UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

MODELADO DE REDES SDN CON MININET

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Francisco Jose Navarro Ojeda

Tutor: Pietro Manzoni

2016-2017

Resumen

Las deficiencias que presentan las arquitecturas actuales en el diseño de redes frente a la aparición de nuevas tecnologías que necesitan de un mejor control del ancho de banda de la red y una mejora en la gestión de estas ha provocado la aparición de las redes SDN. Esta nueva arquitectura centraliza la lógica que gestiona el tráfico de la red a diferencia de otros modelos.

Durante el siguiente proyecto se ha estudiado esta nueva arquitectura de red y los beneficios que aporta, además de observar mediante experimentación en el emulador Mininet-wifi el comportamiento de red de este. Pudiendo evaluar también el funcionamiento de este emulador.

Palabras clave: Mininet-wifi, redes, simulador, SDN, OpenFlow

Abstract

The deficiencies presented by the current architectures in the design of networks against the appearance of modern technologies that need a better control of the bandwidth of the network and an improvement in the management of these has caused the emergence of SDN networks. This new architecture centralizes the logic that manages the traffic of the network unlike other models.

During the next project has studied this new network architecture and the benefits it provides, besides observing through experimentation in the emulator Mininet-wifi network behavior of this. You can also evaluate the operation of this emulator.

Key words: Mininet-wifi, network, simulator, SDN, OpenFlow

Índice

Lista de imagenes

Listado de abreviaturas

Abreviatura	Significado
SDN	*Software-Defined Networking*
SDWN	*Software-Defined Wireless Network*
CAPEX	*Capital expenditures*
OPEX	*Operating Expense*
OF	*OpenFlow*
VANET	*Redes vehiculares ad-hoc/Vehicular ad-hoc network*
ITU	International Telecomunication Union
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
ARP	Addres Resolution Protocol
ACK	Acknowledgement
sta	*Station/estacion*
ap	*Acces Point/ Punto de acceso*
lo	*Loopback*
SSID	*Service Set Identifier*

1. Introducción

En este apartado se procederá a explicar la teoría e información necesaria para entender el desarrollo del proyecto y sus simulaciones. Estará dividido en 3 bloques: Contexto, contenido teórico y Mininet-wifi.

1.1. Contexto

El diseño de las redes actuales se hizo de forma que el control de flujo y enrutamiento es ejercido directamente en los dispositivos de la red, dándole una estructura estática, jerárquica y dependiente de la infraestructura.

La complejidad para acomodar las redes a las necesidades de los usuarios, abarcando todas las políticas de una red y la dificultad a la hora de escalarla son los problemas a los que se enfrentan estos diseños.

El mundo actual donde la aparición de las redes sociales, los dispositivos sociales y el *cloud computing*, las redes tienden a saturarse y consumen un alto ancho de banda debido a estas nuevas tecnologías. Los grandes beneficios de la computación y el almacenamiento virtual se están viendo limitados por las redes. Todo ello a dado lugar al problema de desarrollar nuevas redes capaces de satisfacer las nuevas necesidades que tienen ahora.

Bajo la necesidad de un nuevo modelo de red que mejore sus capacidades ha aparecido las redes SDN (Software-Defined Networking).

1.2. Conceptos teóricos

1.2.1. Software-Defined Networking (SDN)

Software-Defined Networking (SDN) y *Software-Defined Wireless Network* (SDWN) en el caso de las redes wifi, es una nueva arquitectura de diseño de red en la que se separan la capa de control y la capa de datos dentro de una red. Separando el software de gestión del hardware de red y dándole el control a otros dispositivos llamados controladores que convierten el control de la red en un servicio centralizado.

SDN ha surgido de la necesidad por paliar las deficiencias de las redes actuales haciéndolas más escalables, mejorando su CAPEX y OPEX (*Capital expenditures y Operating Expense*). Reduciendo el hardware necesario para su montaje pudiendo reutilizar hardware (CAPEX), facilitando la gestión de los elementos de control de la red haciendo más sencilla su configuración y reduciendo el tiempo de gestión de los administradores (OPEX), agilizando el despliegue de aplicaciones, servicios e infraestructuras y controlando el efecto de estos en la red.

El gran objetivo de las redes SDN es permitir a los ingenieros de redes y los administradores responder rápidamente a los cambios producidos en los negocios centralizando la consola de control, con lo que se mejora los servicios en red haciéndolos más dinámicos, económicos y escalables evitando la gestión a bajo nivel.

Una red SDN tiene tres partes principales. Un controlador, el cerebro de la red y el encargado de indicarle al resto de dispositivos como manejar el tráfico en la red. Southbound APIs, un software como el protocolo OpenFlow, encargado de gestionar la comunicación entre el controlador y los dispositivos. Y por último Northbound APIs, otro software encargado de comunicarse con las aplicaciones y la lógica de negocio.

Para poder llevar acabo esta funcionalidad en una red es necesario que los dispositivos de esta tengan disponible un firmware con OpenFlow (OF) u otro protocolo similar.

Las redes SDN permiten gestionar de forma flexible los dispositivos gracias a que el controlador conoce toda la topología de la red. Entre las posibilidades que da el controlador están la creación de reglas para el tráfico de la red permitiendo al administrador manejar cargas de trafico de manera flexible y eficiente, dando prioridad a procesos, aplicaciones o servicios.

1.2.2. OpenFlow(OF)

OpenFLow(OF) es un protocolo de estándar abierto utilizado para la gestión de redes SDN. Debido a su gran difusión se confunde OpenFlow con SDN, pero no son lo mismo, OpenFlow tan solo es uno de protocolos utilizados, aunque se ha estandarizado por su uso generalizado.

En las redes convencionales, el software del fabricante instalado en el dispositivo le indica que debe hacer y cómo gestionar la red, pero OpenFlow centraliza las decisiones del dispositivo permitiendo la programación de dispositivo y su funcionamiento.

El protocolo cumple la función de conectar el software controlador (dispositivo controlador) con los dispositivos de red, indicándole a los switch donde enviar los paquetes. El controlador utilizando este protocolo tiene acceso a los dispositivos, que puede configurar y elegir rutas para distribuir el tráfico de red.

Pese a que es uno de los protocolos más utilizados no es el único existente, tampoco existe un modelo estándar.

La última versión de OpenFlow es la 1.4, habiendo tenido cinco versiones mayores en total. En la 1.0 presentaba un total de veintiún tipos de paquetes distintos que ha ido creciendo hasta situarse en los treinta y cuatro en la última versión.

1.3. Mininet-wifi

Mininet-wifi es un emulador de redes SDWN, es decir, sustituye la parte hardware de la red por un software manteniendo el realismo en forma virtual. Ha sido desarrollada en Python principalmente con una parte menor en #C además de haber sido publicada como Open-Source.

Antes de la aparición de Mininet-wifi, existía lo que puede llamarse una versión previa llamada Mininet. En ella al igual que en Mininet-wifi se emulan redes SDN, pero en este caso no son wifi, es decir, solo emula redes con conexiones físicas.

La llegada de Mininet-Wifi y Mininet se vio propiciada por la ausencia de aplicaciones similares que fuesen accesibles. Esto se debe a que, aunque existían otras aplicaciones similares no presentaban las características necesarias o debido a que fuesen aplicaciones propietario quedaban fuera del alcance de muchos usuarios y empresas. Su primera versión fue publicada el 12 de septiembre de 2015 en GitHub por sus principales desarrolladores Ramon dos Reis Fontes y Christian Rodolfo Esteve Rothenberg.

La aparición de Mininet-wifi, incorporo todas las utilidades y funciones de Mininet. Por lo que al utilizar Mininet-wifi se tiene acceso a un emulador competo de redes SDN tanto físicas como wifi.

La versión de Mininet-wifi también incorpora utilidades para crear redes *ad-hoc* simuladas con vehículos en ciudad, redes vehiculares (VANET) junto a SUMO (Simulador de tráfico) aunque este ámbito queda fuera de este proyecto.

Mininet-wifi presenta una ventaja importante comparada con otros emuladores y simuladores similares y es que permite emular la red con máquinas reales, las estaciones o dispositivos que emula, generan tráfico y se comportan como maquinas reales, se puede ejecutar código en ellas y aplicaciones a nivel de usuario lo que permite realizar experimentos y pruebas en escenarios reales con código real.

Entre los grandes problemas que presenta Mininet-wifi esta que se diseñó de forma que utilizara un *kernel* de procesador Linux, por lo que solo funciona en estos sistemas. Si se quisiera utilizar en otro sistema operativo deberá utilizarse un emulador para la máquina. Conocedora de este problema en la página web de GitHub Mininet-wifi ofrece un enlace con una máquina virtual perfectamente configurada.

Durante las simulaciones que se llevan a cabo en la aplicación el usuario mantiene un control total de la aplicación permitiéndole modificar parámetros de los dispositivos en la red, como su posición o alcance de la señal, incluso añadir dispositivos nuevos y eliminar algunos existentes.

Las dos principales incorporaciones a Mininet-wifi son la movilidad de los dispositivos y la propagación de la señal, donde se pueden encontrar varios posibles métodos para llevar acabo ambos.

Mininet-wifi también permite mostrar de forma visual la situación física de la red pudiendo localizar rápidamente la ubicación de las estaciones si estas están en movimiento, desgraciadamente no es muy precisa por este mismo motivo, pero permite controlar la situación de la red si se ha generado unos patrones de movimiento a seguir durante la simulación o esta sigue un patrón aleatorio. Y tampoco permite percibir la tercera dimensión pese a que se puede trabajar con ella. Como usar la tercera dimensión no aporta información adicional para las pruebas y simulaciones que se realizaran más adelante, esta se ha obviado para mantener un escenario más sencillo.

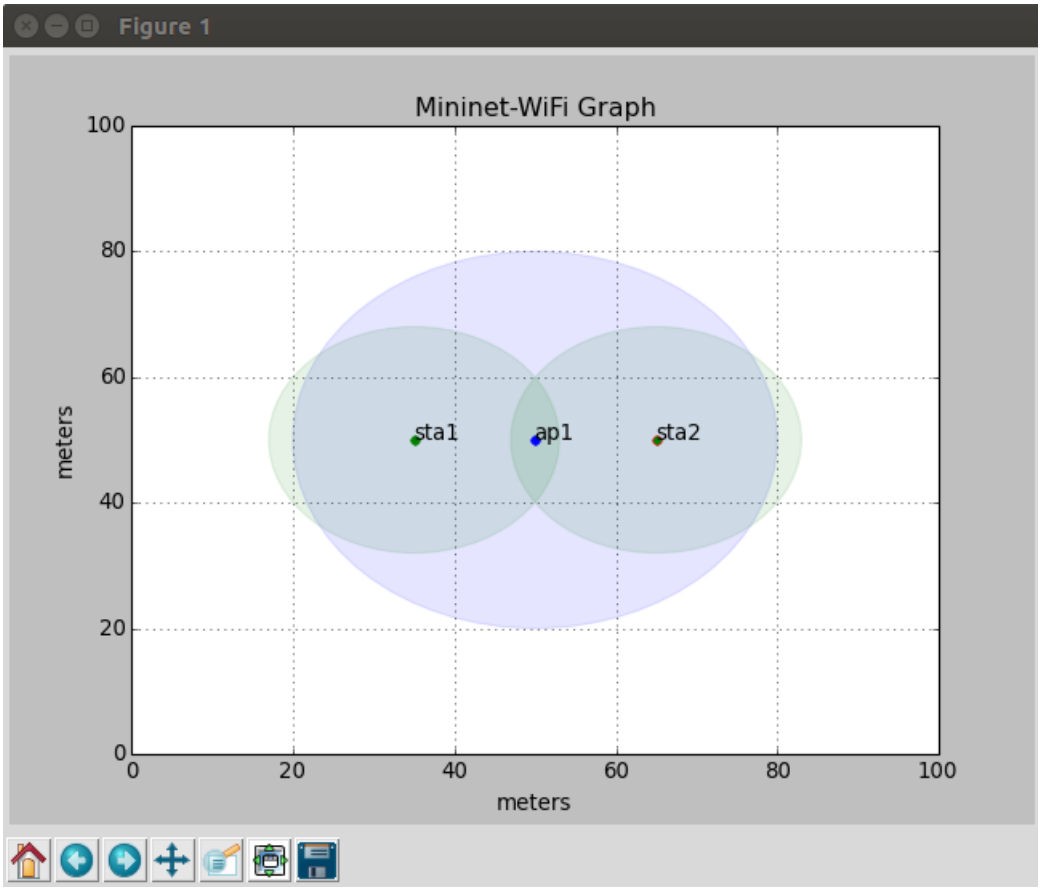

Imagen 1- Visualización de red en Mininet-wifi

1.3.1. Movilidad

Los modelos de movilidad permitidos en Mininet-wifi son los siguientes:

- RandomWalk: Basado en seguir una sucesión de pasos donde todos los posibles movimientos tienen siempre la misma probabilidad de ser llevados a cabo.
- RandomDirection: el nodo se mueve en una dirección hasta que por algún motivo externo no pueda o le indique que no puede continuar, entonces cambia a otra dirección.

- RandomWayPoint: En este modelo el nodo antes de comenzar moverse selecciona unas coordenadas destino a las que posteriormente se moverá, una vez alcanzado este *waypoint* seleccionará otro al que moverse a continuación.
- GaussMarkov: Los nodos que utilicen este algoritmo de movimiento elegirán una dirección y velocidad. Este algoritmo funciona de la siguiente forma. Primero elige una dirección y velocidad aleatoria completamente, pero después de un tiempo indeterminado usara los parámetros de velocidad y dirección más un nuevo parámetro de aleatoriedad para calcular la nueva dirección y velocidad que deberá seguir el nodo

Además, hay que tener en cuenta que si no se usa la función "net.associationControl()" las estaciones no se conectarán automáticamente a los puntos de acceso, aunque estén en rango de igual forma que si no se activa la opción correspondiente para que se actualicen los *flows* durante la simulación, las rutas no se actualizarán manualmente y los cambios en la topología pueden aislar los dispositivos.

Indistintamente, de estos algoritmos de movimiento aleatorio Mininet-wifi también permite limitar la zona de movimiento o generar patrones de movimientos para las simulaciones, permitiendo generar situaciones concretas en la red en tiempo real. Para ello se indica las coordenadas y el tiempo en el que deben alcanzar esas opciones y los valores máximos y mínimos que pueden alcanzar en el eje de coordenadas, esta configuración se puede observar en la simulación de "CruceDeEstaciones" (Anexo 3).

También está preparado para poder repetir simulaciones "iguales" aunque el movimiento se indique como aleatorio gracias a la función "seed()" la cual establece la aleatoriedad de la secuencia de movimientos, es decir, dentro de una red con la línea "seed(20)" siempre devolverá el mismo patrón de movimiento. Algo muy útil si se quieren realizar diversas pruebas que no se pueden ejecutar simultáneamente, permitiéndote mantener la aleatoriedad del movimiento, pero repetir la simulación cuantas veces te haga falta.

1.3.2. Propagación

Los modelos de movilidad permitidos en Mininet-wifi son los siguientes:

- Friis Propagation Loss Model: Ecuación que trabaja bajo condiciones ideales para transmitir entre dos antenas.
- Log-Distance Propagation Loss Model: Modelo de propagación que predice la perdida de señal a través de edificios o zonas de población densas.
- Log-Normal Shadowing Propagation Loss Model: Modelo similar al anterior pero que añade una variable gaussiana de aleatoriedad a la perdida de señal calculada.

- International Telecomunication Union (ITU) Propagation Loss Model: Modelo de propagación especializado en la perdida de señal en habitaciones o salas delimitadas por paredes de cualquier forma, diseña para el uso en interiores.
- Two-Ray Ground Propagation Loss Model: Diseñado para el cálculo de perdida de señal entre dos antenas generalmente situadas a dos alturas distintas.

Aunque Mininet-wifi presente todos estos modelos para la propagación de las señales wifi también permite acceder a una función en la que se realiza el cálculo de la potencia de la señal lo que permite a cada usuario modificarla según sus necesidades si lo quisiera.

2. Simulaciones

2.1. CruceDeEstaciones

En esta primera simulación se ha generado una red de tamaño reducido (Anexo Cruce de estaciones) para comprobar el funcionamiento de la red con el menor número de nodos posibles participando en ella y el modelo de propagación ideal (Friis Propagation Loss Model) donde la perdida de señal es menor.

Para este caso se han generado dos estaciones (sta) denominadas sta1 y sta2 conectadas a un punto de acceso (ap) llamado ap1. Ap1, sta1 y sta2 se han colocado en las coordenadas '50,50,0', sta1 se alejaba de ap1 aproximándose a 0 y sta2 se alejaba de ap1 de forma opuesta. A continuación, se ha procedido a realizar pruebas mediante el comando de iperf en Linux para obtener información de la conexión, así como comprobar su evolución mientras las estaciones se alejan de ap1.

La primera prueba sobre su ancho de banda y capacidad de transmisión sin embargo se ha realizado con los tres dispositivos colocados en las coordenadas '50,50,0' para tener una muestra de ejemplo de la máxima conexión que se podría obtener para este caso. (Imagen 2)

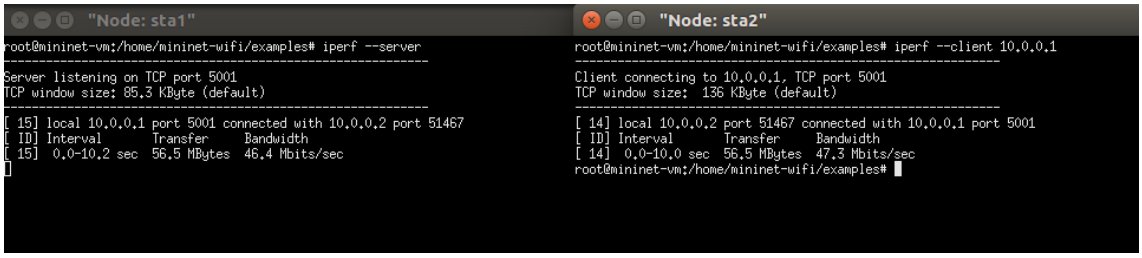

Imagen 2- Iperf entre estaciones con distancia 0

A continuación, se pretendía que gracias a la movilidad de Mininet-wifi que se obtuvieran los datos mientras se movían, pero debido a los problemas para obtenerlos correctamente se ha procedido a mover manualmente las estaciones a los puntos de elegidos para obtener los datos con el objetivo de obtener datos más precisos en coordenadas concretas. Al estar en movimiento las estaciones no realizarían la toma de datos en unas coordenadas concretas sino en un recorrido de ellas, en función de la velocidad de estas y el tamaño del stream a emitir.

Para obtener unos datos válidos para su evaluación se ha mantenido la distancia desde ap1 a ambas estaciones obteniendo así un ancho de banda y transferencia similar entre ellas, no idéntica debido a la propia conexión wifi que presenta perdidas de paquetes. Así se han obtenido datos para los pares de coordenadas de 45 y 55; 35 y 65 y 25 y 75. (Imagen 3)

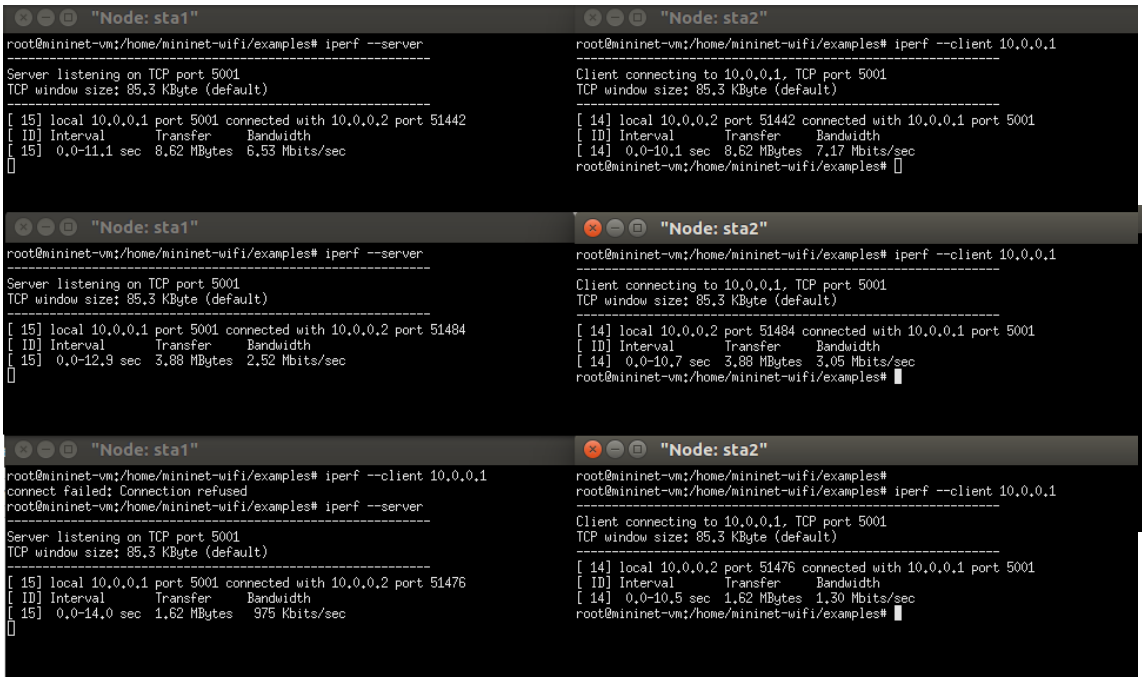

Imagen 3- Iperf entre 2 estaciones a distintas distancias

También para el punto medio de 35 y 65 se han realizado tres pruebas más para obtener más información sobre la calidad de la conexión. La primera de ellas consiste en realizar un envió de datos continuo, para ello se ha enviado cada dos segundos durante diez segundos información entre ambas estaciones. (Imagen 4)

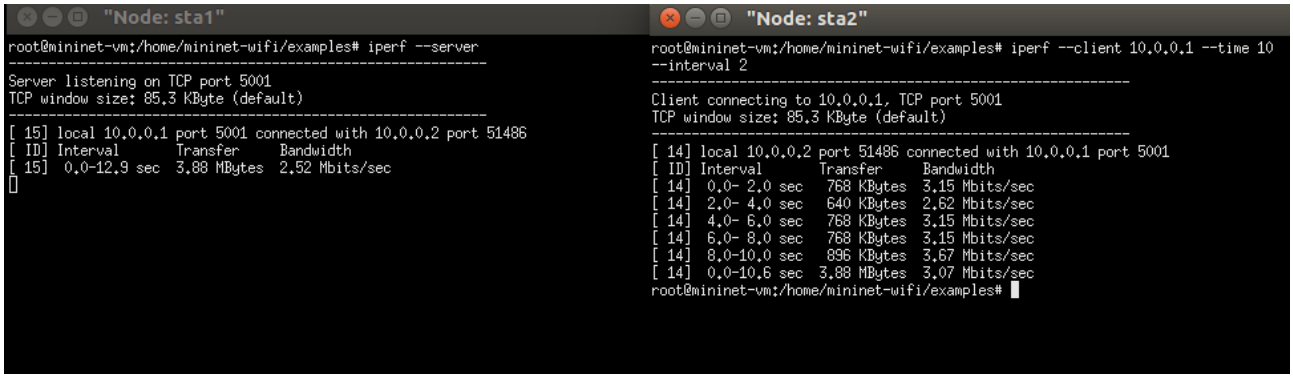

Imagen 4- Iperf de 10 segundos en intervalos de 2 segundos

Por último, se han creado archivos de 1MB y 1GB para ser trasferidos de una estación a otra (Imagen 5 y 6)

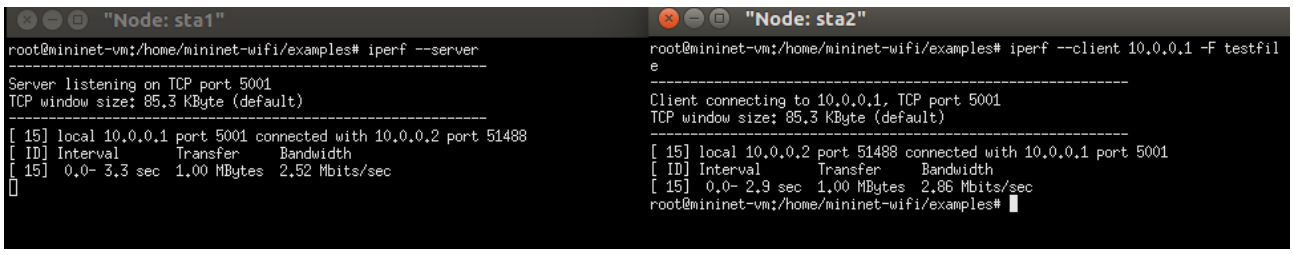

Imagen 5- Iperf de archivo con tamaño 1MB

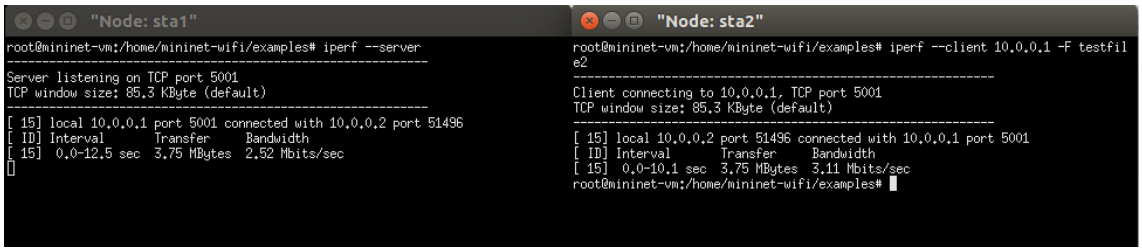

Imagen 7- Iperf de archivo con tamaño 1GB

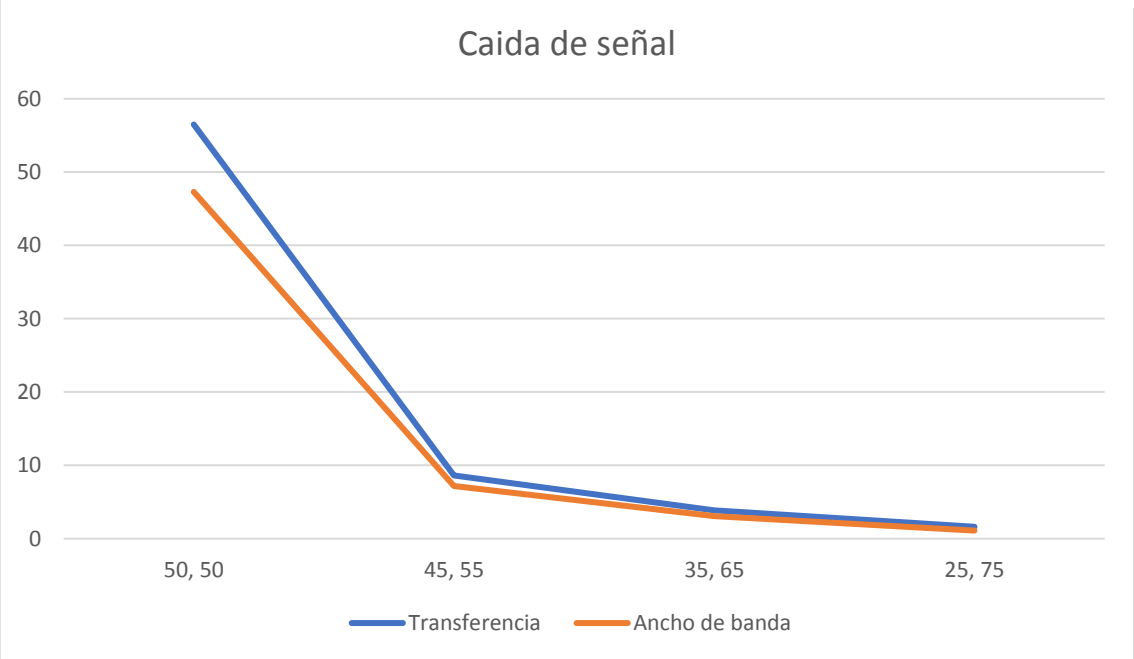

Imagen 6- Degradación de la señal al alejarse

De los resultados obtenidos con los test iperf se puede comprobar la perdida de señal con la distancia que durante los primeros metros es muy rápida, pasando de 47Mb/s a 7,17Mb/s para decrecer a continuación lentamente al igual que las redes wifi reales. También se puede observar en la figura 1.6 que la conexión no mantiene un ancho de banda estable, sino que sufre de pequeñas desviaciones típicas de las redes wifi.

Por otro lado, se puede observar en las capturas de iperf un retraso de dos segundos en el cierre de la conexión, esto se debe al cierre de la conexión una vez la transmisión finaliza y no tiene relación con el intervalo de dos segundos.

Tal y como se ha mencionado en la sección de Mininet-wifi, existen varios modelos de propagación, para comprobar su utilidad durante este proyecto se ha procedido a realizar pruebas de los distintos modelos bajo las mismas circunstancias. Ambas estaciones se colocaron en las posiciones 35 y 65 como en las pruebas anteriores y se calculó el ancho de banda para cada uno de los distintos modelos, Log-Normal Shadowing Propagation Loss, International Telecomunication Union (ITU) Propagation Loss Model, Two-Ray Ground Propagation Loss Model y Log-Distance Propagation Loss Model respectivamente. (Imagen 8)

Friis Propagation Loss Model al ser un modelo de perdida ideal y el utilizado para las pruebas anteriores se tomará como referencia. (Imagen 3)

Imagen 8- Resultados iperf con distintos modelos de propagación

Teniendo en cuenta que nuestra referencia es de 3,88 MB transferidos y un ancho de banda de 3,0Mb/s se puede observar que los modelos de ITU y Two-Ray Ground Propagation Loss Model no presentan ninguna perdida. Esto se debe a que ambos modelos están diseñados para la simulación de transmisiones a larga distancia mientras que los otros están preparados para corto alcance.

Con estos datos se ha tomado la decisión de utilizar el modelo de Friis Propagation Loss Model para realizar el resto de las simulaciones restantes, de esta forma se obtendrá el mejor resultado posible dentro de las simulaciones, aunque no lleguen a representar un valor perfecto.

He comprobado además que en la estación que sirve de servidor presenta un retraso a la hora del cierre de las conexiones cuando finaliza la transmisión. Mientras que los clientes finalizan la transmisión próximos al segundo diez, el servidor la cierra por encima del segundo doce. Para comprobar que sea un tiempo de espera estándar y no guarda relación con esa configuración del iperf he realizado una prueba rápida con distintos intervalos de envío de paquetes (Imagen 9). Con lo se observa que solo se trata de un tiempo de espera estándar de iperf.

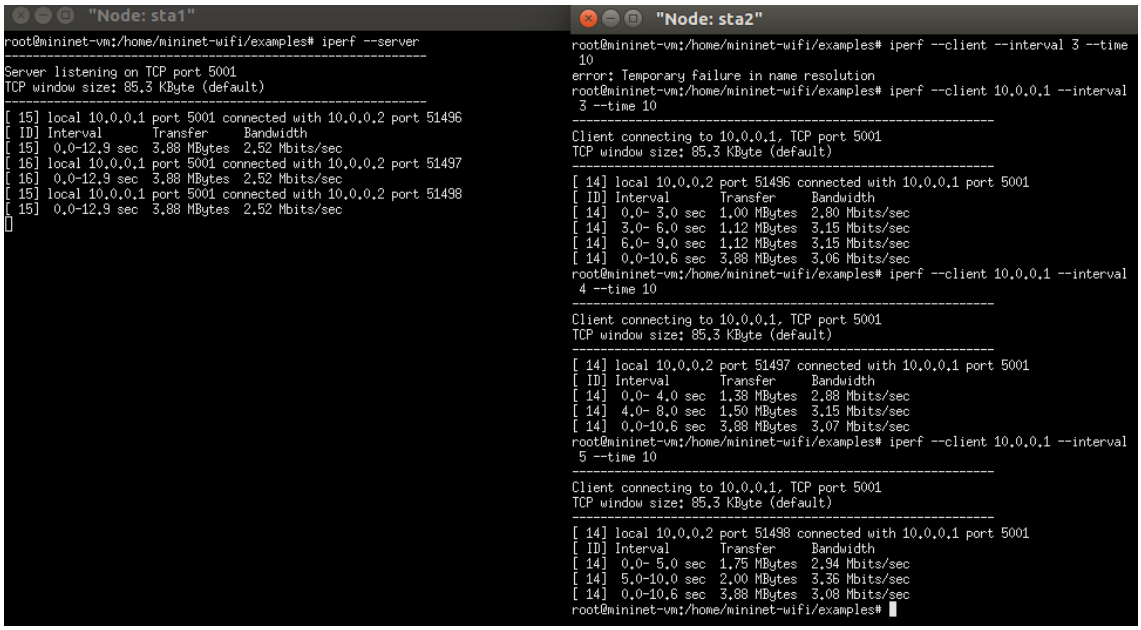

Imagen 9- Iperf con distintos intervalos de envio

2.2. MobilidadMesh

En esta simulación se ha diseñado una red ligeramente más grande, en ella se encuentra un punto de acceso y diez estaciones, nueve en movimiento y una detenida en las coordenadas ‘30,21,0’.

Para la movilidad de la red se ha utilizado el modelo de 'RandomDirection' y para fijar la estación estática se le indicaron las coordenadas de aparición y sus coordenadas máximas y mínimas siendo estas mismas, en caso de no hacer cualquiera de ellas la estación no aparecería o se movería por la red como las demás.

Además, para mantener las estaciones conectadas en todo momento al punto de acceso se les ha restringido el movimiento, lo cual en este caso se puede hacer de tres formas diferentes. Una vez calculado el cuadrado perfecto dentro de la circunferencia de radio 30 que define el alcance del punto de acceso (21,21; -21,21; -21,-21 y -21,21 si ap1 se encuentra en ‘0,0’), se coloca el punto de acceso en las coordenadas ’21,21,0’ y definir un valor máximo de 42 para los ejes x e y. El segundo definir en la movilidad las coordenadas máximas y mínimas para cada eje de forma que las estaciones se mantengan en el alcance. La tercera y última consiste en definir en cada estación sus coordenadas máximas y mínimas.

El método utilizado para las pruebas realizadas en este proyecto ha sido el primero, pero es importante mencionar que el tercer método da la posibilidad de crear zonas aisladas entre si donde se presente movilidad permitiendo simular varias redes al mismo tiempo, simular pisos o salas separadas pero que pertenecen a una misma red.

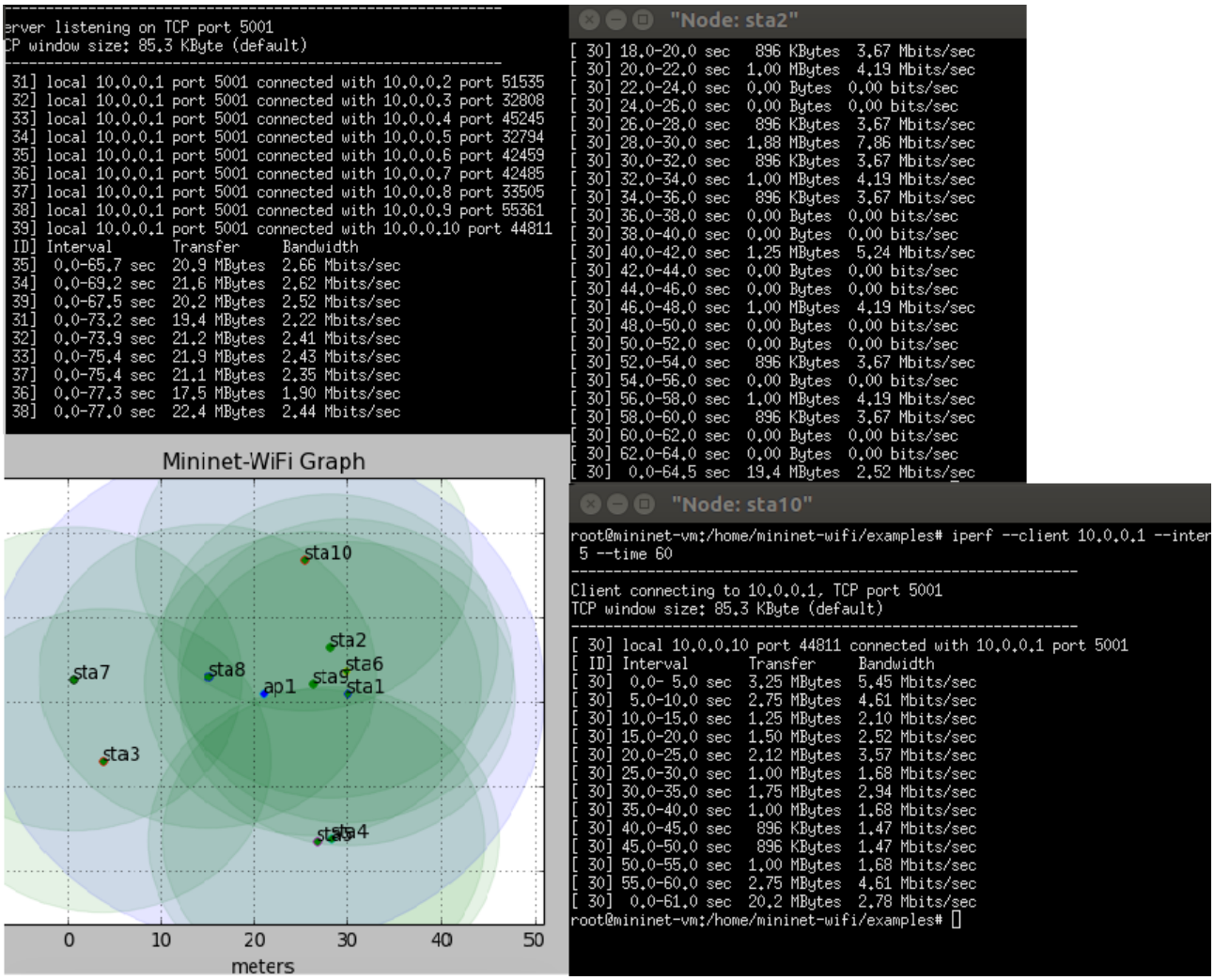

Imagen 10- Iperf TCP durante 60 segundos en intervalso de 5 segundos

En esta simulación con al iperf TCP (Imagen 10) y con un iperf UDP (Imagen 11) podemos observar el proceso de transferencia de datos en tiempo real mientras las estaciones se mueven libremente, a diferencia de la simulación anterior donde las estaciones se colocaban en determinadas coordenadas donde realizar pruebas y tomar datos.

En las capturas podemos observar varios fallos durante la retransmisión del iperf por TCP en el que en determinados intervalos no detecta o realiza transmisión, además a diferencia del iperf UDP. Además, la configuración por defecto ha aprovechado todo el ancho de banda.

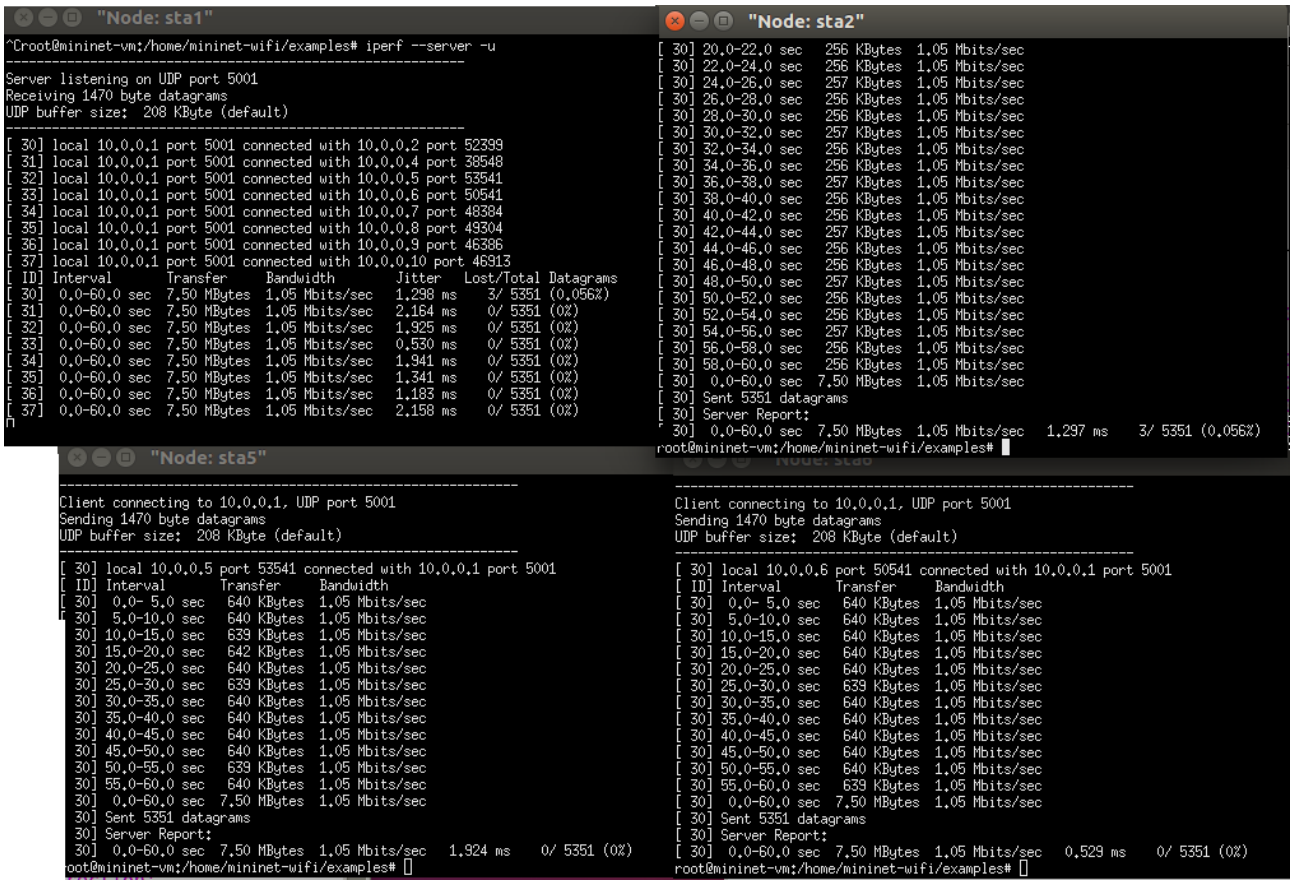

Imagen 11- Iperf UDP durante 60 segundos en intervalso de 5 segundos

2.3. Wireshark

Para trabajar y conocer internamente el funcionamiento de la red SDN dentro de Mininet-wifi hemos utilizado Wireshark una aplicación capaz de capturar los paquetes emitidos en la red creada con Mininet-wifi.

Para este análisis se ha modificado “CruceDeEstaciones” para obtener todas las capturas diferentes de Wireshark. Existe una interfaz en Mininet-wifi que retransmite todos los paquetes que navegan de cualquier forma dentro de la red creada (hwsim0), esta interfaz por defecto esta desconectada por lo que hay que activarla antes de comenzar con las capturas que realiza Wireshark.

La nueva versión de la red coloca sta2 en las coordenadas ’65,50,0’ y a sta1 se le crea una orden de movimiento por lo que se desplaza desde ‘0,50,0’ hasta ’80,50,0’. De esta forma se podrá apreciar toda la comunicación desde que se inicia la simulación.

2.3.1. Puerto hwsim0

Primero realizamos la simulación capturando el puerto hwsim0 del que ya hemos hablado, aquí podemos observar la red como si de una real se tratara, sin mayor diferencia (Imagen 12).

Al principio sta1 está fuera del rango del punto de acceso por lo que lo único que captamos son los "*Beacon*" que produce ap1 para que otros dispositivos lo encuentren. Entonces sta1 entra en el rango de ap1 y detecta los *beacon* tras lo que inicia los pasos necesarios para poder asociarse a él (Imagen 12-13).

En este punto podemos encontrar el *request* solicitado por sta1 y el *response* de ap1, donde sta1 informa de los parámetros con los que puede trabajar en la conexión y el SSID al que se quiere asociar, en caso de que hubiera más puntos de acceso al alcance.

Una vez ambos dispositivos tienen conocimiento de la existencia del otro comienza el proceso de conexión, sta1 se autentifica y solicita asociarse a ap1 (Imagen 14-15). Luego ap1 responde (Imagen 16) y establece la conexión con sta1. A partir de este punto sta1 está conectada a la estación y puede comunicarse normalmente con sta2 como veremos a continuación.

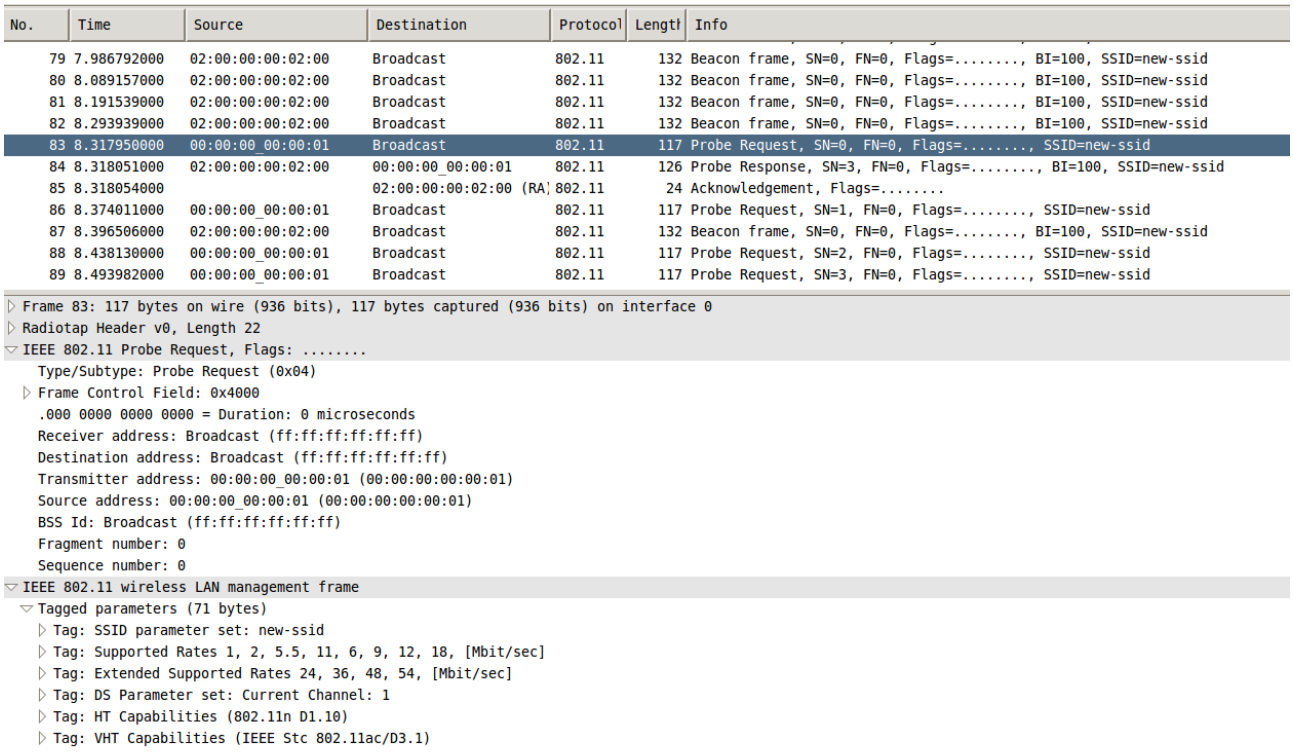

Imagen 12- (hwsim0) Captura del inicio de la simulacion Parte 1

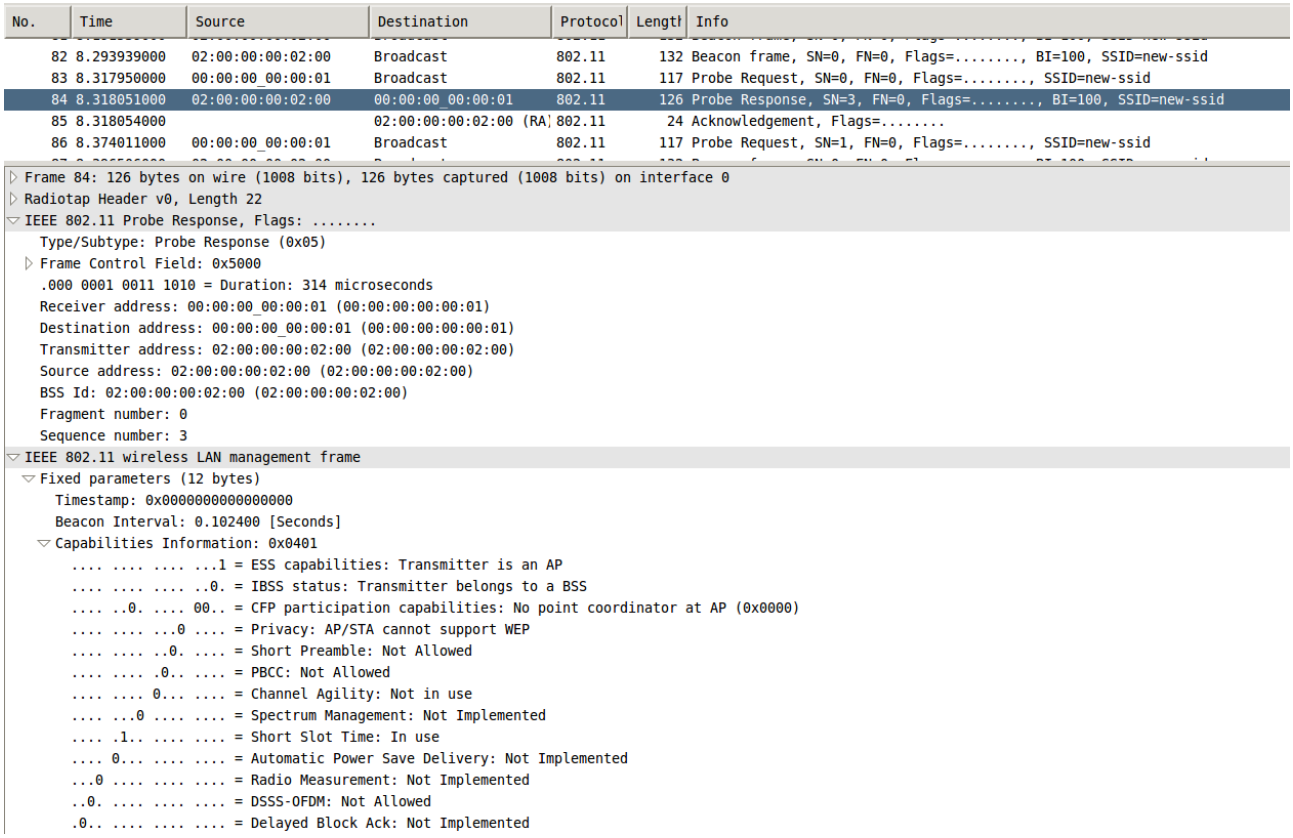

Imagen 13- (hwsim0) Captura de la autentificacion Parte 2

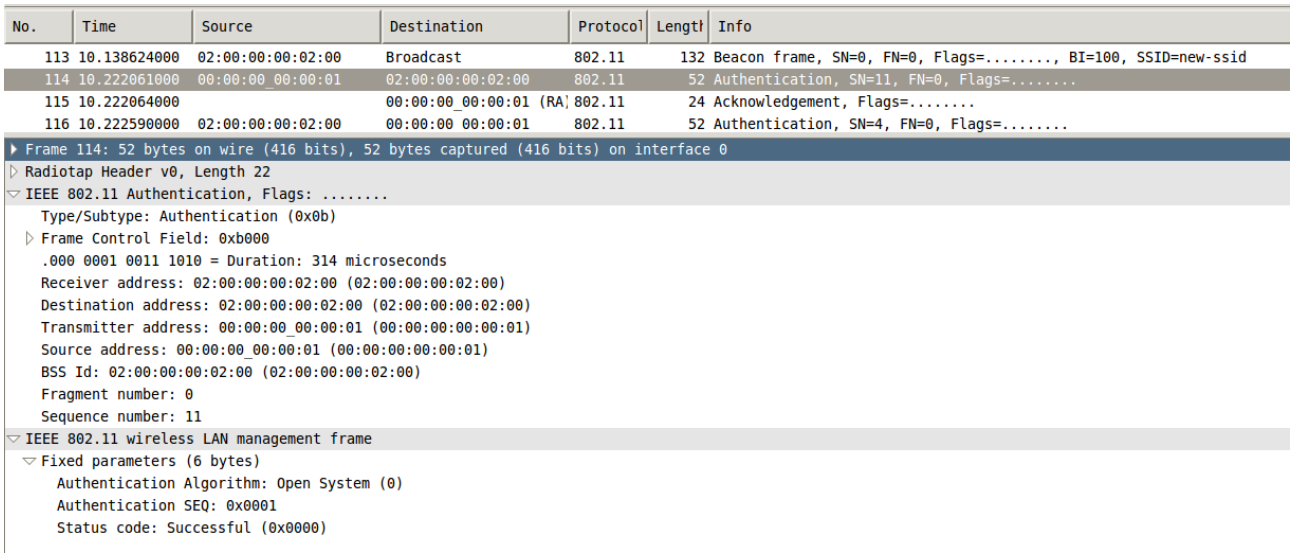

Imagen 14- (hwsim0) Captura de la autentificación Parte 1

```
No.  Time          Source              Destination         Protocol Length Info
 116 10.222590000  02:00:00:00:02:00   00:00:00_00:00:01   802.11       52 Authentication, SN=4, FN=0, Flags=........
 117 10.222593000                      02:00:00:00:02:00 (RA 802.11       24 Acknowledgement, Flags=........
 118 10.226320000  00:00:00_00:00:01   02:00:00:00:02:00   802.11       85 Association Request, SN=12, FN=0, Flags=........, SSID=new-ssid

▽ IEEE 802.11 Association Request, Flags: ........
    Type/Subtype: Association Request (0x00)
  ▷ Frame Control Field: 0x0000
    .000 0001 0011 1010 = Duration: 314 microseconds
    Receiver address: 02:00:00:00:02:00 (02:00:00:00:02:00)
    Destination address: 02:00:00:00:02:00 (02:00:00:00:02:00)
    Transmitter address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Source address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    BSS Id: 02:00:00:00:02:00 (02:00:00:00:02:00)
    Fragment number: 0
    Sequence number: 12
▽ IEEE 802.11 wireless LAN management frame
  ▽ Fixed parameters (4 bytes)
    ▽ Capabilities Information: 0x0421
        .... .... .... ...1 = ESS capabilities: Transmitter is an AP
        .... .... .... ..0. = IBSS status: Transmitter belongs to a BSS
        .... ..0. .... 00.. = CFP participation capabilities: No point coordinator at AP (0x0000)
        .... .... ...0 .... = Privacy: AP/STA cannot support WEP
        .... .... ..1. .... = Short Preamble: Allowed
        .... .... .0.. .... = PBCC: Not Allowed
        .... .... 0... .... = Channel Agility: Not in use
        .... ...0 .... .... = Spectrum Management: Not Implemented
        .... .1.. .... .... = Short Slot Time: In use
        .... 0... .... .... = Automatic Power Save Delivery: Not Implemented
        ...0 .... .... .... = Radio Measurement: Not Implemented
        ..0. .... .... .... = DSSS-OFDM: Not Allowed
        .0.. .... .... .... = Delayed Block Ack: Not Implemented
        0... .... .... .... = Immediate Block Ack: Not Implemented
      Listen Interval: 0x0005
  ▽ Tagged parameters (35 bytes)
    ▷ Tag: SSID parameter set: new-ssid
    ▷ Tag: Supported Rates 1, 2, 5.5, 11, 6, 9, 12, 18, [Mbit/sec]
    ▷ Tag: Extended Supported Rates 24, 36, 48, 54, [Mbit/sec]
    ▷ Tag: Vendor Specific: Microsof: WMM/WME: Information Element
```

Imagen 15- (hwsim0) Captura de la autentificación Parte 2

```
No.  Time          Source              Destination         Protocol Length Info
 119 10.226322000                      00:00:00_00:00:01 (RA 802.11       24 Acknowledgement, Flags=........
 120 10.227177000  02:00:00:00:02:00   00:00:00_00:00:01   802.11      104 Association Response, SN=5, FN=0, Flags=........

▽ IEEE 802.11 Association Response, Flags: ........
    Type/Subtype: Association Response (0x01)
  ▷ Frame Control Field: 0x1000
    .000 0001 0011 1010 = Duration: 314 microseconds
    Receiver address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Destination address: 00:00:00_00:00:01 (00:00:00:00:00:01)
    Transmitter address: 02:00:00:00:02:00 (02:00:00:00:02:00)
    Source address: 02:00:00:00:02:00 (02:00:00:00:02:00)
    BSS Id: 02:00:00:00:02:00 (02:00:00:00:02:00)
    Fragment number: 0
    Sequence number: 5
▽ IEEE 802.11 wireless LAN management frame
  ▽ Fixed parameters (6 bytes)
    ▽ Capabilities Information: 0x0401
        .... .... .... ...1 = ESS capabilities: Transmitter is an AP
        .... .... .... ..0. = IBSS status: Transmitter belongs to a BSS
        .... ..0. .... 00.. = CFP participation capabilities: No point coordinator at AP (0x0000)
        .... .... ...0 .... = Privacy: AP/STA cannot support WEP
        .... .... ..0. .... = Short Preamble: Not Allowed
        .... .... .0.. .... = PBCC: Not Allowed
        .... .... 0... .... = Channel Agility: Not in use
        .... ...0 .... .... = Spectrum Management: Not Implemented
        .... .1.. .... .... = Short Slot Time: In use
        .... 0... .... .... = Automatic Power Save Delivery: Not Implemented
        ...0 .... .... .... = Radio Measurement: Not Implemented
        ..0. .... .... .... = DSSS-OFDM: Not Allowed
        .0.. .... .... .... = Delayed Block Ack: Not Implemented
        0... .... .... .... = Immediate Block Ack: Not Implemented
      Status code: Successful (0x0000)
      ..00 0000 0000 0010 = Association ID: 0x0002
  ▽ Tagged parameters (52 bytes)
    ▷ Tag: Supported Rates 1(B), 2(B), 5.5(B), 11(B), 6, 9, 12, 18, [Mbit/sec]
    ▷ Tag: Extended Supported Rates 24, 36, 48, 54, [Mbit/sec]
    ▷ Tag: Extended Capabilities (8 octets)
```

Imagen 16- (hwsim0) Captura de la autentificación Parte 3

Ahora vamos a realizar un ping entre sta1 y sta2 para comprobar que la conexión es correcta. Podemos observar como ver como el ping funciona normalmente (Imagen 17) y los paquetes circulan sin problema.

Es importante fijarse que también aparecen paquetes ARP para localizar a que mascara pertenece la IP al igual que cuando un dispositivo se conecta a una red nueva y no tiene ningún tipo de información de la red en sus tablas. Y el iperf que pasa por ap1 llega a sta2 (Imagen 18) generando los paquetes ACK de respuesta.

Es aquí cuando nos podemos fijar que Mininet-wifi replica en detalle el comportamiento de los dispositivos que emula, donde se puede ejecutar cualquier aplicación disponible en la maquina emulada y se comporta como tal en la red.

Finalmente, sta1 ha seguido desplazándose hacia el límite del rango de ap1 y cuando llega a este sta1 es quien inicia la desconexión de ap1 (Imagen 19).

A continuación, vamos a realizar el mismo experimento, pero observándolo desde otro punto, desde la interfaz de “lo”, lo que nos permitirá seguir el funcionamiento de OpenFlow por lo que usaremos el filtro de Wireshark para observar solo este protocolo y no los paquetes que la maquina envié por “lo”. Seguiremos el progreso de la simulación desde que se inicia hasta que sta1 entra y sale del alcance de ap1, realizando el ping mientras se encuentra conectado a él.

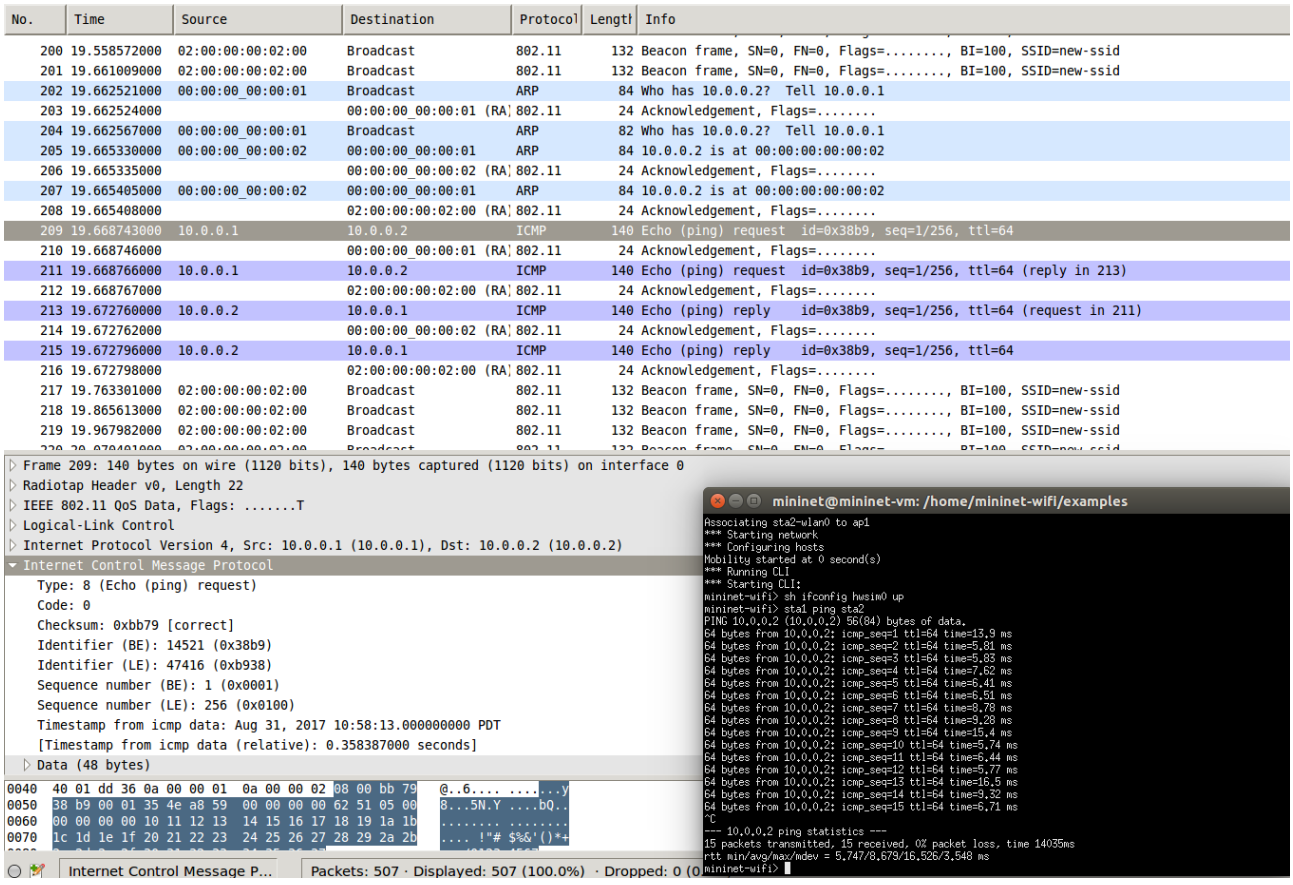

Imagen 17- (hwsim0) Captura de Ping

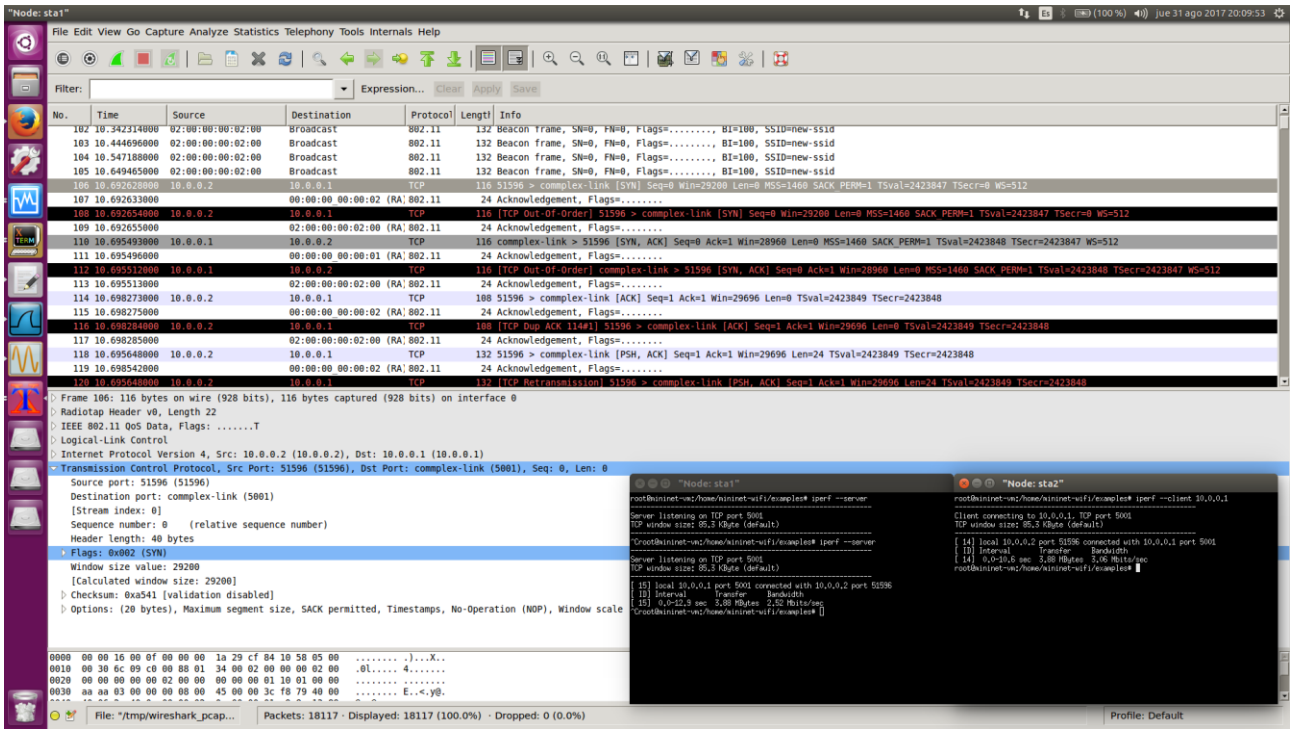

Imagen 18- (hwsim0) Captura de Iperf

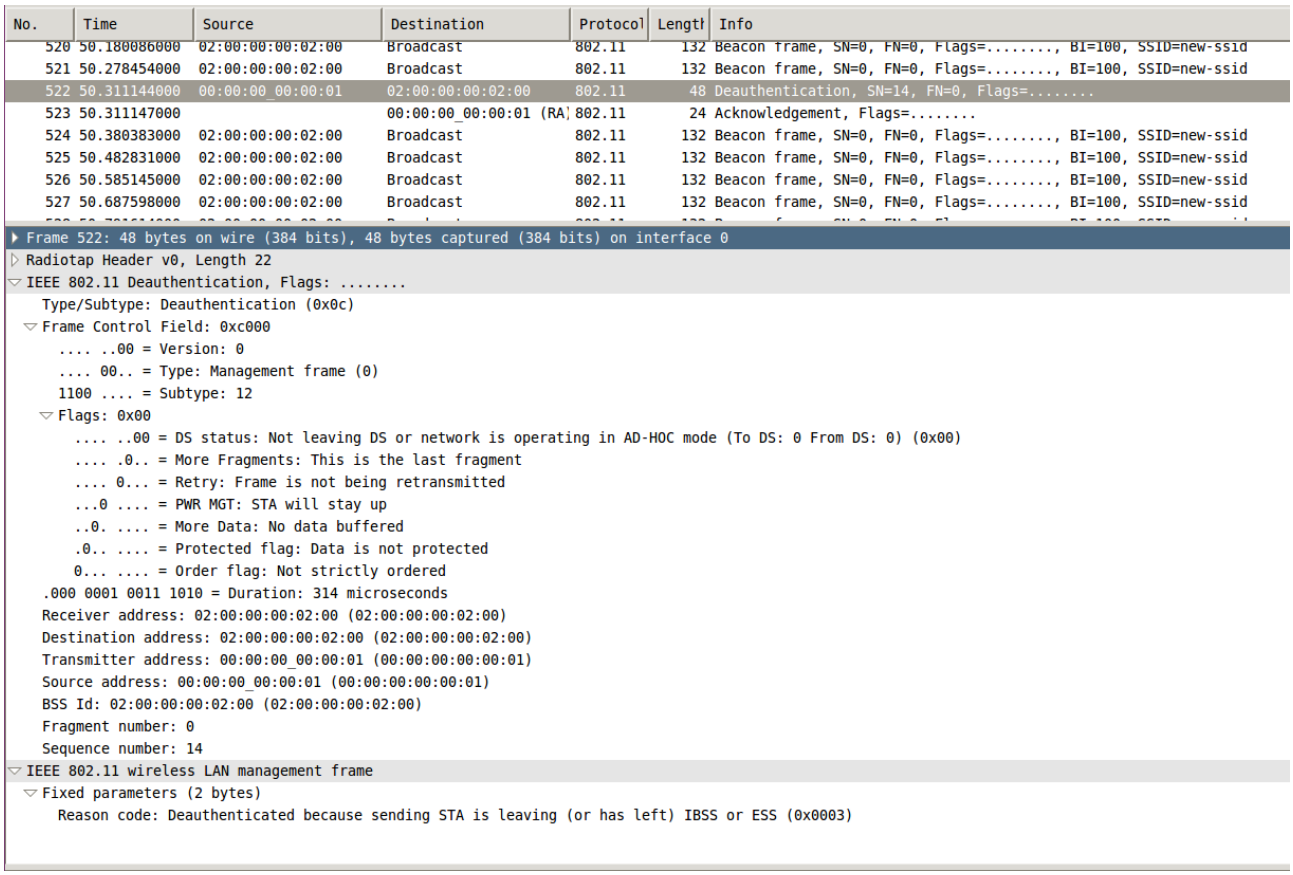

Imagen 19- (hwsim0) Captura de desconexión

2.3.2. Puerto lo

Inmediatamente nada más empezar la simulación el controlador de OpenFlow comienza por establecer una conexión con los dispositivos pertinentes informando con "*Hello*" la última versión con la que puede trabajar, de igual manera deben responder estos dispositivos. Estos una vez conocida la existencia del controlador realizan una solicitud de las características del dispositivo con "*of_features_request*", el controlador prepara una configuración para controlarlo y le responde "*of_features_reply*" (Imagen 20). Es importante fijarse que el puerto destino ya que indica que es el protocolo OpenFlow el destinatario.

A continuación, se inicia el ping desde sta1 a sta2, OpenFlow también genera paquetes para gestionar la nueva información en la red. Cuando esto ocurre el dispositivo que recibe el primer paquete de este tipo envía un mensaje al controlador "*of_packet_in*" al que contesta con "*of_packet_out*", con este intercambio de paquetes el dispositivo envía la información sobre el paquete que debe retransmitir y el controlador le responde con la información necesaria para poder gestionar la transmisión. (Imagen 22 y 23)

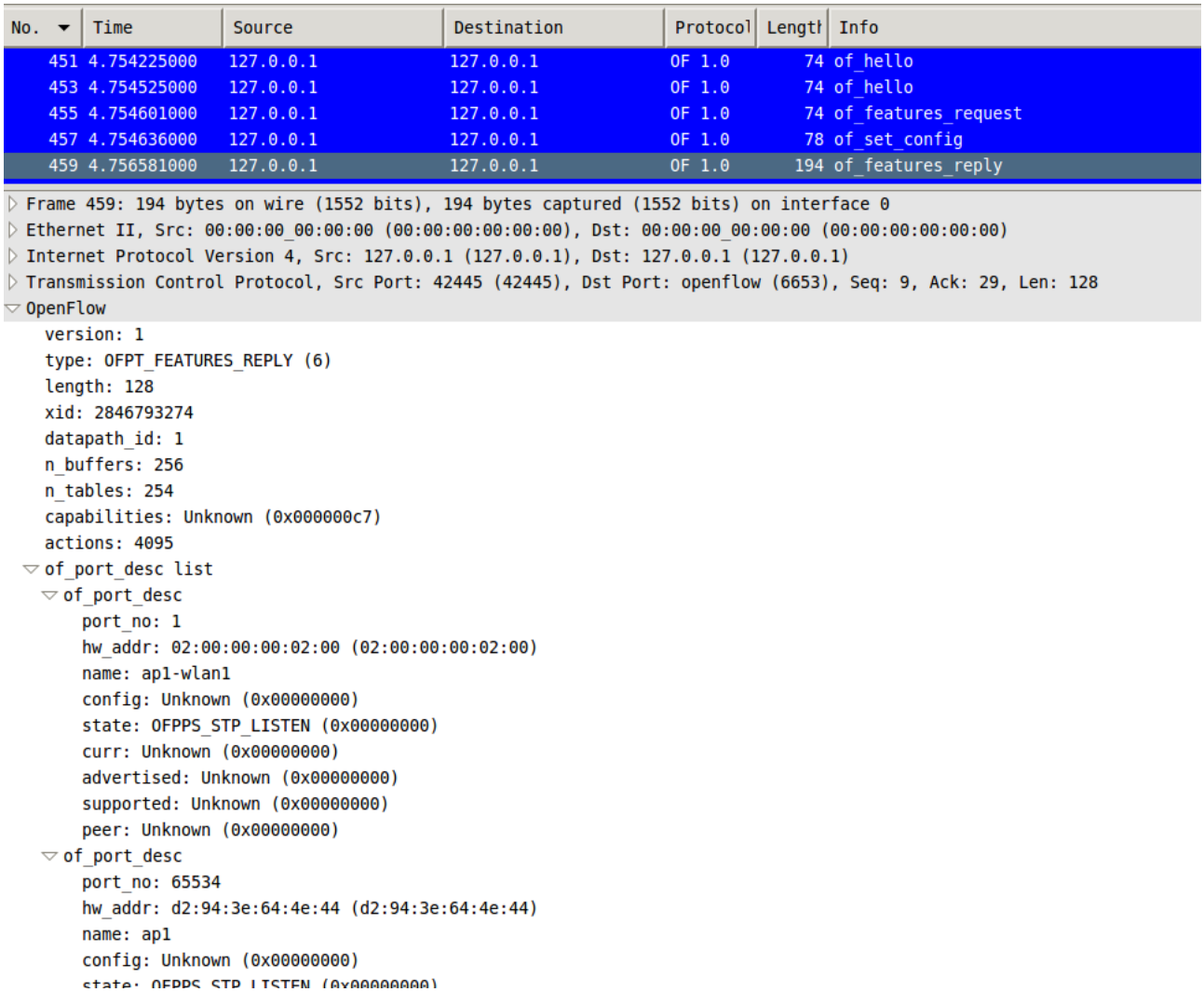

Imagen 20- (lo) Captura del inicio de la simulación

También se puede reconocer fácilmente el inicio del ping en la captura de la red debido a que Wireshark lo remarca de un color azul más claro. También se pueden observar varios pares de paquetes "*of_echo_reply*" y "*of_echo_request*" OpenFlow utiliza estos paquetes para intercambiar información sobre el estado de la red y las conexiones, dentro se puede encontrar información como la latencia o ancho de banda (Imagen 24).

No.	Time	Source	Destination	Protocol	Length	Info
451	4.754225000	127.0.0.1	127.0.0.1	OF 1.0	74	of_hello
453	4.754525000	127.0.0.1	127.0.0.1	OF 1.0	74	of_hello
455	4.754601000	127.0.0.1	127.0.0.1	OF 1.0	74	of_features_request
457	4.754636000	127.0.0.1	127.0.0.1	OF 1.0	78	of_set_config
459	4.756581000	127.0.0.1	127.0.0.1	OF 1.0	194	of_features_reply
1413	6.488179000	00:00:00_00:00:01	Broadcast	OF 1.0	104	of_packet_in
1415	6.488647000	127.0.0.1	127.0.0.1	OF 1.0	90	of_packet_out
1417	6.591902000	00:00:00_00:00:02	Broadcast	OF 1.0	104	of_packet_in
1418	6.592388000	127.0.0.1	127.0.0.1	OF 1.0	90	of_packet_out

```
▷ Frame 1413: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
▷ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▷ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▷ Transmission Control Protocol, Src Port: 42445 (42445), Dst Port: openflow (6653), Seq: 137, Ack: 29, Len: 38
▽ OpenFlow
    version: 1
    type: OFPT_PACKET_IN (10)
    length: 38
    xid: 0
    buffer_id: 256
    total_len: 20
    in_port: 1
    reason: OFPR_NO_MATCH (0)
  ▽ Ethernet packet
      ▷ IEEE 802.3 Ethernet
      ▷ Logical-Link Control
      ▷ Logical-Link Control Basic Format XID
```

Imagen 21- (lo) Captura de llegada de un paquete a la red Parte 1

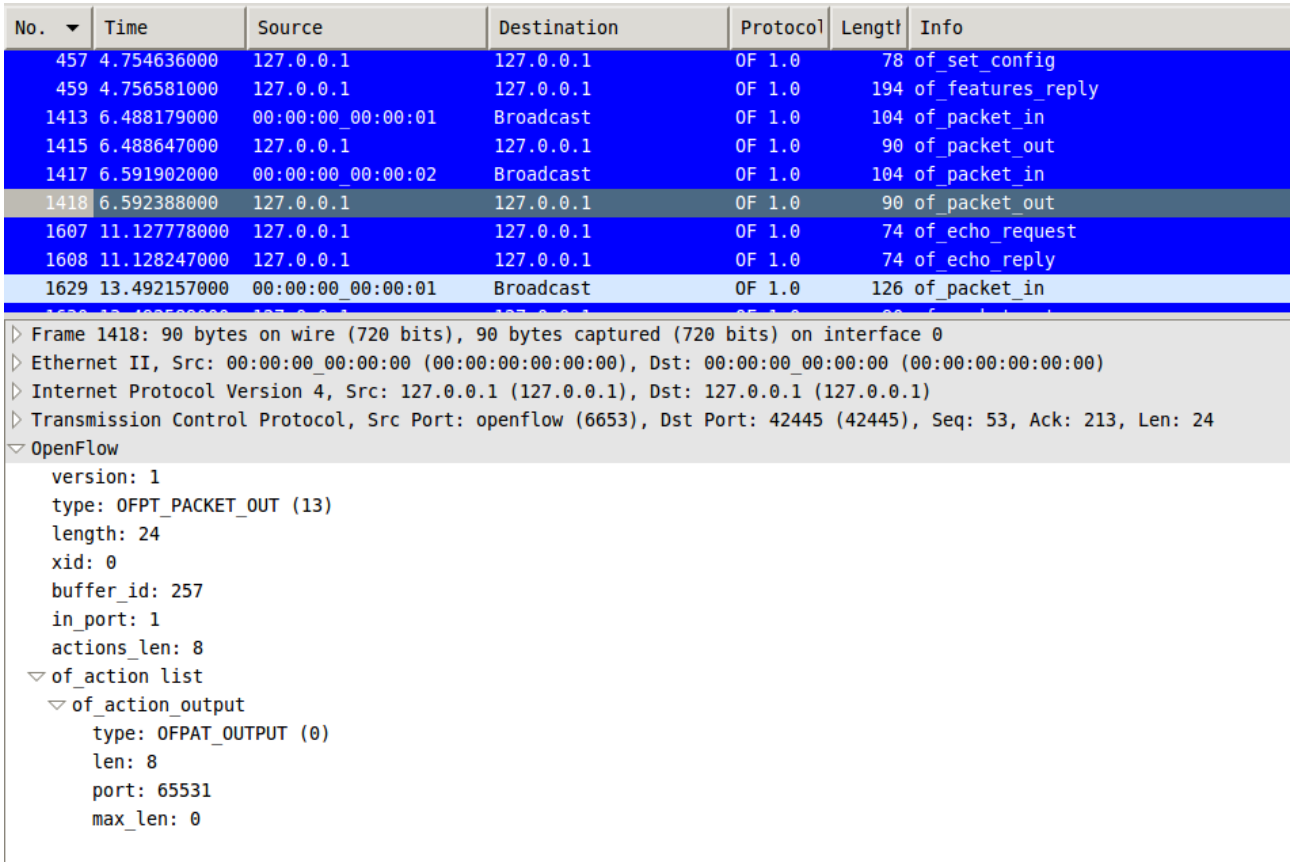

Imagen 22- (lo) Captura de llegada de un paquete a la red Parte 2

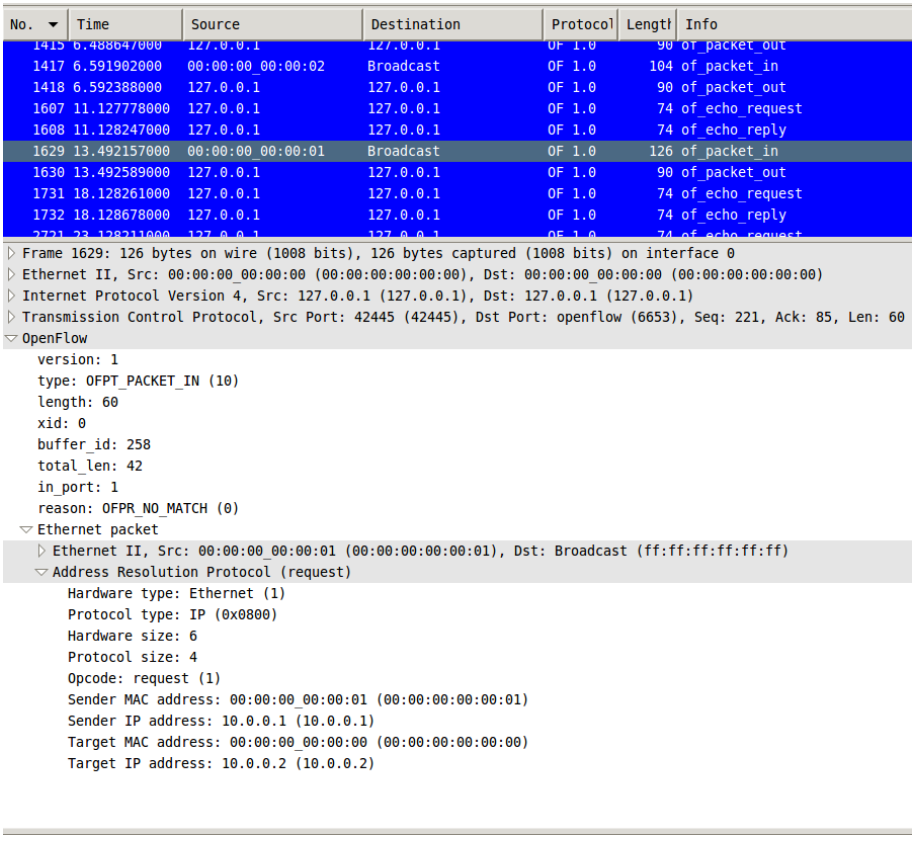

Imagen 23- (lo) Captura de paquete ARP

No.	Time	Source	Destination	Protocol	Length	Info
451	4.754225000	127.0.0.1	127.0.0.1	OF 1.0	74	of_hello
453	4.754525000	127.0.0.1	127.0.0.1	OF 1.0	74	of_hello
455	4.754601000	127.0.0.1	127.0.0.1	OF 1.0	74	of_features_request
457	4.754636000	127.0.0.1	127.0.0.1	OF 1.0	78	of_set_config
459	4.756581000	127.0.0.1	127.0.0.1	OF 1.0	194	of_features_reply
1413	6.488179000	00:00:00_00:00:01	Broadcast	OF 1.0	104	of_packet_in
1415	6.488647000	127.0.0.1	127.0.0.1	OF 1.0	90	of_packet_out
1417	6.591902000	00:00:00_00:00:02	Broadcast	OF 1.0	104	of_packet_in
1418	6.592388000	127.0.0.1	127.0.0.1	OF 1.0	90	of_packet_out
1607	11.127778000	127.0.0.1	127.0.0.1	OF 1.0	74	of_echo_request
1608	11.128247000	127.0.0.1	127.0.0.1	OF 1.0	74	of_echo_reply
1629	13.492157000	00:00:00_00:00:01	Broadcast	OF 1.0	126	of_packet_in
1630	13.492589000	127.0.0.1	127.0.0.1	OF 1.0	90	of_packet_out
1731	18.128261000	127.0.0.1	127.0.0.1	OF 1.0	74	of_echo_request
1732	18.128678000	127.0.0.1	127.0.0.1	OF 1.0	74	of_echo_reply
2721	23.128211000	127.0.0.1	127.0.0.1	OF 1.0	74	of_echo_request
2722	23.128643000	127.0.0.1	127.0.0.1	OF 1.0	74	of_echo_reply

```
▷ Frame 1732: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▷ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▷ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
▷ Transmission Control Protocol, Src Port: openflow (6653), Dst Port: 42445 (42445), Seq: 109, Ack: 289, Len: 8
▽ OpenFlow
    version: 1
    type: OFPT_ECHO_REPLY (3)
    length: 8
    xid: 0
```

Imagen 24- (lo) Captura de paquete IP

2.4. Fábrica

En la última simulación realizada he realizado una prueba rápida de iperf y ping, en la que observar al mismo tiempo los datos que estos dan y los paquetes generados en Wireshark.

Las estaciones 1 a 4 realizan una conexión iperf a sta5, donde se ve perfectamente al controlador recibiendo el aviso de la nueva trama generada en la red y que como se ha explicado en la simulación anterior, remarcando de un color azul claro el inicio de la transmisión generada por el comando iperf. (Imagen 25)

Por otro lado, también se ha realizado un ping desde sta2, sta3 y sta4 a sta1. Aquí se pueden observar las perturbaciones que sufre la red al aumentar la distancia. Mientras que sta2 presenta una oscilación en el tiempo pequeña (Imagen 27) de unas 2ms, sta3 presenta oscilaciones mucho mayores (Imagen 26), de hasta 20ms. También se hace notar que en sta2 presenta un tiempo alrededor de las 5ms y sta 3 de las 8ms.

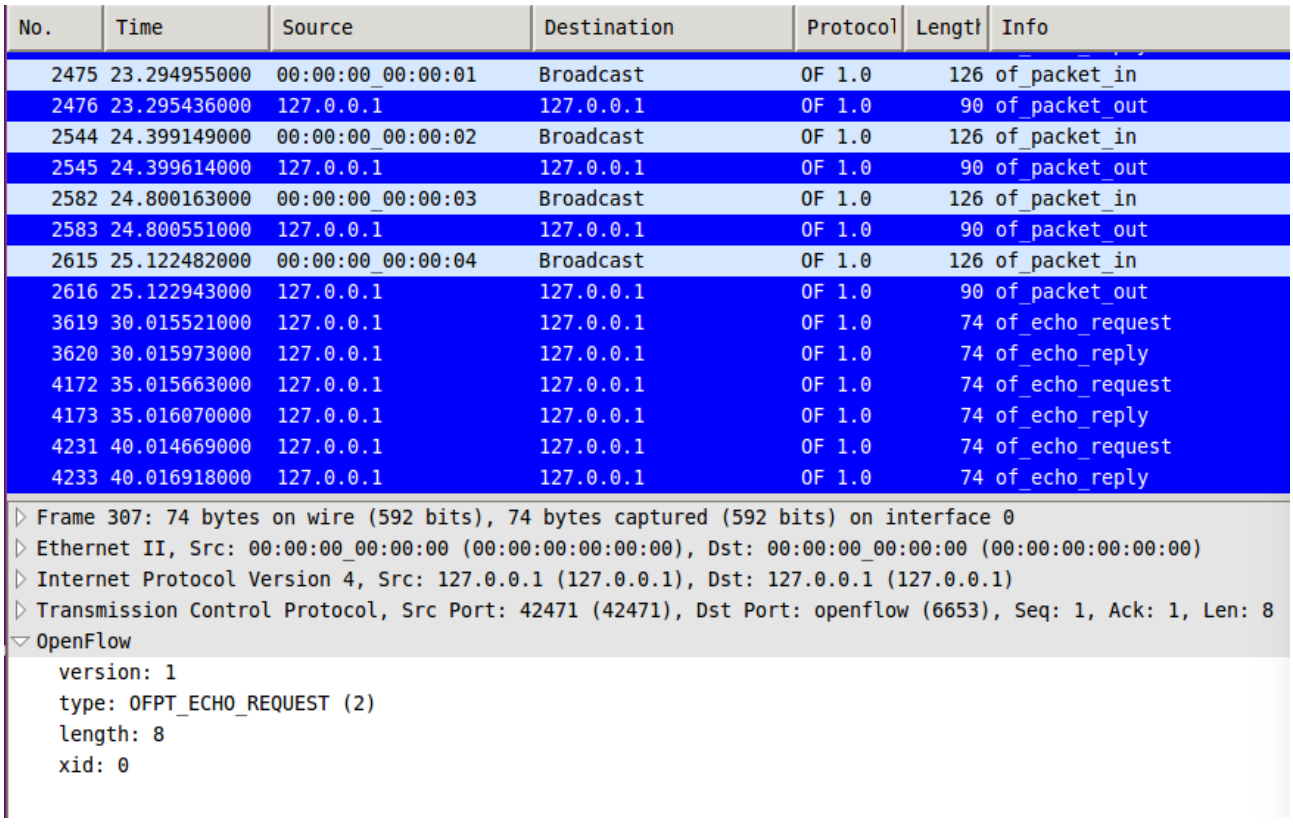

Imagen 25- (lo) Captura de varios paquete IP en Fabrica.py

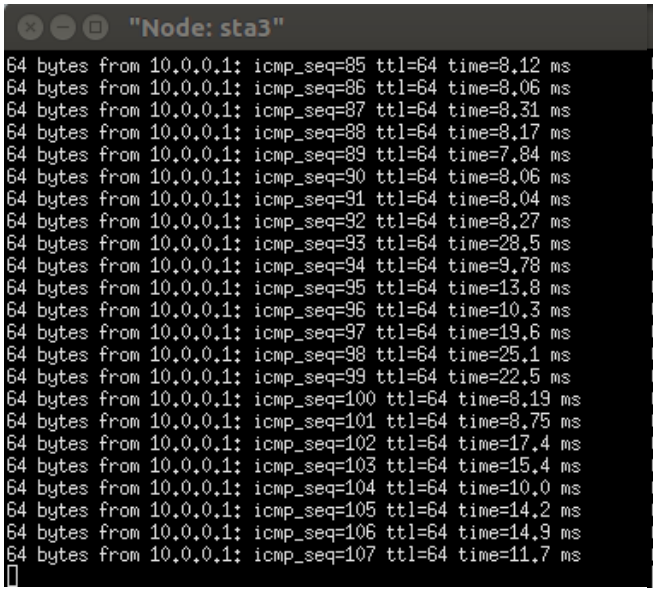

Imagen 26- Ping sta3 a sta1 en Fabrica

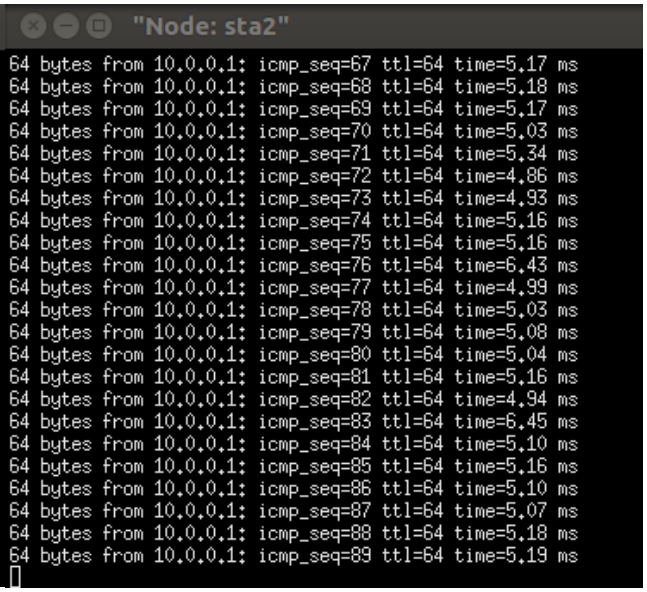

Imagen 27- Ping en sta2 a sta1 Fabrica

3. Conclusión

3.1. Análisis del proyecto

Al comienzo de este proyecto se partió con dos objetivos. El primero entender que eran y cuál era la función de las redes SDN, que las diferenciaba y porque suponían una ventaja sobre las arquitecturas de red actuales. El segundo el uso de la aplicación de Mininet-wifi para el diseño y evaluación de redes SDN emuladas en él.

El primero de los objetivos se ha cumplido durante el estudio previo, los cambios en las necesidades de las redes, es el más teórico y la base del proyecto que se ha llevado a cabo gracias a los *papers* publicados y diferentes páginas de información y noticias con secciones especialidades en SDN.

En cuanto al uso de Mininet-wifi no se ha explotado al máximo las posibilidades de esta aplicación, aunque es algo fuera del alcance de este proyecto debido a que se trata de un simulador destinado a la investigación, donde cada pocos meses se le incorpora utilidades nuevas o se proponen nuevas ideas para experimentar con él.

Como consecuencia de lo expuesto, es difícil explotar toda la capacidad de Mininet-wifi, pero pese a ello se ha cumplido el objetivo de diseñar y evaluar redes creadas con Mininet-wifi para observar, comprender y entender mejor el funcionamiento de las redes SDN.

3.2. Posibles mejoras y trabajos futuros

Hay una variedad de opciones para mejorar o ampliar este proyecto, entre las más sencillas podríamos encontrar la de incorporar Mininet a los diseños y evaluaciones de las simulaciones realizadas o incrementar el tamaño de las redes y el alcance de los puntos de acceso para comprobar cómo responde el simulador a redes con mayor carga.

También se podría experimentar aplicaciones de streaming dentro de las simulaciones o incorporar servidores RADIOUS como se ha propuesto recientemente los desarrolladores de Mininet-wifi además de la sección de redes *ad-hoc* que ya incorpora Mininet-wifi.

Pero sobre todo las posibilidades que nos ofrezca Mininet-wifi en el futuro, junto a la evolución de las redes SDN en los próximos años.

4. ANEXO 1

Las simulaciones realizadas en este proyecto se realizaron en la máquina virtual que proporciona Mininet al que además se le añadieron los conectores de red y aumento la potencia asignada la máquina virtual. Las especificaciones completas se encuentran en la siguiente captura.

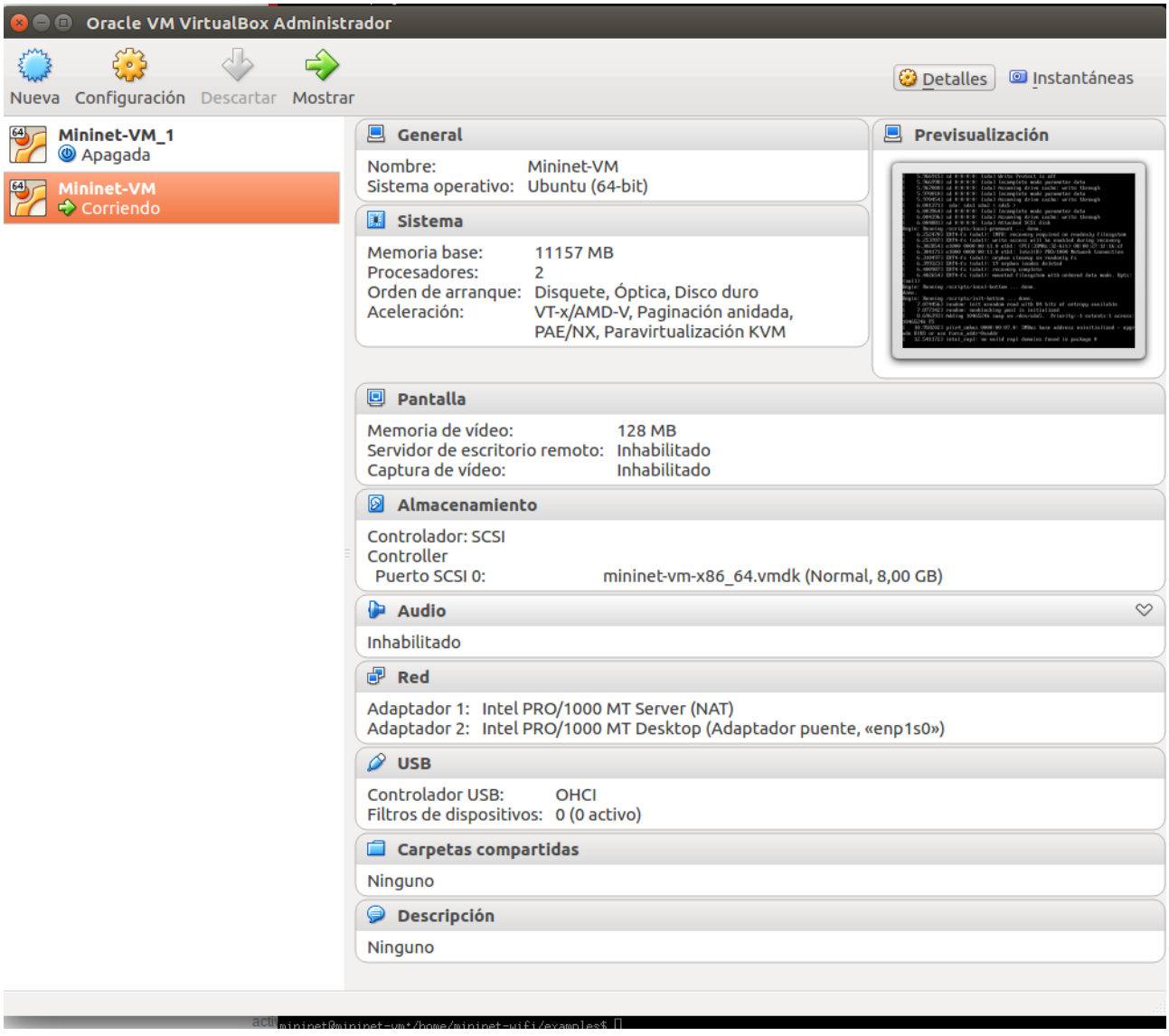

Imagen 13- Características de la maquina virtual

5. ANEXO 2

CRUCE DE ESTACIONES

```
#!/usr/bin/python

'Setting the position of Nodes and providing mobility using mobility models'

from mininet.net import Mininet
from mininet.node import Controller, OVSKernelAP
from mininet.link import TCLink
from mininet.cli import CLI
from mininet.log import setLogLevel

def topology():

    "Create a network."
    net = Mininet(controller=Controller, link=TCLink, accessPoint=OVSKernelAP)

    print "*** Creating nodes"
    sta1 = net.addStation('sta1', mac='00:00:00:00:00:01', ip='10.0.0.1/8',
position='50,50,0')
    sta2 = net.addStation('sta2', mac='00:00:00:00:00:02', ip='10.0.0.2/8',
position='50,50,0')


    ap1 = net.addAccessPoint('ap1', ssid='new-ssid', mode='g', channel='1',
position='50,50,0',range=30)
    c1 = net.addController('c1', controller=Controller)

    print "*** Configuring wifi nodes"
    net.configureWifiNodes()

    print "*** Starting network"
    net.build()
    c1.start()
    ap1.start([c1])

    net.plotGraph(max_x=100, max_y=100)

    """Seed"""
    net.seed(20)
```

```
    net.startMobility(time=0)
    net.mobility(sta1, 'start', time=1, position='50,50,0')
    net.mobility(sta1, 'stop', time=59, position='20,50,0')
    net.mobility(sta2, 'start', time=1, position='50,50,0')
    net.mobility(sta2, 'stop', time=59, position='80,50,0')
    net.stopMobility(time=60)


    print "*** Running CLI"
    CLI(net)

    print "*** Stopping network"
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology()
```

---

6. ANEXO 3

MOBILIDAD MESH

```
#!/usr/bin/python

'Setting the position of Nodes and providing mobility using mobility models'

from mininet.net import Mininet
from mininet.node import Controller, OVSKernelAP
from mininet.link import TCLink
from mininet.cli import CLI
from mininet.log import setLogLevel

def topology():

    "Create a network."
    net = Mininet(controller=Controller, link=TCLink, accessPoint=OVSKernelAP)

    print "*** Creating nodes"
    sta1 = net.addStation('sta1', mac='00:00:00:00:00:01', ip='10.0.0.1/8',
min_x=30,max_x=30, min_y=21, max_y=21, position='30,21,0')
    sta2 = net.addStation('sta2', mac='00:00:00:00:00:02', ip='10.0.0.2/8')
    sta3 = net.addStation('sta3', mac='00:00:00:00:00:03', ip='10.0.0.3/8')
    sta4 = net.addStation('sta4', mac='00:00:00:00:00:04', ip='10.0.0.4/8')
    sta5 = net.addStation('sta5', mac='00:00:00:00:00:05', ip='10.0.0.5/8')
    sta6 = net.addStation('sta6', mac='00:00:00:00:00:06', ip='10.0.0.6/8')
    sta7 = net.addStation('sta7', mac='00:00:00:00:00:07', ip='10.0.0.7/8')
    sta8 = net.addStation('sta8', mac='00:00:00:00:00:08', ip='10.0.0.8/8')
    sta9 = net.addStation('sta9', mac='00:00:00:00:00:09', ip='10.0.0.9/8')
    sta10 = net.addStation('sta10', mac='00:00:00:00:00:10', ip='10.0.0.10/8')



    ap1 = net.addAccessPoint('ap1', ssid='new-ssid', mode='g', channel='1',
position='21,21,0',range=30)
    c1 = net.addController('c1', controller=Controller)

    print "*** Configuring wifi nodes"
    net.configureWifiNodes()

    net.associationControl('ssf')
```

```
    print "*** Starting network"
    net.build()
    c1.start()
    ap1.start([c1])

      net.plotGraph(max_x=100, max_y=100)

    """Seed"""
    net.seed(20)


    net.propagationModel('friisPropagationLossModel', sL=2)

    "*** Available models: RandomWalk, TruncatedLevyWalk, RandomDirection,
RandomWayPoint, GaussMarkov, ReferencePoint, TimeVariantCommunity ***"
    net.startMobility(time=0, model='RandomDirection', max_x=42, max_y=42,
min_v=0.5, max_v=0.8)

    print "*** Running CLI"
    CLI(net)

    print "*** Stopping network"
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology()
```

7. ANEXO 4

FABRICA

```
#!/usr/bin/python

'Setting the position of Nodes and providing mobility using mobility models'

from mininet.net import Mininet
from mininet.node import Controller, OVSKernelAP
from mininet.link import TCLink
from mininet.cli import CLI
from mininet.log import setLogLevel

def topology():

    "Create a network."
    net = Mininet(controller=Controller, link=TCLink, accessPoint=OVSKernelAP)

    print "*** Creating nodes"
    sta1 = net.addStation('sta1', mac='00:00:00:00:00:01', ip='10.0.0.1/8',
position='40,45,0')
    sta2 = net.addStation('sta2', mac='00:00:00:00:00:02', ip='10.0.0.2/8',
position='40,55,0')
    sta3 = net.addStation('sta3', mac='00:00:00:00:00:03', ip='10.0.0.3/8',
position='20,45,0')
    sta4 = net.addStation('sta4', mac='00:00:00:00:00:04', ip='10.0.0.4/8',
position='20,55,0')

sta5 = net.addStation('sta5', mac='00:00:00:00:00:05', ip='10.0.0.5/8',
position='55,50,20')


    ap1 = net.addAccessPoint('ap1', ssid='new-ssid', mode='g', channel='1',
position='50,50,0',range=50)
    c1 = net.addController('c1', controller=Controller)

    print "*** Configuring wifi nodes"
    net.configureWifiNodes()


    print "*** Starting network"
    net.build()
    c1.start()
```

```
    ap1.start([c1])

    net.plotGraph(max_x=100, max_y=100)


    print "*** Running CLI"
    CLI(net)

    print "*** Stopping network"
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    topology()
```

8. Bibliografía

[1] Manual de Mininet-wifi.

https://github.com/intrig-unicamp/mininet-wifi

[2] Mailing list de Mininet-wifi.

https://groups.google.com/forum/#!forum/mininet-wifi-discuss

[3] “Mininet-Wifi: Emulating Software-Defined Wireless Networks” Ramon R. Fontes; Samira Afzal; Samuel H. B. Brito; Matcus A. S. Santos y Christian Esteve Rothenberg. School of Electrical and Computer Engineering (FEEC) University of Campinas (UNICAMP)

[4] “How far can we go? Towards Realistic Software-Defined Wireless Networking Experiments” Ramon Dos Reis Fontes; Mohamed Mahfoudi; Walid Dabbous, Thierry Turletti y Christian Rothenberg. University of Campinas (UNICAMP). Universite Côte d’Azur, 2017

[5] “Impact of the Gauss-Markov Mobility Model on network connectivity, lifetime and hop count of routes for mobile Ad hoc networks” Natarajan Meghanathan. Jackson State University, 2010

[6] “Emulacion de escenarios virtuales, en una SDWLAN (Software Defined Wireless local Area Network), de un campus universitario” Gerlyn Eduardo Duarte y Richard J. Lobo U., 2015

[7] Teoria SDN. Conceptos y arquitectura.

https://www.sdxcentral.com/sdn/?c_action=num_ball

[8] Teoria OpenFlow y funciones de los paquetes.

http://flowgrammable.org/sdn/