Click is a software architecture for building flexible and configurable routers. A Click router is assembled from packet processing modules called elements. Individual elements implement simple router functions like packet classification, queueing, scheduling, and interfacing with network devices. A router configuration is a directed graph with elements at the vertices; packets flow along the edges of the graph. Several features make individual elements more powerful and complex configurations easier to write, including pull connections, which model packet flow driven by transmitting hardware devices, and flow-based router context, which helps an element locate other interesting elements.

5. CONCLUSIONS AND FUTURE EXPANDABILITY

In this thesis, a further experimentation with Wireless SDN was achieved, using the EmPOWER Platform's capabilities. The purpose of this, was to better conceptualize the Packet Flow of wirelessly connected clients, in order to make it more manageable. SDN is a pragmatic compromise that allows researchers to run their experiments adding further functionality to the future network.

Unlike other SDN platforms the EmPOWER contributes to the evolution of wireless networks providing a new scheme for the Northbound API and acts as a network operating system with a REST API besides that the creation of new application based on this model becomes easier due the open source platform.

Loki application was created to support real time events preventing accidents and informing clients faster than in traditional networks. On the other hand, the administrator and users have better view of traffic flow on their network in order to interact and configure it as they please.

As for the future work, a plethora of further expansions in the application has been scheduled, in order to support all kind of wireless transmitting packets such as TCP and UDP, achieving faster wireless transmission of video data as well. Furthermore, each packet's payload could contain a "client type", in order to categorize each client in a different client group. Finally, packet multicast is scheduled to be implemented in the near future.

In conclusion EmPOWER is a remarkable platform, providing experimentation on the topic of wireless SDN and its continuous support and evolution broadening experiments horizons.

ABBREVIATIONS – ACRONYMS

5G	5th generation of mobile networks
A-CPI	Application-Controller Plane Interface
Ajax	Asynchronous Javascript and Xml
AP	Access Point
API	Application Programming Interface
BSSID	Basic Service Set Identifier
CPP	Click Packet Processor
CSS	Cascading Style Sheets
D-CPI	Data-Controller Plane Interface
HTML	Hyper Text Markup Language
IoT	Internet of Things
IP	Internet Protocol
L2/L3	Layer 2/Layer 3 Frame
LVAP	Light Virtual Access Point
MAC	Media Access Control
NBI	North Bound Interface
NE	Network Element
NFV	Network Function Virtualization
NOS	Network Operating System
OSI	Open System Interconnection model
OSS	Operation Support System
QoS	Quality of Service
RDB	Resource Data Base
REST	Representational State Transfer
SDN	Software Defined Network
WLAN	Wireless Local Area Network
WTP	Wireless Termination Point

ANNEX I

The EmPOWER Platform can be found: https://github.com/5g-empower

Following the instruction of the: https://github.com/5g-empower/5g-empower.github.io/wiki

For the **Controller**, the instructions from https://github.com/5g-empower/5g-empower.github.io/wiki/Setting-up-the-Controller were followed.

First, the Controller installation on the Debian based system requires the following packages installed in the system:

- python3-tornado (Version 4.2.1)
- python3-sqlalchemy (Version 1.0.8)
- python3-construct (Version 2.5.2)
- protobuf (Version 3.0.0)
- protobuf3-to-dict (Version 0.1.2)

The platform can be downloaded from GitHub. The terminal command is: git clone https://github.com/5g-empower/empower-runtime.git

The EmPOWER WLAN controller must be executed from a central server that can be reached from all wireless APs. From the main repository enter the controller directory: cd empower-runtime

In order to create a directory for the database named deploy, type the command: mkdir deploy

For the Loki Application installation in the empower-runtime directory, run the loki_setup.sh script, given the following installation options:

./loki_setup.sh -i server: to install the app to the specified server

./loki_setup.sh -r server: to uninstall the app from the specified server

Finally, start the controller with: python empower-runtime.py

To prepare the **WTP** we followed the instructions from https://github.com/5g-empower/5g-empower.github.io/wiki/Setting-up-the-WTP

First, on the Debian based system the following packages must be installed:

- ncurses (Version 6 or above)
- zlib (Version 1.2 or above)
- openssl library (Version 1.0 or above)
- GNU awk (Version 4.1 or above)

Now the empower-openwrt must be cloned from LokiNetworks repository: git clone https://github.com/LokiNetworks/empower-openwrt-15.05.git

Then, to install the feeds:
cd empower-openwrt-15.05

./scripts/feeds update -a
./scripts/feeds install -a

EmPOWER WTPs need to be connected to the Controller for all their operations. Therefore, it is recommended to reserve an IP address for the Controller on the DHCP server. You can refer to the documentation of your router in order to learn how to reserve static IP addresses. To avoid having to configure manually all access points, you can automatically create an image with the configuration that suits your needs (note, the following configuration is for a PCEngines ALIX 2D board equipped with a single Wireless NIC). In order to do so, create a directory named "files":

mkdir files

The OpenWRT buildroot will copy all the contents of the "files" directory to the files image. Create a "etc/config" directory:

mkdir -p files/etc/config

Then create the three following files: (screenshots from empower site)

1) EmPOWER Configuration (/etc/config/empower)

```
config empower general
    option "master_ip" "IP ADDRESS OF YOUR CONTROLLER ASSIGNED USING STATIC DHCP"
    option "master_port" "4433"
    option "network" "wan"
    list "ifname" "empower0"
    list "debugfs" "/sys/kernel/debug/ieee80211/phy0/ath9k/bssid_extra"
```

2) Wireless configuration (/etc/config/network)

```
config interface 'loopback'
    option ifname 'lo'
    option proto 'static'
    option ipaddr '127.0.0.1'
    option netmask '255.0.0.0'

config device
    option name 'br-ovs'
    option type 'ovs'
    list ifname 'eth0'

config interface 'wan'
    option ifname 'br-ovs'
    option proto 'dhcp'

config interface lan
    option ifname   eth1
    option proto    static
    option ipaddr   192.168.250.1
    option netmask  255.255.255.0
```

3) Wireless configuration (/etc/config/wireless)

```
config wifi-device  radio0
        option type     mac80211
        option channel  36
        option hwmode   11a
        option path     'pci0000:00/0000:00:0c.0'

config wifi-iface empower0
        option device   radio0
        option mode     monitor
        option ifname   moni0

config wifi-device  radio1
        option type     mac80211
        option channel  6
        option hwmode   11g
        option path     'pci0000:00/0000:00:0e.0'

config wifi-iface empower1
        option device   radio1
        option mode     monitor
        option ifname   moni1
```

Run the configuration application:
make menuconfig

From the menu select the Target System (e.g. x86) and the Subtarget (e.g. AMD Geode based systems). Then select "Network -> empower-agent" and "Network -> openvswitch". You may also want to compile the LuCI web interface by selecting "LuCI -> Collections -> luci". Save the configuration and exit, then start the compilation. If you have a multi core machine you can increase the compilation speed by increasing the number of parallel builds with the "-j" option.

make -j N (number of system cores)

Once the compilation is done, the compiled image can be found in the bin/directory. For example, in the case of PCEngines Alix platform, the image will be: bin/x86/openwrt-x86-geode-combined-squashfs.img

In the case of the PCEngines Alix board you need to insert the Compact Flash card in a compact flash reader attached to your laptop and then run:
dd if=./bin/x86/openwrt-x86-geode-combined-squashfs.img of=/dev/sdb

Note this assumes that the compact flash device is "/dev/sdb".

Finally, to connect the Arduino Yuns as clients just follow the previous described implementation, plug them in the plug in and they are ready to go. Their code can be found at: https://github.com/LokiNetworks/arduino.git

REFERENCES

[1] Nick McKeown et al., OpenFlow: Enabling Innovation in Campus Networks, March 2008 - http://archive.openflow.org/documents/openflow-wp-latest.pdf

[2] Wireless Software Defined Networks: Challenges and Opportunities https://www.researchgate.netpublication/261021392_Wireless_Software_Defined_Networks_Challenges_and_opportunities

[3]. http://empower.create-net.org/

[4] Riggio, R., T. Rasheed, and F. Granelli, "EmPOWER: A Testbed for Network Function Virtualization Research and Experimentation", IEEE SDN4FNS 2013 (Software Defined Networks for Future Networks and Services), November 2013, Conference Paper.

[5] Riggio, R., T. Rasheed, and M. Marina, "Interference Management in Software-Defined Mobile Networks", IFIP/IEEE Integrated Network Management Symposium (IM 2015), Ottawa, May 2015, Conference Paper.

[6] Open Networking Foundation, SDN architecture Issue 1, ONF TR-502, June 2014.

[7] Adam Drescher, “A Survey of Software-Defined Wireless Networks”, April 2014, http://www.cse.wustl.edu/~jain/cse574-14/ftp/sdwn.pdf

[8] The Click Modular Router, February 2001,

https://pdos.csail.mit.edu/papers/click:tocs00/paper.pdf

[9] Understanding the SDN Architecture, sdxcentral,

https://www.sdxcentral.com/sdn/definitions/inside-sdn-architecture/

[10] A General SDN-based IoT Framework with NVF Implementation, Jie Li, Eitan Altman, Corinne Touat , https://hal.inria.fr/hal-01197042/document

[11] Open Networking Foundation, Software-Defined Networking: The New Norm for Networks, ONF White Paper, April 2012