UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Estudio y validación de Mininet-WiFi en base a pruebas reales

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Panzuela Perez, Francisco

Tutor/a: Tavares de Araujo Cesariny Calafate, Carlos Miguel

CURSO ACADÉMICO: 2021/2022

Resumen

Las redes definidas por software (SDN) permiten un elevado nivel de gestión de la red, razón por la cual son cada vez más utilizadas, especialmente en centros de datos y por parte de operadores de telecomunicaciones. La herramienta de referencia para el estudio y desarrollo de soluciones SDN es Mininet, un emulador muy realista para este tipo de redes. La extensión Mininet-WiFi permite dotar a esta herramienta de capacidades de gestión de puntos de acceso WiFi mediante SDN usando una arquitectura que busca ser realista. No obstante, hasta el momento no se han realizado estudios del grado de corrección con que esta herramienta implementa el estándar *IEEE 802.11*, ni tampoco de la representatividad de su modelo de canal físico.

En este TFG se busca realizar un estudio comparativo donde se analice el rendimiento alcanzado en las comunicaciones entre punto de acceso y estación móvil usando Mininet-WiFi, y usando hardware real, así como de la secuencia de tramas generada, buscando de esa manera validar el correcto funcionamiento de esta herramienta de emulación.

Palabras clave: Mininet, WiFi, modelado, virtualización de redes, redes definidas por software.

Resum

Les xarxes definides per software (SDN) permeten un elevat nivell de gestió de les xarxes, raó per la qual són cada vegada més utilitzades, especialment en centres de dades i per part d'operadors de telecomunicacions. L'eina de referència per a l'estudi i el desenvolupament de solucions SDN és Mininet, un emulador molt realista per a aquest tipus de xarxes. L'extensió Mininet-WiFi permet dotar aquesta eina de capacitats de gestió de punts d'accés WiFi mitjançant SDN utilitzant una arquitectura que busca ser realista. No obstant això, fins ara no s'han realitzat estudis del grau de correcció amb què aquesta eina implementa l'estàndart *IEEE 802.11*, ni tampoc de la respresentativitat del model de canal físic.

Aquest TFG busca realitzar un estudi comparatiu on s'analitze el rendiment assolit a les comunicacions entre punt d'accés i estació móvil utilitzant Mininet-WiFi, i usant hardware real, així com de la seqüència de trames generada, buscant així validar el funcionament correcte d'aquesta eina d'emulació.

Paraules clau: Mininet, WiFi, modelat, virtualització de xarxes, xarxes definides per software.

Abstract

Software-defined networking (SDN) allow a high level of network management, which is why they are increasingly used, especially in data centers and by telecom operators. The reference tool for the study and development of SDN solutions is Mininet, a very realistic emulator for this type of network. The Mininet-WiFi extension allows you to provide this tool with Wi-Fi access point management capabilities through SDN using an architecture that seeks to be realistic. However, to date no studies have been carried out on the degree of correctness with which this tool implements IEEE 802.11 standard, nor on the representativeness of its physical channel model.

This TFG seeks to carry out a comparative study where the performance achieved in communications between access point and mobile station is analyzed using Mininet-WiFi, and using real hardware, as well as the sequence of frames generated, thus seeking to validate the proper functioning of this emulation tool.

Keywords: Mininet, WiFi, modeling, network virtualization, software-defined networking.

Tabla de contenidos

Tabla de figuras

1. Introducción

Actualmente la tecnología WiFi es conocida por todo el mundo debido a su gran popularidad entre los usuarios, ya sea para uso doméstico o en el ámbito laboral, grandes empresas, etc. Cuando surgió causó grandes repercusiones debido a que funcionaba sin cables, es decir, de forma inalámbrica. Una de las razones por las que ha crecido tanto en los últimos años es su flexibilidad y soporte a la movilidad para conectarse a Internet, ya que cada vez hay más dispositivos conectados. Además, con la aparición del concepto *IoT* —del inglés *Internet of Things*— es aún más importante ser consciente de cuántos dispositivos se conectan a Internet, como por ejemplo cámaras de seguridad o electrodomésticos inteligentes y, por lo tanto, existe la necesidad de conocer cómo funciona esta tecnología. Además, en los últimos años, la conectividad inalámbrica ha mejorado con la aparición del 5G, tecnología que permite mejorar las velocidades —y tener menores latencias—, junto con otras mejoras, permitiendo así estar más cerca de crear entornos inteligentes, es decir, el acercamiento a lo que hoy en día se conoce como *smart cities*.

El crecimiento exponencial del uso de datos móviles ha provocado la necesidad de implementaciones de red escalables y manejables en las tecnologías inalámbricas actuales. Las SDN pueden resultar una buena elección ya que pueden crear y controlar una red virtual o controlar una red de hardware tradicional mediante software. Mientras que la virtualización de red ofrece la posibilidad de segmentar distintas redes virtuales dentro de una red física, o de conectar dispositivos de diferentes redes físicas a una red virtual, las SDNs proporcionan una nueva forma de controlar el enrutamiento de los paquetes de datos a través de un servidor centralizado bajo el que podemos controlar todo el entorno, mejorando incluso la seguridad de nuestra organización.

Por esta razón, surge la necesidad de tener sistemas que verifiquen su funcionamiento y realicen pruebas antes de implantar esta tecnología en sistemas informáticos, ya sea en entornos empresariales, instituciones públicas, etc. Al igual que un piloto de avión necesita, antes de volar por primera vez, tener un número determinado de horas de simulación completadas para “evitar sustos”, con este trabajo se pretende resaltar las funciones de los sistemas de emulación, con el objetivo de poder realizar las pruebas antes de implantar la tecnología en la realidad. Así pues, se procederá con la evaluación del emulador Mininet-WiFi, proyecto *open-source* popularmente utilizado en el mercado y en constante desarrollo, para poder entender su potencial y, además, poder determinar si de verdad es fiel a la realidad con el deseo de acercar esta tecnología a la sociedad por todos los beneficios que aporta.

1.1 Motivación

Este proyecto pretende estudiar el grado de corrección y representatividad de las redes definidas vía software mediante la herramienta de emulación Mininet-WiFi con el propósito de evaluar su rendimiento y validar su funcionamiento, realizando un estudio comparativo con la realidad para poder determinar si de verdad es útil esta tecnología, o por el contrario, deja mucho que desear.

1.1.1 Motivación personal

La tecnología inalámbrica marcó “un antes y un después” en nuestra forma de vivir en el mundo debido a que vivimos en una sociedad conectada las veinticuatro horas y en constante movimiento. Por esta razón, siempre he tenido mucha curiosidad por saber cómo funciona internamente la tecnología WiFi, ya que está presente en cualquier lugar, todo el mundo la utiliza diariamente, pero pocos saben cómo funciona realmente.

1.2 Objetivos

El objetivo de este trabajo es poder evaluar las redes definidas por software (SDN) mediante el análisis de la herramienta más utilizada actualmente, Mininet-WiFi, con el propósito de analizar su rendimiento en las comunicaciones entre punto de acceso y estaciones móviles utilizando para ello modelos dinámicos de los que poder extraer información para su posterior comparación con la realidad y poder elaborar una conclusión acerca de si esta tecnología se acerca a la realidad, o por el contrario, está lejos de ésta. Para ello, se realizarán distintos métodos de análisis a distintos niveles para validar su funcionamiento. A continuación, se describirán los objetivos definidos por este estudio:

Los objetivos primarios de este trabajo son:

- Evaluar de forma empírica el emulador Mininet-WiFi justificando su grado de semejanza con la realidad.
- Validar el funcionamiento de este emulador en base a un escenario emulado semejante a un escenario real.

Para poder justificar los objetivos primarios de una forma medible, se deberán de satisfacer los siguientes objetivos específicos:

- Definir un escenario de pruebas representativo.
- Capturar el tráfico generado en la realidad y en la emulación mediante la herramienta *Wireshark*.
- Medir los rendimientos alcanzados en la realidad y en la emulación y estudiar posibles diferencias.

1.3 Impacto esperado

Mediante este trabajo se pretende dar visibilidad a las herramientas de emulación para conseguir mejorar la tecnología que nos rodea y, especialmente, crear emulaciones más realistas para poder ser de utilidad y ayudar a nuestra sociedad, beneficiando directamente al medio ambiente, evitando la compra innecesaria de hardware y equipos informáticos, y asegurando que cumplan su objetivo en entornos adecuados. Además, ofrecería beneficios desde un punto de vista empresarial, debido a que reduciría los costes tanto de equipamiento como de operación, acelerando la introducción de nuevos servicios.

1.4 Estructura

A lo largo de este trabajo, el lector podrá descubrir los siguientes apartados organizados de la siguiente forma:

En el capítulo 2, se definirá el estado del arte acerca del contexto tecnológico de la tecnología estudiada, describiendo brevemente cómo surgió, cuál es su propósito, y se explicarán conceptos relacionados con su arquitectura, finalizando con un análisis sobre trabajos relacionados.

En el capítulo 3, se ofrecerán detalles más específicos sobre Mininet-WiFi que afectarán en gran medida al estudio, como por ejemplo qué modelos ofrece, y cómo es capaz de virtualizar dispositivos.

En el capítulo 4, se realizará un análisis del tráfico mediante la monitorización de tramas mediante *Wireshark* para poder justificar si la virtualización tiene un comportamiento semejante a la realidad.

En el capítulo 5, se medirán rendimientos tanto en la realidad como en la simulación, con el fin de poder aportar más justificaciones acerca de si existen diferencias entre ambos escenarios.

En el capítulo 6, se descubrirá una mejora recientemente adquirida del software Mininet-WiFi descubierta en el estudio.

En el capítulo 7, se presentarán las conclusiones obtenidas del estudio, detallando si se lograron los objetivos y exponiendo las ventajas y desventajas encontradas.

En el capítulo 8, se ofrecerán algunas ideas acerca de posibles trabajos futuros relacionados con este ámbito.

Finalmente, se incluirán anexos con el fin de aportar más información técnica acerca del estudio realizado, detallando un manual de usuario para poder conocer los comandos esenciales de Mininet-WiFi, una explicación de los campos de una trama WiFi, una ampliación para poder replicar el estudio ya que a nivel de hardware real es necesaria la

actualización de ciertos drivers y, por último, se presentarán aquellos objetivos de desarrollo sostenible involucrados en este trabajo.

2. Estado del arte

A continuación, se detallará brevemente el estado actual de la tecnología estudiada para que el lector pueda comprender de forma más completa qué características ofrece, y se darán algunas vías de estudio alternativas del pasado.

2.1 Software-defined networking (SDN)

Actualmente, las redes SDN están siendo utilizadas en numerosos ámbitos ya que nos encontramos en una sociedad completamente digitalizada. Por ello es muy conveniente, antes de instalar un sistema informático, estudiar previamente cuáles serán sus funciones, si la infraestructura está preparada para su cometido, posibles configuraciones, hardware alternativo, etc., y es en este momento cuando tiene gran importancia este tipo de aplicaciones software.

Los primeros pasos hacia las redes definidas mediante software (SDN) se han dado en la década de los 90 por una necesidad de programar las redes convencionales debido a que la tecnología avanzaba muy rápido y los investigadores querían crear nuevos protocolos de red, pero no podían permitírselo debido a que no eran programables, y la organización IETF debía de aprobarlos, hecho que causaba una gran frustración entre ellos por la lentitud del procedimiento. Por eso, surgieron las "redes activas" (1), predecesoras de las SDN actuales, que satisfacían algunas de las necesidades, y comenzaron a desarrollar funcionalidades que ofrecerían la virtualización de redes por primera vez en la historia, respaldados por organizaciones como DARPA[1].

Más adelante, a partir del año 2000, se decidió separar el plano de control del plano de datos debido a que cada vez había más tráfico, y las redes de aquella época no gestionaban adecuadamente todo el volumen de datos. Por lo tanto, los investigadores decidieron separar ambos planos con el objetivo de optimizarlos por separado, ya que la unión de ambos planos ocasionaba muchos problemas de integración. Es decir, se decidió separar el hardware del software, permitiendo así que otros dispositivos llamados controladores tomaran el control de la red.

Actualmente, el último gran avance en este campo ha sido el desarrollo del protocolo *OpenFlow,* que fue originado a raíz de un experimento para crear redes universitarias más personalizables en la Universidad de Stanford. Se puede apreciar que, a lo largo de la historia, el objetivo de las SDN siempre ha sido conseguir redes más flexibles, automatizables y programables, para tener la capacidad de modificarlas dinámicamente

[1] Agencia de Proyectos de Investigación Avanzada en Defensa de EEUU, en la que se puede observar sus proyectos más recientes en https://www.darpa.mil/.

mediante la creación de un entorno virtual para comprenderlas mejor, y aplicar las mejoras descubiertas en la realidad (2).

En conclusión, las SDN son utilizadas a día de hoy por infinidad de empresas de telecomunicaciones e informática, y pueden ayudar a resolver problemas complejos, relacionados con la escalabilidad al incorporar nuevos sistemas informáticos en la infraestructura de la red, e incluso eliminar las dependencias con proveedores ya que, mediante la virtualización, no hay contratos con millones de euros en juego esperando a ser recibidos para comprar equipos, licencias, etc. En resumen, el objetivo de las SDN sigue siendo el mismo que cuando surgieron por primera vez: mejorar nuestra sociedad partiendo de un escenario manipulable para, posteriormente, aplicarlo en un escenario real. Un símil de esta situación podría ser cuando un escalador entrena en el rocódromo antes de escalar la montaña de verdad.

2.1.1 Arquitectura SDN

La arquitectura SDN, descrita por la ONF[2], especifica un total de tres capas, como se detalla a continuación en la Figura 1 (3):

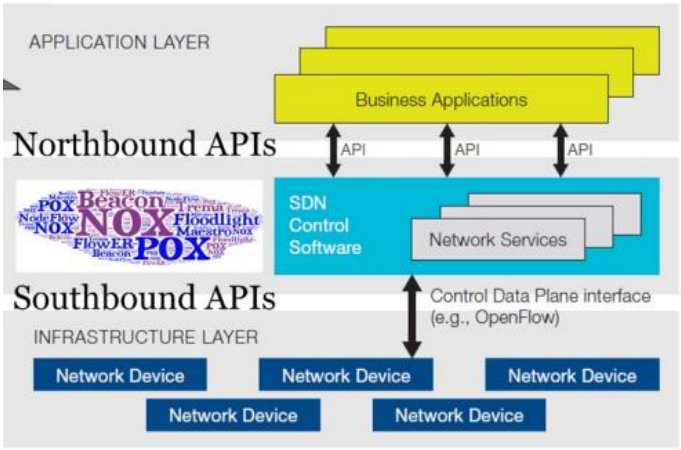

Figura 1 - Arquitectura de SDN.

Capa de aplicación: consiste en las aplicaciones de negocio destinadas a los usuarios finales que utilizan servicios de comunicación SDN a través de las APIs hacia arriba (Northbound APIs), como por ejemplo XML y JSON para comunicarse con la capa de control. Las ventajas que ofrecen las Northbound APIs son muy variadas, ya que sirven para conectar el controlador SDN a los servicios y aplicaciones por encima, o permite a los servicios y aplicaciones simplificar y automatizar las tareas de configuración, provisión, y gestionar nuevos servicios en la red, ofreciendo a los operadores nuevas vías de ingresos,

[2] ONF: *Open Networking Foundation*. Consorcio sin ánimo de lucro que está liderando la evolución de *SDN* y estandarizando los elementos críticos de esta nueva arquitectura.

diferenciación e innovación, además de suplir las necesidades de las diferentes aplicaciones a través de la programabilidad de la red SDN.

Capa de control: proporciona una funcionalidad centralizada de control que supervisa el comportamiento de la red de datos a través de una interfaz abierta, permitiendo a los desarrolladores de aplicaciones utilizar capacidades de red, pero abstrayéndose de su topología o funciones. El principal protagonista de esta capa es el controlador, que es la entidad encargada de traducir las peticiones del servicio SDN a las rutas de datos inferiores, dando a la capa de aplicación una visión abstracta de la red mediante estadísticas y posibles eventos. Se puede decir de los controladores son el cerebro de este tipo de redes ya que tienen control exclusivo sobre la forma de controlar y configurar los nodos de red para dirigir correctamente los flujos de tráfico; además, la arquitectura le permite generar un amplio rango de recursos del plano de datos, lo cual ofrece el potencial de unificar y simplificar su configuración (4).

Capa de infraestructura: está constituida por los nodos de red que realizan la conmutación y encaminamiento de paquetes. Proporciona un acceso abierto programable a través de la Southbound API, como por ejemplo *OpenFlow*. Las southbound API facilitan el control en la red, permitiendo al controlador realizar cambios dinámicos de acuerdo a las demandas en tiempo real y las necesidades (5).

2.2 OpenFlow

Se trata de un protocolo que permite a un servidor decirle a los conmutadores de red dónde enviar paquetes. En una red convencional, cada dispositivo de red tiene software propietario que le dice qué hacer. Sin embargo, con *OpenFlow* se centralizan las decisiones de migración de paquetes, de modo que la red se puede programar independientemente de los conmutadores individuales y el equipo de centro de datos (6). Existen tres partes esenciales como se puede apreciar en la Figura 2:

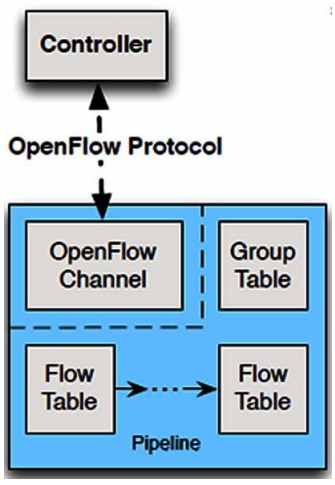

Figura 2 - Arquitectura de un switch OpenFlow.

Tabla de flujos: cada entrada de la tabla dispone de campos en los que buscar coincidencias con los paquetes entrantes, contadores (para estadísticas de los paquetes) e instrucciones sobre qué hacer con los coincidentes.

Canal seguro: conecta el dispositivo al controlador, permitiendo el envío de comandos y paquetes entre ambos mediante el protocolo *OpenFlow*.

Protocolo *OpenFlow*: provee una manera abierta y estándar en la comunicación entre el conmutador y el controlador, permitiendo a este último insertar, eliminar, modificar y buscar en las entradas de la tabla de flujos a través del canal seguro

2.3 Crítica del estado del arte

Respecto a otros trabajos realizados en la ETSINF (7) sobre esta misma temática, cabe destacar que la gran mayoría estudian la tecnología desde puntos de vista diferentes debido a que profundizan más en cómo funciona Mininet-WiFi, qué protocolo implementa a bajo nivel —concretamente *OpenFlow*—, todas las capas involucradas junto con los drivers que ejecute en segundo plano para imitar el comportamiento de un entorno real, e incluso existen trabajos dedicados exclusivamente a la seguridad *WiFi* explicando los diferentes protocolos, sus deficiencias y mejoras —como es el caso de la evolución a lo largo de los años de los protocolos *WEP*, *WPA* y *WPA2*— e incluso para buenas prácticas y normas legales que rigen el protocolo.

Este trabajo pretende ser innovador debido a que nunca antes se había realizado un análisis comparativo entre la realidad y lo virtual, incluso en Internet tampoco existen publicaciones estudiando este punto de vista, por lo que resulta de gran interés poder aportar nuevas ideas y oportunidades de mejora en este campo.

3. Mininet-WiFi

A continuación se procede a detallar el funcionamiento de Mininet-WiFi, profundizando más en los aspectos que trata de analizar este trabajo, como son el funcionamiento interno de la emulación.

3.1 Instalación del software

Para instalar Mininet-Wifi, se procederá a seguir los pasos recomendados por la página oficial[3], concretamente mediante la segunda opción para realizar una instalación nativa directamente desde el código fuente almacenado en *GitHub* por el mismo autor[4]. Para ello, clonaremos desde el repositorio oficial el código fuente (1), accederemos a la carpeta principal (2) y ejecutaremos el script "install.sh" con los argumentos que se muestran en el recuadro naranja (3), como se puede apreciar en la Figura 3:

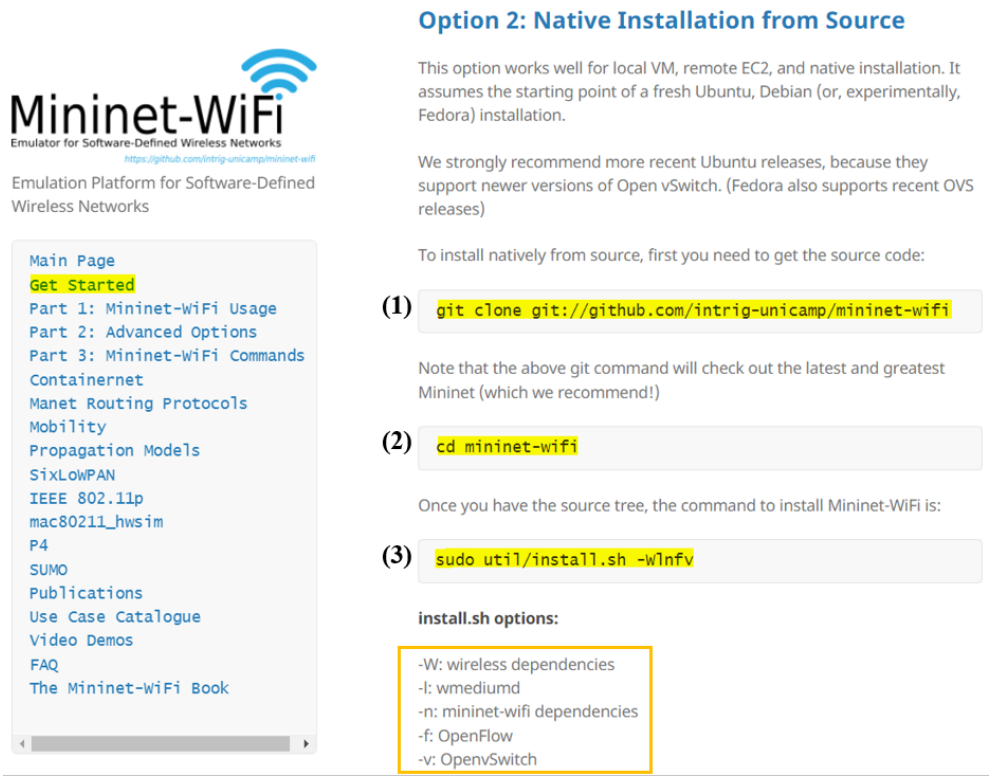

Figura 3 - Instalación de Mininet-WiFi.

[3] Disponible en: https://mininet-wifi.github.io/get-started/
[4] Disponible en: https://github.com/intrig-unicamp/mininet-wifi.git

Una vez instalado el software, se pueden apreciar el listado de directorios que contiene —en la ruta donde ejecutamos las órdenes mostradas anteriormente de la Figura 3—, con los siguientes archivos mostrados en la Figura 4:

```
dclan@dclan2020:/home/mininet-wifi$ ls -la
total 164
drwxr-xr-x 17 root root  4096 may 30 09:25 .
drwxr-xr-x  6 root root  4096 may 29 17:54 ..
drwxr-xr-x  2 root root  4096 oct 20  2021 bin
drwxr-xr-x  5 root root  4096 oct 20  2021 build
-rw-r--r--  1 root root   636 oct 20  2021 CONTRIBUTORS
drwxr-xr-x  2 root root  4096 oct 20  2021 custom
drwxr-xr-x  3 root root  4096 oct 20  2021 debian
drwxr-xr-x  2 root root  4096 oct 20  2021 demos
drwxr-xr-x  2 root root  4096 oct 20  2021 dist
drwxr-xr-x  2 root root  4096 oct 20  2021 doc
drwxr-xr-x  7 root root  4096 may 30 11:01 examples
drwxr-xr-x  9 root root  4096 oct 20  2021 .git
-rw-r--r--  1 root root    17 oct 20  2021 .gitattributes
-rw-r--r--  1 root root   120 oct 20  2021 .gitignore
-rw-r--r--  1 root root    67 oct 20  2021 .gitmodules
drwxr-xr-x 14 root root  4096 oct 20  2021 hostap
-rw-r--r--  1 root root  5432 oct 20  2021 INSTALL
drwxr-xr-x  3 root root  4096 oct 20  2021 iw
-rw-r--r--  1 root root  1877 oct 20  2021 LICENSE
-rwxr-xr-x  1 root root  2003 oct 20  2021 Makefile
drwxr-xr-x 11 root root  4096 oct 20  2021 mininet
drwxr-xr-x  2 root root  4096 oct 20  2021 mininet_wifi.egg-info
-rw-r--r--  1 root root  4130 oct 20  2021 mn.1
-rwxr-xr-x  1 root root 14064 oct 20  2021 mnexec
-rw-r--r--  1 root root   754 oct 20  2021 mnexec.1
-rwxr-xr-x  1 root root  6023 oct 20  2021 mnexec.c
drwxr-xr-x  8 root root  4096 oct 20  2021 mn_wifi
-rw-r--r--  1 root root  9240 oct 20  2021 .pylint
-rwxr-xr-x  1 root root  3873 oct 20  2021 README.md
-rwxr-xr-x  1 root root  2215 oct 20  2021 setup.py
-rw-r--r--  1 root root   532 oct 20  2021 TODO.md
-rw-r--r--  1 root root   663 oct 20  2021 .travis.yml
drwxr-xr-x  9 root root  4096 oct 20  2021 util
```

Figura 4 - Directorios por defecto de Mininet-WiFi.

Mininet-WiFi proporciona una serie de ejemplos con configuraciones de red definidas mediante scripts en Python, en el directorio *examples*, como se puede observar en la Figura 5:

```
dclan@dclan2020:/home/mininet-wifi/examples$ ls -la
total 468
drwxr-xr-x  7 root root   4096 may 30 11:01 .
drwxr-xr-x 17 root root   4096 may 30 09:25 ..
-rwxr-xr-x  1 root root   2309 oct 20  2021 4address.py
-rwxr-xr-x  1 root root   1619 oct 20  2021 6GHz.py
-rwxr-xr-x  1 root root   1678 oct 20  2021 6LoWPan.py
-rwxr-xr-x  1 root root   1501 oct 20  2021 active_scan.py
-rwxr-xr-x  1 root root   2290 oct 20  2021 adhoc.py
-rwxr-xr-x  1 root root   1510 oct 20  2021 associationControl.py
-rwxr-xr-x  1 root root   1034 oct 20  2021 authentication.py
-rwxr-xr-x  1 root root    964 oct 20  2021 battery.py
-rwxr-xr-x  1 root root   4179 oct 20  2021 clustercli.py
-rwxr-xr-x  1 root root  35094 oct 20  2021 cluster.py
-rw-r--r--  1 root root   1298 oct 20  2021 coherenceModel.py
drwxr-xr-x  3 root root   4096 oct 20  2021 eap-tls
-rwxr-xr-x  1 root root   3246 oct 20  2021 forwardingBySSID.py
-rwxr-xr-x  1 root root   2363 oct 20  2021 handover_bgscan.py
-rwxr-xr-x  1 root root   1642 oct 20  2021 handover.py
-rwxr-xr-x  1 root root   1542 oct 20  2021 ieee80211p.py
-rwxr-xr-x  1 root root     48 oct 20  2021 __init__.py
-rwxr-xr-x  1 root root   1433 oct 20  2021 meshAP.py
-rwxr-xr-x  1 root root   1952 oct 20  2021 mesh.py
-rwxr-xr-x  1 root root 247299 oct 20  2021 miniedit.py
-rwxr-xr-x  1 root root   1498 oct 20  2021 mobilityModel.py
-rwxr-xr-x  1 root root   1949 oct 20  2021 mobility.py
-rwxr-xr-x  1 root root   1203 oct 20  2021 multipleWlan.py
drwxr-xr-x  2 root root   4096 oct 20  2021 p4
-rwxr-xr-x  1 root root   1585 oct 20  2021 physicalMesh.py
-rwxr-xr-x  1 root root   1879 oct 20  2021 physicalWifiDirect.py
-rwxr-xr-x  1 root root   1263 oct 20  2021 position.py
-rwxr-xr-x  1 root root   1240 oct 20  2021 propagationModel.py
drwxr-xr-x  2 root root   4096 oct 20  2021 radius
drwxr-xr-x  6 root root   4096 oct 20  2021 replaying
-rw-r--r--  1 root root   3148 oct 20  2021 sflow.py
-rwxr-xr-x  1 root root   1880 oct 20  2021 simplewifitopology.py
-rwxr-xr-x  1 root root    582 oct 20  2021 socket_client.py
-rwxr-xr-x  1 root root   1380 oct 20  2021 socket_server.py
-rwxr-xr-x  1 root root   2668 oct 20  2021 sta_ap_mode.py
-rwxr-xr-x  1 root root   1548 oct 20  2021 telemetry.py
drwxr-xr-x  3 root root   4096 oct 20  2021 uav
-rwxr-xr-x  1 root root   1441 oct 20  2021 userap_managed_mode.py
-rwxr-xr-x  1 root root   2115 oct 20  2021 vanet.py
-rwxr-xr-x  1 root root   3183 oct 20  2021 vanet-sumo.py
-rwxr-xr-x  1 root root   1708 oct 20  2021 wifiDirect.py
-rwxr-xr-x  1 root root   1543 oct 20  2021 wmediumd_error_prob.py
-rwxr-xr-x  1 root root   1485 oct 20  2021 wmediumd_interference.py
-rwxr-xr-x  1 root root   1536 oct 20  2021 wmediumd_mobility.py
-rwxr-xr-x  1 root root   1674 oct 20  2021 wps_auth.py
-rwxr-xr-x  1 root root   1109 oct 20  2021 wwan.py
dclan@dclan2020:/home/mininet-wifi/examples$
```

Figura 5 - Escenarios por defecto en la carpeta examples.

Ejecutando el script *mobility.py* se puede observar la potencia del software de emulación, ya que es capaz de simular tanto redes *WiFi* como estaciones en movimiento, carga de ruido en la red, etcétera, como se aprecia en la Figura 6:

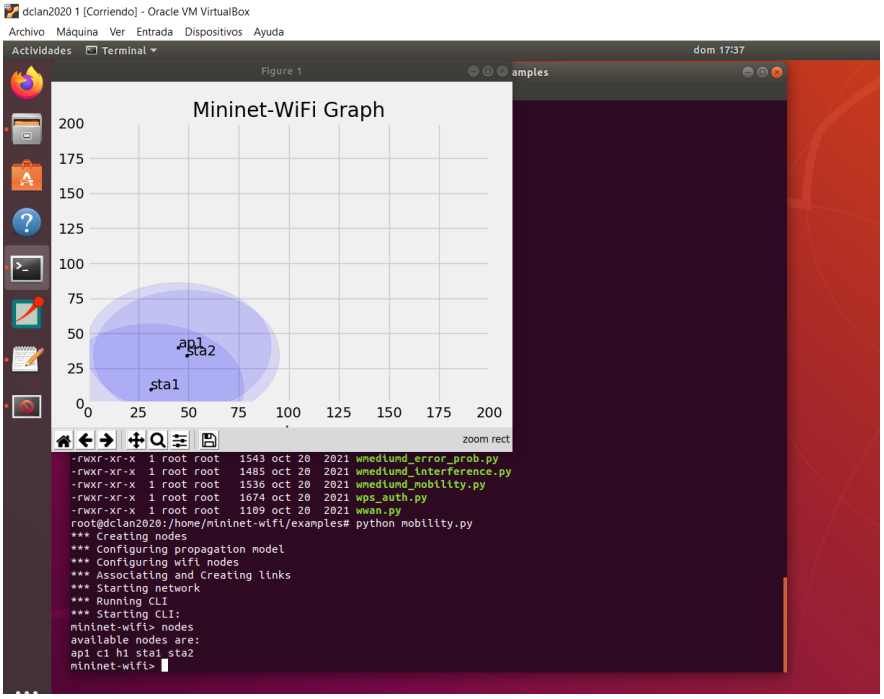

Figura 6 - Ejecución del script mobility.py.

En el siguiente apartado titulado “3.2 Modelos de movilidad y propagación” se entrará más en detalle acerca del funcionamiento de los distintos modelos que Mininet-WiFi nos ofrece, con el propósito de poder recrear un escenario lo más parecido a la realidad para satisfacer uno de los objetivos primarios: “Validar el funcionamiento de este emulador en base a un escenario emulado semejante a un escenario real”.

3.2 Modelos de movilidad y propagación

Mininet-WiFi ofrece muchas herramientas integradas bajo su software con la intención de que cada usuario pueda personalizar al máximo sus escenarios de simulación, objetivo perseguido desde la aparición de las primeras SDN, como se detalló en el apartado “2.1 Software-defined networking (SDN)”.

En cuanto a la movilidad, como indica dicha palabra, Mininet-WiFi ofrece distintos modelos para que nuestras estaciones se desplacen siguiendo diferentes algoritmos matemáticos. Para poder utilizarlos, basta con nombrar en el script python al método “setMobilityModel()” junto con una serie de parámetros especificados en la página oficial (8) que dependerán del modelo escogido, como se observa en la Figura 7. Existe un total de siete modelos: *‘RandomWalk’*, *‘TruncatedLevyWalk’*, *‘RandomDirection’*, *‘RandomWayPoint’*, *‘GaussMarkov’*, *‘ReferencePoint’*, *‘TimeVariantCommunity’*.

```
net.setMobilityModel(time=0, model='RandomWalk', min_x=10, max_x=300,
    min_y=10, max_y=300, constantDistance=1, constantVelocity=1)
```

Figura 7 - Definición modelo movilidad RandomWalk.

Más allá, también se indica en el manual (9) que es posible describir un movimiento *Straight-Line*, es decir, desplazar una estación en línea recta mediante la definición de dos puntos, es decir, proporcionando para ello dos coordenadas, origen y destino. Así pues, para poder cumplir de forma exitosa uno de los objetivos primarios "Evaluar de forma empírica el emulador Mininet-WiFi justificando su grado de semejanza con la realidad" de forma que el estudio sea lo más fiel a la realidad, se escogerá este modelo de movilidad ya que, al ser un movimiento en línea recta, es más fácil de reproducir en la vida real; de lo contrario, si se utilizase un modelo diferente, los resultados entre la realidad y lo virtual estarían muy alejados por la dificultad de reproducir un algoritmo matemático en la realidad cuando se realicen los cálculos de rendimiento en el apartado "5 Rendimiento real vs emulado".

Para la creación de nuestro propio escenario de emulación, que se detallará en el apartado "5.1 Reproducción emulada y recogida de datos", se han utilizado los métodos "startMobility()" y "stopMobility()" para iniciar y detener la simulación, y el método "mobility()" para especificar las coordenadas de origen y destino que guiarán la recta por la que se desplazará la estación. Un ejemplo extraído del manual (9) sería el que se muestra en la Figura 8:

```
net.startMobility( time=0 )
net.mobility( sta1, 'start', time=1, position='10,20,0' )
net.mobility( sta1, 'stop', time=59, position='30,50,0' )
net.stopMobility( time=60 )
```

Figura 8 - Métodos para desplazar una estación en línea recta.

En cuanto a la propagación, se refiere a la pérdida de potencia de la señal recibida debido a que, cuanto más se aleje la estación del AP, menor potencia de señal podrá recibir. Para emular dicho comportamiento, Mininet-WiFi menciona en su sitio web (10) hasta cuatro modelos, pero desgraciadamente solo dos de ellos están implementados y disponibles para los usuarios. Éstos son: 'Friis Propagation Loss Model' y 'Log-Distance/Log-Normal Shadowing Propagation Loss Model'. Los dos modelos restantes, que aún están en desarrollo, son 'International Telecommunication Union (ITU) Propagation Loss Model' y 'Two-Ray Ground Propagation Loss Model'; por lo tanto, se puede afirmar que Mininet-WiFi es un proyecto muy activo y en constante crecimiento, como se descubrirá con la realización de ciertas pruebas en el apartado *"6 Mejorando las tablas de flujo de Mininet-WiFi"* sobre actualización dinámica de las tablas de flujo.

Para escoger el más adecuado, primero se deben exponer sus características, por lo que se procederá a enumerarlas:

Friis Propagation Loss Model: se utiliza para modelar la pérdida de trayectoria de la línea de visión (*LoS*) incurrida en un espacio libre de obstáculos, sin objetos que ocasionen difracción, absorción, reflejos o cualquier otro fenómeno que altere las características de una onda radiada (11). Es posible ajustar la pérdida dependiendo del entorno en el que se realice el estudio con un exponente llamado "sL" (System Loss), aunque el valor recomendado es "2", como se aprecia en la Figura 9:

Environment	Path Loss Exponent (n)
Free space	2
Urban area cellular radio	2.7 to 3.5
Shadowed urban cellular radio	3 to 5
Inside a building - line-of-sight	1.6 to 1.8
Obstructed in building	4 to 6
Obstructed in factory	2 to 3

Figura 9 - Exponente de pérdida de trayectoria para varios entornos.

Log-Distance/Log-Normal Shadowing Propagation Loss Model: es una extensión del modelo anterior que se acerca más a la realidad debido a que admite una gama más amplia de entornos con obstáculos y objetos alrededor, ya que el modelo de Friis está restringido a una trayectoria clara sin obstáculos entre el transmisor y el receptor. El modelo abarca, además, efectos de sombreado aleatorios debido al bloqueo de la señal por colinas, árboles, edificios, etc. Para variar la pérdida también posee un exponente "sL" (System Loss) y este modelo sí puede utilizar cualquiera de los valores mostrados en la Figura 9.

Cómo es lógico, se escogerá este último modelo para la simulación debido a que es el que más se acerca a la realidad. En definitiva, se puede afirmar que Mininet-WiFi ofrece una amplia gama de modelos de movilidad y propagación, y es decisión del usuario elegir qué modelos se adaptan más dependiendo de sus prioridades. Para cumplir los objetivos de este trabajo, se ha decidido elegir un modelo de movilidad "straigth-line" y un modelo de propagación "Log-Distance", por ser aquellos que se acercaban más a la realidad.

3.3 Wmediumd

A lo largo de este trabajo ya se han mencionado algunos detalles sobre cómo funciona este simulador a bajo nivel, involucrando protocolos como *OpenFlow*. Más adelante, en el apartado "4.2 Monitorización de tramas 802.11 en un entorno simulado y justificación", se verificará la existencia de los controladores inalámbricos como "mac80211hwsim" (12) con *Wmediumd* activando la interfaz *hwsim0* para observar con *Wireshark* su funcionamiento —debido a que el kernel de Linux elimina algunos mensajes—. Algunos ficheros internos que facilitan la puesta en marcha de la simulación están relacionados con recrear, de la manera más fiel posible a la realidad, un entorno en el cual la red se vea afectada por parámetros como el ruido, distancias entre dispositivos, atenuaciones, etc.

Esto es posible gracias a que existen scripts como los resaltados en la Figura 10, cuyos autores son los responsables de todo el proyecto:

```
root@dclan2020:/home/mininet-wifi/mn_wifi# ls
associationControl.py  devices.py   manetRoutingProtocols.py  node.py              sixLoWPAN     vanet.py
bmv2.py                energy.py    mobility.py               plot.py              sumo          wmediumdConnector.py
clean.py               examples     module.py                 propagationModels.py telemetry.py  wwan
cli.py                 __init__.py  net.py                    __pycache__          test
data                   link.py      nodelib.py                replaying.py         topo.py
root@dclan2020:/home/mininet-wifi/mn_wifi# cat wmediumdConnector.py
"""Helps starting the wmediumd service

authors:    Patrick Grosse (patrick.grosse@uni-muenster.de)
            Ramon Fontes (ramonrf@dca.fee.unicamp.br)"""

import ctypes
import os
import socket
import tempfile
import subprocess
from time import sleep
import struct
import pkg_resources
from sys import version_info as py_version_info

from mininet.log import info, debug


class wmediumd_mode(object):
```

Figura 10 - Ficheros internos de Mininet-WiFi.

Mediante los pasos descritos en el apartado “3.1 Instalación del software”, además de instalar todo el software de Mininet-WiFi, también se instalaron ciertos paquetes fuera del propio directorio *mininet-wifi* que son de vital importancia para que se puedan realizar con éxito las simulaciones, como son los directorios *mac80211_hwsim_mgmt* y *openflow*, como se aprecia en la Figura 11. Cabe remarcar que no se detalla el contenido del directorio *openflow* debido a que se sale del ámbito de este trabajo, pero está estrechamente relacionado con el funcionamiento del propio protocolo, explicado brevemente en el apartado “2.2 OpenFlow*”*.

```
root@dclan2020:/home# ls
dclan  mac80211_hwsim_mgmt  mininet-wifi  openflow
root@dclan2020:/home# cd mac80211_hwsim_mgmt/
root@dclan2020:/home/mac80211_hwsim_mgmt# ls
cmake  CMakeLists.txt  hwsim_mgmt  LICENSE  Makefile  README.md
```

Figura 11 - Ficheros imprescindibles para la simulación.

El directorio *mac80211_hwsim_mgmt* tiene como finalidad hacer compatible Mininet-WiFi con el módulo “mac80211_hwsim”, que forma parte del kernel de Linux (13). Su funcionamiento consiste en utilizar el mismo medio virtual para todos los nodos inalámbricos y, en consecuencia, se consigue que todos ellos estén internamente dentro del alcance de los demás, y se puedan descubrir en un escaneo inalámbrico con las interfaces virtuales disponibles[5]. Es capaz de simular un número arbitrario de radios *IEEE 802.11* para mac80211 en un solo dispositivo. Éstas no tienen las limitaciones del hardware *WLAN* real, por lo que es fácil generar una configuración de prueba arbitraria, y reproducir

[5] aspecto muy relevante ya que centraliza la administración de toda la red en un componente, el controlador, como se vio en la explicación de la “capa de control” en el apartado “2.1.1 Arquitectura SDN”.

siempre la misma configuración para futuras pruebas. Además, dado que se simula todo el funcionamiento de la radio, se puede utilizar cualquier canal en las pruebas, independientemente de las normas reglamentarias.

A su vez, el controlador está estrechamente relacionado con *Wmediumd* debido a que está integrado también en su código fuente (14). La finalidad de *Wmediumd* es mejorar el controlador ofreciendo más funcionalidades mediante el soporte a la interferencia [6], capturando la posición de los nodos de Mininet-WiFi y calculando el nivel de la señal en función de la distancia entre la fuente y el destino basándose en el modelo de propagación de distancia logarítmica, modelo elegido en el apartado "3.2 Modelos de movilidad y propagación" (15).

[6] Aislando las interfaces inalámbricas entre sí consiguiendo un comportamiento bastante próximo a la realidad cuando dos estaciones están a una distancia lo suficientemente grande como para no ser visibles entre ellas, tomando además decisiones sobre cuándo se debe deautenticar una asociación en función del nivel de señal poseyendo una matriz en la que almacena valores de ancho de banda, pérdida, latencias y retardos de forma dinámica.

4. Análisis del tráfico generado mediante Wireshark

A continuación, se procederá a realizar algunas pruebas de concepto para satisfacer nuestro segundo objetivo secundario "Capturar el tráfico generado en la realidad y en la emulación mediante la herramienta *Wireshark*".

Por un lado, se capturarán tramas 802.11 —siendo éstas las correspondientes al protocolo *WiFi*— con una interfaz *TP-Link* para examinar con detalle sus cabeceras, estructura, formato, etc., con el propósito de poder inspeccionar qué tipo de tramas se generan para poder realizar posteriormente comparaciones con las tramas capturadas en el entorno de emulación, generadas mediante la creación de un escenario implementado con Mininet-WiFi. Es resumen, se realizará una captura del tráfico que circula por la red, específicamente por el punto de acceso de este estudio, con el fin de observar las cabeceras, los tipos de trama, qué estructura tienen, etc., con el objetivo de comparar el estándar de la implementación real, con las muestras tomadas en la simulación, obteniendo así más criterios de evaluación de esta tecnología.

El formato que siguen las tramas 802.11 (16) es el que se muestra en la Figura 12. En el anexo "*Explicación de los campos de una trama 802.11*" se podrá encontrar la función que realiza cada campo, con sus valores e interpretaciones, debido a que se sale del ámbito de este trabajo, pero resulta de interés a nivel académico para ser conscientes de la importancia de éstos.

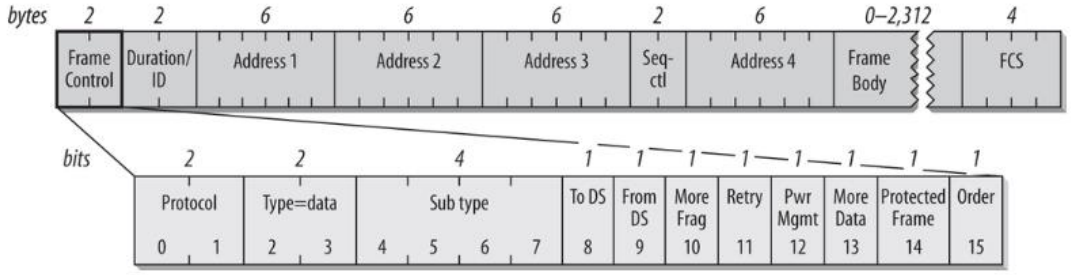

Figura 12 - Estándar de las tramas 802.11.

Para facilitar el análisis, se ha decidido comparar la trama *beacon* ya que se emiten periódicamente y se puede asegurar que se mantienen tienen las mismas condiciones en ambos entornos. Éstas son uno de los marcos de administración en redes WLAN que contienen toda la información sobre la red, es decir, son un medio de transmisión de capacidades y características que admite un BSS generalmente enviado por el "maestro de la red". De esta forma los AP difunden su presencia, y las estaciones crean su lista de AP disponibles con el objetivo de asociarse con uno de ellos. Generalmente, por defecto, se transmiten 10 *beacons*/segundo.

4.1 Monitorización de tramas 802.11 en un entorno real

A continuación se utilizará una máquina virtual Kali Linux de 64 bits con las siguientes características ofrecidas por la Figura 13:

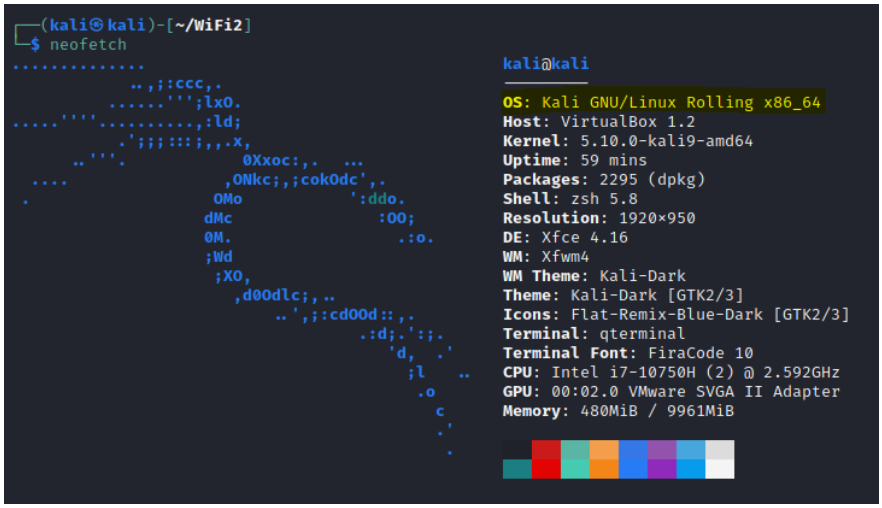

Figura 13 - Características máquina virtual Kali Linux.

Además, se utilizará una interfaz de red *TP-Link*, concretamente el modelo *TL-WN722N v2/v3*, con un aspecto como se puede observar en la Figura 14 (17).

Figura 14 - Interfaz USB WiFi TP-Link TL-WN722N v2/v3.

Para realizar con éxito las pruebas, se necesitará configurar la interfaz de red en modo monitor. Así, se escuchará todo el tráfico de la red debido a que será capaz de capturar todos los tipos de tramas como *Management*, *Data* y *Control*. También se conoce este comportamiento como *modo escucha* o *modo promiscuo* (18).

Existe cierta confusión con esta gama de modelos de tarjetas de red ya que el fabricante, en su versión inicial, del mismo nombre y similar apariencia —concretamente la versión uno con *chipset Atheros AR9271*—, sí soportaba el modo monitor por defecto, pero en sus versiones a la venta actuales cambió el chipset por otro diferente —concretamente las versiones dos y tres con *chipset Realtek RTL 8188EUS*— con el mismo nombre de modelo

de interfaz, ocasionando la incompatibilidad del funcionamiento en modo monitor de la interfaz de fábrica, impidiendo así la monitorización e inyección de paquetes, a pesar de que su hardware sí es compatible.

Este modo va a ser de gran utilidad debido a que solo así podremos escuchar el tráfico por la red, y por ello, la interfaz debe ser compatible con el modo monitor. En el anexo "Actualización drivers de la interfaz de red TP-Link TL-WN722N v2/v3" se describirán todos los pasos a seguir para resolver este problema actualizando sus drivers mediante software ya que, de lo contario, sería imposible realizar la captura de tramas en un entorno real. Así pues, ejecutando un script realizado por el alumno de nombre "*monitorMode.sh*" también descrito en el anexo mencionado, se observa cómo el modo monitor está habilitado, en la Figura 15:

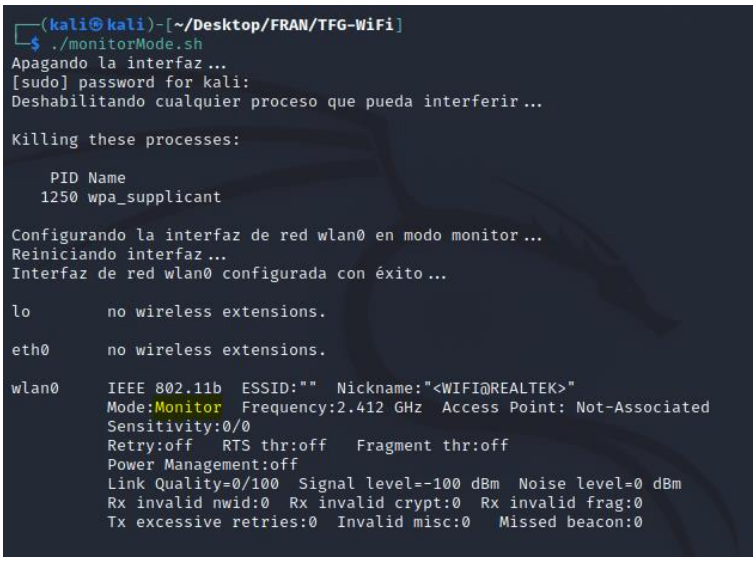

Figura 15 - Modo monitor habilitado.

Así pues, se procederá a observar el resultado de la captura que se ha obtenido, dónde la trama capturada tiene como origen el *router* del estudio y como destino, la dirección *broadcast*[7], debido a que se difunde por toda la red. Además, *Wireshark* identifica que el tipo de encapsulado es *IEEE 802.11* y que se trata de un *beacon frame*, como se observa en la primera línea remarcada de la Figura 16:

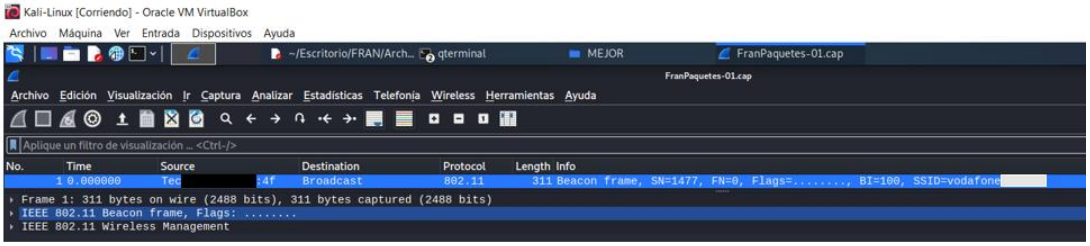

Figura 16 - Captura de una trama beacon en un entorno real.

[7] Dirección de red 255.255.255.255 reservada para transmitir los paquetes a toda la red independientemente de qué dispositivo sea, sin necesidad de esperar respuesta por parte del receptor.

Si se analiza el propio mensaje *beacon*, se puede apreciar que sí sigue la estructura definida por el estándar correspondiente al definido por la Figura 12 ya que el primer campo es *Frame Control Field* que alberga los campos *version*, *type* y *subtype*, como se observa en la Figura 17. Más allá, podemos observar que tiene asignada la versión cero, con tipo *00*, que se corresponde con una trama *Management* —las tramas de tipo gestión en 802.11—; y que pertenece al subtipo *1000* en binario, que es un *8* en decimal correspondiente a un *beacon*, como se podría esperar.

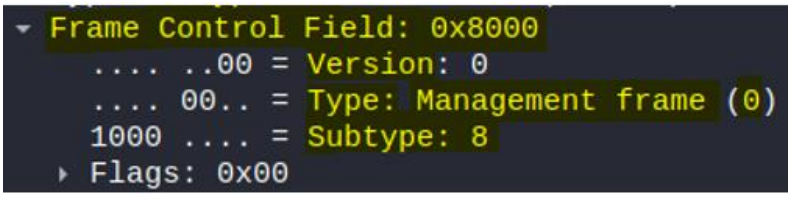

Figura 17 - Primer octeto de la trama 802.11 real.

A continuación, si se despliega la línea correspondiente a *flags*, se puede comprobar en la Figura 18 que se tienen los bits 8-15[8] [9] del estándar de la Figura 12.

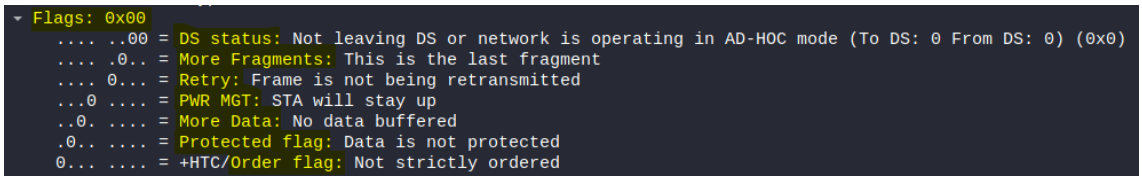

Figura 18 - Segundo octeto de la trama 802.11 real.

Si se sigue con el análisis, se puede verificar en la Figura 19 que el resto de los octetos se encuentran a partir de la línea que contiene *Duration*[9], junto con las direcciones de origen, destino, etc.

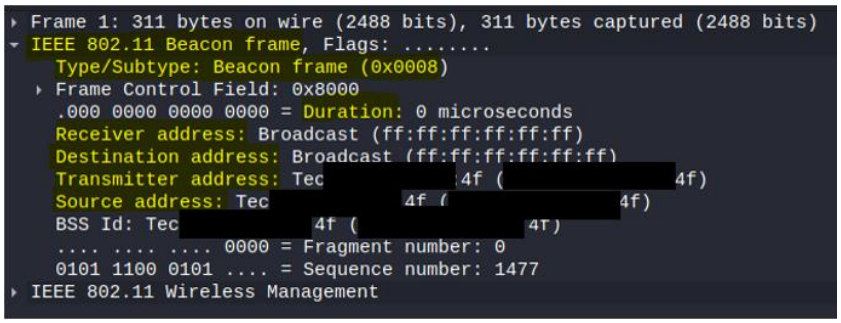

Figura 19 - Final de la trama 802.11 real.

[8] Los primeros dos bits correspondientes a *DS status* se corresponden con los bits 8-9 de la Figura 12.

[9] Cabe mencionar que la explicación de cada uno de los campos, junto con sus posibles valores e interpretaciones se pueden encontrar en el anexo "Explicación de los campos de una trama 802.11".

Por otro lado, dentro de la sección *IEEE 802.11 Wireless Management*, se observa que la información se clasifica en parámetros fijos y etiquetados (19) como se muestra en la Figura 20:

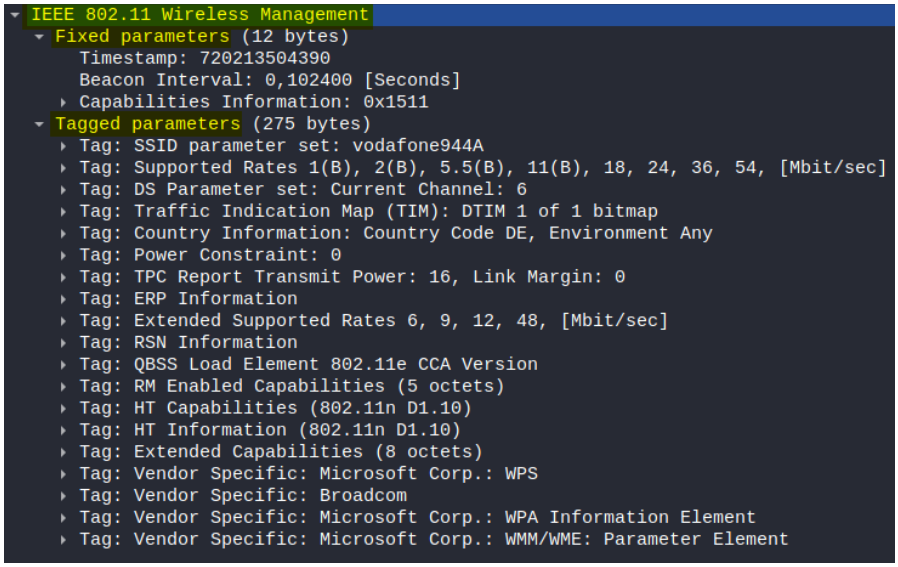

Figura 20 - Sección IEEE 802.11 Wireless Management.

Dentro de los parámetros fijos, se observan dos parámetros: *timestamp*[10] y *beacon interval*[11].

A su vez, dentro de *Capabilities Information* se observa también que el transmisor es un AP, que pertenece a un BSS, que no existe un coordinador en el AP, que se soporta la tecnología WEP, que está en uso *Short Slot Time*[12] o que no se permite *DSSS-OFDM*[13], todo ello mostrado en la Figura 21:

[10] Valor numérico que se correspondiente con un contador que indica cuánto tiempo ha estado encendido el AP y que permite la sincronización de las estaciones mediante *Time Sync Function*.

[11] Indica que se transmiten *beacons* cada 0.1s equivalente a 10 *beacons*/s, como se detalló al final de la introducción del apartado "4 Análisis del tráfico generado mediante Wireshark".

[12] Indica el tiempo que un dispositivo espera después de una colisión antes de retransmitir un paquete.

[13] Direct Sequence Spread Spectrum-Orthogonal Frequency-Division Multiplexing: así funcionaban las primeras versiones del protocolo 802.11 en el pasado, codificaban la señal para que solo la entendiera el destino y que los demás dispositivos solo recibieran ruido, sin saber interpretarlo en bandas ortogonales. Gran descubrimiento por parte de Hedy Lamarr.

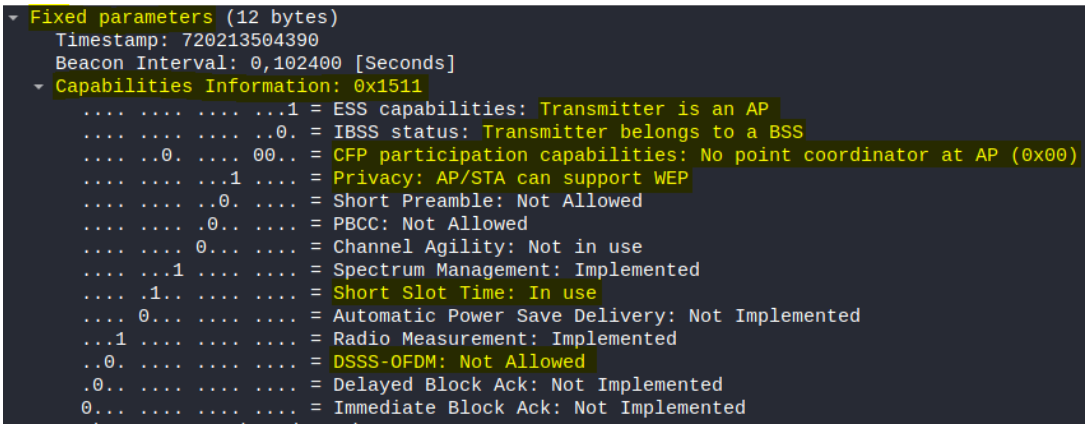

Figura 21 - Parámetros fijos de una trama 802.11 real.

Por otro lado, dentro de los parámetros etiquetados será dónde se almacene la información que comparte el AP con sus vecinos, como son su SSID y más información[9].

En definitiva, se ha realizado un análisis exhaustivo acerca del contenido de una trama *beacon* en la realidad, a continuación, en el apartado "4.2 Monitorización de tramas 802.11 en un entorno simulado y justificación*"* se investigará si el software de simulación se acerca o dista de la realidad.

4.2 Monitorización de tramas 802.11 en un entorno simulado y justificación

A continuación, se procederá a realizar una captura del tráfico generado en el software Mininet-WiFi, concretamente en el escenario creado —cuyos modelos escogidos se detallaron en el apartado "3.2 Modelos de movilidad y propagación" y, cuya explicación de su contenido será presentada en la Figura 31, la Figura 32 y la Figura 33—. Cabe mencionar, que una vez ejecutado el programa y la simulación en tiempo real, se activan sus interfaces correspondientes, y se pueden observar mediante *Wireshark*, tanto en la máquina virtual como también en las estaciones virtuales, debido a que, mediante Mininet-WiFi, es posible acceder a una terminal individual con el comando *xterm sta1*, aspecto muy importante y que mejora la calidad de los datos recogidos por la simulación, como se apreciará en el apartado actual y en "6 Mejorando las tablas de flujo de Mininet-WiFi".

Las interfaces de red que detecta *Wireshark* en la máquina virtual antes de ejecutar Mininet-WiFi son las que muestra la Figura 22:

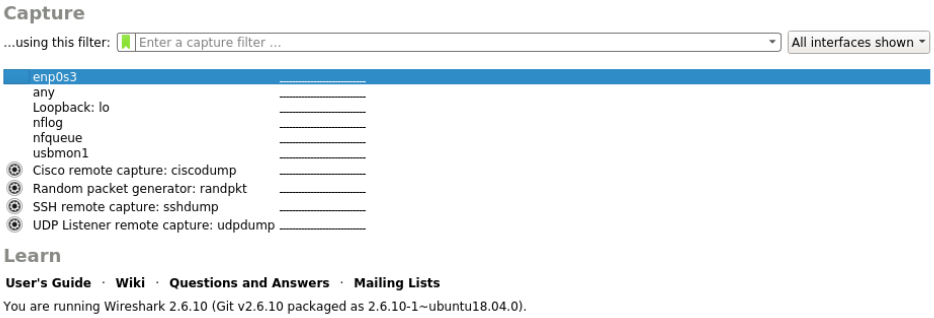

Figura 22 - Interfaces de red detectadas por Wireshark.

Además, se puede observar cómo están formadas las tramas generadas, con posibilidad de analizarlas y compararlas con el estudio realizado mediante la captura de *beacons* realizada con la interfaz *TP-Link* en el apartado "4.1 Monitorización de tramas 802.11 en un entorno real". Así, un AP definido mediante software con dos estaciones generaría las interfaces mostradas en la Figura 23:

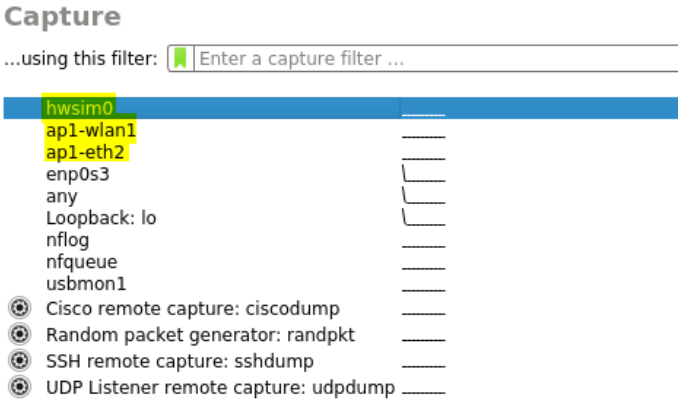

Figura 23 - Interfaces de red detectadas en Wireshark a posteriori.

Así pues, se puede observar que Mininet-WiFi genera más niveles de capas en comparación a la muestra tomada de la realidad en la Figura 16, una causa podría ser que la interfaz de red utilizada no capturara tanta información; por lo tanto, es un aspecto positivo, teniendo la siguiente estructura de tramas *beacon* mostrada en la Figura 24:

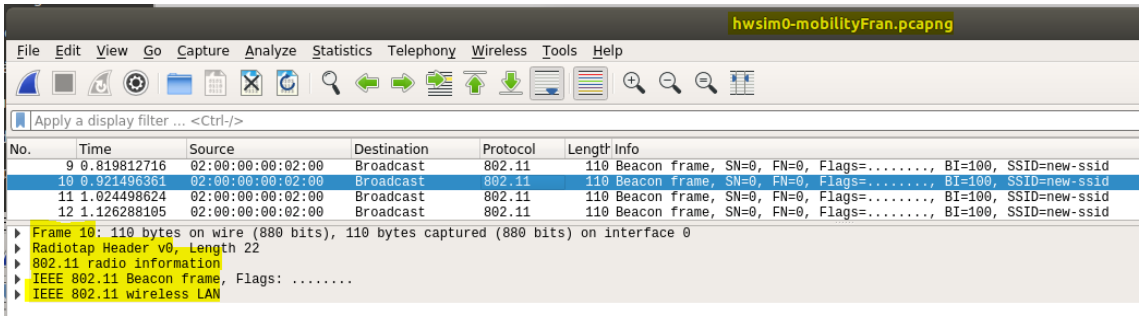

Figura 24 - Captura de una trama beacon en un entorno simulado.

Si se accede directamente a la trama que *Wireshark* detecta con encapsulado *IEEE 802.11 Beacon frame*, se observa que se reciben exactamente los mismos datos a los obtenidos en la Figura 17, como se observa en la Figura 25:

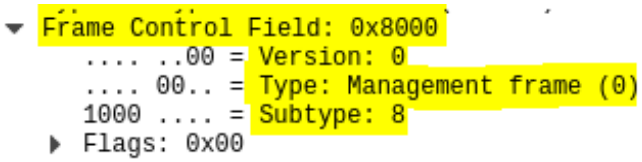

Figura 25 - Primer octeto de la trama 802.11 virtual.

A continuación, si se despliega la línea correspondiente a *flags*, se puede observar una vez más como los datos obtenidos en la realidad de la Figura 18 se corresponden con los obtenidos en el escenario virtual, mostrados en la Figura 26:

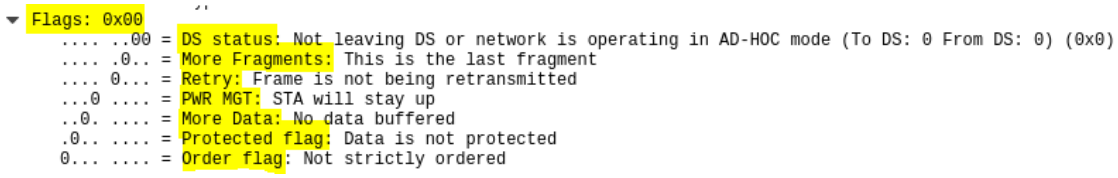

Figura 26 - Segundo octeto de la trama 802.11 virtual.

Continuando con el análisis, se puede verificar que el resto de los octetos son semejantes a los de la Figura 19, como se aprecia en la captura obtenida en la simulación en la Figura 27:

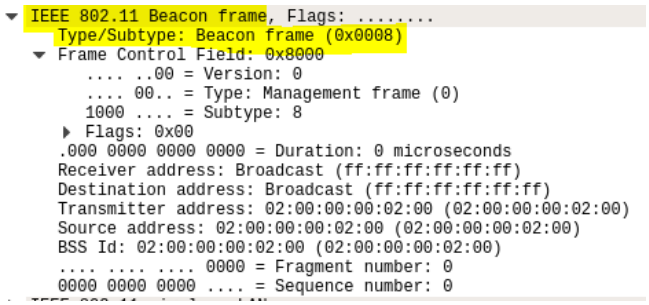

Figura 27 - Final de la trama 802.11 virtual.

Sin embargo, aparece una sección diferente nombrada *IEE 802.11 wireless LAN*, sin aparecer la palabra *Management* que aparecía en la Figura 20, pero muestra internamente una apariencia casi idéntica ya que se tiene el mismo valor para el intervalo de envío de *beacons*, pero sin embargo tenemos menor cantidad de parámetros etiquetados, dividiéndose en parámetros fijos y etiquetados, como se muestra en la Figura 28:

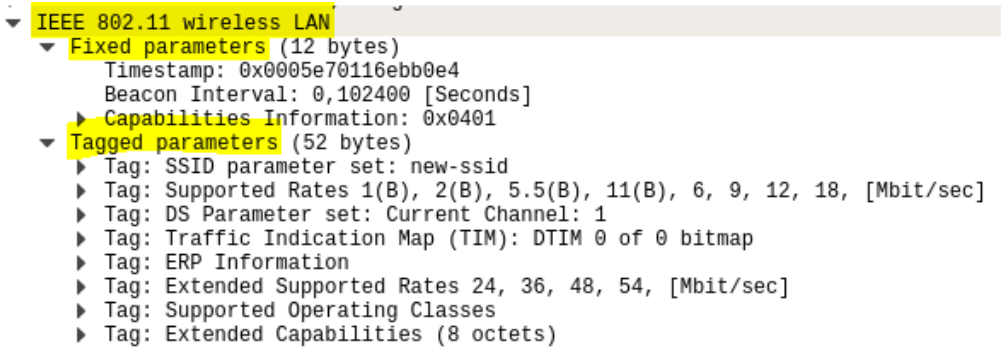

Figura 28 - Sección IEEE 802.11 wireless LAN.

Dentro de *Capabilities Information* se obtienen algunas diferencias con respecto a la Figura 21 destacadas en color rojo en la Figura 29, como la característica de no soportar WEP –ofrece mayor seguridad— y que no está implementada la gestión del espectro y la recogida de datos de aspectos relacionados con radiofrecuencias, quizá sea la razón por la cual no se podrán obtener medidas actualizadas dinámicamente de la potencia de la señal recibida en dBm, como se detallará en el apartado "5.2 Reproducción en la realidad y recogida de datos".

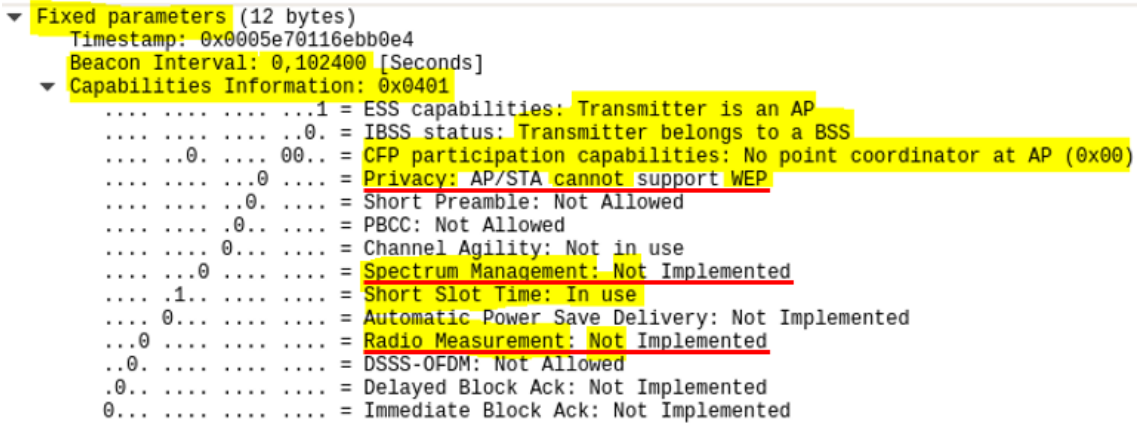

Figura 29 - Parámetros fijos de una trama 802.11 virtual.

Más allá del análisis realizado, se han detectado ciertas limitaciones que poseen las redes definidas mediante software en algunas capturas realizadas debido a que hay algunos paquetes que no llegan a formarse adecuadamente, y por ello *Wireshark* los etiqueta como *Malformed Packet*, como ha sucedido en la Figura 30 con la trama *Probe Request*.

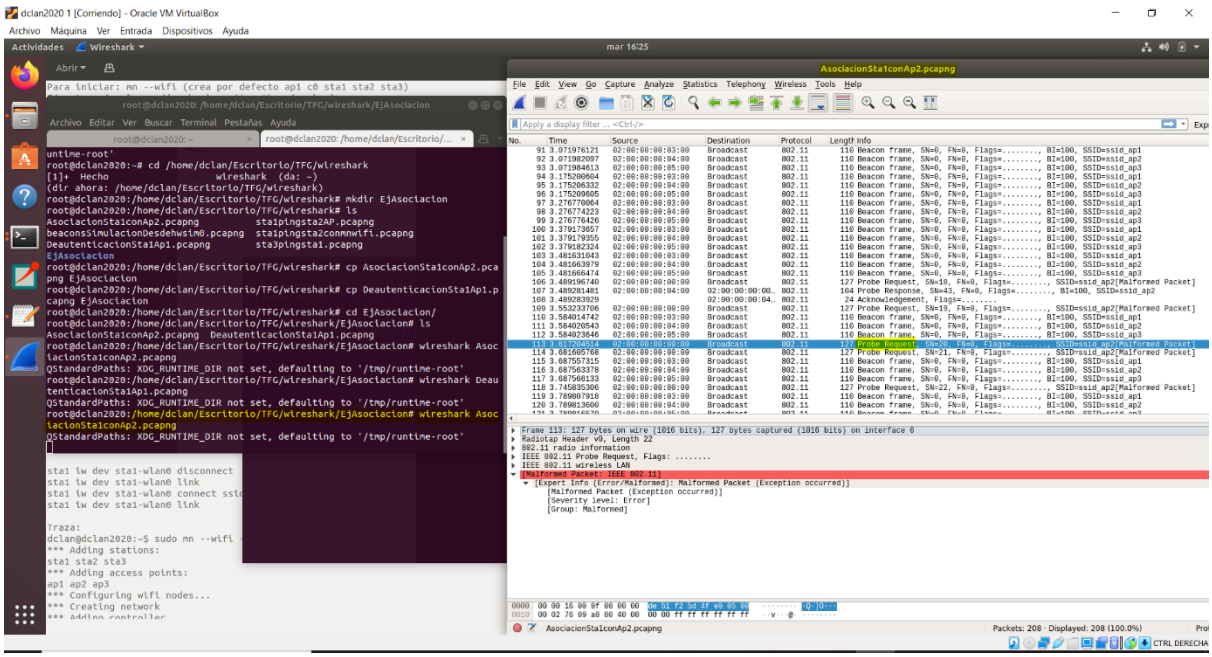

Figura 30 - Malformación de tramas en Mininet-WiFi.

5. Rendimiento real vs emulado

5.1 Reproducción emulada y recogida de datos

Como ya se mencionó anteriormente en el apartado "3.2 Modelos de movilidad y propagación", se tomó la decisión de escoger aquellos modelos que se acercasen más a la realidad, por ello se eligió *Straight-Line* como modelo de movilidad y *Log-Distance* como modelo de propagación. Así pues, para crear el escenario que posteriormente se ejecutará mediante la interfaz gráfica de Mininet-WiFi para analizar rendimientos y recoger datos, se presentarán las partes del script con mayor relevancia, detallando a su vez el comportamiento esperado.

Inicialmente se deberán crear y configurar los nodos de la red (20). Para nuestro estudio, bastará con crear un host —que actúe como servidor para calcular rendimientos a posteriori—, dos estaciones —que serán las que se irán desplazando por el mapa— y un punto de acceso, sin olvidar que toda SDN necesita de un controlador para gestionar toda la red desde un único lugar[14], como se observa en la Figura 31:

```
# Crea un host 'h1' para actuar como servidor para calcular rendimientos:
h1 = net.addHost('h1', mac='00:00:00:00:00:01', ip='10.0.0.1/8')

# Crea dos estaciones, 'sta1' and 'sta2':
sta1 = net.addStation('sta1', mac='00:00:00:00:00:02', ip='10.0.0.2/8')
sta2 = net.addStation('sta2', mac='00:00:00:00:00:03', ip='10.0.0.3/8')

# Crea un controlador 'c1' y un punto de acceso 'ap1' con SSID 'new-ssid',
# funcionando bajo el protocolo 802.11g con 2.4GHz y 54Mbps como máx,
# transmitiendo en el canal 1 y la posición en nuestra interfaz
# gráfica será  (x = 87; y = 100):
ap1 = net.addAccessPoint('ap1', ssid='new-ssid', mode='g', channel='1', position='87,100,0')
c1 = net.addController('c1')
```

Figura 31 - Creación de los nodos del escenario de simulación.

A continuación, se deberá de especificar que se desea que las estaciones vayan perdiendo potencia de señal siguiendo nuestro modelo, que reducirá la potencia según el logaritmo de la distancia, a medida que se van alejando, junto con el exponente para acercarse más a la realidad[15], como se observa en la Figura 32:

[14] Como se detalló con la explicación de la capa de control en el apartado "2.1.1 Arquitectura SDN".

[15] Recordar la tabla mediante la cual es posible definir entornos variados en la Figura 9, en nuestro caso estaríamos dentro de un edificio con obstáculos.

```
info("*** Configurando el modelo de propagación escogido\n")
net.setPropagationModel(model="logDistance", exp=4.5)
```

Figura 32 - Definición del modelo de propagación Log-Distance.

Una vez creados los dispositivos de la red y definido el modelo de propagación, se proseguirá con las órdenes para utilizar el modelo de movilidad propuesto, como se vio en el ejemplo de la Figura 8. Para ello, se utilizarán los métodos "mobility()", "startMobility()" y "stopMobility()" siempre con el tiempo en el que se desea que suceda la acción, como puede observar en la Figura 33. El modelo funciona de la siguiente forma: como existen dos estaciones en nuestro escenario, y se debe definir por cada una de ellas su origen y su destino, se definirán cuatro posiciones, las cuáles se utilizarán cuando se nombre al método "mobility()", pasándolas como un puntero, y decidiendo la acción mediante las palabras clave "start" y "stop".

```
# Comienza la simulación con el modelo de movilidad Straight-Line:
net.startMobility(time=0, mob_rep=1, reverse=False)

# Definición de las posiciones involucradas en la trayectoria de ambas estaciones,
# p1 y p2 son los puntos de los que parten las estaciones,
# p3 y p4 los puntos donde se detendrán:

p1, p2, p3, p4 = dict(), dict(), dict(), dict()
if '-c' not in args:
    p1 = {'position': '75.0,100.0,0.0'}
    p2 = {'position': '100.0,100.0,0.0'}
    p3 = {'position': '45.0,100.0,0.0'}
    p4 = {'position': '115.0,125.0,0.0'}

# En el 1s empieza a moverse 'sta1' partiendo desde el punto (x = 75; y = 100):
net.mobility(sta1, 'start', time=1, **p1)
# A los 2s empieza a moverse 'sta2' partiendo desde el punto (x = 100; y = 100):
net.mobility(sta2, 'start', time=2, **p2)

# Fin del movimiento en los segundos 40 y 41:
net.mobility(sta1, 'stop', time=40, **p3)
net.mobility(sta2, 'stop', time=41, **p4)

# Fin del modelo de movilidad:
net.stopMobility(time=45)
```

Figura 33 - Definición del modelo de movilidad Straight-Line.

Teniendo en cuenta algunas órdenes esenciales[16] para el uso de Mininet-WiFi, se ejecutará el script con la orden "*python mobilityFran.py*" apareciendo en el escenario de simulación, como se observa en la Figura 34, coincidiendo así cada elemento con las posiciones definidas en la Figura 33.

[16] Se adjuntará una serie de órdenes sencillas para el uso básico de esta herramienta en el anexo "Órdenes esenciales para el uso de Mininet-WiFi".

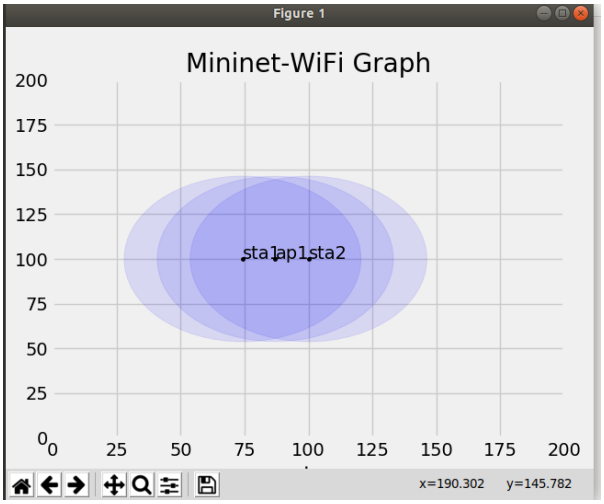

Figura 34 - Interfaz gráfica de Mininet-WiFi.

Mientras, en la terminal, aparecen algunos mensajes, mostrados en la Figura 35, junto con el prompt[17] de Mininet-WiFi esperando a recibir órdenes. Será mediante esta forma como se podrá calcular sus rendimientos ya que el software ofrece hasta cierto punto datos de forma dinámica. Por lo tanto, la estrategia a seguir será recabar datos acerca del rendimiento en diferentes puntos de la trayectoria, para poder comprobar si de verdad se cumple que cuánto más se aleja una estación del punto de acceso, menor potencia de señal puede recibir.

```
root@dclan2020:/home/mininet-wifi/examples# python mobilityFran.py
*** Creación de los nodos
*** Configurando el modelo de propagación escogido
*** Configurando los nodos WiFi
*** Asociación del host 'h1' con  el punto de acceso 'ap1'
*** Construyendo la red
*** Ejecutando CLI
*** Starting CLI:
mininet-wifi>
```

Figura 35 - Mensajes por terminal e interacción de forma dinámica.

Para calcular los rendimientos, se hará uso de la herramienta *iperf*[18] ya que Mininet-WiFi ofrece este servicio para determinar, entre dos dispositivos, la velocidad de la red de bajada y de subida, mediante la arquitectura cliente-servidor. Es decir, la forma mediante la cual se procederá para la recogida de los datos será la siguiente: se ejecutará *iperf* en diferentes momentos del recorrido de la simulación, alejándose la estación "sta1" del punto de acceso "ap1", actuando como cliente la estación y como servidor, "ap1" mediante "h1" —debido a que "ap1" no posee ninguna IP y es necesaria para el funcionamiento de *iperf*; por tal razón se crea "h1", y se conecta vía ethernet con "ap1"—. Cabe remarcar que las distancias que se tomarán serán las mismas que posteriormente se probarán cuando se

[17] Se llama prompt al carácter o conjunto de caracteres que se muestran en una línea de comandos para indicar que está a la espera de órdenes.

[18] Herramienta de línea de comandos que realiza pruebas de rendimiento en la red cuyo manual detalla cómo utilizar cada uno de sus parámetros en https://linux.die.net/man/1/iperf

analice en un entorno real[19], con el fin de tener el mayor grado de precisión, como se detallará en el apartado “5.2 Reproducción en la realidad y recogida de datos”.

En el origen, situando “sta1” a una distancia de 0.7 metros de distancia de “ap1”, se puede observar en la Figura 36 la interfaz gráfica de Mininet-WiFi y en la Figura 37, el rendimiento obtenido —correspondiente a 10.5Mbps de bajada y a 13Mbps de subida a la distancia mencionada—, además de obtener el tiempo mínimo, promedio y máximo con la herramienta *ping* desde la estación “sta1” hasta “h1” como muestra la Figura 38 —concretamente con el intercambio de cinco trazas ICMP un tiempo mínimo/promedio/máximo de 1.165/2.021/5.265 milisegundos respectivamente—.

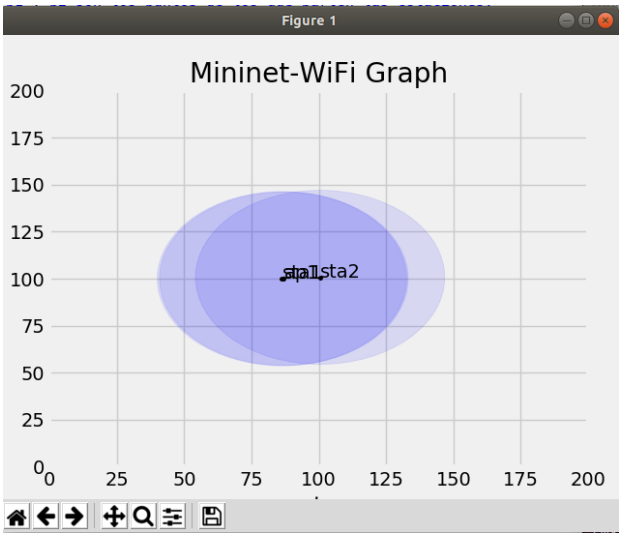

Figura 36 - Interfaz gráfica con distancia de 0,7m entre “sta1” y “ap1”.

```
root@dclan2020:/home/mininet-wifi/examples# python mobilityFran.py
*** Creación de los nodos
*** Configurando el modelo de propagación escogido
*** Configurando los nodos WiFi
*** Asociación del host 'h1' con  el punto de acceso 'ap1'
*** Construyendo la red
*** Ejecutando CLI
*** Starting CLI:
mininet-wifi> iperf sta1 h1
*** Iperf: testing TCP bandwidth between sta1 and h1
*** Results: ['10.5 Mbits/sec', '13.0 Mbits/sec']
mininet-wifi> distance sta1 ap1
The distance between sta1 and ap1 is 0.7 meters
mininet-wifi>
```

Figura 37 - Rendimiento con una distancia de 0,7m.

[19] En realidad, cuando se realizó el experimento, primero se recogieron los datos del entorno real, debido a la flexibilidad para cambiar distancias en el entorno virtual.

```
mininet-wifi> sta1 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=5.26 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=1.28 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=1.17 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=1.16 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=1.21 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 1.165/2.021/5.265/1.622 ms
```

Figura 38 - Ping con una distancia de 0,7m.

Este proceso se repetirá para obtener todos los datos requeridos para el estudio midiendo estas variables, aumentando la distancia a 3.5m, 8.8m, 12.4m, 22.1m, 28m, 35.1m, 43m y 50m.

Cabe remarcar que a una distancia de 43m, resulta de interés ya que "sta1" sí puede observar a "ap1", pero al contrario no. Debido a que "ap1" es quién recibe la petición del cliente, "sta1", averigua de dónde proviene y puede enviarle la respuesta, a pesar de que desde la perspectiva de "ap1", "sta1" está fuera de rango, como se observa en la Figura 39, obteniendo los resultados mostrados en la Figura 40 y la Figura 41.

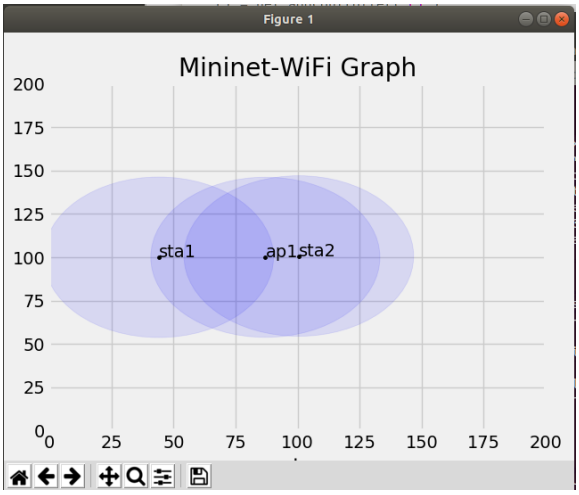

Figura 39 - Interfaz gráfica con distancia de 43m entre "sta1" y "ap1".

```
mininet-wifi> distance sta1 ap1
The distance between sta1 and ap1 is 43.0 meters
mininet-wifi> iperf sta1 h1
*** Iperf: testing TCP bandwidth between sta1 and h1
*** Results: ['6.87 Mbits/sec', '8.67 Mbits/sec']
mininet-wifi>
```

Figura 40 - Rendimiento con una distancia de 43m.

```
mininet-wifi> sta1 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=7.75 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=4.22 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=3.35 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=3.40 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=3.76 ms
^C
--- 10.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 3.354/4.499/7.756/1.658 ms
mininet-wifi>
```

Figura 41 - Ping con una distancia de 43m.

Además, a la distancia de 50 metros se puede observar cómo los rangos de distancia que abarcan tanto "sta1" como "ap1" coinciden únicamente en una pequeña zona común de intersección como se muestra en la Figura 42, y ninguno de los dispositivos está situado ahí, por lo tanto, al lanzar a ejecución *iperf sta1* como cliente no podrá enviarle a "h1" datos, y el servidor nunca podrá devolverle la respuesta, ocasionando así un error que ocasionará el final abrupto de la simulación, como se aprecia en la Figura 43 y la Figura 44.

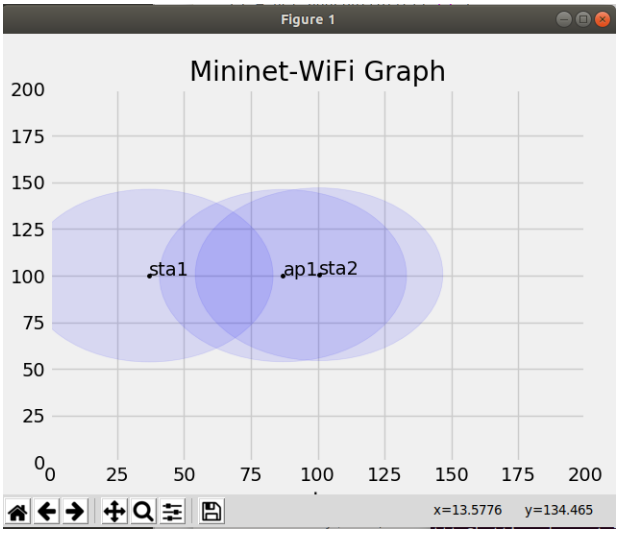

Figura 42 - Interfaz gráfica con distancia de 50m entre "sta1" y "ap1".

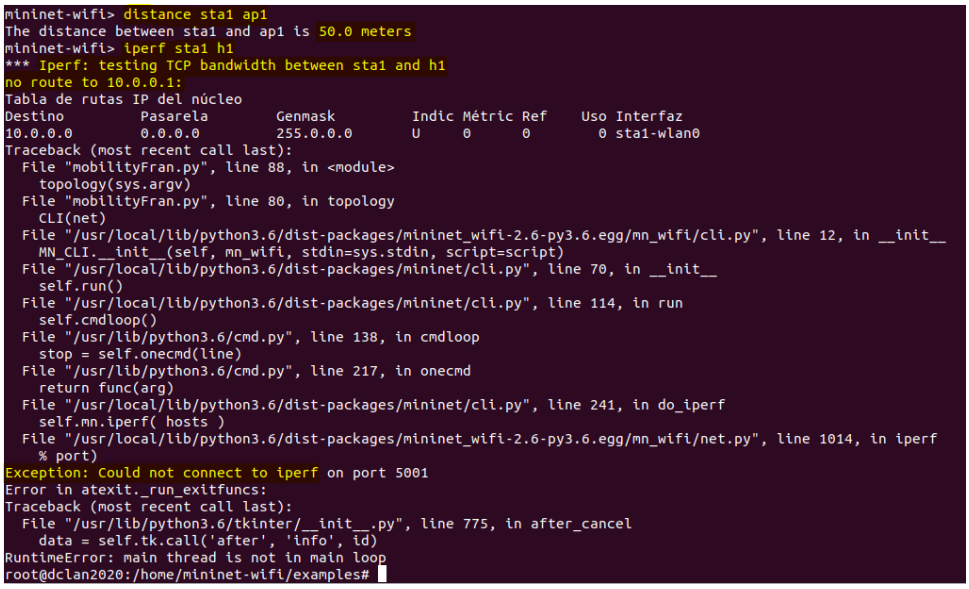

Figura 43 - Rendimiento con una distancia de 50m.

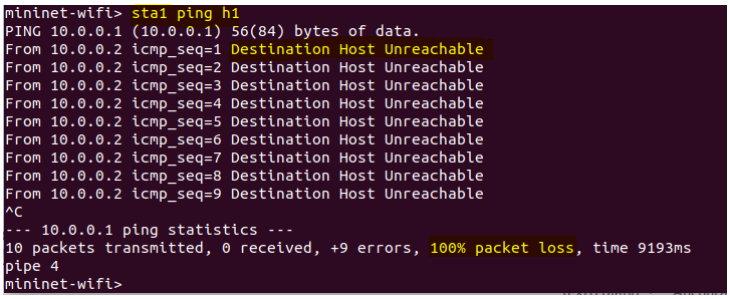

Figura 44 - Ping con una distancia de 50m.

Una desventaja que se ha descubierto en Mininet-WiFi es el hecho de que no soporta ofrecer de forma dinámica el dato de la potencia de señal, dBm, ya que independiente de la distancia, en nuestro escenario siempre es -36dBm, ya sea con una distancia de 12.4m o 22.1m, por ejemplo, en la Figura 45 y la Figura 46:

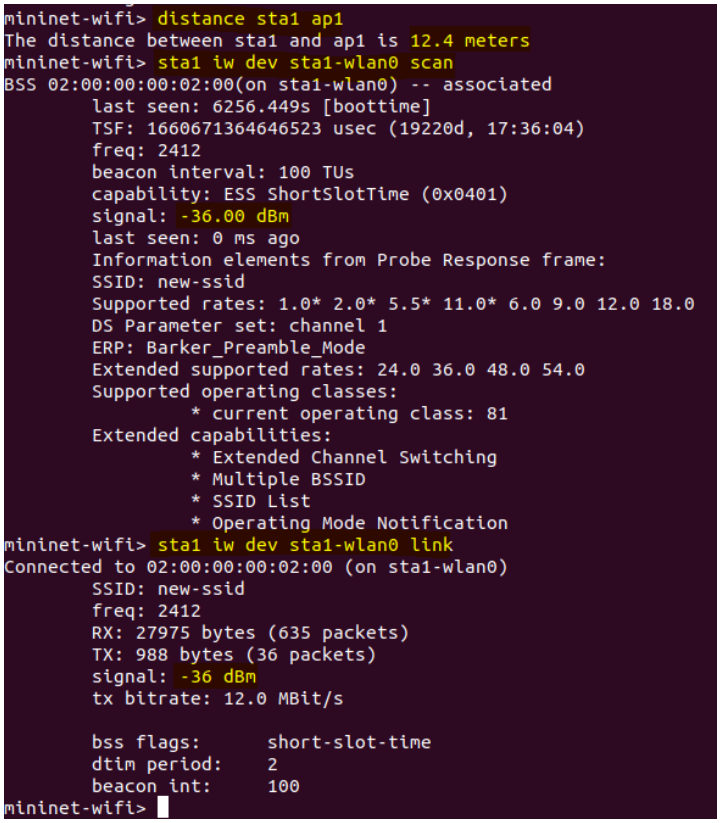

Figura 45 - Potencia de la señal sin variar a 12.4m de distancia.

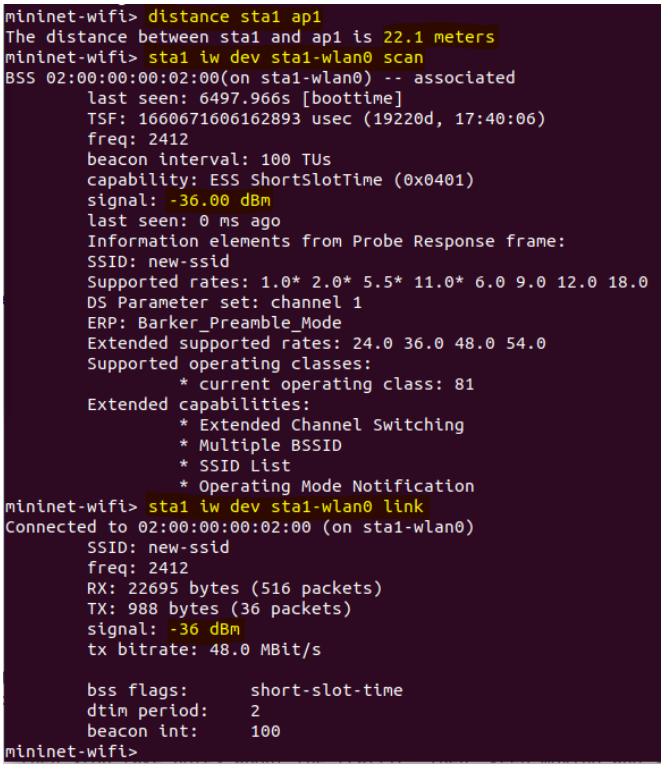

Figura 46 - Potencia de la señal sin variar a 22.1m de distancia.

Así pues, el propio manual de Mininet-WiFi también afirma que los comandos ejecutados con la herramienta *iw* solo deben de ser usados en escenarios estáticos y no en simulaciones más realistas debido a que no está soportado (9).

Siempre, para finalizar cada simulación y comenzar una nueva, se utilizarán las órdenes "exit" y "mn -c" para finalizar con éxito y evitar problemas con los controladores, limpiando así archivos temporales, etc., como se muestra en la Figura 47:

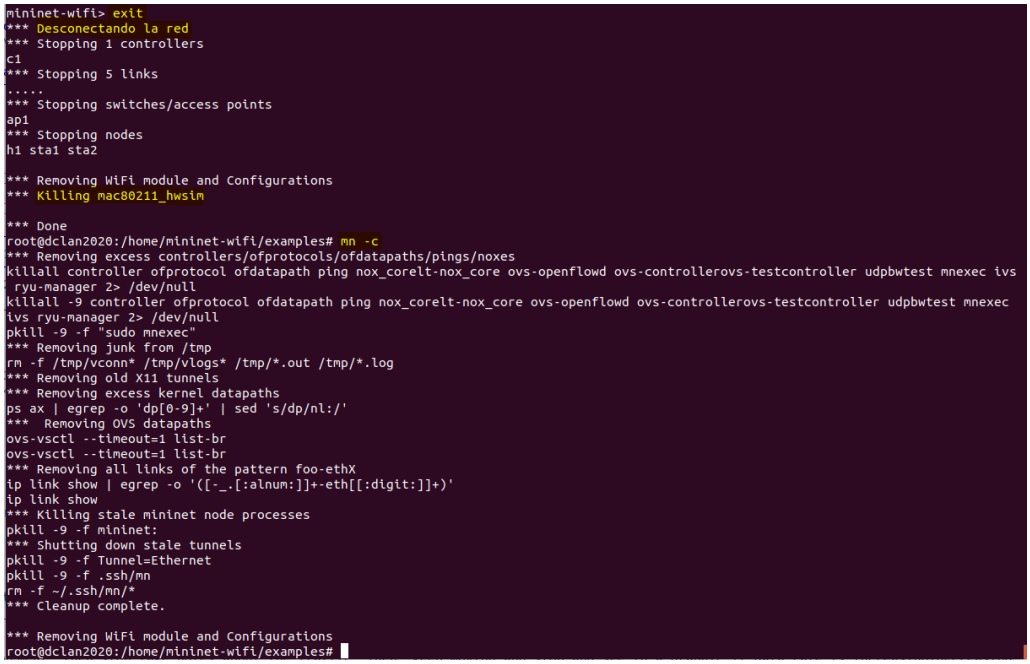

Figura 47 - Finalización de la simulación.

En definitiva, y para presentar los datos de una forma más visual, se presentarán una serie de tablas y gráficas con el fin de comprender el comportamiento que ha seguido la simulación, con todos los datos recogidos, con el fin de lograr que el lector pueda entender con plenitud el comportamiento de la red.

Tabla 1 - Datos del rendimiento de la simulación.

Distancia (m)	ping (ms)				iperf (Mbps)	
	min	max	promedio	perdidos	descarga	subida
0,7	1,165	5,265	2,021	0%	10,5	13
3,5	1,361	4,584	2,029	0%	10,2	12,5
8,8	1,609	4,842	2,283	0%	9,65	11,9
12,4	1,921	5,135	2,586	0%	9,31	11,6
22,1	2,281	5,89	3,035	0%	8,45	10,5
28	2,615	7,095	3,546	0%	7,97	9,84

35,1	2,963	7,358	3,88	0%	7,43	9,23
43	3,354	7,756	4,499	0%	6,87	8,67
50	∞	∞	∞	100%	0	0

Mediante las muestras recogidas, podemos deducir que sí se asemejan bastante a la realidad debido a que, conforme se va aumentando la distancia entre al estación y el punto de acceso, el rendimiento va bajando, observándose en la caída de las estadísticas *ping* tardando más milisegundos tanto en mínimo, como en máximo y en promedio. Además, mediante *iperf* se puede observar como se va reduciendo la velocidad cada vez que aumentamos la distancia. Cabe destacar que cuando la distancia se hace superior, se pierden el 100% de los paquetes, perdiendo la conexión y, en consecuencia, se obtiene una velocidad nula ya que no es posible la transferencia de archivos, como se observa en la última fila de la Tabla 1.

Para una mejor interpretación, se ha decidido crear dos gráficas a partir de la Tabla 1 con el fin de representar los datos de forma más visual. En la Figura 48, se oberva que se tienen tres rectas que representan cada uno de los datos ofrecidos por la herramienta *ping*, siendo la de color azul la correspondinte al menor tiempo obtenido cuando se realizó la prueba, la verde la de tiempo máximo y, por último, la morada con el promedio de los parámetros anteriores. Se observa como a medida que se incrementa la distancia, los tiempos aumentan, tardando más tiempo en recibir los datos, empeorando así el rendimiento de la red. Remarcar que cuando llega la distancia de 50 metros, se pierde la conexión y, por esta razón, no se puede realizar con éxito el *ping*.

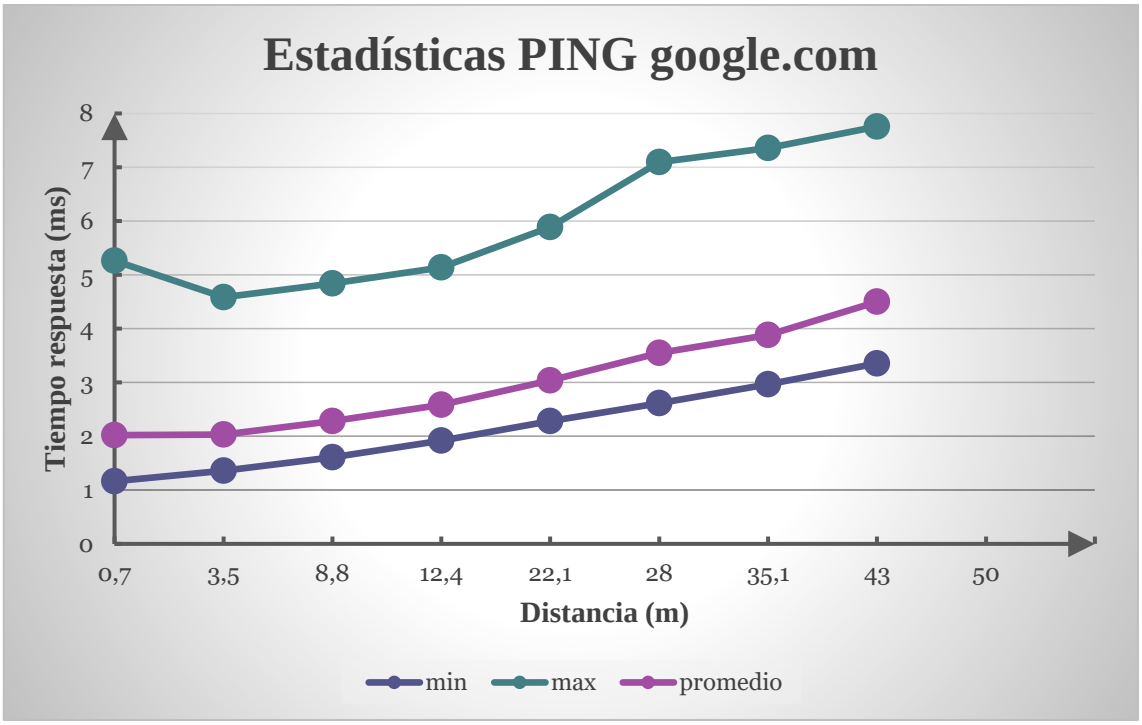

Figura 48 - Gráfica estadísticas ping en el entorno emulado.

Por otro lado, en la Figura 49, se observan los resultados obtenidos por la herramienta *iperf*, mostrando así como a medida que se va aumentando la distancia, el rendimiento va decreciendo hasta que llega un punto en el que se hace nulo debido a que se pierde la conexión.

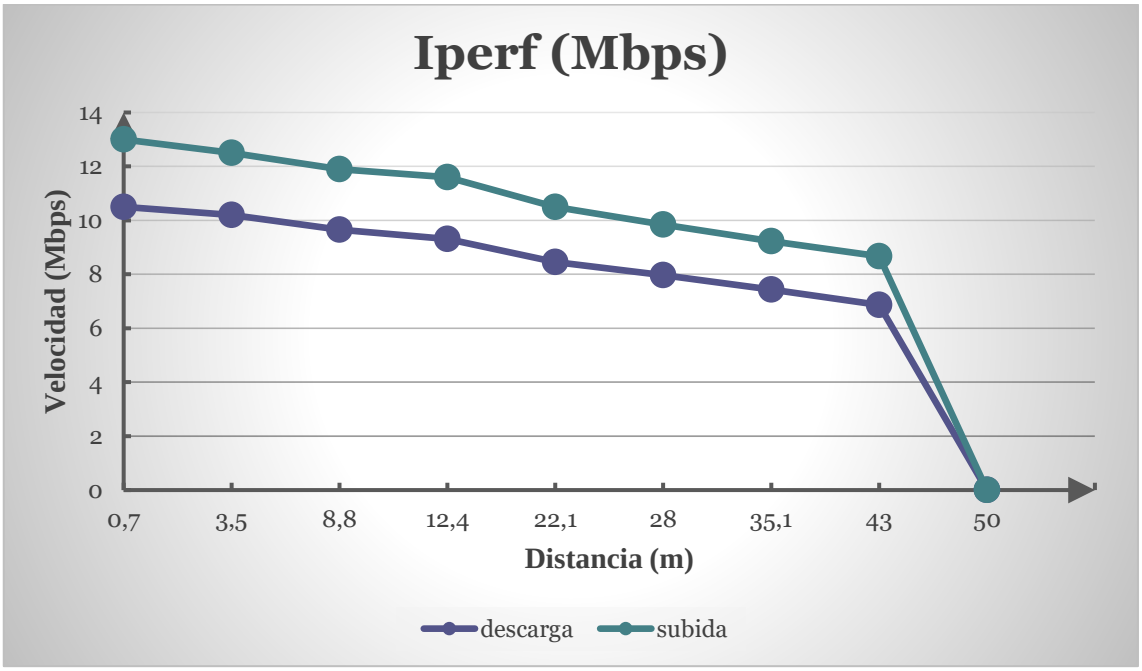

Figura 49 - Gráfica iperf en el entorno simulado.

5.2 Reproducción en la realidad y recogida de datos

A continuación, se van a realizar las pruebas en un entorno real y, para ello, utilizaremos un punto de acceso que emitirá en el mismo canal que el punto de acceso virtual, utilizando el canal 1, y se utilizará como estación móvil un smartphone con cierto software que permitirá realizar las pruebas requeridas, llamado *Fing*[20], como se detallará en breve.

La fuerza de la señal se representa en formato "dBm" y se calcula como el índice de potencia en decibeles (dB) de la potencia medida en referencia a un miliwatio. Mientras más cercano sea el valor a 0, más fuerte será la señal, por ejemplo -30 dBm obtendría un mejor rendimiento que -50 dBm. Para realizar las pruebas, trataremos de medir la señal y otros parámetros partiendo desde el origen del punto de acceso hasta alejarnos lo suficiente como para salir del rango máximo que abarca el punto de acceso.

Antes de empezar con la recogida de datos, se ha analizado la señal recibida del punto de acceso con el portátil del estudiante mediante el software de *InSSIDer*[21], obteniendo las características mostradas en la Figura 50 y la Figura 51:

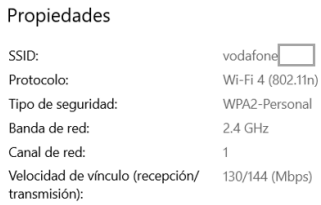

Figura 50 - Propiedades del punto de acceso real.

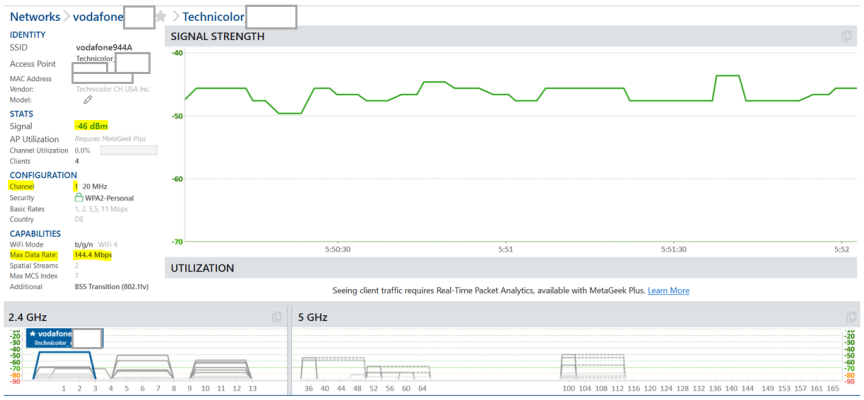

Figura 51 – Gráfica de la potencia de señal recibida (dBm) del punto de acceso real.

[20] Herramienta de auditoría y descubrimiento de redes inalámbricas. Disponible en https://www.fing.com/

[21] Software que permite escanear redes WiFi. Disponible en https://www.metageek.com/inssider/

A continuación, se mostrarán los datos recogidos con el dispositivo móvil mediante el software *Fing*, obteniendo parámetros como la calidad de la señal recibida (dBm), el canal y la banda en la que emite —exactamente la misma que en el entorno virtual vía en canal 1 en la banda 2412MHz—, la distancia (m); además, se realizará un *ping* para ver valores mínimos, máximos y promedios con su gráfica, para poder comparar las gráficas del virtual y el real más adelante y ,por último, se realizará un test de rendimiento como *iperf*, integrado en la herramienta, para ser conscientes de las velocidades de bajada y subida, así como detallando su correspondiente gráfica.

A una distancia de 0.7m, se obtienen los resultados mostrados en la Figura 52, la Figura 53 y la Figura 54:

Figura 52 - Potencia de la señal recibida a 0.7m.

Figura 53 - Ping con una distancia de 0.7m.

Figura 54 - Rendimiento con una distancia de 0.7m.

Como se ha realizado anteriormente en el apartado "5.1 Reproducción emulada y recogida de datos", se procederá a mostrar una tabla con todos los datos de forma ordenada, en la Tabla 2, junto con las gráficas más relevantes de este estudio.

Tabla 2 - Datos del rendimiento del entorno real.

		ping (ms)				iperf (Mbps)		
Distancia (m)	dBm	min	max	promedio	perdidos	descarga	subida	latencia
0,7	-37	1	35	6	0%	51,2	57,7	37ms
3,5	-51	15	49	18	0%	43,7	52,6	36ms
8,8	-59	14	50	17	0%	35,6	45,4	36ms
12,4	-62	16	51	25	20%	28,7	25,6	33ms
22,1	-67	16	54	28	3%	15,7	20,1	38ms
28	-69	16	116	32	3%	7,2	12,4	56ms
35,1	-71	15	434	48	13%	1,6	0,8	70ms

Mediante la siguiente gráfica, se puede estudiar la relación que existe entre la distancia y la potencia de la señal real recibida ya que a medida que aumenta la distancia, la potencia de la señal recibida por parte del punto de acceso va decreciendo. Por lo tanto, si se tienen distancias pequeñas, el valor obtenido es más próximo a 0 y, por el contrario, cuando se aleja, se obtiene un valor muchísimo menor, lejano del 0, como se puede observar en la Figura 55:

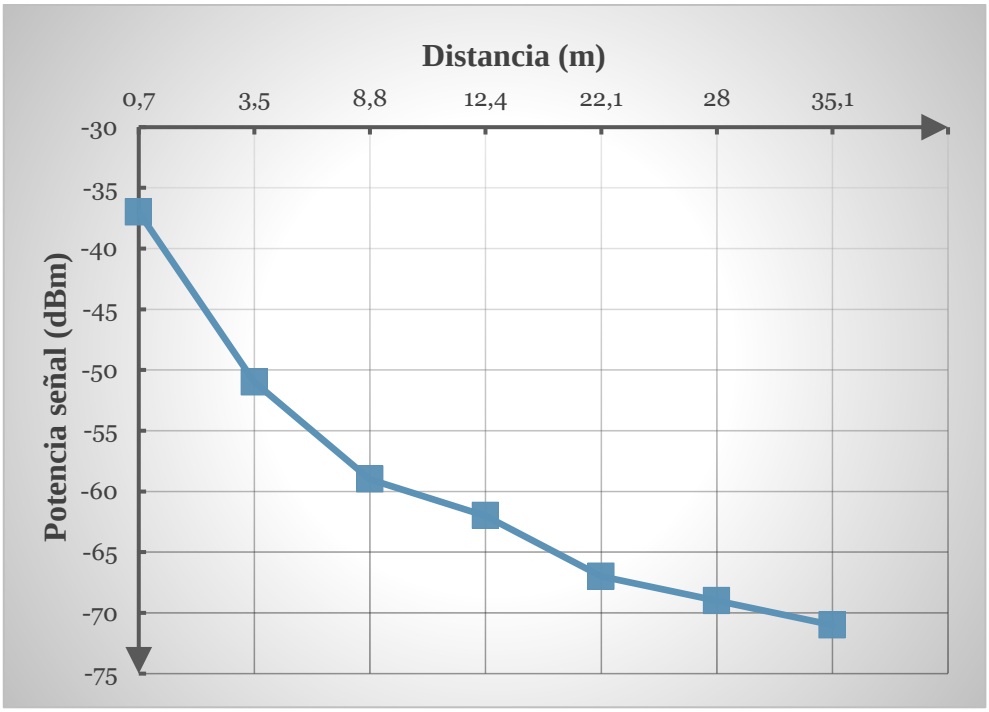

Figura 55 - Gráfica de la potencia de la señal recibida frente a la distancia.

Más allá, si se analizan todos los valores mínimos, máximos y promedios obtenidos mediante *ping*, se obtiene el comportamiento descrito por la Figura 56, en el que se observa que a medida que aumentamos la distancia, el tiempo de respuesta aumenta considerablemente.

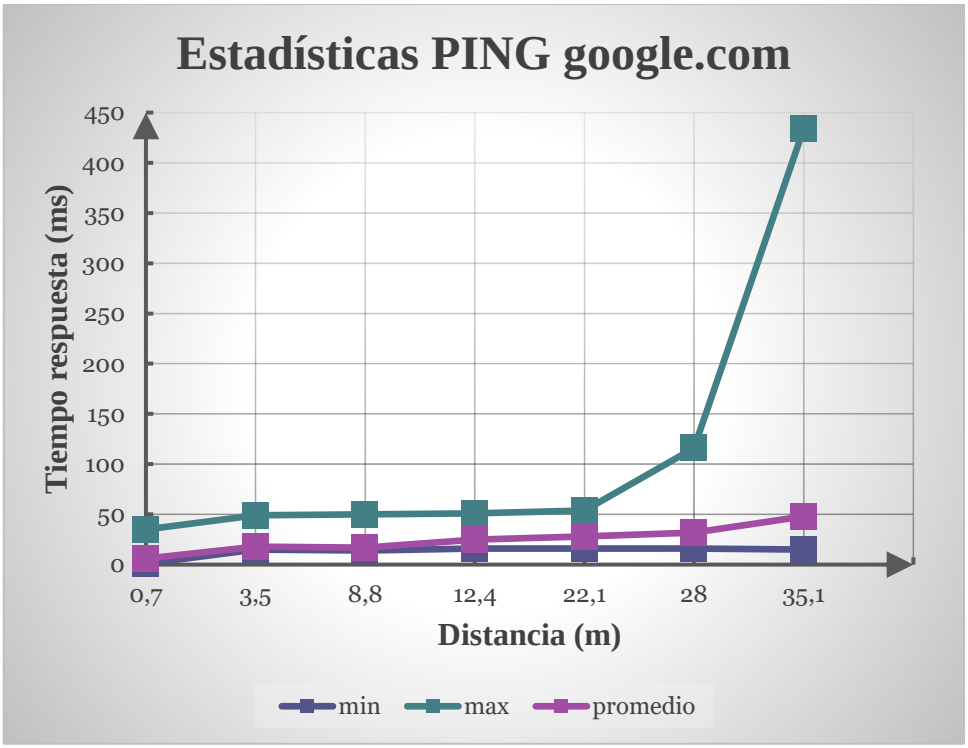

Figura 56 - Gráfica estadísticas ping en el entorno real.

5.3 Resultados del análisis comparativo

A continuación, se procederá a realizar la evaluación a partir de los rendimientos obtenidos. Si se comparan los resultados obtenidos mediante *ping*, se puede apreciar como individualmente, ambos siguen un comportamiento lógico en la Figura 48 y la Figura 56, debido a que los retardos van aumentando a medida que aumentamos la distancia, perjudicando así el rendimiento y el tiempo de respuesta. Pero, si se recogen ambos rangos de valores en una misma gráfica, como se muestra en la Figura 57, se puede apreciar como existe demasiada distancia entre los valores ya que, para las mismas distancias, los números distan demasiado entre ellos, en otras palabras, las dos rectas deberían de mostrarse más próximas entre ellas.

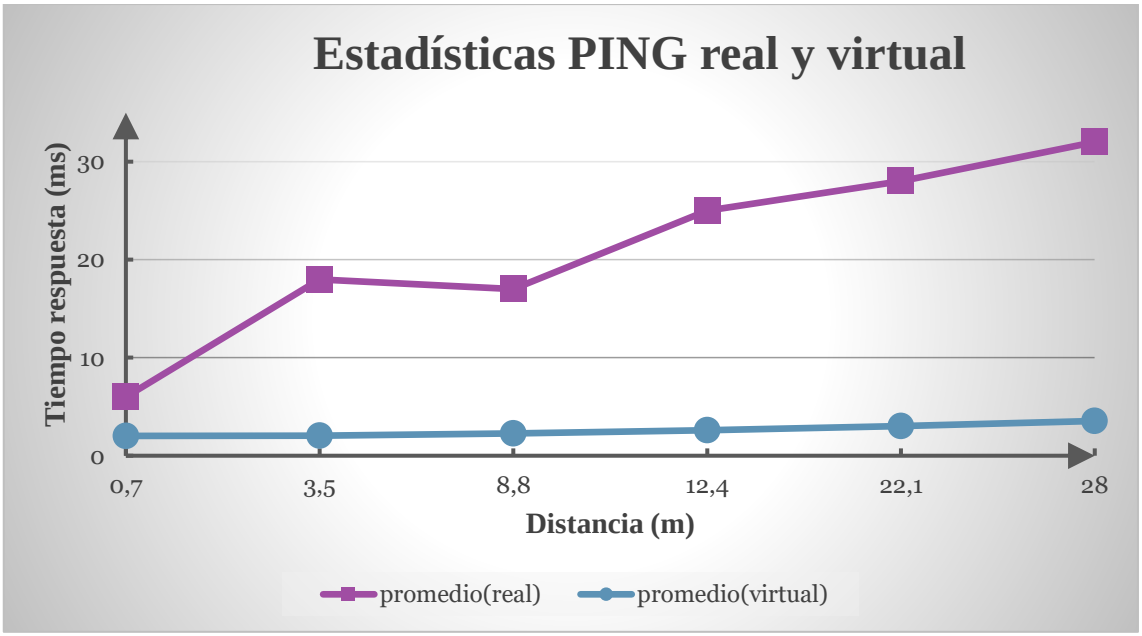

Figura 57 - Gráfica estadísticas ping en el entorno real y virtual.

Por otro lado, si se comparan los datos recogidos por la herramienta *iperf*, se llega a la misma conclusión que en la evaluación de la herramienta *ping* ya que individualmente, tanto el real como el virtual siguen un mismo patrón, al aumentar la distancia el rendimiento se hace próximo a 0, pero si se recogen ambos en la misma gráfica, ambas rectas distan demasiado, es decir, en cuanto a valores numéricos, se alejan demasiado de la realidad, como se observa en la Figura 58:

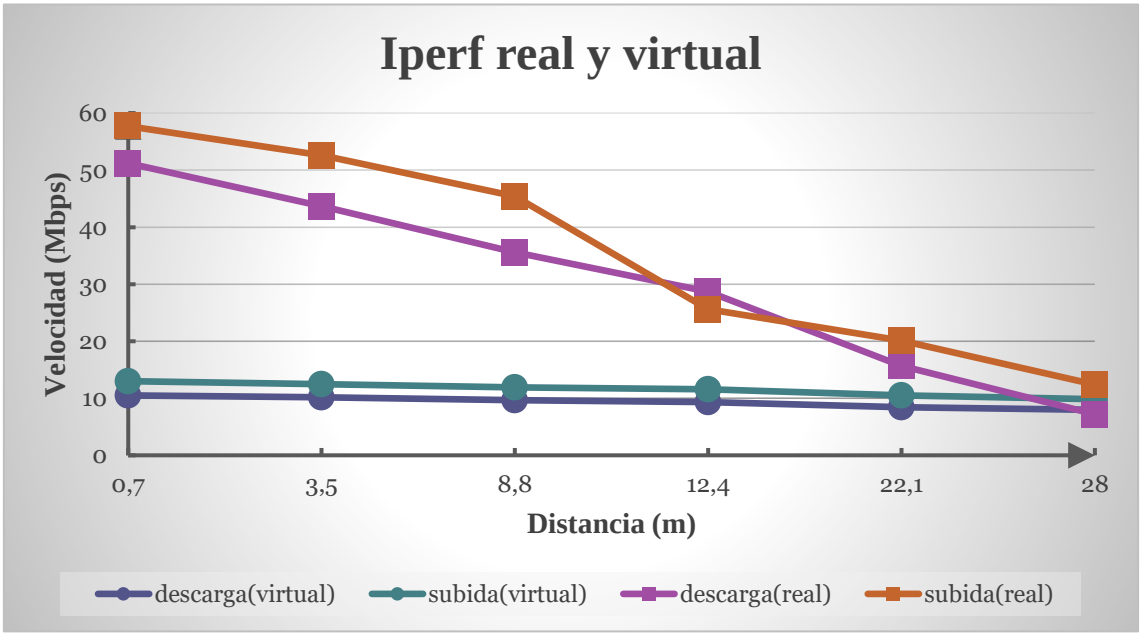

Figura 58 - Gráfica iperf en el entorno real y virtual.

En definitiva, a pesar de que ambos individualmente siguen un comportamiento semejante, ascendiendo con la prueba del *ping* y decreciendo en la prueba de *iperf*, al compararlos, los valores distan demasiado y, en consecuencia, la simulación de redes vista desde la perspectiva numérica no se acerca a la realidad, obteniendo un comportamiento no deseado.

6. Mejorando las tablas de flujo de Mininet-WiFi

Mininet-WiFi está en constante desarrollo, razón por la cuál es el software más popular entre los usuarios que utilizan SDN. Con el fin de poder realizar más pruebas para evaluar el software, se ha realizado una verificación del funcionamiento de las tablas de flujo, intrínsecamente relacionadas con el controlador, para entender mejor cómo funcionan. Inesperadamente se ha hallado una mejora sustancialmente importante que acerca más a la realidad este tipo de simuladores. Para ello, se procederá a crear una red con topología lineal mediante el comando mostrado en la Figura 59, observando la siguiente salida por la terminal.

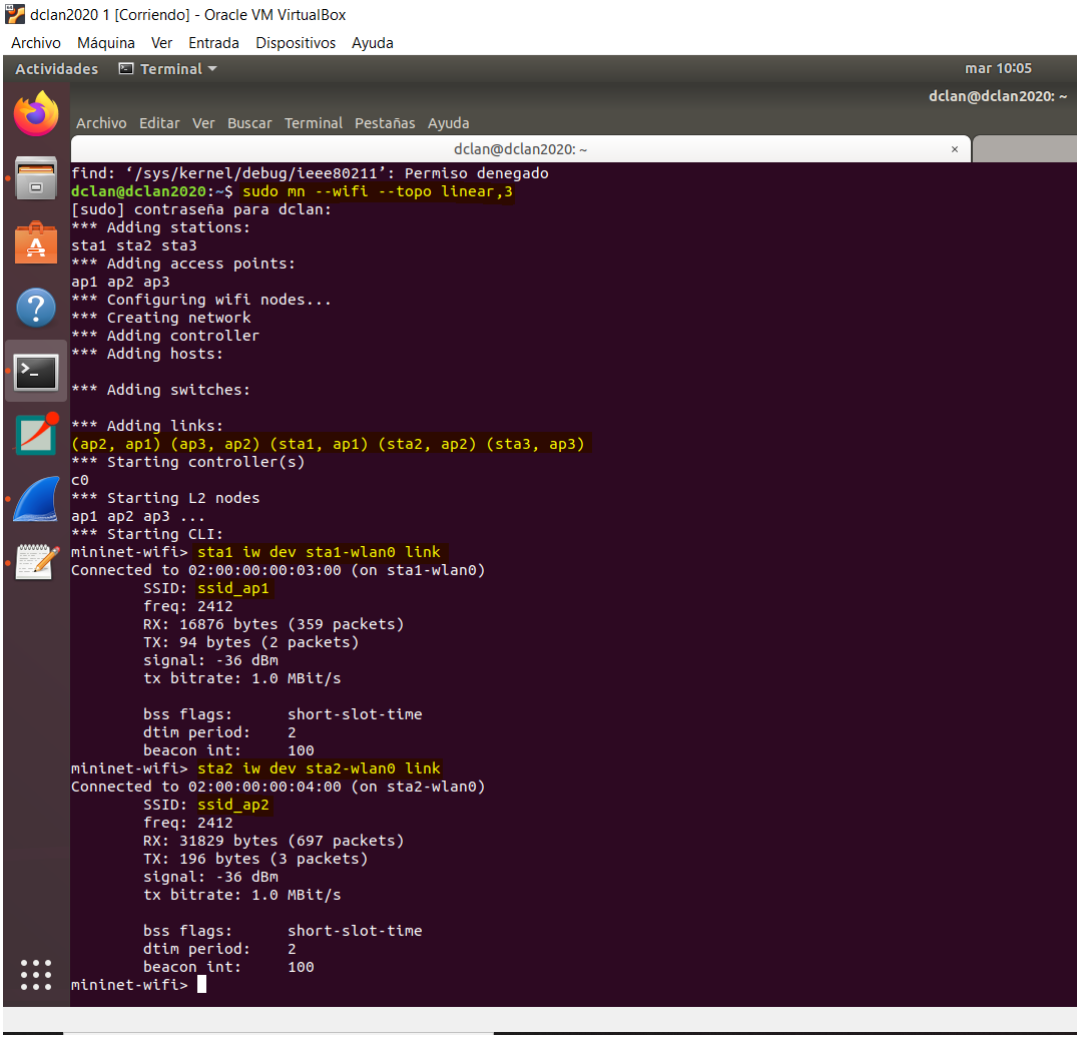

Figura 59 - Creación de una red con topología lineal.

Como se puede observar en la Figura 59, se tienen tres puntos de acceso conectados linealmente mediante ethernet —de forma que "ap1" se conecta con "ap2" y, a su vez, "ap2" con "ap3" formando una conexión lineal en la que para comunicarse "ap1" y "ap3",

es obligado pasar por “ap2”— y tres estaciones, cada una de ellas conectada a uno de ellos. Concretamente, la estación “sta1” está asociada con “ap1” de ssid “ssid_ap1”, la estación “sta2” está asociada con “ap2” cuyo ssid es “ssid_app2” y, finalmente, la estación “sta3” está asociada de igual forma al punto de acceso “ap3”.

Si, por ejemplo, se desautentica “sta1” y verificamos sus conexiones mediante los siguientes comandos, observamos como realmente ha tenido éxito en la Figura 60:

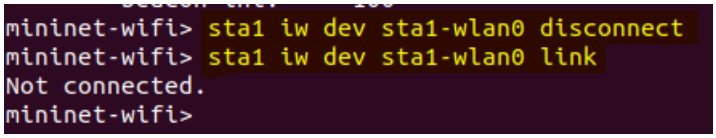

Figura 60 - Desautenticación de "sta1" de "ap1".

Más allá, en la Figura 61, se monitoriza con *Wireshark* lo que sucede en realidad, observándose así el mensaje *Deauthentication* generado mediante una SDN. Para poder monitorizar con tanto detalle los paquetes que circulan por la red, se deberá de activar manualmente la interfaz *hwsim0* para que *Wireshark* la detecte y pueda acceder al tráfico, como se aprecia.

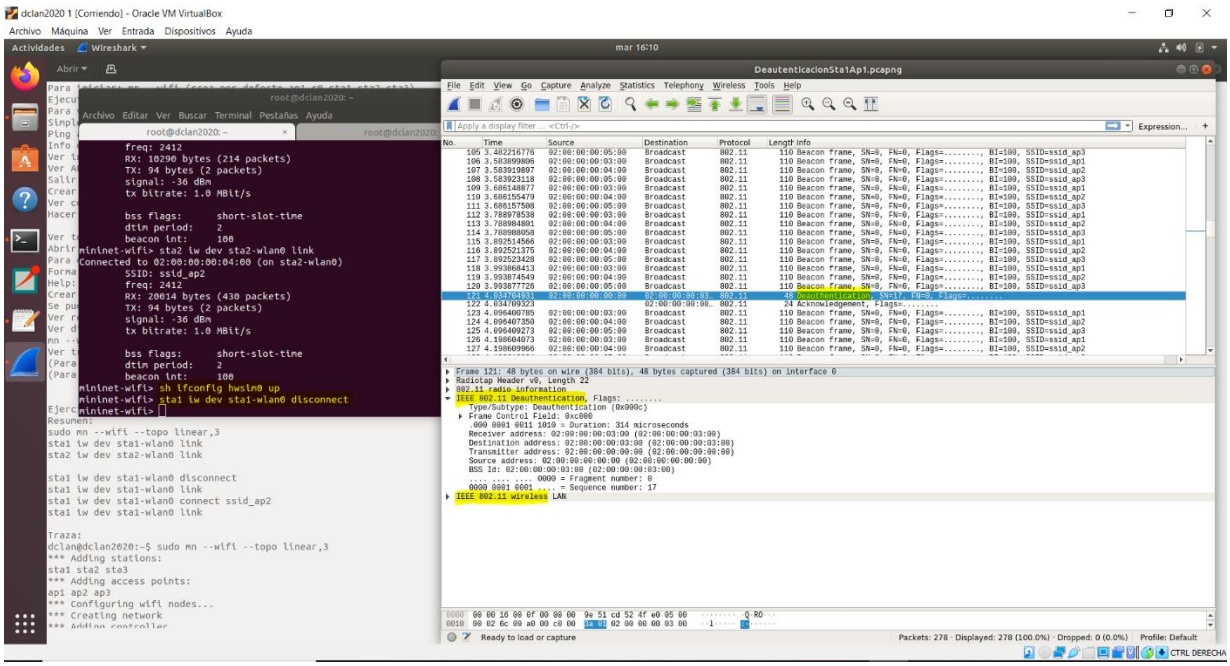

Figura 61 - Activación de la interfaz hwsim0 y mensaje Authentication.

Si, a continuación, en la Figura 62 , se reasocia “sta1” con otro punto de acceso como “ap2”:

```
mininet-wifi> sta1 iw dev sta1-wlan0 connect ssid_ap2
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:04:00 (on sta1-wlan0)
        SSID: ssid_ap2
        freq: 2412
        RX: 7307 bytes (168 packets)
        TX: 94 bytes (2 packets)
        signal: -36 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
mininet-wifi>
```

Figura 62 - Reasociación de "sta1" con "ap2".

También se puede ver el flujo de tramas, concretamente los mensajes “probe request/response” en la Figura 63:

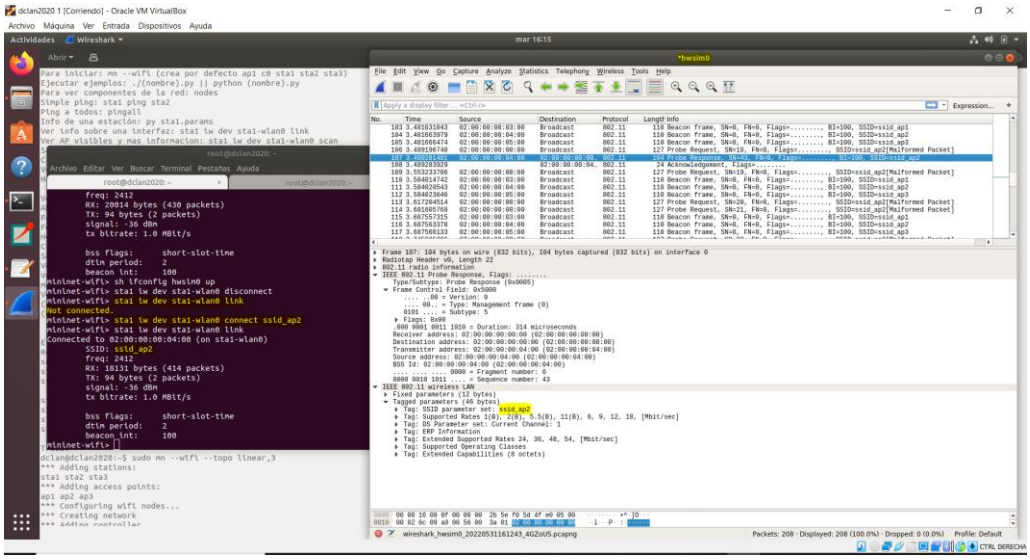

Figura 63 - Captura del tráfico mediante la interfaz hwsim0.

Para poder verificar el funcionamiento de las tablas de flujo, se puede ver el tráfico que se genera en los AP si, por ejemplo, se realiza un *ping* entre estaciones. De momento no se ha realizado ninguno, por ello las tablas donde se almacena esta información están vacías. Estas tablas se pueden consultar con el comando de la Figura 64:

```
mininet-wifi> dpctl dump-flows
*** ap1 ------------------------------------------------------------------------
*** ap2 ------------------------------------------------------------------------
*** ap3 ------------------------------------------------------------------------
```

Figura 64 - Consulta de las tablas de flujo de Mininet-WiFi.

A continuación, se realizará un *ping* desde “sta3” hacia “sta1” mediante una terminal ejecutada en la propia estación. Para poder abrir una terminal en una estación, se ejecutará el comando de la Figura 65:

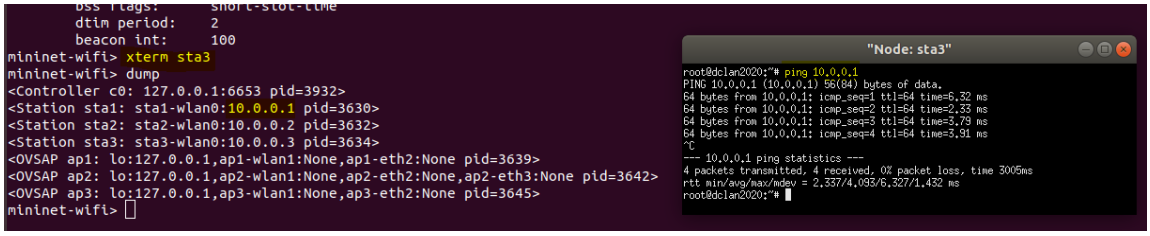

Figura 65 - Terminal en la estación virtual "sta3" y realizar un ping a “sta1”.

Si se vuelve a observar las tablas de flujo con el mismo comando mostrado en la Figura 64, se tendrá el siguiente contenido mostrado en la Figura 66:

Figura 66 - Información de las tablas de flujo después del ping.

La razón por la cual no ha pasado nada de tráfico vía “ap1” es porque la estación “sta1” está asociada con “ap2” y “ap2” con “ap3” vía ethernet. Y como “sta3” está asociada con “ap3”, no pasa la información por “ap1”.

Más allá, es posible observar el tráfico a dos niveles[22] ya que está activo el protocolo *OpenFlow*, visible desde la interfaz de *loopback* de la máquina anfitriona.

[22] Debido a que anteriormente en la Figura 63 se observó el tráfico a nivel virtual como si se tratase de la realidad, en cambio en la Figura 67 se aprecia la existencia del protocolo *OpenFlow*, responsable de la virtualización desde una perspectiva más exterior.

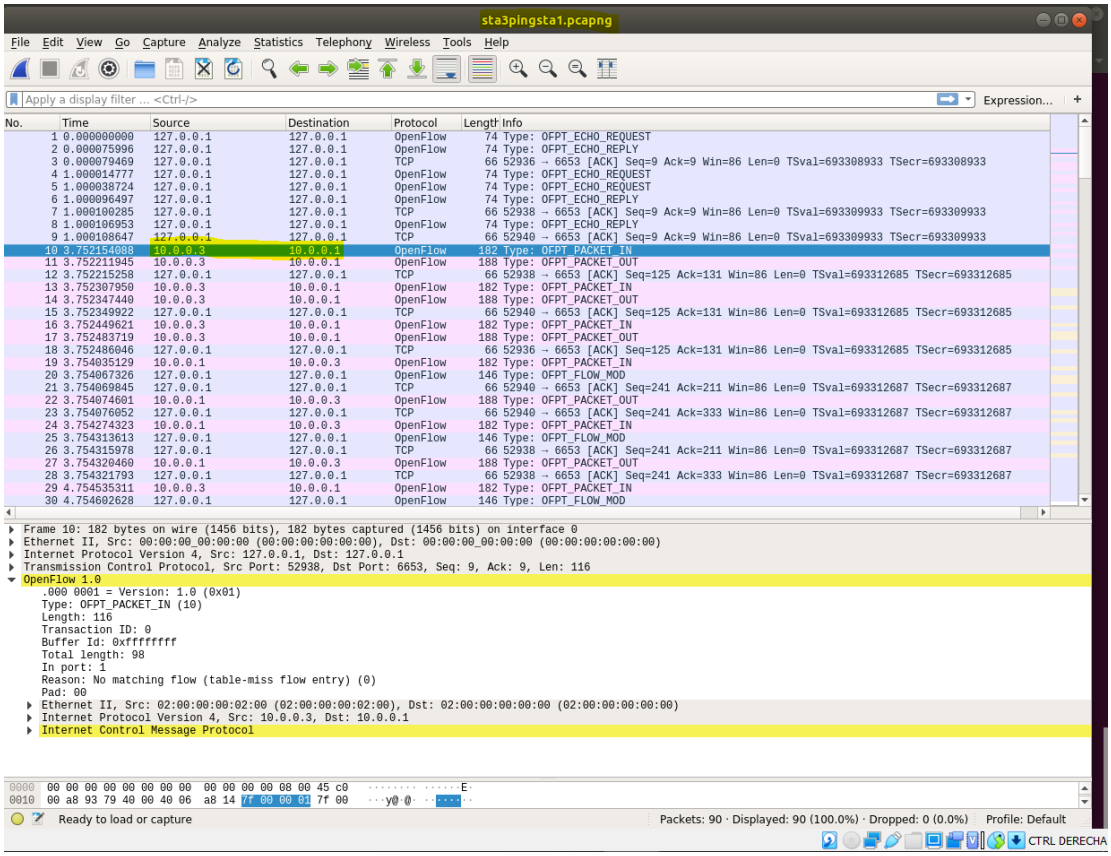

Figura 67 - Captura del tráfico mediante la interfaz loopback.

Y, al mismo tiempo, se pueden ver las tramas *ICMP* desde la interfaz *hwsim0* como si se tratara de un entorno real en la Figura 68:

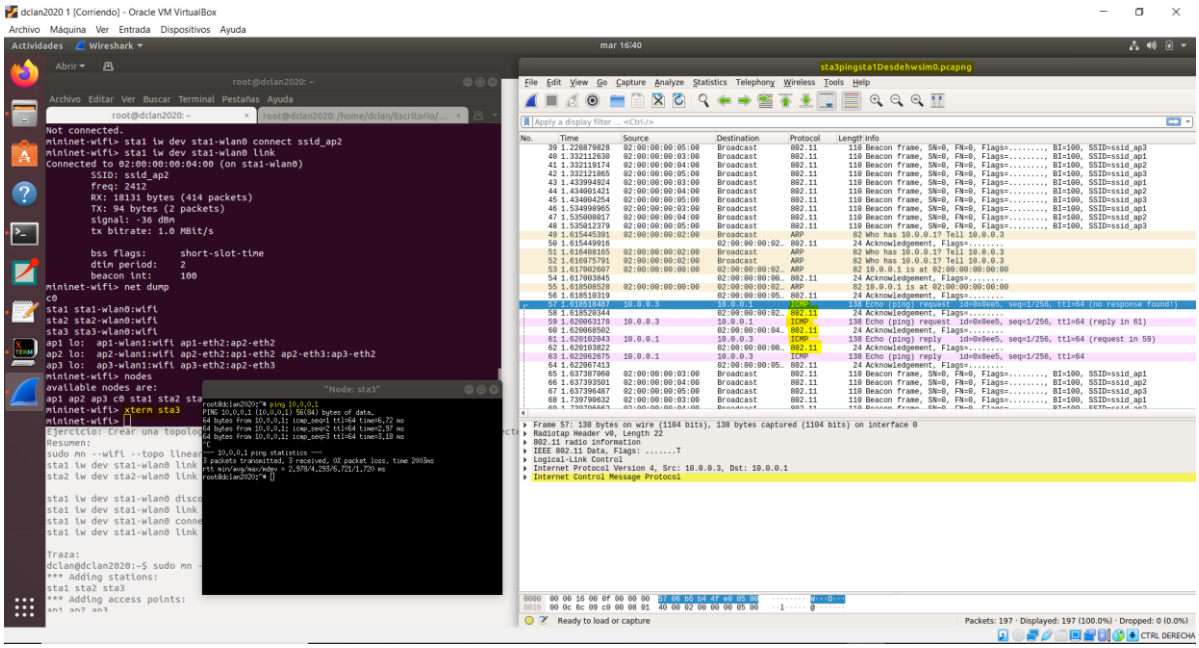

Figura 68 - Captura de paquetes ICMP mediante la interfaz hwsim0.

A continuación, se demostrará la constante evolución de esta tecnología ya que, a pesar de haberse desarrollado en la última década, sigue mejorando año tras año. Para ello, se volverá a configurar la estructura de la red como estaba originalmente, es decir, con “sta1” asociada a “ap1”, como se muestra en la Figura 69:

```
mininet-wifi> sta1 iw dev sta1-wlan0 disconnect
mininet-wifi> sta1 iw dev sta1-wlan0 link
Not connected.
mininet-wifi> sta1 iw dev sta1-wlan0 link
Not connected.
mininet-wifi> sta1 iw dev sta1-wlan0 connect ssid_ap1
mininet-wifi> sta1 iw dev sta1-wlan0 link
Connected to 02:00:00:00:03:00 (on sta1-wlan0)
        SSID: ssid_ap1
        freq: 2412
        RX: 4139 bytes (96 packets)
        TX: 94 bytes (2 packets)
        signal: -36 dBm
        tx bitrate: 1.0 MBit/s

        bss flags:      short-slot-time
        dtim period:    2
        beacon int:     100
```

Figura 69 - Reasociación de "sta1" con "ap1".

Si se realiza un *ping* como de la misma forma que la Figura 65, se obtendrá en la tabla de flujos que sí ha pasado información por “ap1” y que, aunque parezca obvio, ha tenido éxito el *ping*, como se muestra en la Figura 70:

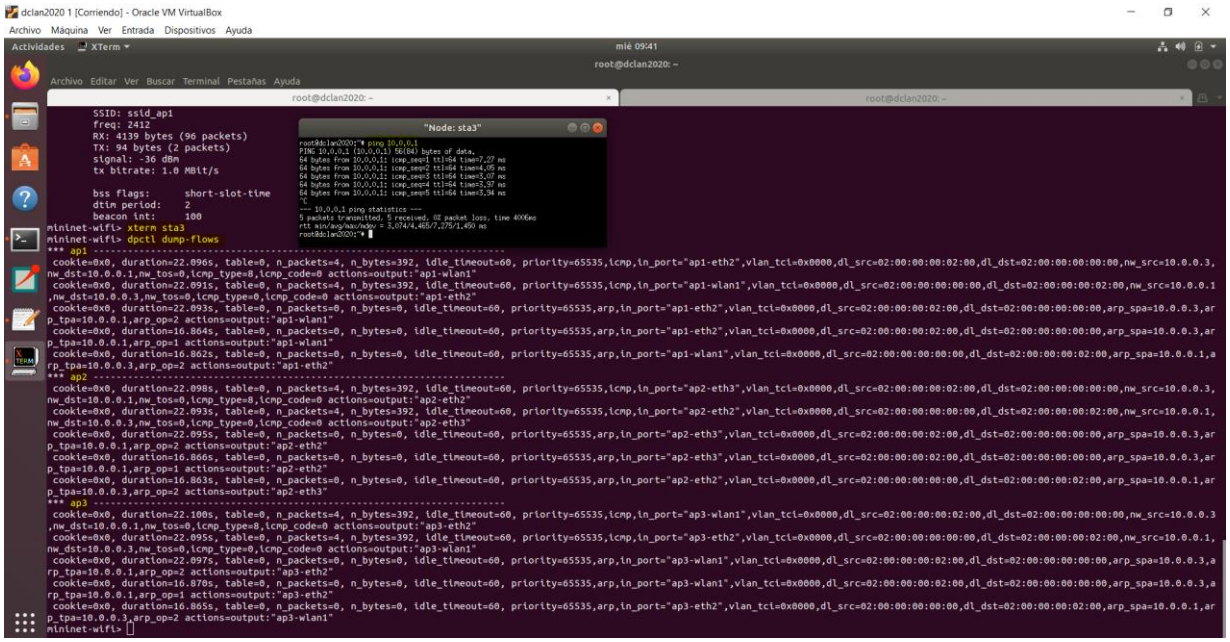

Figura 70 - Ping con éxito después de haber cambiado la topología de la red.

Hay que destacar que los flujos de *OpenFlow* en un escenario que no es estático, sino de movilidad debido a las asociaciones de las estaciones, están implementados mediante software, y éste puede tener límites en cuanto a asemejarse a la realidad. En el año 2019 (21), esta misma prueba no funcionaba debido a que hay muchos estudios que remarcaban que los flujos de *OpenFlow* no estaban tan avanzados; por ello, una vez se cambiaba la estación “sta1”, y se había realizado el primer *ping* de la Figura 65, la tabla de flujos se

quedaba estática, a pesar de que la "sta1" volviera a asociarse con su AP original. Es decir, al volver a asociarse con "ap1", el segundo *ping* de la Figura 70 fallaba debido a que "ap2" y "ap3" ya tenían definidos los flujos para los mensajes ICMP procedentes de "sta3", por lo que seguían enviando paquetes hacia la interfaz "ap2-wlan0" para llegar a dónde creían que se alojaba "sta1" —manteniendo de forma estática el contenido de la Figura 66—. En conclusión, no solicitaba ninguna actualización del flujo del controlador. La única forma de solucionar el problema era borrando "a mano" la tabla mediante la orden "*dpctl del-flows*", aspecto que dejaba mucho que desear y que afectaba negativamente a esta tecnología. Actualmente, como se ha demostrado, se ha mejorado y las tablas de flujos se actualizan automáticamente, viendo que "sta1" ha cambiado de posición y redirigiendo el flujo correctamente. Cabe remarcar que es una tecnología muy novedosa y en constante mejora que nos puede ayudar a mejorar, incluso el medio ambiente, debido a que el objetivo de cualquier simulación es predecir qué problemas alberga el sistema en estudio con el objetivo de que, cuando se reproduzca en la realidad, surjan los mínimos problemas, pudiendo así aprovechar al máximo todo el hardware adquirido por una empresa, evitando la compra de dispositivos innecesarios, evitando emisiones de dióxido de carbono, etc.

7. Conclusiones

Se ha cumplido con éxito el objetivo secundario “Definir un escenario de pruebas representativo” en el que se ha justificado qué modelos se adaptaban mejor a la realidad en el simulador y se ha creado un script para poder evaluar su funcionamiento, capturando con la herramienta *Wireshark* el tráfico generado en él y en la realidad para poder tener diferentes perspectivas y ver el grado de corrección de la herramienta Mininet-WiFi, cumpliendo además el objetivo secundario “Capturar el tráfico generado en la realidad y en la emulación mediante la herramienta *Wireshark*”.

En consecuencia, se han conseguido los objetivos primarios “Evaluar de forma empírica el emulador Mininet-WiFi justificando su grado de semejanza con la realidad” y “Validar el funcionamiento de este emulador en base a un escenario emulado semejante a un escenario real” debido a que se han recogido muestras en la realidad y la simulación para evaluar su rendimiento, satisfaciendo el objetivo secundario “Medir los rendimientos alcanzados en la realidad y en la emulación y estudiar posibles diferencias”.

En definitiva, se han explorado los límites de la herramienta y en base a las pruebas realizadas en este trabajo se puede afirmar que la herramienta Mininet-WiFi simula un comportamiento próximo a la realidad, pero procede mejorar el modelo del simulador para ajustarlo mejor a datos reales. Por un lado, se han encontrado ventajas ya que las tablas de flujo han mejorado y permiten actualizarse de manera dinámica, sin necesidad de borrarlas manualmente, acercándose más a un comportamiento real ya que en el año 2019 era la única forma posible de utilizar la herramienta. Sin embargo, por otro lado, se han descubierto cinco desventajas: cuando se ha deseado comparar la potencia de la señal recibida en dBm, el entorno virtual estaba limitado debido a que los valores no se actualizaban dinámicamente y devolvían únicamente un valor a pesar de aumentar la distancia; la interfaz gráfica que ofrece Mininet-WiFi cuando se ejecutan los scripts no permite el uso de *zoom* y los botones situados debajo para mejorar la experiencia del usuario no funcionan muy bien; al ejecutar la simulación, no se ha descubierto ningún método para pausar la simulación con opción de reanudarla, dejando como única opción la detención de la red sin opción de continuar la ejecución; se ha podido observar como la generación del tráfico a veces falla, ocasionando la malformación de algún paquete con la herramienta *Wireshark*; y por último, los valores numéricos en los rendimientos obtenidos distaban demasiado.

En conclusión, a pesar de que el software se está mejorando anualmente, la existencia de las cinco desventajas nombradas hace que el estudio concluya con la sentencia de que el emulador no tiene un comportamiento satisfactorio.

8. Trabajos futuros

En un futuro convendría investigar posibles mejoras en cuanto a las desventajas encontradas en este trabajo mediante la creación de diferentes escenarios de emulación. Además, se podrían crear escenarios más complejos que requieran el uso de más dispositivos virtuales, con rangos preestablecidos, y sería interesante ver si el movimiento dinámico entre ellos, una vez ejecutado el escenario, causa problemas, ya sea con la generación de tramas virtuales, o mediante la simulación de estadísticas ya que, cuantos más dispositivos se estén ejecutando a la vez, mayor carga tendrá el controlador, y las tablas de flujos podrían verse afectadas, además del propio software en general. Por tal razón, explorar el límite máximo que el emulador funciona sin perder la semejanza con la realidad podría ser una buena vía de investigación, debido a que este software está pensado para incluso emular ciudades inteligentes y, en consecuencia, convivirían a la vez un gran número de dispositivos.

Referencias bibliográficas y citas

1. **D. Larrabeiti, M. Calderón, A. Azcorra, A. García, J. E. Kristensen.** Redes activas con IPv6. *it.uc3m.es.* [En línea] Universidad Carlos III de Madrid, Enero de 2001. [Citado el: 8 de Agosto de 2022.] https://www.it.uc3m.es/maria/papers/redact_com_world01.pdf.

2. **FEAMSTER, Nick. REXFORD, Jennifer. ZEGURA, Ellen.** *The Road to SDN: An Intellectual History of Programmable Networks.* [En línea] cs.princeton.edu, 2013. [Citado el: 5 de Marzo de 2022.] https://www.cs.princeton.edu/courses/archive/fall13/cos597E/papers/sdnhistory.pdf.

3. **OPEN NETWORKING FOUNDATION.** *SDN architecture.* [En línea] youtube.com, 2014. [Citado el: 8 de Febrero de 2022.] https://opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf.

4. **CISCO.** *Demystifying SDN for the Network Engineer.* [En línea] mkto.cisco.com, 2017. [Citado el: 6 de Mayo de 2022.] https://mkto.cisco.com/rs/564-WHV-323/images/DemystifyingSDN_Whitepaper.pdf.

5. **CRAVEN, Connor.** *What is Software Defined Networking (SDN)? Definition.* [En línea] sdxcentral.com, 2020. [Citado el: 5 de Junio de 2022.] https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/.

6. **COKER, Oswald. AZODOLMOLKY, Siamak.** *Software-Defined Networking with Openflow.* 2nd edition. Livery Place : Packt, 2017. págs. 246. ISBN 978-1783984282.

7. **SERRANO CARRERA, David Andrés.** *Redes Definidas por Software (SDN): OpenFlow.* [En línea] riunet.upv.es, 2015. [Citado el: 15 de Marzo de 2022.] https://riunet.upv.es/bitstream/handle/10251/62801/SERRANO%20-%20Redes%20Definidas%20por%20Software%20(SDN):%20OpenFlow.pdf.

8. **FONTES, Ramon.** *Mobility Models.* [En línea] mininet-wifi.github.io, 2022. [Citado el: 6 de Noviembre de 2021.] https://mininet-wifi.github.io/mobility/.

9. **FONTES, Ramon. ESTEVE ROTHENBERG, Christian.** *The User Manual of Mininet-WiFi.* [En línea] usermanual.wiki, 2018. [Citado el: 10 de Octubre de 2021.] https://usermanual.wiki/Pdf/mininetwifidraftmanual.297704656.pdf.

10. **FONTES, Ramon.** *Propagation Models.* [En línea] mininet-wifi.github.io, 2022. [Citado el: 5 de Diciembre de 2021.] https://mininet-wifi.github.io/propagation/.

11. **MATHURANATHAN.** *Friis Free Space Propagation Model.* [En línea] gaussianwaves.com, 2013. [Citado el: 10 de Enero de 2022.] https://www.gaussianwaves.com/2013/09/friss-free-space-propagation-model/.

12. **SNIDER, Dave.** *Introduction.* [En línea] mn-wifi.readthedocs.io, 2022. [Citado el: 15 de Abril de 2022.] https://mn-wifi.readthedocs.io/en/latest/intro.html.

13. **MALINEN, Jouni.** *mac80211_hwsim - software simulator of 802.11 radio(s) for mac80211.* [En línea] kernel.org, 2008. [Citado el: 25 de Abril de 2022.] https://www.kernel.org/doc/readme/Documentation-networking-mac80211_hwsim-README.

14. **TORVALDS, Linus.** *mac80211_hwsim.* [En línea] github.com, 2008. [Citado el: 28 de Abril de 2022.] https://github.com/torvalds/linux/blob/master/drivers/net/wireless/mac80211_hwsim.c.

15. **SNIDER, Dave.** *Propagation Models.* [En línea] mn-wifi.readthedocs.io, 2022. [Citado el: 30 de Abril de 2022.] https://mn-wifi.readthedocs.io/en/latest/propagationmodels.html.

16. **GAST, Matthew S.** *802.11 Wireless Networks: The Definitive Guide.* 2nd edition. California : O'Reilly, 2005. págs. 670. ISBN 978-0596100520.

17. **TP-Link.** *TP-Link TL-WN722N - Adaptador Wi-Fi USB inalámbrico.* [En línea] amazon.es. [Citado el: 25 de Septiembre de 2021.] https://www.amazon.es/TP-LINK-TL-WN722NV2-Adaptador-inal%C3%A1mbrico-10-6-10-11/dp/B002SZEOLG?th=1.

18. **ANGULO AGUIRRE, Luis.** *Hacking & Cracking Redes inalámbricas wifi.* Lima : MACRO, 2019. págs. 254. ISBN 978-84-267-2695-7.

19. **GAST, Matthew S.** Chapter 4 Framing in Detail. *802.11 Wireless Networks: The Definitive Guide.* [En línea] O'Reilly, 2005. [Citado el: 6 de Octubre de 2021.] Disponible desde los servicios de la UPV en la biblioteca virtual de O'Reilly.

20. **RATAN, Abhishek. CHOU, Eric. KATHIRAVELU, Pradeeban. FARUQUE SARKER, Dr. M. O.** *Python Network Programming.* Livery Place : Packt, 2019. págs. 776. ISBN 978-1-78883-546-6.

21. **MANZONI, Pietro.** *Mininet Lab 3: WiFi networks.* [En línea] hackmd.io, 2019. [Citado el: 7 de Mayo de 2022.] https://hackmd.io/@pmanzoni/rkMF11rIH.

22. **STAFANICK, George.** *802.11 - TIM and DTIM Information Elements.* [En línea] blogs.arubanetworks.com, 2016. [Citado el: 4 de Mayo de 2022.] https://blogs.arubanetworks.com/industries/802-11-tim-and-dtim-information-elements/.

23. **ARUBA HEWLETT PACKARD ENTERPRISE COMPANY.** *Country Codes List.* [En línea] arubanetworks.com, 2022. [Citado el: 9 de Mayo de 2022.] https://www.arubanetworks.com/techdocs/InstantWenger_Mobile/Advanced/Content/Instant%20User%20Guide%20-%20volumes/Country_Codes_List.htm.

24. **dot11zen.** *Exploring DTPC and 802.11h Transmit Power Control.* [En línea] dot11zen.blogspot.com, 2017. [Citado el: 13 de Mayo de 2022.] https://dot11zen.blogspot.com/2017/05/exploring-dtpc-and-80211h-transmit.html.

25. **LAKSHMANAN, Ravie.** *Lumos system can find hidden cameras and IoT Devices in your Airbnb or hotel room.* [En línea] thehackernews.com, 2022. [Citado el: 15 de Mayo de 2022.] https://thehackernews.com/2022/05/lumos-system-can-find-hidden-cameras.html.

26. **GAST, Matthew S.** Chapter 8 Management Operations. *802.11 Wireless Networks: The Definitive Guide.* [En línea] O'Reilly, 2005. [Citado el: 15 de Mayo de 2022.] Disponible desde los servicios de la UPV en la biblioteca virtual de O'Reilly.

27. **BOMBAL, David.** *Kali Linux TP-Link TP-WN722N.* [En línea] youtube.com, 2022. [Citado el: 5 de Mayo de 2022.] https://www.youtube.com/watch?v=tYnjMiTTdms.

28. **OPEN SOURCE.** *Aircrack-ng.* [En línea] aircrack-ng, 2022. [Citado el: 12 de Mayo de 2022.] https://www.aircrack-ng.org/doku.php?id=airmon-ng.

29. **RAMOS VARÓN, Antonio. BARBERO MUÑOZ, Carlos A. FERNÁNDEZ HANSEN, Yago. DASWANI DASWANI, Deepak. MARUGÁN RODRÍGUEZ, David.** *Hacking práctico de redes wifi y radiofrecuencia.* Madrid : Ra-Ma, 2014. págs. 272. ISBN 978-84-9964-296-3.

Anexos

A continuación, se muestran los anexos de este trabajo:

Órdenes esenciales para el uso de Mininet-WiFi

Algunas órdenes sencillas para utilizar este software son las que se muestran en la Figura 71:

```
Para iniciar: mn --wifi (crea por defecto ap1 c0 sta1 sta2 sta3)
Ejecutar ejemplos: ./(nombre).py || python (nombre).py
Para ver componentes de la red: nodes
Ver procesos creados por mininet asociados a cada estacion: py locals()
Simple ping: sta1 ping sta2
Ping a todos: pingall
Info de una estación: py sta1.params
Ver info sobre una interfaz: sta1 iw dev sta1-wlan0 link
Ver AP visibles y mas informacion: sta1 iw dev sta1-wlan0 scan
Salir: exit y después mn -c
Crear topologia lineal: mn --wifi --topo linear,3
Ver conexiones: net
Hacer ifconfig desde sta1: sta1 ifconfig
                           sta1 iwconfig
Ver todas las IPs: dump
Ver interfaces de red activas: net dump
Abrir terminal de una estación: xterm sta1
Para comandos sin terminal extra, se nombra disp + orden: sta1 ls
Forma Gráfica: (dentro de la carp /home/mininet-wifi/examples) ./miniedit.py
Help: mn -h
Crear 4 hosts y 1 switch: mn --topo=single,4
Se puede ejecutar Wireshark dentro de cada maq: sta1 wireshark
Ver rendimientos: iperf sta1 sta2
Ver distancias: distance sta1 sta2
mn --wifi --ssid=prueba --mode=g --channel=1

Analisis trafico simulado:
Ver tráfico que pasa por los AP: dpctl dump-flows
(Para observar el tráfico de OpenFlow en wireshark, monitorizar la interfaz "lo")
(Para cabeceras 802.11): sh ifconfig hwsim0 up (desde mininet-wifi>)
```

Figura 71 - Órdenes esenciales Mininet-WiFi.

Explicación de los campos de una trama 802.11

Como se analizó en el apartado "4.1 Monitorización de tramas 802.11 en un entorno real", existe gran cantidad de información que viaja dentro la trama *beacon*, por ello, en este anexo se detallarán los campos más relevantes junto con sus posibles interpretaciones.

En primer lugar, el campo *flags* de la Figura 18 ofrece la siguiente información:

DS Status: identifica si la trama está entrando o saliendo del entorno inalámbrico, además de mostrar si forma parte de una red ad-hoc o parte de un sistema de distribución inalámbrica. Es decir, mediante estos bits es posible averiguar incluso si el paquete ha sido transmitido hacia/desde la estación, como se muestra en la Tabla 3:

Tabla 3 - Posibles valores para los campos DS Status y las cuatro direcciones IP.

Escenario	To DS	From DS	Address 1	Address 2	Address 3	Address 4
a)	0	0	DA	SA	BSSID	X
b)	0	1	DA	BSSID	SA	X
c)	1	0	BSSID	SA	DA	X
d)	1	1	RA	TA	DA	SA

La nomenclatura seguida es la siguiente: *DA* y *SA*, que denotan la direcciones de destino y origen —del inglés *Destination Address* y *Source Address* —; *RA* y *TA*, que se relaciona con las direcciones destino y origen entre puntos de acceso —del inglés *Recipient Address* y *Transmitter Address*—; *BSSID*, que tiene el valor de la MAC del punto de acceso o un valor aleatorio en una red ad-hoc; y por último, *X*, que no tiene valor, todo a ceros, como se muestra en la Figura 72:

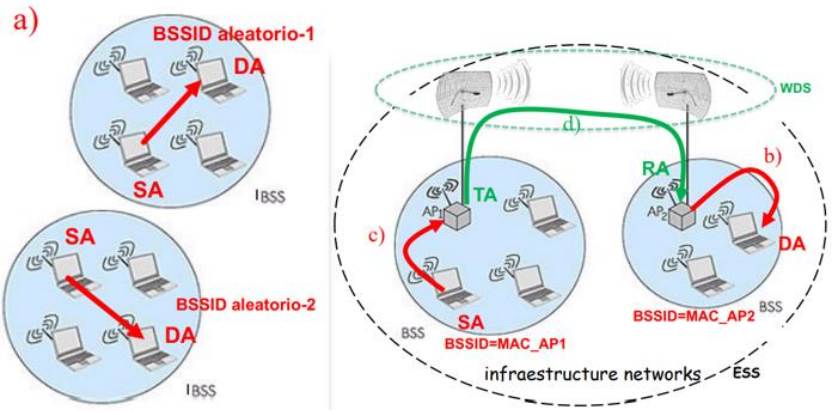

Figura 72 - Esquema asociado a las cuatro direcciones en flags

Así pues, se dispone de los cuatro escenarios posibles en los que se tiene: caso a) red sin AP, se trata de una red ad-hoc; caso b), se estudia la comunicación STA-DS; caso c), se observa la comunicación DS-STA; y finalmente caso d), se analiza cómo se comunicarían entre APs siendo una comunicación DS-DS.

Continuando con el análisis de los *flags*, se tiene:

More Fragments: si tiene activado el bit a "1" significa que el paquete ha sido fragmentado; en caso contrario, tendrá un valor de "0" y significará que es el último fragmento o, que no se ha habido fragmentación.

Retry: si tiene activado el bit a "1" significa que la trama está siendo retransmitida y ayuda a la estación receptora a eliminar las tramas duplicadas; en caso contrario, tendrá un valor de "0" significando que es el primer intento.

PWR MGT: se encarga de administrar la energía, así si tiene el bit activado a "1" significará que la estación transmisora, para ahorrar batería, apagará partes de su interfaz de red tras la finalización del intercambio de la trama; en caso contrario tendrá un "0", no se permitirá ahorrar energía y estará siempre activa debido a que realiza funciones importantes.

More Data: si tiene este bit activado a "1" significa que el AP ha recibido del sistema de distribución tramas destinadas a ser enviadas a una estación y las tiene almacenadas en un buffer disponible para ser enviadas; en caso contrario, tendrá un valor "0" indicando que no tiene más tramas y, en ese caso, le facilita a la estación su ahorro de energía.

Protected Flag: si tiene activado este bit a "1" significa que la trama está protegida con protocolos de seguridad de capa de enlace y la trama cambia ligeramente. Anteriormente, recibía el nombre de bit WEP.

HTC/Order Flag: si tiene activado este bit a "1" significa que la entrega se realiza con "pedido estricto", de forma Strictly-Ordered Service Class.

En segundo lugar, el campo *Duration* de la Figura 19 recibe las siguientes interpretaciones y sus bits se pueden interpretar como se detalla en la Figura 73:

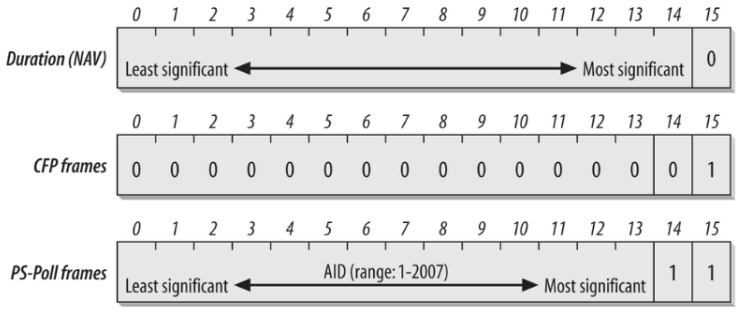

Figura 73 - Posibles interpretaciones del campo Duration.

Si el bit 15 está desactivado, con un valor "0", que sería nuestro caso, el campo se utiliza para establecer el NAV —del inglés *Network Allocation Vector*—, el cual tiene la función de representar el número de microsegundos que se espera que el medio permanezca ocupado para la transmisión actualmente en curso, y todas las estaciones deberán actualizarlo. Cualquier valor que extienda la cantidad de tiempo que el medio está ocupado actualiza el NAV y bloquea el acceso al medio durante un tiempo adicional.

Si, por el contrario, el bit 15 está activo, entonces se tendrá en cuenta en el bit 14. Si está desactivado, entonces el campo se utilizará para establecer NAVs para *frames* transmitidos durante periodos sin contención CFP —del inglés Content-Free Periods— con un valor de "32768". Permite que cualquier estación que no reciba *beacons* pueda actualizar su *NAV* con un valor más grande para evitar interferir con las transmisiones sin contención.

Por último, si tanto el bit 14 como el bit 15 están activos, se tendría bajo estudio una trama *PS-Poll*. La existencia de estas tramas surge de la necesidad garantizar que no se pierde información al haber estado durmiendo durante cierto periodo de tiempo por cuestiones de ahorro de energía. Así, la estación transmitirá una trama de tipo *PS-Poll* para recuperar cualquier trama almacenada en el AP y aportará en este campo de duración su "AID", "ID de asociación", que indicará a qué BSS pertenecía, con valores en el rango 1-2007, como se observa en la Figura 73.

Finalmente, los últimos campos son *Destination Address*, *Source Address* y *BSS Id*, como se ha explicado en los *flags ToDS-FromDS*. En este caso, parten del AP y se envían a toda la red, mediante *broadcast* o difusión.

En tercer lugar, el campo *tagged parameters* de la Figura 20 se encuentra información como el SSID, dentro de la variable *SSID parameter set*; el canal por el que se está transmitiendo, dentro de *DS Parameter set*, como se puede observar en la Figura 74:

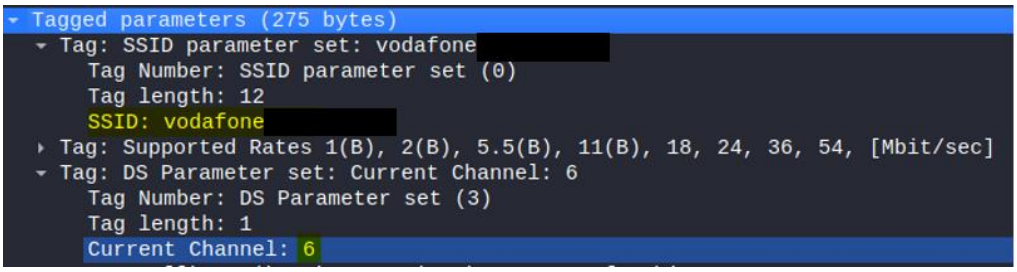

Figura 74 - SSID compartido con el entorno.

Además, se presentan otros datos como "TIM" —del inglés *traffic indication map* que significa mapa de indicación de tráfico—, que es utilizado por las tramas de gestión e indica a las estaciones dormidas que el AP tiene datos en buffer esperando para ser enviados, cuya estructura está definida por la Figura 75:

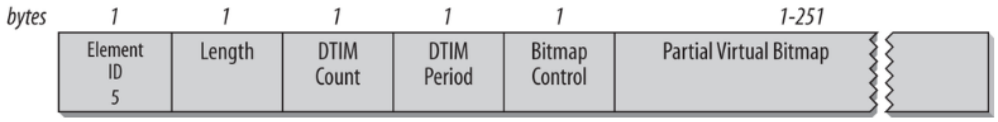

Figura 75 – Estructura del campo TIM.

Tiene cuatro campos principales que son: *DTIM count*, *DTIM period*, *Bitmap control* y *Partial Virtual Bitmap*, como se muestra en la Figura 76:

```
▾ Tag: Traffic Indication Map (TIM): DTIM 1 of 1 bitmap
    Tag Number: Traffic Indication Map (TIM) (5)
    Tag length: 4
    DTIM count: 0
    DTIM period: 1
  ▾ Bitmap control: 0x00
      .... ...0 = Multicast: False
      0000 000. = Bitmap Offset: 0x00
    Partial Virtual Bitmap: 00
```

Figura 76 - Campos principales del campo TIM.

DTIM count indica el número de balizas que quedan antes de un *DTIM*[23] debido a que no todos los *beacons* la incluyen. *DTIM* es un tipo de *TIM* que informa a los clientes sobre la presencia de datos de difusión en el AP. Después de éste, el AP enviará los datos por el canal siguiendo las normas CSMA/CA, ayudando a minimizar las colisiones y, en consecuencia, aumentando el rendimiento. En cuanto a *DTIM period*, cuanto mayor sea, más tiempo podrá dormir una estación y, por lo tanto, más energía podrá ahorrar. Para ordenadores, es recomendable un valor bajo "1" como se observa en la figura ya que requieren un rendimiento de comunicación bastante alto y tiene baja sensibilidad al ahorro de energía. Sin embargo, en dispositivos móviles, en los que sí son sensibles al ahorro de energía, las balizas tienen un periodo *DTIM* relativamente alto, con un valor de "8", debido a la limitación de los móviles por sus baterías. Además, al ser "1" indica que cualquier otra baliza es un *DTIM*, si por el contrario fuera "8", indicaría que una de cada ocho balizas es un DTIM. Así pues, se ha podido observar que las estaciones saben y son conscientes de los intervalos de transmisión *DTIM* cuando procesan *beacons* (22).

En cuanto a *Bitmap control*, significa control del mapa de bits y si hubiera un "1", se reflejaría en el campo *Traffic Indication* e indicaría que aún queda tráfico de multidifusión en el buffer del AP.

Sobre el parámetro *Partial Virtual Bitmap*, mapa de bits virtual parcial, será el lugar dónde se almacene un listado que contiene para cada estación asociada, sus "AIDs". Así, cuando una estación reciba la baliza, sabrá si el AP tiene tráfico *unicast* para ésta, debido a que en su posición correspondiente se indicará. En caso contrario, *aparecerá traffic is not buffered*. En nuestro caso, tiene el valor 0x00, luego no ha registrado ninguna entrada.

[23] Incluida esta trama, por lo que 0 significa que esta trama es una DTIM, si fuera "1" significaría que la siguiente baliza es un DTIM. Proviene de los términos en inglés *Delivery Traffic Indication Message*.

Además, otro campo que observamos es *Country information*, como se observa en la Figura 77:

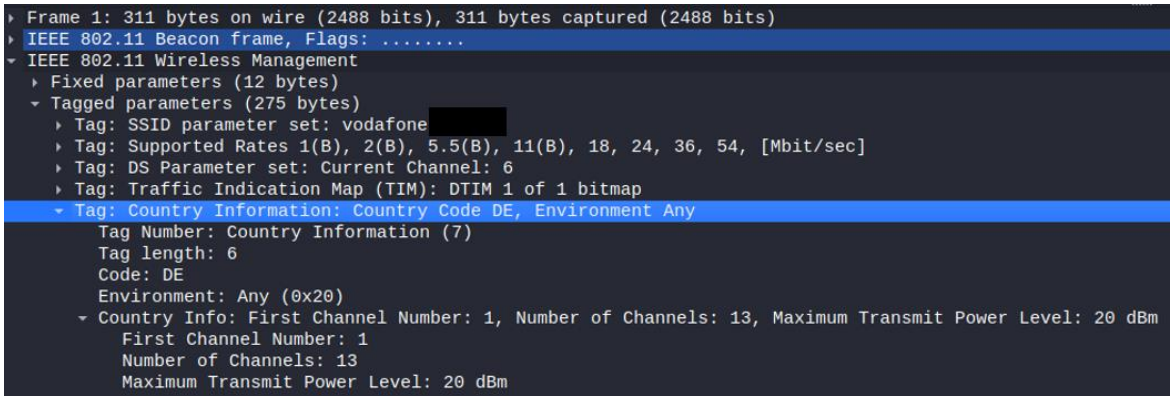

Figura 77 - Campo country information.

Este campo surgió para resolver algunos problemas derivados de las restricciones reglamentarias y dominios reguladores existentes entre países creándose a partir de la norma 802.11d-2001. Así, las estaciones están informadas acerca de los canales permitidos y las salidas de transmisión máximas para cumplir las regulaciones impuestas por el país. Su estructura es la que se muestra en la Figura 78:

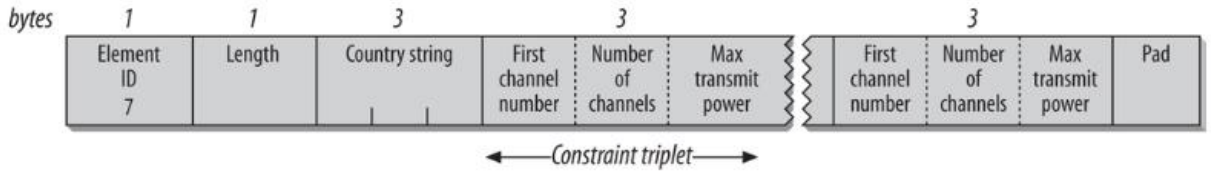

Figura 78 - Estructura del campo country information.

A través del string *Country Code* se observa qué reglas se siguen dependiendo del país. En nuestro caso, "*DE*" pertenece a Alemania, según la tabla (23) :

Además, se tienen otros elementos como *First channel*, que indica el canal más bajo disponible; *Number of channels*, que indica el número de canales totales disponibles y *Maximum transmit power level* que indica la potencia máxima de transmisión de salida expresada en dBm.

En la Figura 79 se tiene otro campo es *TPC*[24] *Report Transmit Power* (24):

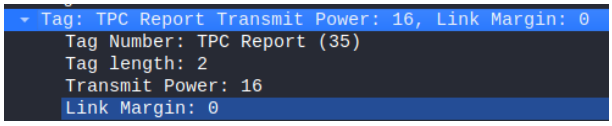

Figura 79 - Campo TPC RTP.

[24] Del inglés Transmit Power Control, que significa que alberga el control de la potencia transmitida.

Se encarga como su propio nombre indica, de controlar la potencia que se transmite y serán los AP los que mediante la norma 802.11k aplicarán estas modificaciones para informar a las estaciones de los niveles máximos. Está muy relacionado con el campo descrito anteriormente *maximum transmit power level* y, si no se controlan estos límites, pueden surgir problemas de rendimiento en la red ya que si un AP transmite con mucha potencia, pero el cliente no dispone de la suficiente, podría darse el caso de que el AP no recibiera el mensaje de la estación. La tecnología *Lumos* (25) funciona mediante estos indicadores de potencia con el objetivo de detectar cámaras e IoT ocultos en hoteles, etc.

El siguiente campo a analizar sería *RSN*[25] *Information*, como se observa en la Figura 80:

```
▾ Tag: RSN Information
    Tag Number: RSN Information (48)
    Tag length: 24
    RSN Version: 1
  ▸ Group Cipher Suite: 00:0f:ac (Ieee 802.11) TKIP
    Pairwise Cipher Suite Count: 2
  ▸ Pairwise Cipher Suite List 00:0f:ac (Ieee 802.11) AES (CCM) 00:0f:ac (Ieee 802.11) TKIP
    Auth Key Management (AKM) Suite Count: 1
  ▸ Auth Key Management (AKM) List 00:0f:ac (Ieee 802.11) PSK
  ▸ RSN Capabilities: 0x000c
```

Figura 80 - Campo RSN Information.

Su estructura es la que se muestra en la Figura 81:

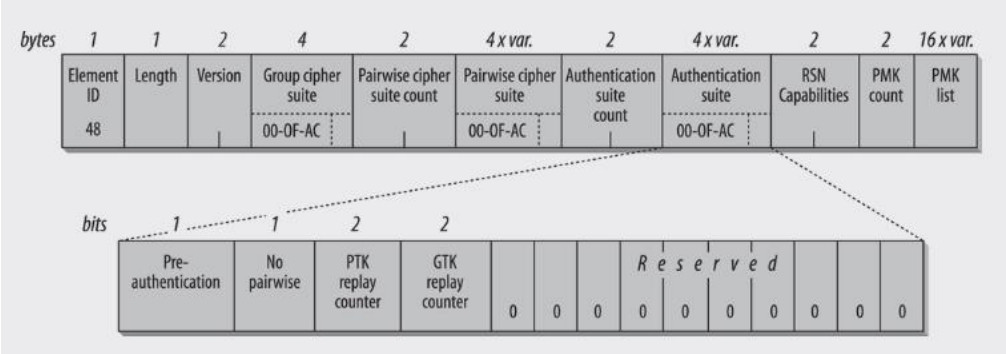

Figura 81 - Estructura del campo RSN Information.

Como se puede observar, la versión que está disponible actualmente es la "1", definida por el estándar 802.11i. Por el contrario, el valor "0" está reservado. Cualquier otro valor no está definido aún.

A continuación, se tiene *Group Cipher Suite*, como se observa en la Figura 82 cuya función principal es la de tener un único cifrado de grupo común soportado por todas las estaciones para proteger las tramas *multicast* y *broadcast*. La razón de la existencia de este campo es debido a que el tráfico de difusión se cifra de manera diferente al tráfico *unicast*. Entonces, cuando un *SSID* dispone de varios métodos de cifrado, deberá escoger aquel "más bajo", es decir, aquel que esté soportado por todas las estaciones asociadas a él ya

[25] Del inglés *Robust Secure Networks*, protocolo utilizado para realizar el apretón de manos cuando una estación decide asociarse con un AP, almacenando todos los cifrados soportados.

que, de lo contrario, no podrían descifrar el tráfico. Por ejemplo, si un AP utiliza como cifrado de grupo *CCMP* y existen estaciones que solo disponen de *TKIP*, jamás entenderán el tráfico. En nuestro caso, se corresponde con el cifrado *TKIP*, con tipo de suite "2", como se observa en la Tabla 4.

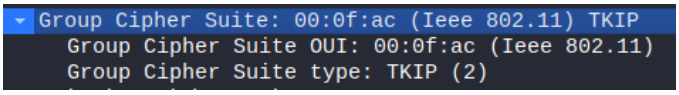

Figura 82 - Campo Group Cipher Suite

A continuación, se muestra la tabla de números únicos asociados a cada tipo de cifrado:

Tabla 4 - Suites de cifrado disponibles en 802.11

Suites de cifrado		
OUI	Tipo de suite	Cifrado
00-0F-AC	0	Cifrado grupo (Cifrado por pares)
00-0F-AC	1	WEP-40
00-0F-AC	2	TKIP
00-0F-AC	3	Reservado
00-0F-AC	4	CCMP (AES)
00-0F-AC	5	WEP-104
Proveedor	Cualquier valor	Definido por el proveedor

Pairwise Cipher Suite (count + list), sirve para definir el cifrado para el tráfico *unicast* y posee un listado con los posibles valores de la tabla anterior, no existe un límite. En nuestro caso, soporta dos tipos, *CCMP (AES)* o *TKIP* como se observa en la Figura 83:

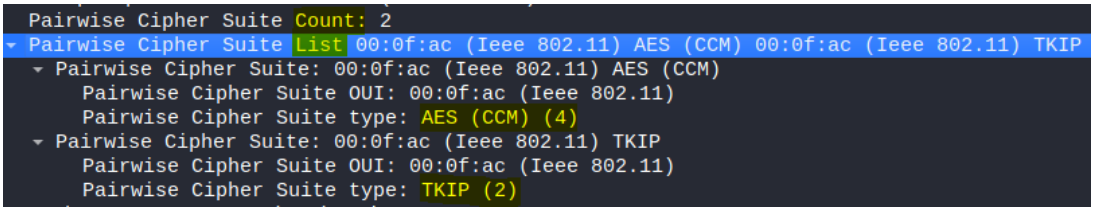

Figura 83 - Campo Pairwise Cipher Suite List.

Auth Key Management (count + list) se encarga de identificar la autenticación admitida en un SSID, siendo así compatible con las dos formas de autenticación mencionadas, 802.1X o PSK, correspondientes con los siguientes valores mostrados en la Tabla 5:

Tabla 5 - Suites de autenticación disponibles en 802.11.

Suites de Autenticación		
OUI	Tipo de suite	Cifrado
00-0F-AC	1	802.1X (EAP)
00-0F-AC	2	PSK
Proveedor	Cualquier valor	Definido por el proveedor

En nuestro caso, se puede ver en la Figura 84 que solo se soporta la autenticación PSK:

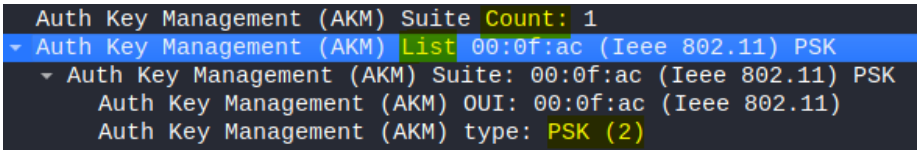

Figura 84 - Campo Auth Key Management.

RSN Capabilities describe lo que el transmisor es capaz de hacer mediante la activación de ciertos bits, descritos en la Figura 85:

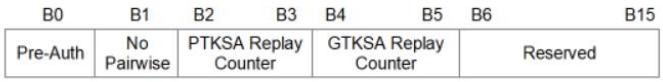

Figura 85 - Estrucutra de bits del campo RSN Capabilities.

Por ejemplo, *Pre-Auth capabilities* está relacionado con la preautenticación en la que las estaciones pueden autenticarse con varios APs durante el proceso de escaneo para que cuando se requiera asociación, la estación ya esté autenticada (26). En la Figura 86 no está soportada. *No Pairwise capabilities* establece cuando una estación puede admitir una clave WEP manual para datos de difusión junto con una clave unicast más fuerte, no es recomendable excepto que sea estrictamente necesario.

PTKSA —del inglés *Pairwise Transient Key Security Association*— y *GTKSA* —del inglés *Group Transient Key Security Association*— *Replay Counter Capabilities* —que significa contador de reproducción por pares y contador de reproducción en grupo— está relacionado con prioridades para servicios *QoS* cuando se tienen que retransmitir tramas. Dependiendo del valor asignado, se realizarán más o menos intentos, como se puede apreciar en la Tabla 6:

Tabla 6 - Números de reintentos dependiendo de la calidad del servicio.

Number of Replay Counters value	Meaning
0	1 replay counter per PTKSA/GTKSA
1	2 replay counter per PTKSA/GTKSA
2	4 replay counter per PTKSA/GTKSA
3	16 replay counter per PTKSA/GTKSA

En nuestro caso está asignado este bit con un “3”, luego hay 16 intentos.

El resultado de la captura es la que se muestra:

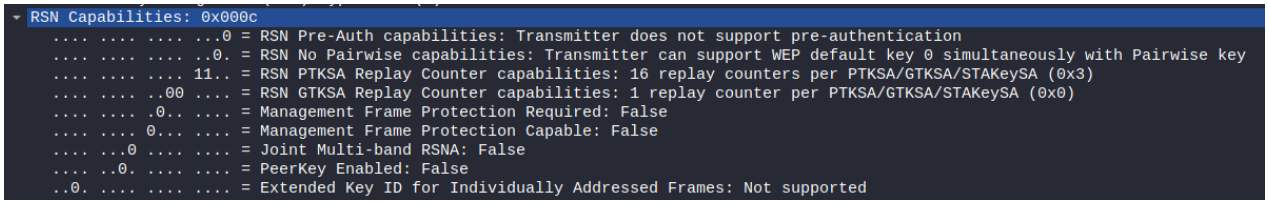

Figura 86 - Campo RSN Capabilities.

Así, hemos podido entender cómo funciona esta tecnología y sus mecanismos para transmitir información a bajo nivel.

Actualización drivers de la interfaz de red TP-Link TL-WN722N v2/v3

A continuación, se describirán una serie de instrucciones a seguir con el objetivo de actualizar los drivers en la propia interfaz mediante software para conseguir que admita modo monitor.

Conectamos la interfaz a nuestro ordenador y podemos observar en la Figura 87 que no aparece:

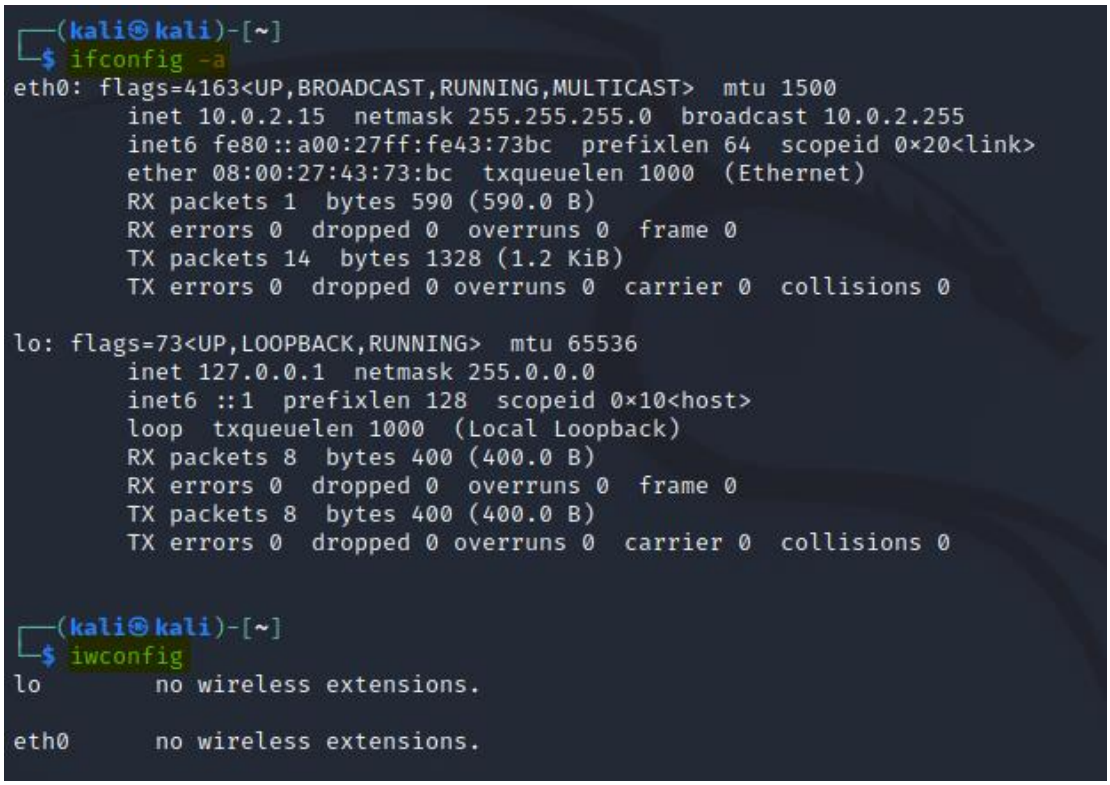

Figura 87 - Resultado del comando ifcongif -a.

La razón por la que sucede es debido a que como estamos en una máquina virtual, debemos habilitar el puerto USB correspondiente. Para ello, clicamos en el icono con imagen de USB y mediante el botón derecho seleccionamos “Realtek 802.11n NIC”, como se puede apreciar en la Figura 88:

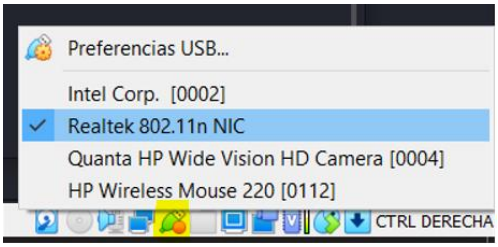

Figura 88 - Habilitación del puerto USB en la máquina virtual.

Así pues, mediante los comandos que se muestran en la Figura 89 y la Figura 90 se puede observar la presencia de nuestra interfaz de red “wlan0” en la máquina virtual. Cabe remarcar que, con los drivers de fábrica, la interfaz aparecía en modo *Managed*, sin embargo, una vez actualizados aparece en modo *Auto*:

Figura 89 - Aparición de la interfaz wlan0.

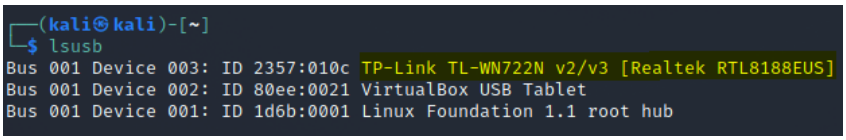

Figura 90 - Descripción completa del driver.

Una vez conectada la interfaz, procederemos a configurar sus drivers debido a que, si se intenta activar el modo monitor para escuchar el tráfico que circula por la red, el sistema responde que no es posible, como se observa en la Figura 91:

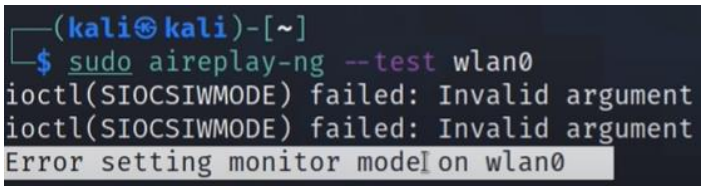

Figura 91 - Error al intentar activar el modo monitor.

Mediante los comandos de la Figura 92, se conseguirá nuestro objetivo (27).

```
DAVID BOMBAL: https://www.youtube.com/watch?v=tYnjMiTTdms
!Use these commands to get the adapter working on Kali for packet injection and monitoring:
!Commands:
==========
sudo apt update
sudo apt upgrade
sudo apt install bc //aqui al acabar reincia pc
sudo apt-get install build-essential
sudo apt-get install libelf-dev

!Try either of these commands to see which works:
sudo apt-get install linux-headers-`uname -r`
sudo apt-get install linux-headers-5.10.0-kali6-amd64

sudo apt install dkms
sudo rmmod r8188eu.ko
git clone https://github.com/aircrack-ng/rtl8188eus
cd rtl8188eus
sudo -i
echo "blacklist r8188eu" > "/etc/modprobe.d/realtek.conf"
exit
sudo reboot
sudo apt update
cd rtl8188eus
sudo make
sudo make install
sudo modprobe 8188eu
```

Figura 92 - Comandos para actualizar los drivers.

Una vez ejecutadas las órdenes anteriores, se habrá conseguido con éxito actualizar los drivers de la interfaz. A continuación, se procederá a activar el modo monitor. Para ello, deberemos apagar y encender la interfaz mediante la orden *ifconfig* con los parámetros *up* y *down*, como indica el manual en la Figura 93:

```
up      This  flag  causes  the interface to be activated.  It is implicitly specified if an address is as-
        signed to the interface; you can suppress this behavior when using an alias interface by  appending
        an  -  to  the alias (e.g.  eth0:0-).  It is also suppressed when using the IPv4 0.0.0.0 address as
        the kernel will use this to implicitly delete alias interfaces.

down    This flag causes the driver for this interface to be shut down.
```

Figura 93 - Manual ifconfig.

Además, mediante la orden *airmon-ng check kill* se matará cualquier proceso que pueda interferir con el modo monitor (28). Por último, especificaremos el modo monitor desde la orden *iwconfig wlan0 mode monitor*. Todas estas órdenes necesitarán privilegios de *root*, por esta razón se utilizará *sudo*. Con el fin de agilizar el proceso y debido a que estas órdenes se deben ejecutar siempre ya que la interfaz por defecto arranca en modo “auto”, se ha creado el script de la Figura 94:

monitorMode.sh

```
#!/bin/sh

echo Apagando la interfaz ...
sudo ifconfig wlan0 down

echo Deshabilitando cualquier proceso que pueda interferir ...
sudo airmon-ng check kill

echo Configurando la interfaz de red wlan0 en modo monitor ...
sudo iwconfig wlan0 mode monitor

echo Reiniciando interfaz ...
sudo ifconfig wlan0 up

echo Interfaz de red wlan0 configurada con éxito ...
echo ""
iwconfig

```

Figura 94 - Script monitorMode.sh.

Se otorgan permisos de ejecución con *chmod +x monitorMode.sh* y se ejecuta como muestra la Figura 95:

```
┌──(kali㉿kali)-[~/Desktop/FRAN/TFG-WiFi]
└─$ ./monitorMode.sh
Apagando la interfaz ...
[sudo] password for kali:
Deshabilitando cualquier proceso que pueda interferir ...

Killing these processes:

    PID Name
   1250 wpa_supplicant

Configurando la interfaz de red wlan0 en modo monitor ...
Reiniciando interfaz ...
Interfaz de red wlan0 configurada con éxito ...

lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11b  ESSID:""  Nickname:"<WIFI@REALTEK>"
          Mode:Monitor  Frequency:2.412 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality=0/100  Signal level=-100 dBm  Noise level=0 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Figura 95 - Ejecución del script y activación con éxito.

De esta forma ya tenemos nuestra interfaz en modo monitor y podemos inyectar paquetes sin errores (29), como se muestra en la Figura 96:

```
┌──(kali㉿kali)-[~/Desktop/FRAN/TFG-WiFi]
└─$ sudo aireplay-ng --test wlan0
[sudo] password for kali:
02:09:10  Trying broadcast probe requests...
02:09:10  Injection is working!
02:09:12  Found 9 APs

02:09:12  Trying directed probe requests...
02:09:12  D8:7D:[redacted]:5F - channel: 1 - 'MIWIFI[redacted]
02:09:13  Ping (min/avg/max): 7.726ms/22.437ms/54.678ms Power: -82.21
02:09:13  28/30:  93%

02:09:13  10:C2:[redacted]:4F - channel: 1 - 'vodafone[redacted]
02:09:13  Ping (min/avg/max): 3.875ms/11.047ms/27.965ms Power: -25.23
02:09:13  30/30: 100%

02:09:13  E8:AD:[redacted]:3E - channel: 1 - 'vodafone[redacted]
02:09:14  Ping (min/avg/max): 3.915ms/12.159ms/24.275ms Power: -69.87
02:09:14  30/30: 100%

02:09:14  DA:7D:[redacted]:5C - channel: 1 - 'MIWIFI[redacted]
02:09:17  Ping (min/avg/max): 7.772ms/44.132ms/117.100ms Power: -82.12
02:09:17  16/30:  53%

02:09:17  2C:EA:[redacted]:1F - channel: 1 - 'MOV[redacted]
02:09:21  Ping (min/avg/max): 9.666ms/24.144ms/42.816ms Power: -93.00
02:09:21  14/30:  46%

02:09:21  14:B7:[redacted]:B7 - channel: 1 - 'vodafone[redacted]
02:09:27   0/30:   0%

02:09:27  10:C2:[redacted]:40 - channel: 1 - 'Lowi[redacted]
02:09:28  Ping (min/avg/max): 6.257ms/23.965ms/42.787ms Power: -74.00
02:09:28  28/30:  93%

02:09:28  4C:1B:[redacted]:18 - channel: 1 - 'Mi[redacted]
02:09:34   0/30:   0%

02:09:35  50:0F:[redacted]:01 - channel: 2 - 'Ten[redacted]'
02:09:41   0/30:   0%
```

Figura 96 - Inyección de paquetes.

Objetivos de Desarrollo Sostenible (ODS)

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**	X			
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**	X			
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

El estudio realizado se acerca en gran medida a algunos de los objetivos de desarrollo sostenibles debido a que gracias al software de simulación Mininet-WiFi, se evita el hecho de comprar dispositivos electrónicos qué finalmente se desecharán debido a su mal rendimiento en el sistema de información preestablecido.

De los anteriores objetivos de desarrollo sostenibles mencionados, el proyecto está relacionado con: “Educación de calidad”, “Industria, innovación e infraestructuras” y “Acción por el clima”. A continuación, se detallarán las razones por las cuales este trabajo tiene un gran impacto positivo para la sociedad en los ODS:

Educación de calidad: gracias a las redes definidas por software se tiene la capacidad de manipular y programar las redes de manera flexible, aspecto que permite facilitar el aprendizaje acerca de su funcionamiento de manera didáctica y dinámica, pudiendo aplicar las mejoras descubiertas en la realidad.

Industria, innovación e infraestructuras: se tiene un impacto positivo debido a que el propósito de la tecnología involucrada siempre sigue la dirección de proponer mejoras e innovaciones mediante configuraciones óptimas que permitan sacarle el mayor partido al hardware real, diseñando en consecuencia mejores infraestructuras.

Acción por el clima: esta tecnología de simulación evita la compra innecesaria de equipos informáticos, routers, switches y más dispositivos de red debido a que tiene como propósito el hecho de realizar pruebas virtualmente antes de implantar el sistema en la realidad, pudiéndose medir los rendimientos y la calidad de funcionamiento en la simulación con el fin de hallar la mejor elección y poder instalarla con una probabilidad de éxito cercana al 100%, reduciendo así la huella de carbono y la retirada prematura de aparatos electrónicos, es decir, evitando la incesante generación de residuos electrónicos del mundo.