Software Defined Networking en redes Inalámbricas

Ari Chamlian[1], Gastón Telfeyan[1] y Nicolás Piquerez[1]

[1]Facultad de Ingeniería, Universidad de la República

Diciembre 2016

Resumen

La situación actual de crecimiento exponencial de las redes inalámbricas WiFi, está generando grandes cambios en el paradigma existente debido a las nuevas necesidades y problemas que se deben enfrentar. La incorporación de nuevas funcionalidades y servicios a dichas redes está condicionada a varios factores y las reglas de mercado juegan un rol fundamental. Bajo estas circunstancias es que surgen disciplinas como las Redes Definidas por Software -SDN por sus siglas en inglés- y la Virtualización de Funciones de Red -NFV por sus siglas en inglés- que comienzan a pisar fuerte intentando tomar un rol en lo que se presenta como una nueva forma de concebir las redes informáticas, distribuyendo capacidades de cómputo de forma distinta a la existente, incorporando herramientas de software, creando entidades centralizadas para el manejo del Plano de Control, e intentando abstraerse del hardware mediante virtualización que facilita la implementación de nuevas funcionalidades de forma más simple evitando los problemas típicos de la estandarización.

En este trabajo se hará foco en el estudio de SDN en redes inalámbricas, un campo de estudio reciente y bastante inmaduro al momento en comparación con el mismo paradigma en redes cableadas. De esta forma, se comenzará con un relevamiento general sobre herramientas e investigaciones existentes sobre SND en redes inalámbricas, realizando una comparativa entre los elementos encontrados. A partir de dicho relevamiento, se profundizará el estudio de un framework basado en SDN y NFV disponible *open source* llamado EmPOWER, que permitirá implementar un prototipo mediante una API para la definición de funcionalidades programadas sobre un controlador centralizado. El prototipo implementará la optimización de asignación de canales de frecuencia en los APs de la red, funcionalidad no existente en dicho framework al momento. La implementación estará basada en la Tesis de Maestría *Self Management of High Density Wireless Networks* de Guillermo Apollonia, sobre estrategias de asignación de canal para la optimización de interferencia en redes inalámbricas. Finalmente, mediante un despliegue de prueba se estudio la ejecución del prototipo realizando un análisis posterior de resultados.

Glosario

AP Access Point. 1, 6

API Application Programming Interface. 1

BSSID Basic Service Set IDentifier. 23

CPP Click Packet Processor. 20

DNS Domain Name System. 10

DSSS Direct Sequence Spread Spectrum. 43

IaaS Infrastructure as a Service. 10

LVAP Light Virtual Access Point. 14

LVNF Light Virtual Network Function. 22

NAT Network Address Translation. 8

NCQM Network Channel Quality Maps. 34

NFV Network Function Virtualization. 1

NV Network Virtualization. 10

QoS Quality of Service. 8

RSSI Received Signal Strength Indication. 17

SDN Software Defined Networks. 1

SNR Signal-to-Noise Ratio. 38

SSID Service Set Identifier. 40

UCQM User Channel Quality Maps. 34

UE User Equipment. 19

VAP Virtual Access Point. 14

VLAN Virtual Local Area Network. 9

VNF Virtual Network Function. 10

WTP Wireless Termination Point. 14

Índice general

Capítulo 1

Introducción

En la actualidad, el masivo crecimiento de las comunicaciones ha convertido a las redes informáticas en un factor crítico. El desarrollo de dispositivos móviles, la virtualización de dispositivos de red y el auge de servicios en la nube, además del crecimiento acelerado de la trasmisión de datos aumenta significativamente la congestión de flujos de información, por lo que la necesidad de administrar y optimizar la transmisión de éstos grandes caudales de una forma ágil, requiere de nuevos elementos de infraestructura.

El avance y la necesidad de incorporar nuevas funcionalidades y servicios a dichas redes de comunicación está condicionada a las reglas de mercado, donde evidentemente se puede observar que las características de diseño de hardware y software siguen patrones que no se rigen en pro del avance técnico sino que se ven determinados de forma particular según el fabricante. Dicha condicionante genera grandes limitaciones e incrementos en los costos de administración y operación en las redes actuales.

La posibilidad de implementar de forma alterna herramientas de software que se adapten funcionalmente en los sistemas de redes existentes se está convirtiendo en una alternativa eficaz para resolver los problemas de infraestructura. Bajo este rumbo, disciplinas como las Redes Definidas por Software (*Software Defined Networks* o SDN) y la Virtualización de Funciones de Red (*Network Functions Virtualization* o NFV) comienzan a tomar un rol importante en lo que se presenta como un nuevo paradigma, distribuyendo funcionalidades y capacidades de cómputo en los distintos elementos de la red, incentivando la producción de hardware de propósito general, no tan específico, quitando "inteligencia" del lado de los dispositivos de red y creando otras entidades que de forma central toman la responsabilidad de administrar los flujos de datos implementando funcionalidades específicas.

Actualmente, el plano de datos (la información que envían o reenvían los dispositivos) y el plano de control (donde se manipulan los dispositivos que envían información) residen dentro de los dispositivos de red, siendo este último implementado mediante protocolos distribuidos ejecutados en todos ellos y obligando a estandarizar debido a los diversos diseños de hardware. La idea de las Redes Definidas por Software (SDN), es proporcionar una configuración más eficiente y de mejor rendimiento, flexibilizando la posibilidad de nuevos diseños sin las dificultades de estandarización presentes hoy en día, con el propósito de aumentar la capacidad de los sistemas de comunicación en su totalidad.

Las estrategias para afrontar los desafíos teniendo en cuenta el volumen de datos, apuntan a la automatización de las redes, aprovechando las ventajas proporcionadas por las SDN que permiten incorporar inteligencia de software, desacoplando las funcionalidades del plano de control de los dispositivos de red, utilizando un dispositivo de cómputo externo y centralizado. La capacidad de programar con software los distintos switches y routers para redirigir el tráfico y tomar diferentes acciones según sea necesario resulta muy beneficioso y hasta las compañías más importantes de redes y tecnologías comienzan a tomar como válidos éstos conceptos, virando hacia la producción de hardware compatible con algunos de los protocolos utilizados por el nuevo paradigma. Bajo este contexto, la rapidez con la que se pueden implementar nuevas funcionalidades crece y permite a cualquier operador de red especializado desarrollar sin atarse a los tiempos de estandarización.

En este proyecto en particular se busca explorar las posibilidades de SDN en redes inalámbricas, o sea exportar parte de las funcionalidades del plano de control de un dispositivo inalámbrico a controladores externos programables. Se requirió entonces implementar agentes en dispositivos inalámbricos (APs), que permitan la comunicación con el controlador, y definan una API para la definición de funcionalidades que se programan en el controlador.

Por lo tanto podemos plantear los objetivos principales de este proyecto como:

- conocer el estado del arte de SDN en redes inalámbricas, y
- desarrollar estrategias de implementación de funcionalidades bajo el paradigma SDN en redes inalámbricas.

De esta forma es que a lo largo de este informe desarrollaremos el proceso de estudio general de las SDN en redes inalámbricas, a partir del cual hemos tomado una herramienta en particular para la implementación de funcionalidades sobre el Plano de Control. Es así como el siguiente capítulo estará enfocado al estado del arte de dichas redes, para luego pasar específicamente a la descripción detallada del framework EmPOWER en el capítulo 3 (herramienta escogida para el segundo ítem de los objetivos planteados). Posteriormente se proseguirá con la especificación del prototipo programado sobre EmPOWER que optamos realizar para implementar una funcionalidad no prevista en la herramienta, concretamente una estrategia particular para la asignación de canales en los APs de la red con el propósito de optimizar el nivel de interferencia de la misma. Luego el informe continúa con un capítulo íntegro detallando las pruebas realizadas sobre el prototipo. Finalmente el informe cuenta con el capítulo de las conclusiones finales.

Capítulo 2

Software Defined Networking

Software Defined Networking (SDN) es un paradigma donde la lógica de control de la red se encuentra implementada en un controlador centralizado. La separación del Plano de Control del Plano de Datos es lograda mediante la implementación de una API de comunicación entre los dispositivos de *forwarding* y el controlador. Este tipo de API es conocida como *interfaz southbound*.

2.1. Interfaces Northbound y Southbound

La función principal de una interfaz *southbound* es proporcionar la comunicación entre el controlador SDN y los nodos de red (ya sean dispositivos de red físicos o virtuales) para que el controlador pueda realizar tareas como descubrir la topología de red, definir flujos de red y realizar las solicitudes de las aplicaciones de red. Las interfaces *northbound* sirven para programar el control de la red desde las aplicaciones de red que se ejecutan sobre el controlador.

Las aplicaciones a través de esta API tienen la capacidad -entre otras- de:

- Configurar flujos para encaminar paquetes
- Balancear trafico a través de distintos caminos
- Reaccionar ante cambios en la red, ejemplo fallos en enlaces
- Redireccionar tráfico para realizar distintas tareas de seguridad

En un Data Center empresarial, las funciones de la interfaz *northbound* suelen incluir soluciones de gestión, orquestación y automatización.

En un sentido figurado, el flujo de las interfaces *northbound* puede ser visto dirigido hacia arriba, mientras que el flujo del *southbound* en sentido contrario. Por ello en diagramas de arquitectura, las interfaces *northbound* son dibujadas sobre el componente aplicable, mientras que las interfaces *southbound* debajo como se puede apreciar en la figura 2.1. Mientras que los términos *southbound* y *northbound* pueden aplicarse casi que a cualquier tipo de red o sistema de computadora, son conceptos clave en SDN.

OpenFlow es uno de los estándares más usados como interfaz *southbound*, debido a que se ha vuelto el estándar de facto como plataforma y protocolo de SDN en *switches*, se describirá a continuación.

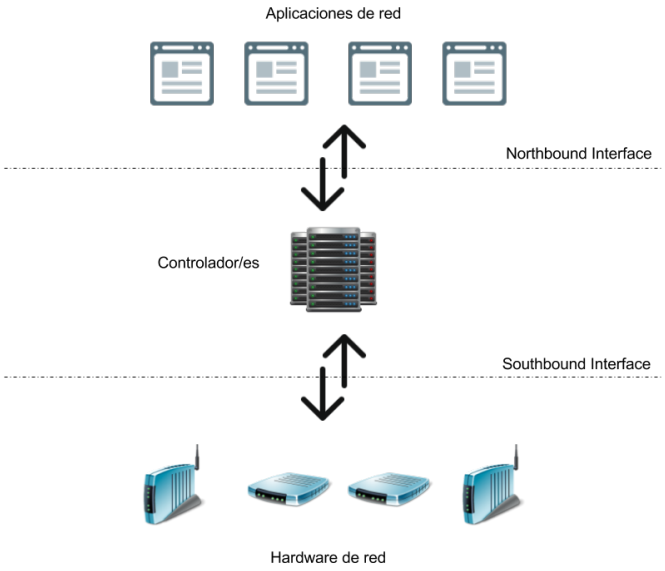

Figura 2.1: Interfaces Northbound y Southbound

2.2. OpenFlow

OpenFlow[4] es un protocolo y estándar abierto que permite a controladores de red determinar el camino de los paquetes a través de los *switches* de red. Su protocolo nos permite implementar SDN en switches. El objetivo principal de OpenFlow es proveer una plataforma virtual, abierta y programable para la red.

Cuando hablamos de OpenFlow, hablamos tanto del protocolo de comunicación entre el controlador y los switches como de la sintaxis de los mensajes que permiten programar el comportamiento de los flujos.

Las soluciones comerciales de SDN son muy cerradas e inflexibles, y otras soluciones de investigación no tienen el rendimiento necesario o son muy caras. Debido a ello con OpenFlow se busca que los switches sean flexibles y tengan un mejor rendimiento a un bajo precio.

OpenFlow explota el hecho de que la mayoría de los switches Ethernet modernos y routers contienen tablas de flujos que se pueden consultar para tomar acciones sin agregar retardo de procesamiento, para la implementación de *firewalls*, NAT, QoS y recolección de estadísticas. Mientras cada tabla de flujo de estos dispositivos son diferentes (según cada vendedor), OpenFlow identifica y explota el conjunto de funciones que corren en muchos de estos dispositivos.

En definitiva, OpenFlow permite programar las tablas de flujo de los dispositivos de red. El conjunto de acciones que soporta un Switch OpenFlow es extensible, pero hay ciertos requerimientos mínimos que debe cumplir:

1. Una tabla de flujo, con una acción asociada a cada entrada que le diga al switch cómo procesar cada paquete

2. Un canal seguro que conecte al switch con un proceso remoto de control (Controlador) permitiendo el envío de paquetes e instrucciones entre el

switch y dicho controlador, utilizando,

3. El protocolo OpenFlow, que provee una forma abierta y estándar de comunicación, especificando una interfaz por donde cada entrada en la tabla pueda ser definida externamente

2.2.1. Switches OpenFlow

Existen dos categorías de Switches OpenFlow, los Switches-dedicados (que no soportan el procesamiento normal de capa2 y capa3) y los Switches híbridos (que pueden ser utilizados como switches normales para el tráfico de producción y además agregan como funcionalidad a OpenFlow).

Todo switch dedicado debe soportar las siguientes tres acciones básicas:

1. Realizar el *forward* de paquetes según el puerto/s asociado. Esto permite enrutar los paquetes por la red

2. Encapsular y hacer *forward* de los paquetes a un controlador. Los paquetes son enviados por un canal seguro, donde son encapsulados y llegan al controlador. Típicamente esto sucede con el primer paquete de un flujo, entonces el controlador puede decidir si agregarlo en la tabla. En casos especiales, todos los paquetes son enviados previamente al controlador para su procesamiento

3. Eliminar los paquetes de un flujo. Puede ser usado por seguridad o para eliminar tráfico basura. Una de las mayores ventajas de OpenFlow es permitirle a los investigadores experimentar sobre redes existentes con tráfico y aplicaciones regulares. Por lo tanto, los switches híbridos deben aislar el tráfico de producción del tráfico experimental, y una de las formas fue sumando a estos switches una cuarta acción

4. Realizar el *forward* de paquetes de un flujo mediante el procesamiento de un *pipeline* de un switch normal

Los Switches híbridos deben respetar dos propiedades: todos los paquetes que no provengan del tráfico experimental, deben utilizar un protocolo de ruteo estándar; y únicamente se puede agregar entradas en las tablas de flujos del tráfico autorizado por el administrador de la red.

A los switches que cumplan con estas cuatro acciones, se les denomina switches *Type0*, se definen nuevos tipos de switches según las nuevas funcionalidades que logre.

2.2.2. Controladores OpenFlow

Un controlador es quien agrega y elimina acciones para flujos OpenFlow en la tabla de los dispositivos. Existen varios tipos de controladores según las funcionalidades que se requieran.

Una aplicación básica, sería un controlador que aporte la funcionalidad de VLAN a los switches, definiendo flujos estáticos en la tabla OpenFlow de acuerdo al tag de *VLAN*.

Con un controlador más avanzado, se podrían implementar funcionalidades más complejas como el manejo de tráfico prioritario.

Existen al momento de realizar el estudio varias implementaciones de controladores OpenFlow, por listar alguna de ellas:

- **NOX**[3] Fue el primer controlador OpenFlow. Proporciona una API *northbound* en C++ y Python la cual luego quedo obsoleta.
- **POX** Análogo al controlador NOX pero con soporte para Python
- **Floodlight** Controlador basado en Java, desarrollado por una comunidad abierta
- **Ryu** Controlador basado en Python, de código abierto. Es soportado por OpenStack, software para virtualización y creación de *clouds*
- **Beacon** Controlador extensible basado en Java. Se basa en el framework OSGI, el cual permite que las aplicaciones creadas sobre el mismo puedan ser inicializadas, detenidas, refrescadas e instaladas en tiempo de ejecución, sin perder la conexión con los switches
- **Bigswitch** Controlador basado en Beacon de código cerrado enfocado en redes empresariales de producción

En «Software-Defined Networking: A Comprehensive Survey»[21] se puede apreciar un estudio reciente y completo sobre SDN y las distintas implementaciones disponibles.

2.3. Software Defined Networking y Network Function Virtualization

Virtualización de Funciones de Red -NFV por sus siglas en inglés- es una manera de diseñar, desplegar y manejar servicios de red. Consiste en desacoplar las funciones de red del hardware propietario de manera de que puedan ser realizadas por software. Ejemplo de estas funciones de red pueden ser la traducción de direcciones de red (NAT), *firewalling*, sistemas de detección de intrusos (IDS), DNS y utilización de caché (web por ejemplo).

Con las NFV se puede consolidar y entregar los componentes de red necesarios para soportar una infraestructura completamente virtualizada. Una función de red virtualizada -VNF por sus siglas en ingles- consiste en realizar una función de red que normalmente se realizaría en hardware de red especializado, sobre hardware de alta capacidad de servidores, switches y almacenamiento.

Es aplicable a cualquier procesamiento del Plano de Datos tanto como a las funciones del Plano de Control, tanto en redes cableadas como inalámbricas. NFV se enfoca en el desacople y optimización de los servicios de red mientras que SDN separa el Plano de Datos con el de Control para un control y visión centralizada de la red. Los conceptos de SDN y NFV son independientes, pero en general se complementan junto a Virtualización de Red -NV por sus siglas en ingles-, por ejemplo es muy común gestionar infraestructuras de red como servicio (IaaS) con técnicas en conjunto de SDN, NV y NFV. SDN se utiliza para tener una visión global de la red y orquestarla, contribuyendo con la automatización al decir hacia donde va el tráfico y desde donde, basado en políticas

de la red, mientras que NVF se enfoca en los servicios en sí y NV en virtualizar y particionar elementos de la red.

Ejemplo de aplicaciones y frameworks que usan en conjunto de SDN y NFV son:

- **OpenStack**[11] Un proyecto de código abierto de *cloud computing* para ofrecer IaaS (Infraestructura como servicio)
- **Open vSwitch (OVS)** OVS es una implementación de código abierto de un switch virtual distribuido y multicapa. Su principal propósito es proveer un *stack* de switching para virtualización de hardware. Soporta el protocolo OpenFlow
- **OpenDayLight**[17] OpenDayLight es una plataforma SDN de código abierto y sirve de base por ejemplo para implementaciones de NFV y NV. Esta escrito en Java y es promocionado por la Linux Foundation y muchos de los principales actores en temáticas de SDN y NFV
- **Open Platform for NFV (OPNFV)** OPNFV es un plataforma para facilitar el desarrollo y evolución de componentes NFV, integrando diferentes plataformas abiertas como OpenDayLight, OpenStack, KVM, Open vSwitch y Linux
- **OPEN-O** OPEN-O es un proyecto de código abierto respaldado por la Linux Foundation que habilita a los operadores de cable y telecomunicaciones entregar efectivamente servicios de punta a punta a través de infraestructura NFV así como SDN y servicios de red legados.

2.4. SDN Inalámbrico

Se define SDN Inalámbrico como la aplicación de los conceptos de SDN a dispositivos de red inalámbricos, o sea la separación entre el plano de datos y el plano de control de forma que éste último sea gestionado por medio de un controlador de lógica centralizada. Para poder aplicar éstos conceptos en el medio inalámbrico, se presentan ciertos problemas y particularidades que no se encuentran en el medio cableado. Por ejemplo, parte del plano de control no puede condicionarse a los tiempos que maneja el controlador central para tomar decisiones, por lo tanto, debe ser resuelto por el dispositivo de red inalámbrico, como el caso de los ACK en las redes WiFi. Otro de los problemas es la falta de un framework estandarizado como OpenFlow para los medios inalámbricos. A partir de ésta apreciación, es que se realizó un relevamiento de los estudios existentes sobre SDN en dicho medio.

2.4.1. OpenRoads

OpenRoads[8], desarrollado en Stanford, fue el primer trabajo sobre SDN en redes inalámbricas. El objetivo de este proyecto fue desarrollar una plataforma abierta donde investigadores puedan experimentar con soluciones de movilidad, controladores de red y nuevos protocolos de enrutamiento. La arquitectura es presentada en la figura 2.2.

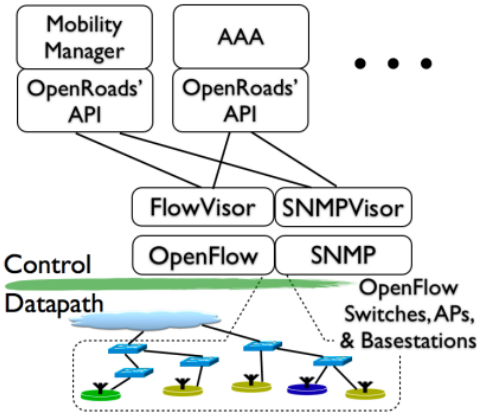

Figura 2.2: Arquitectura OpenRoads[7]

OpenFlow es utilizado en los dispositivos de red (AP, switches, routers) para controlar el camino de datos, FlowVisor para crear porciones aisladas de red permitiendo realizar múltiples experimentos simultáneamente y SNMPVisor para configurar los dispositivos entre las distintas porciones de la red. De esta manera la red es virtualizada y cada experimento se ejecutará utilizando los recursos que le fueron asignados sin interferir con los demás experimentos (particionamiento de red).

FlowVisor funciona como un *proxy* transparente entre los dispositivos OpenFlow y los controladores del experimento. Los switches ven a FlowVisor como un controlador y los controladores de los experimentos ven una red privada virtual.

SNMPVisor funciona de manera similar a FlowVisor, particionando el camino de datos de la configuración de los AP permitiendo a un experimento configurar los dispositivos presentes en su porción de la red.

2.4.2. Odin

Odin[10] es un framework SDN para redes inalámbricas WiFi. El principal aporte de este trabajo es la introducción de la abstracción LVAP (explicada en detalle en el siguiente capítulo) y el prototipo implementado, que luego fueron utilizados por trabajos que extendieron la herramienta agregando y mejorando funcionalidades.

A continuación se presenta un diagrama (figura 2.3) de la arquitectura propuesta por Odin y los detalles de los componentes.

- **Controlador** Permite la ejecución de aplicaciones de red que controlan la red física. Expone un conjunto de interfaces a las aplicaciones y las llamadas a estas son traducidas en mensajes a los dispositivos de red. También mantiene la visión de la red, la cual incluye clientes, APs y switches OpenFlow, los cuales las aplicaciones pueden controlar

- **Agente** Se ejecuta en los AP inalámbricos y se comunica con el controlador para permitirle controlar la red inalámbrica. Los aspectos críticos del

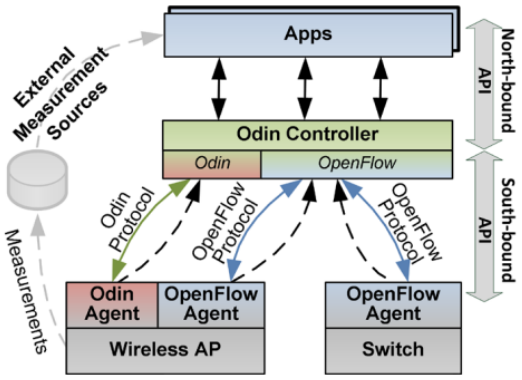

Figura 2.3: Odin [19]

protocolo MAC[1] Wifi son realizados por el hardware inalámbrico, como el envío de ACKs. Por otro lado, las funcionalidades donde el tiempo no es crítico (asociación de clientes) es implementada en software entre el controlador y el agente

- **Aplicaciones** Programan la red a través de la API *northbound* brindada por el controlador. El acceso a los datos de estadísticas y mediciones pueden ser programados tanto de forma reactiva como proactiva. Se brinda acceso a las mediciones realizadas por los agentes, estadísticas OpenFlow y mediciones realizadas por herramientas externas

En el prototipo presentado de Odin, el controlador esta implementado como una extensión del controlador Floodlight de OpenFlow. Para la comunicación con los agentes se mantiene una conexión a través de la cual se envían los comandos del protocolo Odin. En lo que respecta al agente, éste fue implementado usando Click Modular Router y se ejecutan en APs con OpenWrt. Además del agente, los APs ejecutan Open vSwitch para programar el Plano de Datos en estos dispositivos.

2.4.3. AeroFlux

AeroFlux[18], basado en Odin, propone un diseño en dos capas para implementar el Plano de Control inalámbrico. La arquitectura tiene los siguientes componentes:

- **Controlador Global** Maneja eventos donde el tiempo no es critico o pertenecientes a tareas necesariamente globales. Ejemplos son la autenticación, movilidad, balanceo de cargas y otros. Además controla los NSC
- **Controladores Near-Sighted (NSC)** La segunda capa del Plano de Control incluye un conjunto de controladores ubicados cerca a los APs para realizar funciones donde el tiempo es crítico

[1] Medium Access Control

- **Agente de Radio** Este agente se encarga de alojar los LVAPs junto con reglas especificas para la transmisión WiFi para cada uno de los clientes. Estas reglas definen los atributos para la transmisión en el medio inalámbrico. El agente expone todas las funcionalidades de los LVAPs junto con la reglas hacia los NSC
- **Reglas de transmisión del camino de datos inalámbrico (WDTX)** Estas reglas extienden las reglas de OpenFlow definiendo por flujo propiedades de transmisión para las tramas 802.11. Permiten configurar el *rate*, los reintentos, potencia, uso de ACK y RTS/CTS

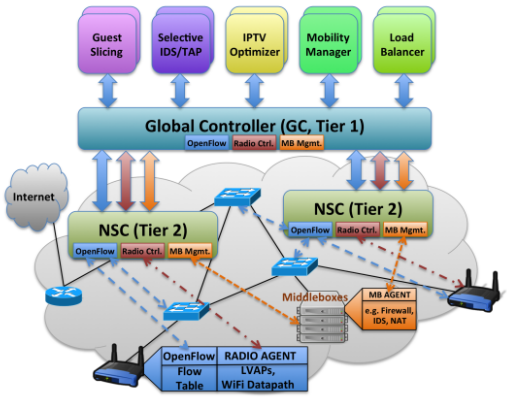

Figura 2.4: AeroFlux [18]

2.4.4. CloudMAC

CloudMAC[16] es una arquitectura distribuida donde el procesamiento de las tramas 802.11 es parcialmente realizado en servidores virtuales conectados entre ellos mediante una red utilizando OpenFlow. Un despliegue CloudMAC consiste de APs virtuales (VAPs), WTPs, un switch y un controlador OpenFlow.

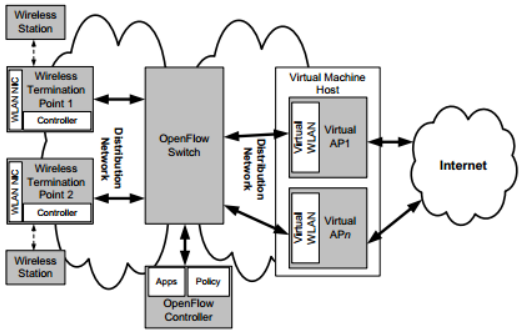

Figura 2.5: CloudMAC[16]

Los VAPs son máquinas virtuales con una o más tarjetas de red inalámbricas virtuales que ejecutan software de administración de APs para generar tramas beacons o responder a las asociaciones de los clientes. Las distintas tarjetas inalámbricas (virtuales) de un VAP pueden estar conectadas a diferentes tarjetas físicas en distintos WTPs.

Los WTPs son APs con tarjetas inalámbricas que permiten enviar y recibir tramas WiFi. En el *downlink* las tarjetas virtuales de los VAPs generan las tramas MAC, agregan encabezados de control y opcionalmente realizando cifrado. Los WTPs envían las tramas con el esquema de modulación y potencia indicados en el encabezado de control. Además de esto, los WTPs se encargan de enviar las tramas que tienen restricciones de tiempo como los ACK. En el *uplink*, los WTP reciben las tramas y las reenvían al VAP que corresponda.

Los VAPs y WTPs se conectan a través de un switch OpenFlow donde su tabla de *forwading* representa la comunicación entre estos.

CloudMAC también permite el control sobre comandos de configuración de las tarjetas de red inalámbricas, como pueden ser el cambio de canal. Para esto el driver de la interfaz virtual intercepta los comandos y genera un paquete especial que el switch reenviará al controlador. Si el comando es permitido será reenviado a una aplicación de control que se encuentra en el WTP y este lo ejecutará.

2.4.5. OpenSDWN

OpenSDWN[22] es una arquitectura de redes WiFi basada en los enfoques de SDN y NFV que extiende los trabajos de Odin y AeroFlux. Su principal aporte es la introducción de la abstracción vMB que representa una *middlebox* virtualizada que puede ser transferida a través de la red.

Los componentes de esta arquitectura son los siguientes:

- **SDN inalámbrico** Este componente es basado en Odin del cual utiliza la abstracción LVAP, la capacidad de cambiar a los usuarios de AP de manera transparente, y el particionamiento de la red. De AeroFlux utiliza las reglas para las transmisiones inalámbricas (WDTX), las cuales permiten configurar características (*rate*, potencia, uso de ACKs) de la transmisión a nivel de flujo

- **Middleboxes virtuales (vMB)** Es la abstracción introducida para brindar virtualización de funciones a nivel de cliente. Una vMB mantiene la información especifica del usuario y puede ser transferida a otra instancia de MB. Sobre un MB físico se ejecuta un agente, encargado de brindar comunicación con los recursos, manejar los vMBs, y exponer el control del MB hacia el controlador

- **Interfaz participativa** Permite a los usuarios priorizar los flujos de acuerdo a sus preferencias. El controlador se encarga de traducir estas prioridades en reglas de transmisión. Se utiliza un *IDS* basado en firmas que inspecciona los datos del paquete para identificar los servicios de interés. Una vez detectado el controlador es informado y este aplica las políticas necesarias

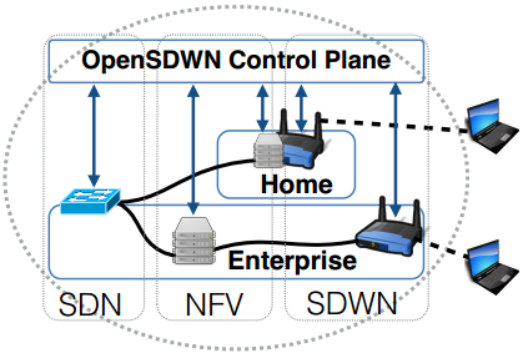

Figura 2.6: OpenSDWN[22]

2.4.6. EmPOWER

EmPOWER[15] es un *Sistema Operativo de Red* (Network Operating System) para SDN, que funciona para redes heterogéneas. EmPOWER toma como base los trabajos de Odin[10] a modo de concretar un *testbed* para investigación y experimentación de *Virtualización de Funciones de Red*.

La motivación de EmPOWER se genera por la escasez de facilidades experimentales completamente virtualizadas que exploten los conceptos de NFV en el campo de las redes inalámbricas; y más allá, en el actual ecosistema OpenFlow formado por el controlador y la plataforma particionada, el software y hardware de sus dispositivos de red no proveen mucho soporte para el dominio WiFi.

Las APIs *northbound* eran muy primitivas y obstaculizaban el desarrollo de aplicaciones de red más modulares y flexibles. Por ejemplo, aún siendo OpenFlow una abstracción concreta y práctica para *forwarding*, los esfuerzos requeridos para la definición de otros modelos de programación son considerables. Por ende, muchos actores en el campo de las SDN argumentan a favor de la creación de lenguajes de alto nivel los cuales serían implementados por el *Sistema Operativo de Red*.

De esta forma se crea EmPOWER, como un testbed que ayuda a llenar los vacíos de facilidades experimentales proporcionadas por SDN y NFV.

Los requerimientos que se plantean al definir EmPOWER son los siguientes:

- **Particionamiento y modelo de programación** La plataforma debe soportar primitivas programables de alto nivel con respecto al control de la red, con el objetivo de dejar abstraer detalles específicos del estándar WiFi como mecanismos de asociación/disociación, intercambio de paquetes de control y la gestión de estado. Es importante que la plataforma no implique la modificación de software/hardware de los clientes WiFi, salvo que se esté pensando en implementar para ellos exclusivamente. Finalmente debe contar con un mecanismo de particionamiento que permita asignar a *tenants* porciones de la red (APs y switches)

- **Consultar el estado de red** La plataforma debe proveer al *tenant* un conjunto de primitivas para consultar el estado de la red. Los switches OpenFlow proveen una colección de estadísticas relacionadas a los puertos y los flujos con respecto al tamaño y la cantidad de paquetes, en el caso

de implementaciones inalámbricas se deben contemplar otros aspectos característicos del medio como por ejemplo RSSI y la pérdida de paquetes (de mayor impacto en redes inalámbricas)

- **Arquitectura de federación** La instanciación de una red virtual sobre la plataforma, debe ser realizada a través de un framework de control web o por línea de comandos que permita una fácil reserva de recursos de red, ya sean nodos o enlaces. Idealmente el usuario deberá estar habilitado para seleccionar los APs y switches para usar durante el experimento y luego el sistema deberá instanciar la red de una forma transparente para el usuario

2.5. Comparación entre las distintas herramientas

A continuación se presentará un resumen comparativo entre los estudios analizados a modo de referencia.

Criterios de comparación:

1. Southbound: interfaces de comunicación entre los dispositivos de red y el controlador soportados
2. VNF: La herramienta brinda soporte para VNFs
3. Transmisión: Permite configurar propiedades de la transmisión inalámbrica a nivel de cliente
4. Modificaciones en clientes
5. Controlador utilizado
6. Particionamiento de la red (cableada e inalámbrica)

	OpenRoads	**Odin**	**AeroFlux**
Southbound	OpenFlow	OpenFlow y Odin	OpenFlow, Odin y extensión para WDTX
VNF	No	No	No
Propiedades de transmisión	No	No	Si
Modificaciones en clientes	No	No	No
Controlador	NOX	Floodlight	Floodlight
Particionamiento	Si	Si	Si

	CloudMAC	**OpenSDWN**	**EmPOWER**
Southbound	OpenFlow y CloudMAC	Idem AeorFlux y extensión para vMB	EmPOWER
VNF	No	Si	Si
Propiedades de transmisión	Si	Si	No
Modificaciones en clientes	No	No	No
Controlador	CloudMAC	Floodlight	EmPOWER Runtime
Particionamiento	Si	Si	Si

2.6. Elección de herramienta

Vistos los estudios realizados en el área, podemos notar que la gran mayoría radican en diseñar una arquitectura e implementar un prototipo con el fin de evaluarla, y son pocos los que están disponibles para ser utilizados. Debido a estos motivos, las opciones posibles se vieron acotadas a Odin y EmPOWER, donde finalmente se optó por esta última ya que aún continua activo su desarrollo (a diferencia de Odin) y extiende los conceptos aportados por el primero.

De esta forma, el estudio continuará profundizando sobre la herramienta elegida: EmPOWER.

Capítulo 3

Descripción EmPOWER

A partir del estudio y análisis previo, en este capítulo se describirá en profundidad el framework EmPOWER.

3.1. Descripción del framework

El siguiente estudio se basa en la descripción original del testbed EmPOWER[15] e información encontrada en el sitio web[24], a su vez se obtuvo más detalle de la implementación en el repositorio de software[32].

EmPOWER es un sistema operativo de red que provee una red altamente programable implementando conceptos de SDN y VNF para redes de acceso inalámbrico.

Introduce el concepto de *Programmable Network Fabric* (PNF) que abstrae el acceso inalámbrico disponible y los recursos de procesamiento de paquetes como un recurso virtual de red genérico.

EmPOWER define sus propias VNFs, llamadas abstracciones y permite definir VNFs adicionales. Los servicios de red complejos de EmPOWER pueden ser implementados al encadenar VNFs simples y reusables. A modo de ejemplo, imaginemos que la arquitectura del framework nos permite tener varios APs para comunicarse con un cliente, por lo tanto cualquiera de ellos podría recibir los paquetes del mismo. De esta forma, se puede implementar una VNF encargada de eliminar posibles paquetes duplicados (recibidos por dos APs) y encadenarlo con otra VNF distinta encargada de priorizar el tráfico según alguna política definida.

El framework considera una red particionada donde tendrá dos tipos de usuarios. El *tenant* que será un usuario del framework al cual se le reservará una porción de la red, es el encargado de administrar su porción de la red mediante las APIs *northbound*. Y luego estarán los clientes propiamente dichos que serán los que utilizarán la red definida por los *tenants* mediante dispositivos inalámbricos, también llamados *User Equipment* (UE)[1].

[1] *User Equipment* es un termino definido por UMTS/LTE para todo dispositivo que el usuario final utiliza para comunicarse en el entorno inalámbrico

3.1.1. Componentes

EmPOWER brinda tres tipos de recursos de red virtualizados: nodos de *forwarding* (por ejemplo switches compatibles con OpenFlow), nodos de procesamiento de paquetes y también de acceso inalámbrico.

- **Switches OpenFlow** Switches clásicos con soporte OpenFlow. Son los nodos de la red más básicos, donde el Controlador configurará sus tablas de *forwarding*.
- **Click Packet Processors (CPP)** Son nodos de red que incluyen un switch OpenFlow pero además tendrán capacidades extra de procesamiento de paquetes a través de scripts definidos por los tenants (LVNF). El agente de los mismos que permitirá el procesamiento de paquetes es implementado con Click[1].
- **Wireless Termination Points (WTP)** Los WTP son los nodos de red que tienen recursos inalámbricos. El agente del mismo será implementado con Click y contendrá un switch OpenFlow, por lo que se puede afirmar que los WTP contienen a los CPP en cuanto a funcionalidades. El término WTP aplica en la arquitectura a cualquier nodo de radio en la red, pero como lo instanciaremos con la tecnología WiFi, es indistinto hablar de WTP o AP. La diferencia con un CPP es su capacidad de utilizar los recursos inalámbricos.

Estos componentes juntos con el Controlador de EmPOWER y un controlador OpenFlow forman la arquitectura a alto nivel, como se puede apreciar en la figura 3.1.

El Controlador EmPOWER gestionará los CPP y WTP mediante la API *southbound*. Los switches OpenFlow dedicados, así como los contenidos en los CPP y WTP serán gestionados por algún controlador OpenFlow. Este controlador OpenFlow no está abarcado por el framework EmPOWER, se debería elegir uno de acuerdo a las necesidades de la plataforma donde se despliegue EmPOWER, y debería dar soporte por ejemplo al particionamiento de la red cableada de manera coherente a la realizada por EmPOWER, tal vez con ayuda de un framework de nivel superior como OpenStack.

3.1.2. Abstracciones Programables

Software-Defined Networking apunta a simplificar las tareas de control y gestión al proveer a los desarrolladores de visibilidad total sobre el estado de la red, desde un controlador de lógica centralizada, moviendo el foco de atención desde el diseño de nuevos mecanismos y protocolos hacia definir algoritmos sobre el controlador central. EmPOWER define un conjunto de abstracciones de programación de alto nivel que permiten desplegar nuevas características y servicios como aplicaciones de red sobre un controlador de lógica centralizada. En EmPOWER todo es tratado como una *Virtual Network Function* (VNF), incluyendo la funcionalidad de acceso inalámbrico. Las VNFs nativas de EmPOWER son llamadas abstracciones. Al momento EmPOWER soporta abstracciones para WiFi, LTE y VNF definidas por el *tenant* basadas en Click[1].

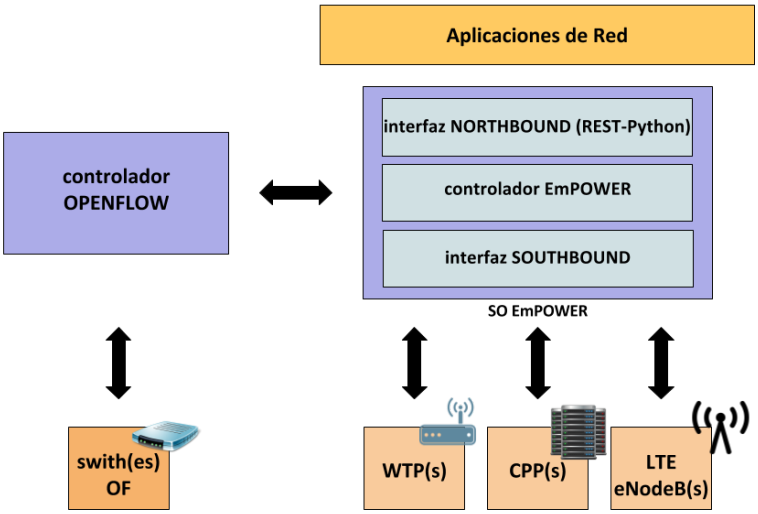

Figura 3.1: Arquitectura EmPOWER

Abstracciones inalámbricas

Estas abstracciones definidas por EmPOWER resuelven cuatro aspectos de control y gestión de las redes inalámbricas: gestión de estado, asignación de recursos, monitoreo y reconfiguración de la red.

- **Light Virtual Access Point(LVAP)** LVAP provee a los programadores de una interfaz de alto nivel para la gestión del estado de los clientes inalámbricos, abstrayéndose de la tecnología del hardware. Se detalla con más detalle en que consiste esta abstracción en la sección 3.1.3.
- **Radio Port** La abstracción *Radio Port* separa los programas autónomos de control en el borde de la red (MAC[2] de bajo nivel) de las tareas de gestión de red corriendo en el controlador central (MAC de alto nivel). En definitiva, es la abstracción que separa los paquetes que deberá procesar un WTP de los que deberá procesar el controlador. La abstracción *Radio Port* la representará un LVAP instanciado en cierto bloque de recursos de un WTP.
- **Resource Pool** La abstracción *Resource Pool* permite a los desarrolladores utilizar recursos de diferentes tecnologías de capa de enlace, usando un modelo lógico común de alto nivel. Nos presentará los recursos inalámbricos que tendremos a disposición. En definitiva, el *Resource Pool* está compuesto por *Resource Elements*, y éstos últimos se componen de las antenas disponibles con sus respectivas frecuencias de canal.

[2]Media Access Control

- **Channel Quality Map** La abstracción *Channel Quality Map* permite a los desarrolladores recolectar el estado de la red usando primitivas de alto nivel.

Abstracciones Virtual Network Functions

Las abstracciones *Virtual Network Functions* definidas por EmPOWER resuelven tres aspectos fundamentales de la composición VNF, estos son: gestión de capas (por ejemplo cómo formar una red virtual), gestión funcional (por ejemplo cómo migrar un VNF de un nodo a otro), y descubrimiento de red (por ejemplo cómo exponer el estado de la red).

- **Light Virtual Network Function(LVNF)** La abstracción LVNF es una generalización de LVAP. Sin embargo a diferencia de LVAPs, LVNFs puede realizar procesamiento de paquetes de forma arbitraria. Los LVNF van a ser las VNF que puede definir el usuario y no están incluidas en el framework. Serán implementadas como scripts de Click[1].
- **Network Graph** El *Network Graph* extiende el *Channel Quality Map* con total visión sobre el estado de la red.
- **Virtual Port** Los *Virtual Port* conectan un puerto de salida de un LVNF a un puerto de entrada de otro al especificar el flujo que debe ser ruteado a través de los puertos.

3.1.3. Light Virtual Access Point

EmPOWER está construido sobre la abstracción LVAP. El mecanismo consiste en crear un AP virtual para un UE, de manera que pueda ser manejada su conexión a una red inalámbrica a través de distintos AP físicos, mientras que para el UE la conexión se verá como si fuera siempre el mismo AP físico. Un LVAP es un AP virtual específico para cada cliente. Cada AP físico mantiene un LVAP para cada UE conectado a él.

Cuando un UE desea conectarse a un AP, debe autenticarse y asociarse con el mismo. Para ello el UE debe conocer el identificador del AP que es el BSSID. Este identificador lo puede obtener de un *beacon frame* el cual los AP mandan periódicamente para anunciar una red disponible o el UE puede enviar un *probe request* solicitando información sobre las redes disponibles y el AP podrá anunciar su red con un *probe response*, el cual contendrá la información del BSSID junto a otras como las capacidades del AP.

Luego de conocer el BSSID el UE realizará los pasos de autenticación y asociación como se muestra en la figura 3.2

Notar que aquí la autenticación no es del tipo WPA o 802.1X. Ese tipo de autenticaciones se realizarán luego de que el UE se autenticó y asocio con un AP, ya que la autenticación inicial no se realiza de forma segura. Por razones de optimización un UE podría autenticarse con varios AP, así al realizar un handover solo necesitará realizar la asociación, pero un UE solo podrá estar asociado a un AP. Por esta razón un UE no se encontrará finalmente conectado a un AP hasta no haber realizado los pasos de autenticación y asociación.

En EmPOWER cuando un UE hace un *probe scan* y es recibido por alguno de los WTP, el paquete, al ser de control, es enviado al controlador central. El

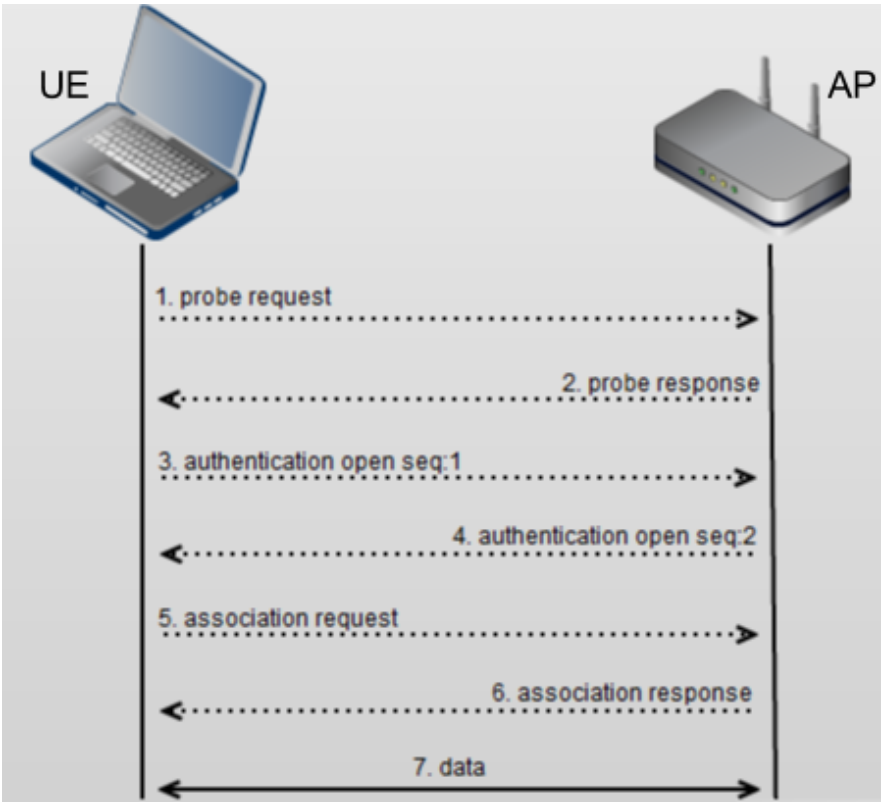

Figura 3.2: Autenticación y Asociación WiFi[25]

Controlador instancia un LVAP en un AP físico cercano al UE. Esta instancia de LVAP es la que responde al *probe scan* del UE con un *probe response*.

Al UE solo le concierne obtener los ACKs de las tramas de datos que genera, y recibir *beacons* de su LVAP cada tanto. Los LVAP pueden ser migrados a otro WTP, si un LVAP es migrado a otro WTP lo suficientemente rápido de manera que retenga todo el estado necesario para mantener las asociaciones, y el cliente continúa recibiendo los *beacons* y ACKs, entonces el cliente puede continuar transmitiendo tramas de datos sin tener que realizar una re-asociación. Por lo que una migración de LVAP tiene el efecto de un handover, sin inducir un cambio en la máquina de estado del cliente, y si es realizado lo suficientemente rápido el cliente siquiera notara algún período de desconexión. Esto funciona porque el cliente solo se preocupa por las respuestas que obtiene de su WTP (identificado por su BSSID), y es inconsciente acerca de que radio físico está generando estas respuestas. Es por ello que un LVAP posee un BSSID único para cada UE que se genera a partir de dos partes. La primera parte es un prefijo MAC llamado *base_mac* el cual es constante para cada *tenant* de EmPOWER y es de tres bytes. La segunda parte la forman los últimos tres bytes de la MAC del UE.

LVAPs entonces separa el estado de asociación de un enlace WiFi del AP físico en la red. Esta configuración hace que los clientes vean el mismo AP virtual consistente, sin importar del WTP actual con el que se encuentra en el rango. Por lo que si se quita el LVAP de un cliente del AP al que está conectado y se

lo crea en otro AP, se genera un *handover* sin que el cliente deba realizar una re-asociación, debido a que el LVAP mantiene la información de la conexión con el cliente.

Ventajas Tiene como ventaja que los handover entre los distintos WTP serán transparentes para el UE por lo que se baja drásticamente la latencia del mismo. Otra ventaja es que un controlador central de la red puede elegir con que WTP se comunicará el UE, simplemente traspasando el estado del LVAP a otro AP a elección, pudiendo optimizar el congestionamiento de los WTPs, el ruido y las interferencias. Adicionalmente, podrá instanciar LVAPs adicionales "de escucha" asociados al mismo UE en otros WTP, de manera que si bien solo un WTP transmite los paquetes hacia el UE, los paquetes enviados desde el UE a la infraestructura pueden ser recibidos en varios WTPs, por ejemplo para tener mas robustez en la comunicación. Al poder mover los LVAP a través de los diferentes WTP, se podrá instanciar el LVAP correspondiente a un UE lo mas cercano posible al mismo, de manera de que utilice menos potencia en la señal para la comunicación y de esta manera disminuir el consumo eléctrico.

3.2. Diseño original del testbed EmPOWER

EmPOWER propone un diseño original para un testbed que se detalla a continuación. Sin embargo, al momento de realizar el estudio muchos componentes no se encontraban aún desarrollados. Esto motivó a que el diseño final fuera levemente diferente, sobre todo en las implementaciones concretas de cada componente.

La arquitectura de EmPOWER, consiste en un único controlador y un conjunto de agentes que corren en los WTPs. El controlador se implementa sobre un controlador OpenFlow el cual tiene visión global de la red en términos de clientes, flujos e infraestructura. El controlador OpenFlow sería Floodlight y el controlador de EmPOWER correría como una aplicación Floodlight, permitiendo tener sus propias *aplicaciones de red* EmPOWER que correrán como hilos. Esta arquitectura evoluciona de Odin[10]. Las *aplicaciones de red* pueden utilizar tanto la interfaz REST embebida en Floodlight o un intérprete intermediario como por ejemplo Pyretic[14]. Cada aplicación de red corre sobre una porción aislada de la red controlando todos o sólo un subconjunto de los WTP disponibles. Los agentes, al implementar LVAPs permiten que múltiples clientes sean tratados como un conjunto de clientes lógicamente aislados, conectados a diferentes puertos de un switch. Cada uno se asociará un AP virtual exclusivo (LVAP), pero los clientes podrían comunicarse entre sí mediante la red. De la misma forma que Odin, el agente corriendo en cada AP está implementado con Click.

EmPOWER posee la capacidad de gestionar el consumo eléctrico; para ello utiliza Energino, un add-on Arduino que permite medir el consumo energético de un dispositivo y funciona también como controlador de chasis permitiendo prender/apagar los dispositivos de la red mediante una interfaz *RESTful HTTP*.

Mientras que Energino permite medir el consumo total de un AP, no provee información sobre el consumo energético de una porción (*slice*) del testbed. Para resolver este problema se extiende el framework para proporcionar información sobre la eficiencia energética de una aplicación de red. Para lograr dicho objetivo,

se desarrollan algunos modelos de consumo energético que se embeben en el framework de control, permitiendo aislar la contribución actual de cada porción al consumo total de un AP. El modelo de consumo energético toma como entrada estadísticas de red medibles tal como: paquetes transmitidos/recibidos sobre cierta interfaz, uso de CPU, etc. Dichas estadísticas son utilizadas por una entidad centralizada que está a cargo de estimar el consumo energético por cada porción. Estos se brindan al usuario por medio de la interfaz *northbound* del controlador.

Más allá de las funcionalidades aplicables al concepto de SDN como "lógica centralizada" y la serie de abstracciones que esto facilita, la disponibilidad del control de consumo energético en tiempo real le brinda al usuario información empírica sobre sus aplicaciones.

En el diseño de referencia, cada WTP está equipado con dos puertos Ethernet, uno de ellos está conectado a la red de gestión y control, esto permite obtener información de la red y realizar tareas administrativas sin afectar el tráfico de los clientes que fluyen a través del segundo puerto Ethernet. En los switches se utilizan VLANs para mantener separado el tráfico de datos y control. Finalmente, otra red recolecta las estadísticas del consumo de energía generadas por los dispositivos Energino.

Al contrario de otros frameworks WiFi, EmPOWER no permite utilizar un sistema operativo personalizado en cada WTP, pero provee un conjunto de APIs con el cual se puede controlar el comportamiento de cada AP desde un controlador centralizado.

En el diseño propuesto para el testbed original de EmPOWER se utilizan herramientas libres y gratuitamente disponibles: *Open vSwitch*[6](ovs) y *Click Modular Router*[1] para el camino de datos; Floodlight como controlador y Arduino como gestor de energía.

3.2.1. Diferencias con el diseño original

Inicialmente el controlador de EmPOWER sería desarrollado como una aplicación sobre Floodlight, tomando como base la implementación de Odin. Pero al momento de realizar el estudio de EmPOWER aún no se encontraba en su versión final. Los desarrolladores optaron por dejar de lado la implementación de los aspectos cableados de la plataforma y se desarrolló un nuevo controlador auto-contenido escrito en Python, el cual presentaría una interfaz REST para la realización de aplicaciones de red o se podrían implementar como módulos Python. El controlador manejaría los aspectos inalámbricos de los agentes de los WTPs, como los LVAPs, el *Channel Quality Map* o los *Radio Port*. Para los agentes de los WTP, efectivamente se tomaría como base lo definido en Odin, que es un agente implementado con Click.

Por lo tanto el particionamiento de la red de manera que cada *tenant* viera su entorno de experimentación aislado no era posible en la primera versión, al menos para la parte cableada. En un principio se pretendía que fuera cubierto con Floodlight, pero quedó abierta la implementación al momento de realizar el estudio. Por lo que para el diseño inicial habría que apoyarse en un framework/controlador OpenFlow como se muestra en la figura 3.3.

Entonces los switches OpenFlow, los CPP y los switches de los WTP funcionarían como *switches tontos* que direccionarían los paquetes de forma tradicional.

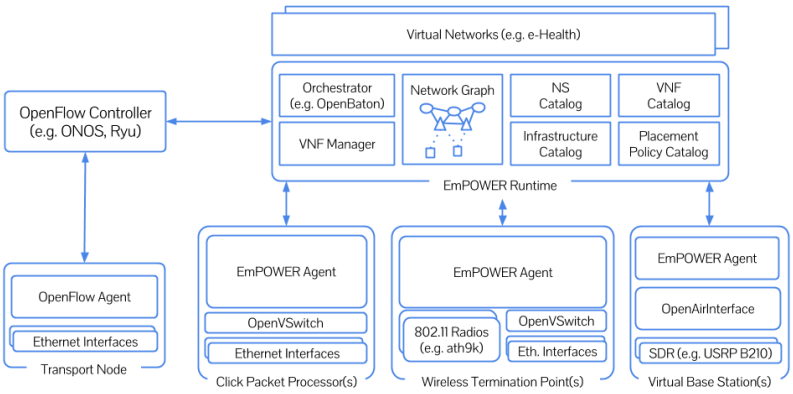

Figura 3.3: Arquitectura EmPOWER[24]

Luego de realizado el estudio, los desarrolladores estabilizaron el controlador EmPOWER y centraron sus esfuerzos en el desarrollo del agente de los CPP, escrito en Python. Finalizado el estudio se agregó el controlador Ryu, para realizar una gestión de la red *Intent-based*[13]. En la figura 3.4 se puede apreciar la arquitecura aproximada de la implementación actual.

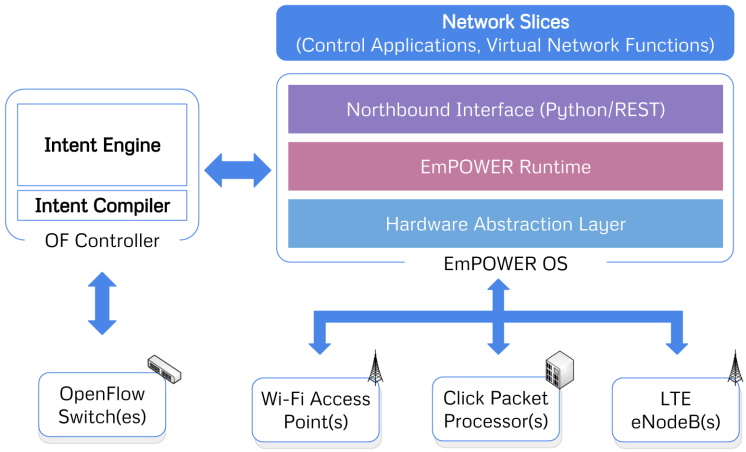

Figura 3.4: Arquitectura EmPOWER intent-based[24]

Entonces los flujos sobre los switches OpenFlow serían gestionados por el controlador Ryu. El agente para los CPP se comunicaría con el controlador EmPOWER y hablaría un protocolo análogo al de los LVAPs pero para definir LVNFs. Estos LVNFs se conectarían a puertos virtuales de los switches OpenFlow encontrados en los CPPs y se podría conectar la salida de una VNF con la entrada de otra mediante la API REST de EmPOWER. Las LVNFs en la practica son scripts de Click, distintos del código Click del agente.

3.3. Agentes EmPOWER para los WTP

Nuestro estudio se centró en el Controlador EmPOWER y los WTP, quienes nos proporcionaban un framework SDN y NFV en entornos inalámbricos. El agente de los WTP es implementado en Click, que se introduce a continuación.

3.3.1. Click Modular Router

Click[1] es una arquitectura de software para construir routers flexibles y configurables. Un router Click se arma con módulos que procesan paquetes de red llamados ***elements***. Estos *elements* se encadenan para manipular y configurar el flujo de los paquetes en un router. Los *elements* individuales se encargan de alguna función del router simple como clasificación de paquetes, encolamiento, planificación o hacer de interfaz con los dispositivos de red. En definitiva, un router Click es un grafo dirigido de *elements*.

Las propiedades más importantes de un *element* son:

- Clase del elemento: Cada *element* pertenece a una clase. Esto especifica el código que debe ejecutarse cuando el *element* procesa un paquete, así como el procedimiento de inicialización y la disposición de los datos
- Puertos: Un *element* puede tener cualquier cantidad de puertos de entrada y salida. Cada *connection* va desde un puerto de salida de un *element* a un puerto de entrada de otro. Diferentes puertos pueden tener distinta semántica, por ejemplo, los segundos puertos de salida se usan generalmente para dirigir paquetes erróneos
- String de configuración: El string de configuración es opcional y contiene argumentos extra que permiten moldear/configurar el *element* según sea necesario

Los paquetes entran por los puertos de entrada a los *elements* para ser procesados y éstos pueden enviar paquetes a través de sus puertos de salida donde eventualmente llegarán a una interfaz de salida.

Hay tres tipos de puertos que determinaran el tipo de conexión entre los *elements*:

1. **push** El *element* fuente inicia la transferencia. Las conexiones entre puertos de este tipo se llaman *event ased packet flow*. En los esquemas de flujos entre *elements* estos puertos se representan con un cuadrado o triangulo relleno. El cuadrado para los puertos de salida y el triángulo para los de entrada
2. **pull** El *element* destino solicita la transferencia del paquete. Las conexiones entre puertos de este tipo se llaman *flow-based router context*. En los esquemas de flujos entre *elements* estos puertos se representan con un cuadrado o triángulo vacío. Este tipo de conexiones "a demanda" entre *elements* se usan cuando se debe hacer *polling* o planificación (*schedulling*) de los paquetes
3. **agnóstico** Los puertos agnósticos pueden funcionar de forma *pull* al igual que *push*. En los esquemas de flujos entre *elements* estos puertos se representan con un doble cuadrado o triángulo. Y se rellenan (el triángulo

o cuadrado interno) o se dejan vacíos dependiendo si se transforman en *push* o *pull* respectivamente

Una configuración Click es un grafo dirigido con *elements* en los vértices. Los bordes, o *connections*, entre dos *elements* representan un posible camino para la transferencia del paquete, que como ya se mencionó pueden ser *pull* o *push*.

Las conexiones entre distintos puertos tienen algunas consideraciones.

- Un puerto *push* debe de estar conectado con un puerto *push* o agnóstico. Se puede hacer una conversión *push* a *pull* teniendo un *element* con una entrada del tipo *push* y una salida del tipo *pull*. Por ejemplo el *element* *queue*(encolamiento)
- Un puerto *pull* debe de estar conectado con un puerto *pull* o agnóstico. Se puede hacer una conversión *pull* a *push* teniendo un *element* con una entrada del tipo *pull* y una salida del tipo *push*. Por ejemplo el *element* *unqueue*(des-encolamiento).

En la figura 3.5 se pueden apreciar como funcionan los mecanismos *push* y *pull* y las conexiones entre los *elements*.

En definitiva Click consiste en conectar *elements* que representan una unidad conceptualmente simple de cómputo, tal como decrementar un campo IP TTL para lograr cómputo complejo y largo como el ruteo IP.

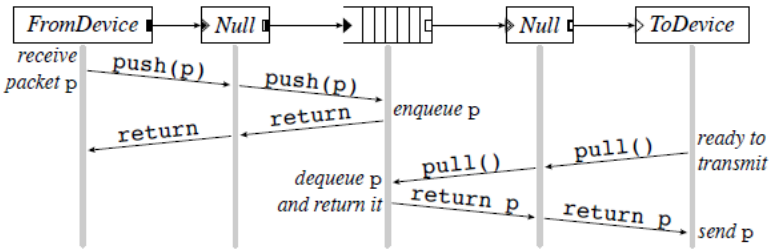

Figura 3.5: Diagrama de flujo entre elements[9]

Hay algunos *elements* básicos y necearios ya definidos por Click para cualquier implementación, como *elements* *FromDevice*, *ToDevice*, *FromHost* y *ToHost*. El objetivo de un router Click es encaminar los paquetes desde/hacia una interfaz de red hacia/desde un dispositivo de red.

Los *elements* *FromHost* y *ToHost* se utilizan para recibir y enviar los paquetes a alguna interfaz (la cual se establece al instanciar el *element*) y tendrán un puerto de salida de tipo *push* si es *FromHost*; y una entrada de tipo *pull* si es del tipo *ToHost*, es el *element* que recibirá/enviará los paquetes a la interfaz establecida en el sistema operativo del router.

Los *elements* *FromDevice* y *ToDevice* se utilizan para recibir y enviar los paquetes a través de un dispositivo de red, por ejemplo una antena WiFi o una interfaz Ethernet. Tendrán un puerto de salida de tipo *push* si es *FromDevice*; y

una entrada de tipo *pull* si es del tipo *ToDevice*. Son los *elements* que finalmente enviarán los paquetes a la red o los recibirán de la misma.

Por lo tanto es de esperar que haya por lo menos un camino en un esquema Click donde se reciban paquetes de un *element FromDevice* y se finalice el camino en un *ToDevice*, esto es, que reciba y envié paquetes por algún dispositivo de red. Cabe destacar que la flexibilidad de Click es debido a que éstos *elements* son reutilizables, al instanciar en un script de configuración Click con diferentes parámetros, por ejemplo se puede usar un *FromDevice* para cada dispositivo de red que tenga en el router, y hacer que tomen caminos distintos en los flujos de los *elements*.

En la implementación de Click los *elements* son objetos de C++ que puede mantener estado privado. Las *connections* son representadas como punteros a objetos *elements*, y pasar un paquete a través de una *connection* es implementado como una única función virtual. Los paquetes pasados entre los elementos son bytes en bruto, los cuales se pasan por referencia y se acceden mediante un puntero *struct* de C++. Esto hace que la implementación sea más eficiente al no tener que copiar la memoria de los paquetes al pasar de un *element* a otro.

Se puede obtener información de los *element* en tiempo real mediante funciones llamadas *handlers*. Estos *handlers* pueden ser llamados por otros *elements* o por un *socket*. Los *handler* son de tipo lectura o escritura, pero no ambos a la vez.

La configuración de un router Click se realiza mediante archivos de texto que especifica como estarán conectados los *elements* y como se inicializarán. Al iniciar Click, él mismo parsea este archivo y crea el grafo de elementos.

Click se puede implementar en macOS, Linux, BSD y parcialmente en Windows. Como lo utilizaremos en OpenWRT que es una distribución especifica de Linux para routers inalámbricos, nos centraremos en la implementación para ese sistema.

Click se puede implementar de dos maneras, como modulo de kernel o en el espacio de usuario.

En el modo kernel sobrescribe completamente el comportamiento de enrutamiento de Linux y funciona a alta velocidad, pero tiene como contrapartida que requiere permisos de root y en caso de falla, puede provocar un *crash* en el *kernel* y por ende en el sistema.

En el modo espacio de usuario, Click corre como un *deamon* (servicio) del sistema Linux. Es fácil de usar e instalar y a paser de correr en modo usuario es rápido[1]. Se recomienda usar cuando se esta experimentando o desarrollando un router.

3.3.2. Agente EmPOWER implementado con Click

El agente de EmPOWER se implementa como un grafo de *elements* Click. El grafo tiene como vértices de entrada y salida la interfaz de red del sistema (*empower0* por ejemplo) con el dispositivo inalámbrico (la antena) del AP. En otras palabras implementa el camino de datos inalámbrico de un WTP y se encarga de separar entre paquetes de datos y de control. El agente tiene un *element* llamado *EmPOWER LVAP Manager* (EL) que establece la conexión con el Controlador y maneja el protocolo EmPOWER, sin formar parte del camino de datos (grafo), pero se comunica con el resto de los nodos/*elements* ya que tiene la información de los LVAPs.

Podemos clasificar el camino de datos de dos formas distintas:

- **Paquetes recibidos desde el dispositivo de red (*FromDevice*)** Los paquetes provenientes del dispositivo de red, se clasifican según paquetes de datos o de control; previamente pasan por algunos *elements* como un filtro de paquetes duplicados y otro encargado de mantener datos estadísticos. Luego de la clasificación, si es un paquete de datos se dirige al *element* *EmpowerWifiDecap* previo a la salida por la interfaz (*ToHost*). El cometido de *EmpowerWifiDecap* es transformar los paquetes Wifi en paquetes Ethernet y descartar los paquetes cuyo BSSID no está asociado a un LVAP en el WTP. Para realizar este filtro se comunica con el EL. Cuando los paquetes son de control, se dirigen a un *element* específico según el tipo. Estos son reenviados al EL, para que el Controlador pueda tomar decisiones a través del protocolo EmPOWER.
- **Paquetes recibidos desde la interfaz (*FromHost*)** Los paquetes que provienen desde la interfaz (*FromHost*), pasan por un *element* llamado *EmpowerWifiEncap* que se encarga de transformar un paquete Ethernet en uno Wifi (con un cabezal LLC), adicionalmente se establece la BSSID apropiada según el LVAP que enviará el paquete (o descartará el mismo). Luego este paquete se encola para salir por el dispositivo de red *ToDevice*

3.3.3. Protocolo EmPOWER

A continuación se describirá el protocolo EmPOWER, esto refiere a los mensajes que se intercambiarán entre los WTP/CPP y el Controlador de EmPOWER. El protocolo EmPOWER contará con 6 grupos de mensajes:

Mensajes LVAP Mensajes para manejar los LVAPs en los WTPs

```
HELLO = 0x03            // wtp -> ac
PROBE_REQUEST = 0x04    // wtp -> ac
PROBE_RESPONSE = 0x05   // ac -> wtp
AUTH_REQUEST = 0x06     // wtp -> ac
AUTH_RESPONSE = 0x07    // ac -> wtp
ASSOC_REQUEST = 0x08    // wtp -> ac
ASSOC_RESPONSE = 0x09   // ac -> wtp
ADD_LVAP = 0x10         // ac -> wtp
DEL_LVAP = 0x11         // ac -> wtp
STATUS_LVAP = 0x12      // wtp -> ac
SET_PORT= 0x13          // ac -> wtp
STATUS_PORT = 0x14      // wtp -> ac
CAPS_REQUEST = 0x15     // ac -> wtp
CAPS_RESPONSE = 0x16    // wtp -> ac
```

Contadores Mensajes para solicitar los contadores de paquetes y bytes

```
COUNTERS_REQUEST = 0x17  // ac -> wtp
COUNTERS_RESPONSE = 0x18 // wtp -> ac
```

Estadísticas de enlace Mensajes para solicitar estadísticas del enlace

```
LINK_STATS_REQUEST = 0x29   // ac -> wtp
LINK_STATS_RESPONSE = 0x30  // wtp -> ac
```

Triggers Mensajes para activar los gatillos de eventos

```
ADD_RSSI_TRIGGER = 0x19     // ac -> wtp
RSSI_TRIGGER = 0x20         // ac -> wtp
DEL_RSSI_TRIGGER = 0x21     // ac -> wtp
ADD_SUMMARY_TRIGGER = 0x22  // ac -> wtp
DEL_SUMMARY_TRIGGER = 0x24  // ac -> wtp
SUMMARY_TRIGGER = 0x23      // ac -> wtp
```

Channel Quality Map Mensajes para obtener la información de la abstracción *Channel Quality Map*

```
UCQM\_REQUEST = 0x25     // ac -> wtp
UCQM\_RESPONSE = 0x26    // wtp -> ac
NCQM\_REQUEST = 0x27     // ac -> wtp
NCQM\_RESPONSE = 0x28    // wtp -> ac
```

Mensajes Internos Mensajes internos del controlador para dar de alta o baja WTPs y LVAPs

```
WTP\_BYE = 0x00
LVAP\_JOIN = 0x01
LVAP\_LEAVE = 0x02
WTP\_REGISTER = 0xFF
```

3.3.4. Camino de datos en EmPOWER

EmPOWER permite a cada *tenant* reservarse una porción de la red, para ello el usuario entra al portal de administración y solicita los CPPs y WTPs que usará, para que un administrador los habilite. Si bien se reserva los nodos de procesamiento y los WTP a usar, no se crean reglas OpenFlow en los switches OpenFlow incluidos en cada uno de manera que haya un correcto particionamiento de la red, al menos la momento de realizar esta investigación, si bien estaba pensado en la arquitectura original del testbed donde Floodlight se encargaría de la tarea.

Por lo que a futuro habría dos posibilidades, que se desarrolle un controlador OpenFlow específico de EmPOWER que cree las reglas OpenFlow adecuadas, o que para manejar la red cableada con OpenFlow se deba usar una herramienta ya existente.

De ser esta última la alternativa, las aplicaciones deberán usar el *northbound* de EmPOWER para gestionar los aspectos inalámbricos y el framework que maneje los switches OpenFlow para el camino de datos cableados, ya sea para un correcto particionamiento de la red o para cualquier otro aspecto de SDN.

Sobre el final del trabajo se encontró en el repositorio de EmPOWER un controlador OpenFlow basado en Ryu, orientado a una implementación *intent-based*, el cual parece basado en el trabajo *Intent–based Mobile Backhauling for 5G Networks*[23]. Por lo tanto se utilizó una solución mixta, ya que se usa un controlador existente -Ryu- para el *backhaul* cableado, con una aplicación sobre el mismo que presenta primitivas *intent-based* a EmPOWER.

A nivel de los CPP/WTP, por defecto se tiene el camino de datos procesado por Click, quien esta conectado a un switch virtual Open vSwitch. Este switch virtual se pone en modo *bridge* con por lo menos una interfaz física del *backhaul*, la interfaz del dispositivo inalámbrico y el switch interno de Open vSwitch.

En el caso de nuestro despliegue se uso un switch para el *backhaul*, por lo que la configuración del switch nos quedo de la siguiente manera:

```
Bridge br-ovs
Controller "tcp:192.168.9.6:6633"
Port br-ovs
    Interface br-ovs
        type: internal
Port "eth0.2"
    Interface "eth0.2"
Port "empower0"
    Interface "empower0"
```

La interfaz virtual *empower0* correspondiente a la antena de radio se pone en modo monitor de manera de poder ver todos los paquetes que llegan a la misma. El agente Click es el que procesa dichos paquetes separando el plano de control del plano de datos. Los datos siguen hacia el switch OpenFlow, y debido a que funciona como *switch tonto* salen directamente al *backhaul*, debido a que aún no hay controlador OpenFlow que ponga reglas en el switch. Si es un paquete de control como un *beacon* o un *probe request*, el mismo es enviado al Controlador EmPOWER, ya que Click mantiene una conexión con el mismo. En nuestro caso la conexión se realiza por el mismo switch físico que van los datos generales, pero se podría realizar la conexión por una boca Ethernet dedicada.

También mediante las VNF, se pueden crear puertos virtuales en el switch OVS, y al recibir paquetes serán procesados por scripts Click que son las VNF en sí mismos. Estas VNF podrán ser conectados entre sí mediante la API REST de EmPOWER, aunque al momento de realizar el estudio no se encontraban desarrollados los agentes LVNF en los CPP ni la compatibilidad para los mismos en los agentes de los WTPs.

Los switches virtuales definidos en cada nodo, se conectan con un controlador OpenFlow en el puerto 6633 en caso de estar definido.

Si bien al momento de realizar el estudio no se encontraban los agentes de los CPP implementados, están definidos los mensajes que intercambiarán los agentes de los CPP y EmPOWER para gestionar los LVNF.

3.4. Controlador EmPOWER

En esta sección se describirán dos aspectos generales en la construcción de las redes EmPOWER, como son las distintas APIs de desarrollo y las aplicaciones

que se encuentran por encima del Controlador implementadas y disponibles para su uso.

EmPOWER brinda dos APIs para su desarrollo, la *API Python* y la *API Rest*.

3.4.1. API Rest

EmPOWER cuenta con una API Rest, que consiste en un conjunto de recursos RESTful y sus atributos.

La URL base de la API es:

http:nombre_ de_ usuario:password@nombre_ de_ host:puerto/api/v1/

La API Rest facilita varias funciones de control y administración de los elementos de la red EmPOWER. Permite visualizar, agregar y borrar los WTPs, CPPs, *tenants* y LVAPs registrados en el Controlador; y de la misma forma los WTPs, CPPs, LVAPs, LVNFs asociados a un *tenant* particular. También manejar los ACLs, por medio de filtros MAC, manteniendo dos listas distintas, dispositivos "permitidos" y dispositivos "no permitidos". En caso de tener ambos registros vacíos, por defecto permite a todo dispositivo vincularse a la red.

Para acceder a la documentación existente de esta API ver el siguiente enlace: [27]

3.4.2. API Python

Una aplicación de red se puede ver como un módulo *Python* cargado y consiste básicamente en una clase con un método *launch*. Todas las aplicaciones de EmPOWER heredan de la clase *EmpowerApp*, y el método *launch* por lo tanto debe retornar una instancia de dicha clase, especificando como parámetro el *tenant_ id* y también puede ir acompañado de un parámetro que determine el período de ciclo de la aplicación.

A continuación hablaremos de algunas primitivas que proporciona EmPOWER, éstas primitivas no son bloqueantes para con la aplicación y agregándoles métodos *callback* pueden modelarse ciertos comportamientos deseados (en caso de ser *pollers* o *triggers*) ejecutándose al momento que se retorna de la primitiva.

- *counters* Un contador de paquetes permite tener control del tráfico de un determinado LVAP y además ordenarlo según información del tamaño de paquete.
- *lvap_ stats* Permite acceder a la información de transmisión (exitosas y no, *throughput* esperado) de cierto LVAP desde que inicia su vínculo con el respectivo WTP.
- *lvapjoin* y *lvapleave* Controlar los eventos de conexión y desconexión de los LVAP permite poder modelar comportamiento activando los métodos *callback* respectivos.
- *wtpup* y *wtpdown* Controlar los eventos de alta y baja de un WTP permite poder modelar comportamiento activando los métodos *callback* respectivos.

- UCQM y NCQM *User/Network Channel Quality Maps* permite consultarle a un WTP sobre sus LVAPs vecinos (con UCQM) como otras redes (con NCQM) para un cierto *resource block* -en nuestro caso el canal de frecuencia-
- *rssi* Esta primitiva activa un *trigger*, en caso que el valor de RSSI de un cierto LVAP se ubica por debajo de un umbral enviado por parámetro se llama a un método *callback*.
- *summary* Esta primitiva permite acceder a un resumen de información de la capa de enlace, a partir de ciertos parámetros como un período de tiempo y cantidad de comunicaciones de un cierto WTP y hasta permite realizar un filtrado por MAC.

Para acceder a la documentación existente de esta API se puede acceder al siguiente enlace: [26]

3.4.3. Aplicaciones implementadas sobre el Controlador

EmPOWER posee algunas aplicaciones sobre el Controlador disponibles para su uso, programadas a partir de las primitivas antes mencionadas.

La aplicación *ConflictGraph* realiza un mapeo de todos los enlaces de interferencia entre las entidades de la red, pudiendo ser tanto clientes como APs. Agrega dichos enlaces en lo que llama *conflict map*, intercalando entre todos los LVAPs registrados, agregando al receptor del enlace sí se encuentra en el rango de interferencia del transmisor.

Joule es otra aplicación disponible programada para medir el consumo energético de los dispositivos. Estima el consumo de los APs a partir de información como el tráfico, de donde se puede establecer una relación con el consumo.

También se cuenta con aplicaciones de utilidad, tales como registradores de eventos (altas y bajas de WTPs, CPPs, clientes, etc), *trackers* (RSSITracker y PowerTracker) que imprimen la información en un archivo tomada de forma periódica del RSSI y el consumo energético, también *pollers* con el que implementa distintos tipos de contadores.

La aplicación que encontramos más trascendente dentro de las ofrecidas por EmPOWER, es el *Mobility Manager*.

Descripción *Mobility Manager*

Concretamente, describiremos la aplicación *Mobility Manager* que implementa el *handover* sobre EmPOWER. La elección de esta aplicación fue debido a la importancia de dicho mecanismo para las redes inalámbricas.

Dicha aplicación deberá medir el *Link Quality* de cada LVAP activo en la red, chequeando que la conexión no se deteriore y el *Link Quality* baje de cierto umbral límite, a partir del cuál se ejecutará un trigger para realizar el *handover*. De forma paralela y periódica (ciclos cada cierto tiempo), la aplicación realizará el *handover* de cada LVAP activo hacia el WTP que presente mejor *Link Quality*.

Para poder ejecutar ambas acciones, la aplicación debe activar dos triggers distintos:

- *lvap_join* trigger, se ejecuta cada vez que un nuevo LVAP se conecta a la red y activa a su vez al trigger *low_rssi* que se ejecuta cuando el RSSI

del LVAP baja el límite determinado, generando entonces lo que llama un *handover* reactivo

- *wtp_ up* trigger, se ejecuta cada vez que un nuevo WTP se conecta con el Controlador y este ejecuta la primitiva *ucqm*, necesaria para medir la visión de la red (LVAPs en este caso) y a partir del tiempo determinado de cada ciclo, evaluar y generar el cambio al mejor WTP, en lo que llama un *handover* proactivo

Los LVAPs están expuestos mediante un objeto Python ("LVAP") y éste provee una interfaz que permite saber en que *Resource Block* se encuentra un LVAP, accediendo a la propiedad *schedule_ on* del mismo objeto. Por lo tanto, la implementación del *handover* radica en asignar otro *Resource Block* a este campo del objeto.

En el caso del *handover* proactivo, en cada ciclo se evalúa el conjunto de *Resource Blocks* disponibles para la partición de red del *tenant*, generando una lista de aquellos que cumplen la condición de superar el umbral de RSSI determinado como límite. De esta forma, siempre y cuando luego de pasar este filtro, queden *Resource Blocks* en la lista, se cambia al que obtenga mejor valor de RSSI.

Prueba realizada Para realizar la prueba del *Mobility Manager*, se dispuso de un despliegue de APs a lo largo del edificio del InCo (Instituto de Computación) de la Facultad de Ingeniería (ver mejor descripción en capítulo de pruebas).

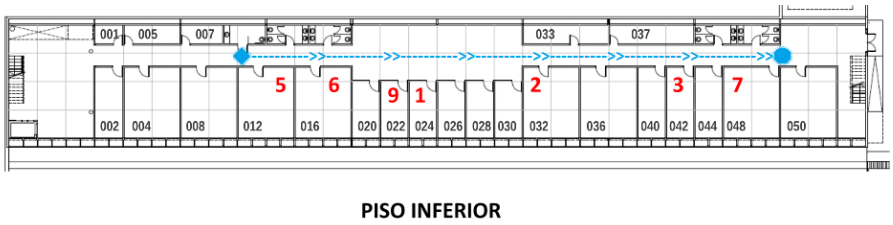

Figura 3.6: Ubicación de los routers en el piso inferior del edificio. Los números identifican cada router.

La prueba consistió en ejecutar la aplicación *Mobility Manager* (con un umbral de -30dBm y un tiempo por ciclo de 5 seg). Conectando un dispositivo móvil a una red generada por EmPOWER, se realizó un recorrido en el cuál se variaba la cercanía con cada WTP conectado al Controlador.

El recorrido establecido se puede visualizar en la figura anterior. Efectivamente a medida que el dispositivo móvil se iba aproximando a los WTPs del recorrido, generaba un *handover*. Dicho cambio se generaba aproximadamente cada 5 seg, o sea que se producía un *handover* proactivo. Debido a la cercanía de los WTPs tiene sentido esta apreciación, ya que a pequeñas distancias recorridas es factible que se active este tipo y no se realice el cambio debido a bajo RSSI.

De esta forma, se pudo constatar el funcionamiento de EmPOWER desde una perspectiva práctica, complementando el estudio teórico y probando una utilidad de la herramienta. A partir de este procedimiento es que nos planteamos extender el framework.

Capítulo 4

Extensión de EmPOWER

Luego de describir detalladamente la herramienta y probar su utilidad, para un completo estudio de la misma resta extenderla y sumarle valor. Al momento, EmPOWER no brinda la posibilidad de realizar cambios de canal en los APs, es en este aspecto que se pretende extender la herramienta, brindando funcionalidades que permitan configurar la red de forma óptima en cuanto a las interferencias de las frecuencias (canales).

Las nuevas funcionalidades serán provistas mediante una API -extendiendo la brindada por el framework- que le permita funcionar desde las aplicaciones que corren sobre el Controlador EmPOWER. Finalmente, las asignaciones de canal se realizarán implementando ciertas estrategias estudiadas en la Tesis de Maestría *Self Management of High Density Wireless Networks* de Guillermo Apollonia[20].

El algoritmo de asignación de canales será implementado en forma de aplicación como tiene previsto el framework.

A continuación se describirán conceptos previos necesarios para posteriormente profundizar sobre los detalles de la aplicación implementada, *ScanApp*.

4.1. Descripción de conceptos previos a la implementación

Es de relevancia definir ciertos conceptos clave previo a la descripción de la implementación. Entre ellos encontramos los distintos tipos de interferencia, las estrategias planteadas por Apollonia[20] para su minimización, la descripción formal del problema y el fundamento teórico de su solución.

4.1.1. Tipos de Interferencia

Interferencia del mismo canal La interferencia del mismo canal ocurre entre dos APs que se encuentran en la misma frecuencia de canal. Este tipo de interferencia puede afectar de forma severa el desempeño de la red inalámbrica. El espectro utilizable para el despliegue del WiFi es limitado. La interferencia del mismo canal es más problemática cuando se realiza un despliegue denso de APs sobre la red inalámbrica, esto significa que los APs están más cerca entre ellos, creando una situación propicia para que dos dispositivos que transmitan

en la misma frecuencia estén lo suficientemente cerca para causar interferencia mutua significativa.

Interferencia de canales adyacentes Los canales adyacentes son aquellos cuya banda de frecuencia de radio son contiguas. Los canales contiguos se solapan entre ellos porque cada canal tiene un ancho de 22 MHz y sus respectivos centros de frecuencia están separados por tan solo de 5 MHz (802.11b/g). La interferencia de canales adyacentes sucede cuando dos o más APs usan canales solapados y están colocados cerca uno del otro. Este tipo de interferencia puede degradar severamente el rendimiento de la red inalámbrica.

Para estos problemas de interferencia se usan comúnmente dos soluciones, separar aquellos APs que transmiten en canales adyacentes lo suficientemente lejos para reducir el solapamiento o utilizar únicamente canales que no se solapen.

4.1.2. Estrategias propuestas

Se pretende minimizar la función denominada $Lran$ (función objetivo). Esta función está definida para minimizar el nivel de interferencia total visto desde los APs. Para calcular $Lran$ debemos definir el término *Interference-level*, escrito como $Il(ap_i, ap_j)$ que describe el nivel de interferencia entre el AP_i y AP_j. Apollonia propone tres estrategias diferentes para minimizar el nivel de interferencia, con la intención de evaluar la implementación práctica y escalabilidad de cada una. Las propuestas pueden ser clasificadas de la siguiente forma:

1. Local y Descoordinada
2. Distribuida y Coordinada
3. Centralizada

Para la Local y Descoordinada, se toma como base *Least Congested Channel* (LCC), los conceptos básicos son:

- Cada AP periódicamente chequea la información de transmisión en el canal que usa
- Si el volumen de tráfico en el canal (generado por otros APs o sus clientes) es mayor que cierto umbral, entonces dicho AP intenta cambiar a un canal menos congestionado

Para la estrategia Distribuida y Coordinada, se determina un líder que va a coordinar la ejecución del protocolo distribuido. A pesar de que la elección del líder puede ser tomada a partir de un algoritmo distribuido, en esta implementación el líder es determinado estadísticamente durante la inicialización del sistema. Una vez que el líder es determinado, entonces este mismo es el encargado de iniciar el proceso. Coordinará un algoritmo distribuido que consiste de dos pasos:

- Primero, un proceso de rankeo, cuando dicho proceso culmina, cada nodo está rankeado según su nivel de interferencia (con enfoque descendente)
- Segundo, en el orden determinado por el rankeo, se le pide a cada nodo que se auto asigne un canal conociendo la asignación de sus vecinos

Esta es una implementación Greedy del algoritmo.

Finalmente, en la implementación Centralizada, un nodo adquiere información de todos los otros en el sistema y sus niveles de interferencia. Entonces este usa un algoritmo de backtracking para obtener la solución óptima en un tiempo razonable. Luego de que el algoritmo está completado, el nodo central le informa a los demás en que canal deben operar.

4.1.3. Definición formal del problema

En esta sección describiremos formalmente el problema a resolver, tomando en cuenta las distintas restricciones que se presentan según las circunstancias planteadas. Tomando k como el número total de canales disponibles que no se solapan, la primera restricción del problema surge de trabajar bajo el estándar 802.11b/g, donde k se iguala a 3 (los canales 1, 6 y 11). Luego, definimos un grafo $G = (V, E)$ donde $V = ap_1, ap_2, \ldots, ap_n$ el conjunto de APs que forman la red. A partir de estos n APs, hay un subgrupo que están bajo nuestro control y otro subgrupo que están en el ambiente distorsionando nuestros nodos. Este último subgrupo afecta nuestra realidad pero no corren nuestros algoritmos. La segunda restricción puede describirse de la siguiente forma: tenemos control sobre un subgrupo V' de V, en otras palabras, podemos realizar el cambio en la asignación de canal sobre estos nodos. Aquellos que pertenecen al grupo $V - V'$ interfieren con los del subgrupo V' y ya tienen un canal asignado que no puede ser cambiado. El conjunto E de aristas, se conformará entre aquellos APs que se interfieran, si ap_j puede interferir en ap_i entonces es muy probable que ap_i interfiera en ap_j también, aunque esta suposición puede ser falsa en ciertos casos. La tercera restricción es que el grafo es dirigido.

Se puede decir entonces, que el problema de asignación de canal $C(ap_i)$ donde $ap_i \in V'$ es un mapeo $C : V' \to \{1...k\}$ desde el subgrupo de APs que podemos asignar el canal hacia el conjunto de canales disponibles.

Definimos el *Interference-level* entre dos APs $Il(ap_i, ap_j)$, como el SNR captado del nodo ap_i con respecto a ap_j.

Finalmente presentamos la función objetivo: $Lran$ minimiza el nivel de interferencia total visto desde los APs.

Minimizar:

$$Lran = \sum_{\forall (ap_i, ap_j) \in E \wedge ap_i \in V'} Il(ap_i, ap_j)$$

Existe una cuarta restricción en nuestro problema que no fue mencionada anteriormente sobre las implicaciones prácticas; la cantidad de nodos que se consideran para este estudio se ubica en el orden de decenas, no se piensa para ambientes donde existen cientos o miles de APs, por la sencilla razón de que está pensado para lugares como escuelas, halls, oficinas, etc., lugares de tamaño reducido.

4.1.4. Diseño de la solución Centralizada

Debido a la naturaleza de este algoritmo, el nodo central debe tener suficiente capacidad de cómputo para poder ejecutar sin problemas el backtracking, por lo

que debe ser un notebook, PC o un router de mayor capacidad que los domésticos. El nodo central envía un mensaje broadcast para que todos los otros nodos le envíen la información necesaria. Cuando el nodo central recibe la respuesta de todos los demás nodos, comienza entonces con la ejecución del backtracking. Debido a la cuarta restricción descrita anteriormente sobre la cantidad de nodos en la red, se asegura que el algoritmo de backtracking no demandará grandes tiempos. El algoritmo se implementa usando tuplas de largo fijo de la forma $< X_0, X_1, ..., X_{n-1} >$, donde $n = |V|$, cada componente X_i representa el canal asignado al vértice $i, \forall i \in (0, n-1)$.
Y tenemos las siguientes restricciones explícitas:

- $X_i \in (1, 6, 11), \forall i \in (0, n-1)$
- $\forall i$, conocemos los nodos externos y los canales C_i utilizados por ellos

La función objetivo es $f = min(interference(G))$, donde $interference(G)$ es la suma de las interferencias producidas dentro de la red para una posible tupla solución teniendo en cuenta también los nodos "externos". El algoritmo intenta construir todas las soluciones posibles, pero en cuanto una solución es mayor que las anteriores halladas, el algoritmo desecha la sección del árbol solución asociado a esta. Luego que finaliza el algoritmo, el canal en el que cada nodo debe operar está almacenado en una estructura y el nodo central itera hasta enviarle la información a todos los demás solicitando la nueva asignación de canal.

4.2. Implementación de la aplicación ScanApp sobre EmPOWER

Luego de realizado el estudio sobre las estrategias planteadas por Apollonia, nos resta entonces analizar que herramientas nos brinda las abstracciones de EmPOWER y cuales faltan para poder implementar dicha aplicación sobre el controlador. De esta forma, definimos etapas claras para el desarrollo de la implementación:

1. Obtención de las métricas necesarias para el cálculo de la interferencia
2. Cambio de canal de los APs en EmPOWER
3. Análisis de las métricas obtenidas para implementar los algoritmos planteados

Obtención de las métricas necesarias para el cálculo de la interferencia Tal como se presentó en secciones anteriores, las estrategias se basan en minimizar la interferencia total de la red, compuesta por la sumatoria de la interferencia de cada nodo. Para hallar la interferencia de cada nodo, se utiliza como métrica la calidad de señal (o *Link Quality*), por lo tanto el cálculo total resulta sencillo luego de obtener esta información.

Dentro de las abstracciones de EmPOWER, como se describió anteriormente, existe una primitiva -*Channel Quality Map*- para obtener información de la intensidad de señal sobre el resto de los LVAPs o WTPs (RSSI) presentadas en

forma de matrices tri-dimensionales respectivamente (UCQM/NCQM - *User/-Network Channel Quality Maps*), para cada WTP en un determinado canal de frecuencia.

Esta información que brinda EmPOWER presenta varias dificultades en función de los datos necesarios para los cálculos previstos (especificados en mayor detalle en la próxima sección de este informe), entre ellos destacamos la diferencia del RSSI conceptualmente con la métrica utilizada por Apollonia (*Link Quality*, cuyas diferencias serán explicitadas detalladamente más adelante) y además la necesidad de iterar sobre los distintos canales para obtener la información de la primitiva en base a cada uno. Debido a estas dificultades se optó por no tomar la información de la interferencia desde la abstracción presentada por EmPOWER.

Finalmente se decidió utilizar el resultado obtenido del comando *iwinfo* para escanear las redes y calidad de señal en cada uno de los canales. Para agregar esta funcionalidad dentro de la herramienta modificamos el controlador, el agente, y el protocolo de comunicación entre estos. Para del controlador agregamos:

- La función *send_scan_request*, encargada de enviar un mensaje *EmPOWER_PT_SCAN_REQUEST* al agente para iniciar el escaneo.
- El evento *scanreceived*, por medio del cual las aplicaciones reciben los resultados del escaneo solicitado con la función anterior. Dentro de los datos retornados se incluye el WTP al que pertenecen los datos. Este evento es disparado en el controlador cuando llegan mensajes del tipo *EmPOWER_PT_SCAN_RESPONSE*.

En el agente, agregamos el manejo para los mensajes *EmPOWER_PT_SCAN_REQUEST* y el envío de los resultados hacía el controlador. Para cada una de las redes reportadas por el comando *iwinfo* los datos enviados hacía el controlador son el BSSID, SSID, canal y calidad de señal.

Cambio de canal de los APs en EmPOWER Para realizar el cambio de canal agregamos en la API la función *send_set_channel*, esta envía un mensaje del tipo *EmPOWER_PT_SET_CHANNEL* al agente incluyendo el canal al que se quiere asignar la interfaz inalámbrica. En el agente, una vez recibido el mensaje, se ejecuta el comando *iw* y se notifica al controlador para mantener sus estructuras sincronizadas. El tipo de este mensaje de respuesta es *EmPOWER_PT_SET_CHANNEL_RESPONSE*.

Análisis de las métricas obtenidas para implementar los algoritmos planteados Llegado a este punto, ya se obtuvo toda la información y se implementaron las funciones necesarias para poder programar una aplicación sobre EmPOWER que evalúe los datos de interferencia y que posteriormente cambie el canal de los APs de la red. En definitiva, ya se cuenta con todas las herramientas necesarias para poder implementar la aplicación sobre el Controlador EmPOWER en base a las estrategias estudiadas. Comienza entonces el análisis sobre cuales estrategias implementar según las características propias del framework y la arquitectura que estamos utilizando.

La estrategia *Local y Descoordinada* es aplicable en un principio bajo cualquier arquitectura de red, ya que simplemente calcula los valores de interferencia de manera local en cada AP, sin considerar ni comunicar cambios en el resto de

la red. Por lo tanto, la implementación de dicha estrategia sobre EmPOWER no tiene mayor sentido ni sumaría valor agregado a la herramienta.

En cambio, la estrategia *Distribuida y Coordinada* requiere de un mayor análisis que la anterior, debido a la necesidad de comunicación entre los APs y de una entidad central (el nodo líder para este caso). Claramente la arquitectura de EmPOWER contempla las restricciones que se plantea esta estrategia para funcionar a nivel de Conectividad y a diferencia de la implementación aplicada por Apollonia, EmPOWER permite abstraerse del protocolo de comunicación necesario entre los APs. Otro aspecto que se resuelve fácilmente utilizando EmPOWER, es la elección del nodo líder, naturalmente adjudicado al controlador.

Esta estrategia se basa en el algoritmo de rankeo, donde cada nodo le envía su información al nodo líder y este determina un orden (decreciente según interferencia) para posteriormente indicarle a cada uno que realice el escaneo y cambio de canal respectivo. Esto requiere un primer escaneo por parte de cada nodo, el envío de información al líder. Posteriormente, volver a realizar otro escaneo para calcular nuevamente la interferencia según los cambios que hayan acontecido en la red, según el turno de cada nodo, más nuevamente el envío de información al líder para notificar acciones y que el algoritmo prosiga con el siguiente. La decisión sobre el canal sigue siendo tomada localmente, simplemente en este caso los nodos se coordinan y ordenan para tomar acciones, por lo que la arquitectura de EmPOWER y las capacidades del controlador quedan únicamente sujetas a proveer el protocolo de comunicación necesario para dicha coordinación. Nuevamente se puede concluir que la implementación de esta solución sobre EmPOWER no conlleva un aporte significativo, entonces queda descartada.

Por último, la estrategia *Centralizada* parece ser la más lógica de implementar ya que el controlador de EmPOWER brinda acceso a los APs conectados y resuelve la comunicación con éstos, a través del protocolo EmPOWER. Además, el controlador posee las capacidades de cómputo necesarias para ejecutar el algoritmo de backtracking.

De esta forma entonces culmina el análisis previo y simplemente resta implementar la solución, por lo que a nuestra separación por etapas previamente mencionada, se agrega una más:

1. Obtención de las métricas necesarias para el cálculo de la interferencia
2. Cambio de canal de los APs en EmPOWER
3. Análisis de las métricas obtenidas para implementar los algoritmos planteados
4. **Implementación de estrategia centralizada**

Implementación de estrategia centralizada En esta sección se describirá la aplicación implementada sobre el Controlador EmPOWER, *ScanApp*. Las siguientes etapas describen un ciclo de la aplicación, que se ejecutará bajo un loop determinado.

Obtención de los datos A diferencia de la implementación original el escaneo de los APs se realiza en forma secuencial. Una vez que se envía el mensaje

solicitud al WTP, se espera la respuesta del mismo para proseguir con el siguiente. De esta forma se obtienen los datos de cada WTP y al obtener la respuesta del último es que culmina esta etapa. El conjunto de WTPs que ejecutarán el escaneo, serán aquellos que se encuentren conectados al controlador.

Generación de las estructuras En esta etapa, a partir de los datos obtenidos en la anterior, se generan las estructuras necesarias para la ejecución del algoritmo. El algoritmo requiere clasificar los datos, aquellas redes bajo el dominio de EmPOWER de las que existen en el entorno. Esta distribución de los datos obtenidos se realiza por medio de una función denominada *isInDomain*. Para clasificar las redes reportadas por el escaneo según las generadas por EmPOWER o no, la función busca la existencia de un LVAP registrado en el controlador utilizando su BSSID.

Ejecución del algoritmo de backtracking Luego de realizar la clasificación, el algoritmo de backtracking comienza a evaluar las distintas tuplas-solución $tupla = [c_1, \ldots, c_n]$ donde c_i refiere al canal del WTP_i siendo n la cantidad total de WTPs. La tupla resultado, será aquella que obtenga el menor valor de interferencia total, calculado según el supuesto canal que adoptaría el WTP con el canal de las redes que este WTP visualiza. Este valor se obtiene multiplicando un factor al *Link Quality* del escaneo. Dicho factor dependerá de la diferencia entre los canales anteriormente mencionados.

Cálculo del *factor*		
factor	$dif_{canales}$	Ejemplo ($canal_i$,$canal_j$)
1.0	0	(1,1)
0.8	1	(1,2)
0.6	2	(1,3)
0.4	3	(1,4)
0.2	4	(1,5)
0.0	5	(1,6)
0.0	+5	(1,7)

Asignaciones de canal Finalizado el backtracking, la aplicación cuenta ya con los canales óptimos para cada WTP, entonces procede a enviar un mensaje a cada uno para que efectivamente se realice el cambio de canal respectivo. De esta manera culmina el ciclo de la aplicación.

4.3. Problemas e inconvenientes durante la implementación

Aquí nos explayaremos en las distintas dificultades y las soluciones encontradas a lo largo de la implementación de la aplicación. De la misma forma, se detallarán algunas opciones que fueron manejadas y realizadas durante el proyecto pero posteriormente se descartaron, pero que de igual forma constituyen una pieza del trabajo de investigación.

4.3.1. Comando iwinfo

iwinfo[31] es una utilidad brindada por OpenWRT, que ensambla información de distintos lugares. Al utilizar el comando *iwinfo* en los agentes surgió una dificultad: el escaneo no puede realizarse con la interfaz que utiliza EmPOWER, ya que debe estar configurada en Modo Monitor. Luego de investigar el código del *iwinfo*, se pudo concluir que internamente envía un comando al driver y es este quien devuelve los datos del escaneo, por lo tanto no existe forma de poder realizar la operación de manera directa. La solución propuesta a este inconveniente, radicó en cambiar el modo de interfaz y ejecutar el *iwinfo* para obtener los resultados requeridos, posteriormente se vuelve la interfaz al modo anterior. Cabe mencionar que implementando esta solución, se estaría cortando la conexión durante el lapso del escaneo, pero a efectos prácticos de este caso no implica grandes problemas, en especial si se tiene en cuenta que de todos modos la conexión se vería afectada al momento de asignar el nuevo canal.

4.3.2. Interrupciones en la conexión

Como se acaba de mencionar, tanto cuando se escanea como cuando se realiza la asignación de canal, se produce una interrupción de la conexión entre los LVAPs y el WTP en cuestión.

Motivos de interrupción:

- Interrupción por escaneo: Ésta se produce por el motivo expuesto en el apartado anterior; para poder utilizar el comando *iwinfo*, se debe realizar un cambio de modo en el AP que indefectiblemente produce un corte en la conexión.

- Interrupción por asignación de canal: De la misma forma, el último paso del algoritmo consiste en asignar los canales a los WTPs. Este suceso también conlleva una interrupción que no puede ser transparente al cliente. El canal es uno de los elementos del *Resource Block*, por lo que un cambio de canal implica un cambio de *Resource Block* en la propiedad *schedule_ on*, derivando en el corte de conexión.

4.3.3. Relación entre RSSI y Signal Quality

Todos aquellos que trabajan sobre IEEE 802.11 están de acuerdo en que el *signal quality* (o calidad de señal) y el *signal strength* (o fuerza de señal) son las métricas adecuadas para evaluar la señal de frecuencia de radio.

El RSSI (Received Signal Strength Indicator), como bien indica su nombre, es una medida que indica el *signal strength* definido en el estándar 802.11. Es un valor entero (de tamaño un byte), por lo tanto su rango es [0,255], aunque cada proveedor utiliza esta escala arbitrariamente. No está asociado con ninguna otra escala particular, es usado internamente por el microcódigo del adaptador y los drivers. Por esta razón los proveedores no están obligados a usar un estándar compatible. Es así, como las medidas de RSSI no son consistentes entre distintos hardwares.

Por otra parte, el *signal quality* puede definirse como una medida de correlación entre la señal DSSS que llega y la señal DSSS ideal. Esta medida refleja la

cantidad de señal dentro del canal formado entre dos elementos en comunicación, por ejemplo un AP y un cliente.

Existen cuatro distintas unidades de medida utilizadas para representar la fuerza de las frecuencias de radio: mW (miliwatts), dBm (decibeles-miliwatts, basado en mW), RSSI y una medida según porcentajes.

Cuando es usado el RSSI para reportar en *signal strength* (dBm), es común verlo representado como un porcentaje, obtenido a partir del valor $RSSI\_MAX$ utilizado por el proveedor y una regla de tres para obtener el respectivo porcentaje según este valor tope. Aunque no todos utilizan este mapeo del RSSI sobre porcentaje, por lo que no permiten una comparación entre distintos proveedores.

El comando *iwinfo* obtiene su *Link Quality* a partir del RSSI, aplicando la siguiente relación:

RSSI a Quality	
RSSI	Quality
$(-, -110)$	0
$[-110, -40]$	(RSSI + 110)
$(-40, 0)$	70
$[0, +)$	RSSI

4.3.4. Problemas en la implementación de la solución centralizada

El manejo de los LVAPs sumado a la problemática antes detallada del modo en que EmPOWER configura los APs, trajeron algunas consecuencias en lo que respecta a la implementación del algoritmo.

Escaneo en los APs La alternativa de realizar el cambio de modo en los APs para poder utilizar el comando *iwinfo*, atrajo consigo otro inconveniente en la ejecución del algoritmo. La solicitud de escaneo por parte del Controlador a los APs se debería realizar en paralelo, por la sencilla razón de que este escaneo debe representar una "foto" del estado de la red en ese preciso momento, a partir del cuál se procesa la información y se ejecuta el backtracking. Durante el tiempo que se cambia la interfaz del AP para realizar el escaneo, el agente pierde la capacidad de capturar y contestar los paquetes que llegan por dicha interfaz. Específicamente, durante este tiempo no se estarán reportando las tramas beacons que informan de los LVAPs presentes en el AP. Por lo tanto, si los escaneos se ejecutan simultáneamente en todos los APs, no se reportaran los LVAPs en el resultado, perdiendo información clave para el cálculo de la interferencia propia de la red EmPOWER.

Alternativa aplicada Para contrarrestar este problema, se decidió realizar un escaneo secuencial, el Controlador le envía la solicitud uno a uno, esperando la respuesta de cada AP para enviar la solicitud al siguiente. Esta implementación altera la representación del estado de la red. Por lo tanto se creó una nueva estructura denominada *LVAP Cache*, que almacena los LVAPs registrados en el Controlador al momento de cada escaneo, para minimizar los efectos del escaneo secuencial.

Manejo de LVAPs escaneados Bajo las condiciones de EmPOWER, para saber si dos APs propios se generan interferencia entre sí, basta con que se capten los respectivos LVAPs de cada uno, y en caso de no registrar ninguno, directamente no existiría dicha interferencia. Pueden captarse varios LVAPs de un mismo AP, por lo tanto, se debe tener el recaudo de utilizar uno de ellos para el cálculo de la interferencia para no alterar los resultados. La explicación de esta consideración radica en que la interferencia en nuestro caso se calcula a partir de la potencia con la que un AP ve a otro, y con un LVAP ya es suficiente para obtener este dato.

Capítulo 5

Pruebas del Prototipo

En esta sección se describirán las distintas pruebas realizadas sobre la aplicación *ScanApp* (Sección 4.2), a modo de probar el funcionamiento de la estrategia implementada.

Podemos clasificar las pruebas en tres tipos:

1. Pruebas de concepto
2. Pruebas de despliegue
3. Pruebas de utilidad

Para cada una de éstas pruebas, se detallará sobre los motivos que la generaron, los resultados de la misma y un posterior análisis. Vale aclarar que dichas pruebas corresponden a un estudio funcional del algoritmo utilizado en la aplicación y no un estudio empírico; por lo que el objetivo de las mismas es mostrar y analizar el comportamiento del algoritmo frente a distintos escenarios planteados, quedando fuera del alcance de las mismas un estudio sobre los beneficios (o no) de la asignación de canales implementada por el algoritmo en la red.

El despliegue se realizó en el Edificio del InCo (Instituto de Computación) de la Facultad de Ingeniería (Universidad de la República).

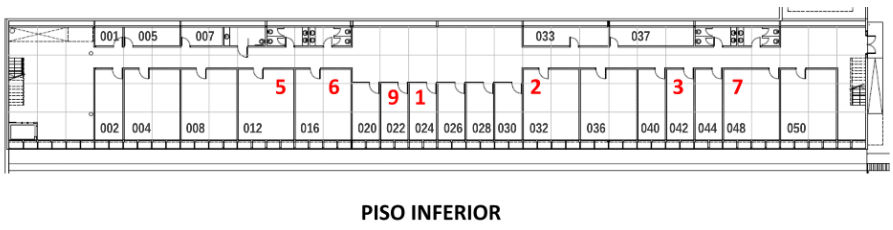

Figura 5.1: Ubicación de los routers en el piso inferior del edificio

El controlador se ejecutara sobre una computadora portátil con el sistema operativo Ubuntu 14.04 LTS conectado a la misma red cableada que los WTP. Los WTP ejecutaran sobre dispositivos TP-LINK TL-WDR4300 con el sistema operativo OpenWRT Chaos Calmer (15.05)

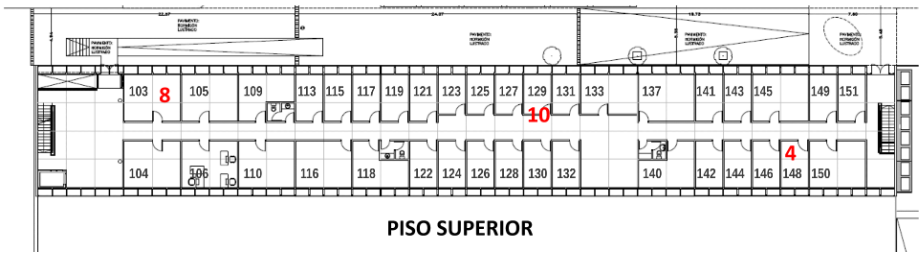

Figura 5.2: Ubicación de los routers en el piso superior del edificio

También vale aclarar el umbral de tiempo utilizado entre cada una de las iteraciones de la aplicación, siendo este seteado en diez minutos a modo de ejecutar las siguientes pruebas.

El detalle de las capturas obtenidas se encuentra en el repositorio GIT, bajo el nombre *Capturas_Pruebas_Realizadas*[28].

5.1. Pruebas de concepto

Para esta prueba se habilitaron únicamente tres WTPs del despliegue, seleccionados debido a su proximidad, para realizar una simple prueba de concepto. Dicha prueba consiste en correr la aplicación *ScanApp* junto al Controlador, donde los tres WTPs estarán transmitiendo en el mismo canal. Los De esta forma, se espera que al momento de correr el algoritmo (luego de pasar el umbral de tiempo), la disposición de canales de los WTPs cambie en busca de una mejor solución.

Los WTPs habilitados fueron:

Disposición de canales		
WTP-5	WTP-6	WTP-9
6	6	6

Esta prueba se realizó bajo dos modalidades a razón de proporcionar más elementos de análisis para los resultados. En una primera instancia, se ejecutó la aplicación de forma normal (considerando para el cálculo de interferencia, los LVAPs captados por nodos propios de la red y las redes externas). En una segunda instancia, se ejecutó la aplicación considerando únicamente los LVAPs captados por nodos propios para el cálculo de interferencia.

5.1.1. Ejecución completa

Los resultados obtenidos por el algoritmo derivaron en la siguiente disposición.

Disposición de canales				
WTP	Inicial	Iteración1	Iteración2	Iteración3
5	6	1	11	11
6	6	11	1	11
9	6	1	1	1

Análisis de los resultados A partir de éstos resultados, podemos notar como dato llamativo, que ningún WTP cambió al canal 6, en todo momento se distribuyeron entre los canales 1 y 11. Realizando un análisis de estos datos, junto a los resultados de los escaneos para cada WTP, se puede concluir que ningún WTP cambió al canal 6 debido a la interferencia generada por las redes externas. La sencilla razón de dicho comportamiento, es que la cantidad de redes captadas del entorno prevalece frente a los LVAPs captados de los nodos propios, además la mayoría de éstas redes transmiten en canales próximos al 6, generando disposiciones mejores omitiendo dicho canal. Debido a éstos resultados, se consideró ejecutar la misma prueba pero descartando la interferencia generada por las redes externas y así analizar el comportamiento del algoritmo frente a distintas circunstancias.

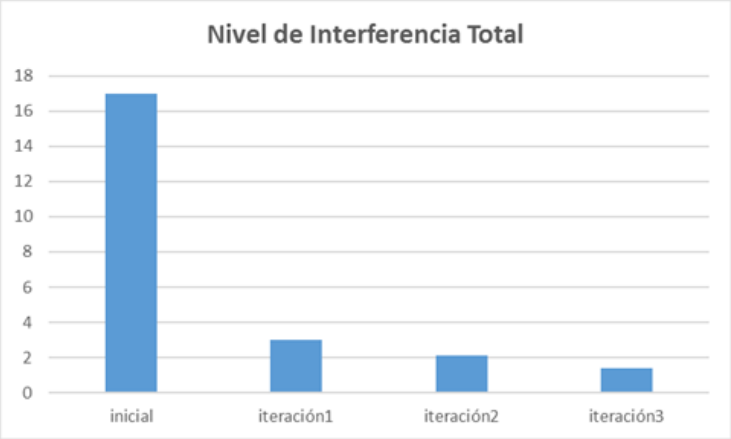

Figura 5.3: Comparación entre los niveles de interferencia total en cada momento

5.1.2. Ejecución sin redes externas

Como se mencionó anteriormente, para la ejecución de ésta prueba se descartó la interferencia generada por las redes externas (modificando el código de la aplicación ScanApp), de esta forma se obtuvieron los siguientes resultados:

Disposición de canales			
WTP	Inicial	Iteración1	Iteración2
5	6	1	1
6	6	11	6
9	6	6	6

Análisis de los resultados En esta segunda ejecución de la prueba podemos visualizar un cambio en los resultados obtenidos. Finalmente luego de la primera iteración de la aplicación se llegó a una disposición (1, 6, 11), además, al eliminar del cálculo la interferencia de redes externas resultó en un cómputo 0.0 como resultado de la interferencia total, tal cuál se esperaba. Lo quizás llamativo de esta segunda ejecución de la prueba puede estar en los resultados de la segunda

iteración del algoritmo. Aquí podemos ver como se repite el canal 6; pero nuevamente el cómputo de interferencia total resultó en 0.0. Entonces corroborando los datos del escaneo, se puede notar que hacia la segunda iteración se captaron una menor cantidad de LVAPs que interfirió en el resultado final, obteniendo en la disposición (1, 6, 6) una solución óptima. De esta forma, la misma implementación del backtracking hace que luego de encontrar una solución óptima (en éste caso inmejorable), se mantenga. Es así como el algoritmo realiza el cambio de canal, no porque la disposición de la segunda iteración (1, 6, 6) sea mejor que la primera (1, 6, 11) -de hecho ambas dan la misma interferencia total- sino porque en la recorrida del backtracking, la disposición (1, 6, 6) se obtiene antes que (1, 6, 11), y a partir de lograr el valor 0.0 ésta primera se mantiene.

Mejora de la aplicación Como se apreció en el análisis anterior, se puede concluir que era innecesario realizar el cambio de canal en el respectivo AP para mantener el valor óptimo de interferencia (0.0), debido a que tanto la disposición (1,6,11) como la (1,6,6) llegan al mismo valor. Una mejora sería evaluar luego del escaneo si la disposición actual de canales mantiene nulo el valor de la interferencia, en dicho caso no sería necesario ejecutar el backtracking y se mantendría la disposición actual, en caso contrario, sí se ejecutaría el backtracking.

5.2. Pruebas de despliegue

Para esta prueba se habilitaron los diez WTPs del despliegue. Dicho número fue tomado del testbed realizado por Apollonia, considerado una buena cantidad tomando en cuenta también el lugar físico (InCo) donde fueron esparcidos. Dicha prueba consiste en correr la aplicación *ScanApp* junto al controlador, donde los diez WTPs estarán transmitiendo en el mismo canal. De esta forma, se espera que al momento de correr el algoritmo (luego de pasar el umbral de tiempo), la disposición de canales de los WTPs cambie a modo de minimizar el cálculo de interferencia total de la red.

Esta prueba también se realizó bajo dos modalidades, tal como la anterior. En una primera instancia, se ejecutó la aplicación de forma normal (considerando para el cálculo de interferencia, los LVAPs captados por nodos propios de la red y las redes externas). En una segunda instancia, se ejecutó la aplicación considerando únicamente los LVAPs captados por nodos propios para el cálculo de interferencia.

Como en todo momento al iniciar la aplicación, todos los WTP transmiten en el mismo canal.

5.2.1. Ejecución completa

Los resultados obtenidos por el algoritmo derivaron en la siguiente disposición.

Disposición de canales				
WTP	Inicial	Iteración1	Iteración2	Iteración3
1	6	11	1	11
2	6	11	11	11
3	6	6	1	11
4	6	1	11	1
5	6	1	11	1
6	6	11	11	1
7	6	1	1	1
8	6	6	6	6
9	6	1	11	11
10	6	1	1	1

Análisis de los resultados De esta prueba se esperaban resultados similares a las pruebas de concepto presentadas previamente, pero ampliando los resultados cuantitativos en los cálculos de interferencia. De esta forma, se puede constatar en los resultados que a nivel de disposición por canales los resultados mantienen un mismo patrón, en este caso encontramos WTPs que transmiten en el canal 6 de forma muy escasa y esto se justifica bajo la misma razón que presentamos en el caso análogo para tres WTPs. También como se puede constatar en los resultados, el cálculo de interferencia total de esta ejecución en comparación con su análoga aumentó debido a la mayor cantidad de WTPs del despliegue.

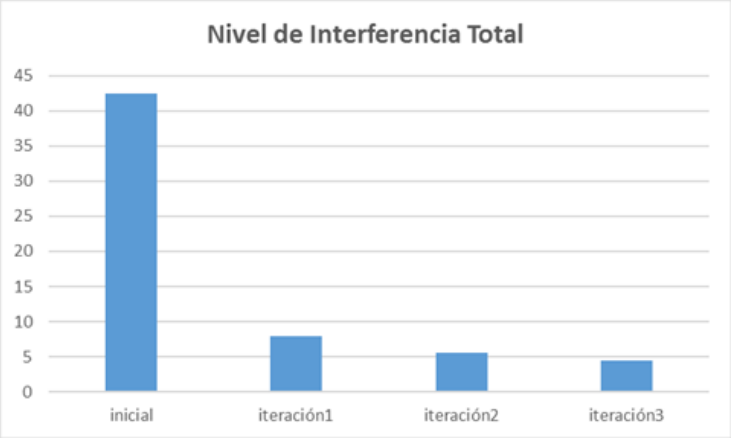

Figura 5.4: Comparación entre los niveles de interferencia total en cada momento

5.2.2. Ejecución sin redes externas

Los resultados obtenidos por el algoritmo derivaron en la siguiente disposición.

Disposición de canales			
WTP	Inicial	Iteración1	Iteración2
1	6	11	6
2	6	6	6
3	6	11	11
4	6	1	6
5	6	1	1
6	6	11	6
7	6	11	11
8	6	1	6
9	6	11	6
10	6	6	11

Análisis de los resultados A partir de esta ejecución podemos confirmar lo expresado por la ejecución anterior y las pruebas anteriores con tres WTPs, en esta oportunidad si encontramos varios WTPs en el canal 6 y nuevamente en ambas iteraciones se obtiene una interferencia total igual a 0.0.

5.3. Prueba de utilidad

Como bien se mencionó previamente, las pruebas ejecutadas sobre la aplicación implementada son estrictamente funcionales y no intentan recabar información a partir de resultados empíricos. Las pruebas anteriores están basadas en el mismo testbed realizado por Apollonia[20], quien investigó sobre los efectos positivos de la Asignación de Canales como una de las técnicas posibles para lidiar con los problemas de las redes inalámbricas de alta densidad.

Su estudio sobre las distintas estrategias de Asignación de Canal puede separarse en dos partes:

- Estudios desde el lado de la industria
- Estudios académicos

En los estudios desde el lado de la industria, presenta dos guías distintas (una publicada por *Aruba Networks* y otra por *Cisco*) con diferentes "buenas prácticas" para el diseño de este tipo de redes, en donde resaltan la importancia de la Asignación de Canales en los APs. En los estudios académicos, realiza una división entre distintas estrategias, entre ellas la Centralizada. También separa según dos categorías, las de optimización estática y las de optimización dinámica.

Nuestro caso de estudio, las estrategias centralizadas, entran dentro de la categoría dinámica. Apollonia presenta dos trabajos basados en este enfoque, por un lado DenseAP[5] que presenta una arquitectura muy similar a la nuestra; y también SMARTA[2].

De esta forma entonces se intenta justificar que la optimización de la Asignación de Canales deriva en una mejora sobre las redes inalámbricas. En base a dicha suposición es que se presenta la siguiente prueba de utilidad de la aplicación.

La prueba radicará en generar tráfico sobre el canal donde transmite EmPOWER, con el propósito de generar interferencia y comparar resultados de

throughput para el momento previo y posterior a la ejecución del algoritmo. Para esta prueba también se utilizará la aplicación *iperf* para medir el *throughput*. En definitiva, aplicaremos dos tipos de tráfico, uno de ellos para la generación de interferencia sobre el canal y el otro a partir de *iperf*.

En esta prueba se podrá evaluar la verdadera funcionalidad de la aplicación *ScanApp*, simulando una situación real y evaluando el comportamiento obtenido. En este caso, se utilizó el Controlador EmPOWER como servidor y un dispositivo conectado a la red EmPOWER como cliente. Independientemente de dicha red, se conectaron otros dos dispositivos generando tráfico en el mismo canal utilizado por la red EmPOWER.

Para evaluar el comportamiento de la aplicación, se registraron cuatro distintos escenarios y se tomó el respectivo ancho de banda entre la comunicación Controlador-Cliente para cada uno, a modo de analizar los distintos resultados. El procedimiento se realizo ejecutando la aplicación y midiendo el ancho de banda de la comunicación Controlador-Cliente, en una primera instancia sin tráfico para el canal (canal 6 -por defecto-), posteriormente aplicando un tráfico de 20Mb, luego aumentando el tráfico a 100Mb. Finalmente se ejecutó el algoritmo y se midió el ancho de banda nuevamente luego de dicha ejecución.

Para toda ejecución de la aplicación *iperf*, se tomó el protocolo UDP (debido a las características propicias para la prueba) y se configuró para transmitir a 50Mb/sec.

Los distintos escenarios fueron registrados en los distintos pasos de la ejecución:

- Escenario 1: Red-EmPOWER: canal 6, sin tráfico
- Escenario 2: Red-EmPOWER: canal 6, con tráfico (20Mb)
- Escenario 3: Red-EmPOWER: canal 6, con tráfico (100Mb)
- Escenario 4: Red-EmPOWER: canal 11, con tráfico (100Mb)

A continuación se mostrarán los resultados de cada escenario:

Resultados por escenario			
Escenario	Canal EmPOWER	Ancho de banda(Mb/sec)	Pérdida(%)
1	6	21.4	55
2	6	19.1	58
3	6	0.8	98
4	11	20.5	57

Análisis de los resultados A partir de los datos clasificados en los distintos escenarios, podemos realizar el siguiente análisis: Al comienzo de la ejecución, en momentos donde no se aplicó tráfico sobre el canal 6 (canal por el que transmiten los WTPs al comienzo de la aplicación *ScanApp*), se registró un valor de 21.4Mb de ancho de banda (con un 55 % de paquetes perdidos), esta medida será tomada como base y a partir de la ejecución del algoritmo se pretenderá volver a dicha base. Posteriormente se aplicó tráfico sobre el canal 6, en una primera instancia de 20Mb, resultando en una muy leve disminución del ancho de banda e incremento de paquetes perdidos (19.1Mb de ancho de banda y 58 % de pérdidas). En una segunda instancia se aumentó significativamente el nivel de tráfico

a 100Mb, produciendo cambios significativos, tan así que casi se llegó a un 100 % de pérdidas (98 % para ser precisos y un ancho de banda ínfimo de 0.8Mb). Del último escenario se esperaba que la ejecución del algoritmo pudiera contribuir a mejorar la situación. Efectivamente se ejecutó nuevamente el algoritmo, se cambió del canal 6 al canal 11 y en una posterior toma de datos se volvieron a registrar resultados prácticamente iguales a la base antes mencionada (20.5Mb de ancho de banda y 57 % de pérdidas).

De esta forma, podemos probar la utilidad de la aplicación *ScanApp* realizando un cambio de canal para mejorar el ancho de banda entre Controlador-Cliente.

Capítulo 6

Conclusiones

A lo largo de esta tesis, se realizó un relevamiento y estado del arte sobre SDN Inalámbrico, a partir del cual se eligió una herramienta para profundizar su estudio e implementar una extensión. A raíz de dicho trabajo, podemos resumir ciertos conceptos.

Hoy en día, no existe un estándar para el uso del paradigma SDN en medios inalámbricos debido al nivel de madurez y las dificultades que este medio presenta:

- Es un medio compartido
- Los enlaces inalámbricos funcionan bajo varios regímenes, por ejemplo, se pueden ajustar los *rates* y el poder de transmisión; y se puede variar en los mecanismos de coordinación utilizados
- Las particularidades de cada tecnología inalámbrica, que hace difícil el desarrollo de un estándar común para todas ellas

SDN en redes cableadas se enfoca en programar los flujos de datos, lo que no es suficiente en redes inalámbricas ya que resulta de interés poder programar las características de la capa MAC (más cercana a los aspectos físicos de los dispositivos), con el fin de hacer frente a las dificultades anteriormente mencionadas.

Por otro lado, en [12] se mencionan algunos de los beneficios que podría implicar el avance de las SDN en redes inalámbricas. Entre ellos, mejorar la conectividad de los usuarios y la calidad del servicio; mejoras a nivel de seguridad, localización y movilidad.

Por su parte, EmPOWER utiliza conceptos de SDN, NFV y NV, definiéndose como un Sistema Operativo de Red. Puede considerarse dentro de lo que es SDN inalámbrico debido a la utilización de un controlador centralizado para programar el plano de datos de la parte inalámbrica, aunque no complementa dichas acciones en la parte cableada, más allá de los planteos teóricos de su definición. También podemos encontrar en EmPOWER algunos de los beneficios mencionados anteriormente; un claro ejemplo se mostró mediante la utilización de la aplicación *Mobility Manager* que se basa en las abstracciones de LVAP y *Channel Quality Map*.

Al ser de código abierto, EmPOWER posibilita un potencial uso masivo, que podría derivar en la estandarización. Luego de haber trabajado sobre este framework, podemos concluir que aún no se encuentra apto para su uso a

nivel productivo. Son muchas las falencias que presenta a todo nivel, desde su orquestación entre los aspectos cableados e inalámbricos, así como errores en la implementación, consecuencia de que aún es un prototipo en desarrollo. Hoy en día, EmPOWER se sostiene sobre ciertas herramientas que podrían ser sustituidas por implementaciones más adecuadas; por ejemplo la utilización de Click Modular Router para el camino de datos dentro de un dispositivo inalámbrico. Otro aspecto que obstaculiza su uso, es la carencia de documentación técnica sobre la herramienta y la distancia entre la definición teórica y la implementación.

Como reflexión final, podemos concluir que no hay un nivel de madurez suficiente que permita definir el rumbo que tomará el paradigma SDN en redes inalámbricas. En la actualidad no existe la herramienta que abarque todos los aspectos y problemas planteados en el área.

El código que acompaña el trabajo realizado en el marco de este proyecto, se encuentra en [29] y [30].

Bibliografía

[1] Eddie Kohler y col. «The Click modular router». En: *ACM Transactions on Computer Systems (TOCS)* 18.3 (2000), págs. 263-297.

[2] Nabeel Ahmed y Srinivasan Keshav. «SMARTA: a self-managing architecture for thin access points». En: *Proceedings of the 2006 ACM CoNEXT conference*. ACM. 2006, pág. 9.

[3] Natasha Gude y col. «NOX: towards an operating system for networks». En: *ACM SIGCOMM Computer Communication Review* 38.3 (2008), págs. 105-110.

[4] Nick McKeown y col. «OpenFlow: enabling innovation in campus networks». En: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), págs. 69-74.

[5] Rohan Murty y col. «Designing High Performance Enterprise Wi-Fi Networks.» En: *NSDI*. Vol. 8. 2008, págs. 73-88.

[6] Ben Pfaff y col. «Extending Networking into the Virtualization Layer». En: *Hotnets*. 2009.

[7] Kok-Kiong Yap y col. «Blueprint for introducing innovation into wireless mobile networks». En: *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*. ACM. 2010, págs. 25-32.

[8] Kok-Kiong Yap y col. «OpenRoads: Empowering research in mobile networks». En: *ACM SIGCOMM Computer Communication Review* 40.1 (2010), págs. 125-126.

[9] Bart Braem y Michael Voorhaen. «Click Concepts». Diapositivas. 2011.

[10] Lalith Suresh Puthalath. «Programming the enterprise WLAN: an SDN approach». Tesis doct. Instituto Superior Técnico, 2012.

[11] Omar Sefraoui, Mohammed Aissaoui y Mohsine Eleuldj. «OpenStack: toward an open-source solution for cloud computing». En: *International Journal of Computer Applications* 55.3 (2012).

[12] C. Chaudet e Y. Haddad. «Wireless Software Defined Networks: Challenges and opportunities». En: *2013 IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS 2013)*. Oct. de 2013, págs. 1-5. DOI: 10.1109/COMCAS.2013.6685237.

[13] Rami Cohen y col. «An intent-based approach for network virtualization». En: *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE. 2013, págs. 42-50.

[14] Joshua Reich y col. «Modular sdn programming with pyretic». En: *Technical Reprot of USENIX* (2013).

[15] Roberto Riggio, Tinku Rasheed y Fabrizio Granelli. «EmPOWER: a testbed for network function virtualization research and experimentation». En: *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. IEEE. 2013, págs. 1-5.

[16] Jonathan Vestin y col. «CloudMAC: towards software defined WLANs». En: *ACM SIGMOBILE Mobile Computing and Communications Review* 16.4 (2013), págs. 42-45.

[17] Jan Medved y col. «Opendaylight: Towards a model-driven sdn controller architecture». En: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. 2014.

[18] Julius Schulz-Zander, Nadi Sarrar y Stefan Schmid. «Aeroflux: A nearsighted controller architecture for software-defined wireless networks». En: *Presented as part of the Open Networking Summit 2014 (ONS 2014)*. 2014.

[19] Julius Schulz-Zander y col. «Programmatic Orchestration of WiFi Networks». En: *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, jun. de 2014, págs. 347-358. ISBN: 978-1-931971-10-2. URL: https://www.usenix.org/conference/atc14/technical-sessions/presentation/schulz-zandery.

[20] Guillermo Apollonia. «Self management of high density wireless networks». En: (2015), pág. 116.

[21] Diego Kreutz y col. «Software-defined networking: A comprehensive survey». En: *Proceedings of the IEEE* 103.1 (2015), págs. 14-76.

[22] Julius Schulz-Zander y col. «OpenSDWN: programmatic control over home and enterprise WiFi». En: *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM. 2015, pág. 16.

[23] Tejas Subramanya, Roberto Riggio y Tinku Rasheed. «Intent-based Mobile Backhauling for 5G Networks». En: (2016).

[24] *5G-EmPOWER*. URL: http://empower.create-net.org/.

[25] *802.11 Association process explained*. URL: https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/802.11_Association_process_explained.

[26] *API PYTHON*. URL: https://github.com/5g-empower/5g-empower.github.io/wiki/Python-API-documentation#lvaps.

[27] *API REST*. URL: https://github.com/5g-empower/5g-empower.github.io/wiki/REST-API-documentation.

[28] *Capturas de pruebas realizadas*. URL: https://github.com/gastontelfe/empower-runtime/blob/master/Capturas_Pruebas_Realizadas.pdf.

[29] *Código del agente EmPOWER (modificado)*. URL: https://github.com/gastontelfe/empower-agent.

[30] *Código del controlador EmPOWER (modificado)*. URL: https://github.com/gastontelfe/empower-runtime.

[31] *iwinfo*. URL: https://wiki.openwrt.org/doc/howto/wireless.utilities.

[32] *Repositorio EmPOWER*. URL: https://github.com/5g-empower.

Apéndice A

Modificar agente de EmPOWER

Para extender EmPOWER y soportar nuevos mensajes en el protocolo de comunicación entre el Controlador y el Agente, fue necesario modificar el código del Agente. A continuación describiremos los pasos realizados para compilar el paquete del Agente con las modificaciones implementadas.

A.1. Compilar paquete empower-lvap-agent

Como primer paso se debe descargar el código fuente de la imagen de OpenWRT y descargar los *feeds*.

```
git clone https://github.com/5g-empower/empower-openwrt-15.05
cd empower-openwrt-15.05
git checkout -b empower origin/empower
./scripts/feeds update -a
./scripts/feeds install -a
```

El paso siguiente es configurar la plataforma para la cual compilaremos (Atheros AR7xxx/AR9xxx Atheros AR9344) y seleccionar el paquete *empower-lvap-agent* a través de la interfaz de configuración de OpenWRT. Esta es accedida a través del comando:

```
make menuconfig
```

Una vez configurada la plataforma, se deben compilar las herramientas utilizadas para compilar hacia la plataforma destino.

```
make -j 8 tools/install
make -j 8 toolchain/install
```

Por último, el paquete es compilado con el siguiente comando:

```
make -j 8 package/empower-lvap-agent/compile
```

A.2. Modificar Agente y compilar

Primero se debe habilitar *Enable package source tree override*. Para esto:

1. Ejecutar make menuconfig
2. Entrar al menú *Advanced configuration options*
3. Seleccionar la opción *Enable package source tree override*
4. Guardar los cambios

Crear link a la carpeta .git del proyecto del agente dentro del feed del agente. Ejemplo:
En ~/empower-lvap-agent tenemos el repositorio del agente
En ~/empower-openwrt el repositorio de OpenWRT

```
cd ~/empower-openwrt
ln -s ~/empower-lvap-agent/.git pacakges/feeds/
    empower/empower-agent/git-src
```

Para compilar los cambios deben estar comiteados, luego ejecutar los siguientes comandos:

```
make package/empower-agent/clean
make package/empower-agent/compile
```

Una vez compilado el paquete, este se instala en los routers con *opkg*, como cualquier otro paquete de *OpenWRT*.