Universidad Zaragoza

Trabajo Fin de Grado

Control y gestión de redes inalámbricas en entornos virtuales distribuidos

Wireless network control and management in distributed virtual environments

Autor

Daniel Naval Alcalá

Director

Julián Fernández Navajas

ESCUELA DE INGENIERÍA Y ARQUITECTURA

Diciembre 2020

Resumen

Desde hace ya varios años las redes inalámbricas forman parte de nuestra vida diaria y no podríamos concebir sin éstas, la sociedad de la información en la que hoy vivimos y que ha revolucionado la forma en la que interactuamos entre nosotros.

Mientras los grandes operadores nos ofrecen conexión a Internet móvil mediante redes 4G o la recién estrenada 5G, hoy en día tanto en hogares, como en oficinas, colegios, universidades y lugares públicos, las redes Wi-Fi son las que predominan y lo seguirán haciendo durante muchos años; no sólo para conectar todos nuestros dispositivos móviles, sino para dar soporte a la revolución de los objetos conectados, también conocida como la industria del IoT o el "Internet de las cosas".

Con el reciente estándar Wi-Fi 6 ya existente (pero que apenas ha comenzado a alcanzarnos), y con las miras ya puestas en el futuro Wi-Fi 7 que permitirá velocidades teóricas de hasta 30 Gbps, se vaticina un interesante y prometedor futuro para esta tecnología.

No obstante, el uso de redes Wi-Fi plantea problemas de movilidad entre puntos de acceso inalámbricos, que pueden degradar considerablemente la experiencia de usuario. Para optimizar y resolver estos problemas, se han desarrollado mecanismos que ayudan a rebajar estos inconvenientes; centrándonos concretamente en el proyecto que da origen a este trabajo denominado LVAP (Light Virtual Access Points), el cual se fundamenta en redes SDWN (Software-Defined Wireless Network).

El uso masivo de las redes intensifica también la necesidad de escalar y gestionar la infraestructura de red de forma flexible y dinámica, permitiendo establecer enrutamientos de tráfico basados en las capacidades de la red, balanceando la carga entre las diferentes rutas disponibles y, en definitiva, mejorando la calidad del servicio ofrecido. Mediante la incorporación de una infraestructura SDN, se permite obtener este control total de la red y tener la capacidad de administrar todo el tráfico, siendo la solución idónea para los problemas citados anteriormente.

Este proyecto tiene como objetivo analizar múltiples alternativas para proporcionar y documentar un escenario óptimo sobre el que se pueda, de forma flexible y sencilla, portar la compleja infraestructura de puntos de acceso inalámbricos del Proyecto LVAP, así como integrar las capacidades de las redes definidas por software a la solución final.

Abstract

For several years now wireless networks have been part of our daily lives and we could not conceive without them, the information society in which we live today, and which has revolutionized the way we interact with each other.

While the big operators offer us mobile Internet connection through 4G networks or the recently released 5G, nowadays in homes, offices, schools, universities and public places, Wi-Fi networks are the ones that predominate and will continue to do so for many years; not only to connect all our mobile devices, but also to support the revolution of connected objects, also known as the IoT industry or the "Internet of things".

With the recent Wi-Fi 6 standard already in existence (but which has only just begun to reach us), and with the sight already set on the future Wi-Fi 7 which will allow theoretical speeds of up to 30 Gbps, an interesting and promising future is forecast for this technology.

However, the use of Wi-Fi networks poses problems of mobility between wireless access points, which can considerably degrade the user experience. To optimize and solve these problems, mechanisms have been developed to help reduce these disadvantages, focusing us specifically on the project that gave rise to this work called LVAP (Light Virtual Access Points), which is based on SDWN (Software-Defined Wireless Network).

The massive use of networks also intensifies the need to scale and manage the network infrastructure in a flexible and dynamic way, allowing the establishment of traffic routing based on network capacities, balancing the load between the different available routes and, in short, improving the quality of the service offered. By incorporating an SDN infrastructure, it is possible to obtain this total control of the network and have the capacity to manage all the traffic, being the ideal solution for the problems mentioned above.

This project aims to analyse multiple alternatives to provide and document an optimal scenario on which, to flexibly and easily port the complex infrastructure of wireless access points of the LVAP Project, as well as integrate the capabilities of software defined networks into the final solution.

Índice

1. Introducción 1

1.1 Motivación 1

1.2 Objetivos 2

1.2.1 Estudio de integración 2

1.2.2 Preparación de escenario 3

1.2.3 Implementación de la tecnología SD-WAN 3

1.3 Estado del Arte 3

1.3.1 Virtualización 3

1.3.2 Adaptadores de red inalámbricos 5

1.3.3 Redes definidas por software 6

2. Virtualización y soporte 8

2.1 Emulador de red 8

2.1.1 GNS3 8

2.1.2 VIRL 8

2.1.3 EVE-NG 8

2.1.4 Elección 9

2.2 Host 9

2.2.1 VMware ESXi 6.0 9

2.2.2 OpenSUSE Leap 15.1 10

2.2.3 Ubuntu Server 20.04 10

2.3 Hipervisor 11

2.3.1 KVM 11

2.3.2 Xen 12

2.3.3 Evaluación 12

2.3.4 Resumen 13

3. Estudio de sistemas operativos 15

3.1 Requisitos 15

3.2 Evaluación 16

4. Escenario e infraestructura de control 19
4.1 Arquitectura propuesta 19
4.2 Descripción del escenario 20
5. Red de datos 22
5.1 Arquitectura propuesta 22
5.2 SD-WAN con Open vSwitch 23
5.3 Metodología de filtrado 25
5.4 Descripción del escenario 28
6. Escenario final 30
6.1 Evaluación 30
7. Continuación 34
8. Conclusiones 35
9. Referencias 36
Anexo I. Preparación del Host 2
Anexo II. Test de Hipervisores 6
Anexo III. Test Debian 8
Anexo IV. Test OpenBSD 11
Anexo V. Test OpenWrt 14
Anexo VI. Test FreeBSD 17
Anexo VII. Entorno GNS3 22
Anexo VIII. Inyección de paquetes 26
Anexo IX. Reglas de flujos OpenFlow 29
Anexo X. Script generador de reglas 31
Anexo XI. Gestión del proyecto 32

Tabla de ilustraciones

Figura 1 Diagrama de funcionamiento de un *switch* OpenFlow 7
Figura 2 Arquitectura simplificada del stack seleccionado 14
Figura 3 Escenario para la red de control en GNS3 20
Figura 4 Esquema la arquitectura en la red de datos 23
Figura 5 Evaluación de reglas como un diagrama de flujo 25
Figura 6 Captura de comunicación de WhatsApp con Wireshark 27
Figura 7 Correspondencia dirección IP con ASN para WhatsApp 27
Figura 8 Escenario para la red de datos en GNS3 28
Figura 9 Agrupación de instancias y grupos por bloques en IPv4 29
Figura 10 Escenario final en GNS3 30
Figura 11 ASN origen y bloque de direcciones de Universidad de Zaragoza 31
Figura 12 Captura de tráfico desde GNS3 32
Figura 13 Captura de tráfico en enlace GRE con Wireshark 32
Figura 14 Captura de tráfico en enlace por defecto con Wireshark 33
Figura 15 Captura de tráfico en enlace crítico con Wireshark 33
Figura 16 Ciclo de vida de las instancias virtuales de GNS3 24
Figura 17 Configuración avanzada en una instancia de GNS3 24
Figura 18 Configuración general en una instancia de GNS3 25

1.Introducción

En el presente documento se expone la memoria del Trabajo Fin de Grado: "Control y gestión de redes inalámbricas en entornos virtuales distribuidos". Este proyecto ha abarcado diferentes áreas: por un lado, el campo de las redes inalámbricas y, por otro lado, el campo de las redes definidas por software o SDN; la combinación de ambas tecnologías da nombre a este proyecto.

La solución planteada se caracteriza por la fragmentación de los recursos necesarios para obtener un adecuado rendimiento permitiendo distribuir la carga de los diferentes componentes que necesitan ser virtualizados, todo ello con un control totalmente centralizado fácilmente gestionable.

Este trabajo cubre por lo tanto la evaluación de las capacidades y viabilidad de algunos de los sistemas existentes para ser utilizados en el despliegue de una infraestructura de redes Wi-Fi, así como la integración de redes definidas por software que aportan un gran valor añadido al escenario principal. Todo ello desplegado en un entorno controlado de laboratorio.

1.1 Motivación

Este trabajo tiene como origen el proyecto de investigación europeo H2020 WI-5[1], ya finalizado, que buscaba construir una arquitectura basada en un conjunto integrado y coordinado de soluciones inteligentes, capaces de reducir las interferencias entre puntos de acceso vecinos y proporcionar una conectividad optimizada para los servicios nuevos y emergentes. [1]

Entre los objetivos que este proyecto abarcaba se encuentran los siguientes:

- Apoyar el traspaso continuo para mejorar la experiencia del usuario con servicios interactivos en tiempo real.
- Desarrollar nuevos modelos de negocio para optimizar el espectro Wi-Fi disponible en zonas urbanas, espacios públicos, y oficinas.
- Integrar nuevas funcionalidades inteligentes en los puntos de acceso, para hacer frente a la congestión del espectro radioeléctrico y a la ineficiencia del uso actual, aumentando así el rendimiento mundial y logrando el ahorro de energía.

[1] https://web.archive.org/web/20190717215410/http://www.wi5.eu/

Dada la madurez y finalización del proyecto H2020 Wi-5, surge una nueva corriente de investigación derivada en el proyecto LVAP[2] (Light Virtual Access Points, o Puntos de Acceso Virtuales Ligeros) a través del grupo de investigación CeNIT (Communications Networks and Information Technologies) del Instituto Universitario de Investigación de Ingeniería de Aragón; cuya línea de investigación es la de explorar y desarrollar una solución SDWN (Software-Defined Wireless Network) para evitar algunos de los problemas que aparecen en las redes Wi-Fi domésticas, donde es habitual la instalación de más de un Punto de Acceso Wi-Fi para suplir los defectos de cobertura e interferencias de redes vecinas.

Esto suele dar lugar a un problema de movilidad, ya que es habitual que los dispositivos se queden asociados a uno de los puntos de acceso con peor cobertura y calidad de señal, pese a que se encuentre disponible, más próximo y con mayor calidad de señal, un segundo punto de acceso.

A continuación, se diseña y ejecuta un escenario idóneo destinado a ser el soporte para un entorno controlado de laboratorio, sobre el que se desplegará la solución de Puntos de Acceso Virtuales Ligeros de forma rápida, flexible y con una gestión centralizada; permitiendo evaluar su comportamiento para detectar y corregir errores. También permitirá proporcionar un empaquetado para que la infraestructura pueda ser migrada y reproducida fácilmente en un entorno real.

Además, la realización de este proyecto va a permitir realizar el despliegue de la solución íntegra, en situaciones en las que no se pueda llevar a cabo de forma física, puesto que es posible distribuir los múltiples puntos de acceso en un escenario bajo un mismo control simulando el mismo entorno real.

1.2 Objetivos

1.2.1 Estudio de integración

El soporte nativo de funciones inalámbricas en los sistemas operativos es crucial para decidir si el despliegue de determinados puntos de acceso Wi-Fi es compatible con el proyecto LVAP. Esto supone hacer uso de funciones no convencionales que los adaptadores inalámbricos de consumo pueden proporcionar a través de controladores desarrollados por el fabricante o por la comunidad. Para verificar este objetivo, es necesario realizar pruebas de funcionamiento con múltiples adaptadores

[22] https://i3a.unizar.es/es/proyectos/wireless-lan-based-use-light-virtual-wifi-access-points

Wi-Fi en todos los sistemas objeto de ser analizados. Estas pruebas serán decisivas para seleccionar o descartar sistemas que serán utilizados posteriormente en el despliegue de la solución final en base a sus capacidades.

Dado que estos sistemas se ejecutarán en un entorno virtualizado, también es conveniente analizar diferentes soluciones y modos de configuración para optimizar el uso de los recursos del sistema anfitrión, así como de las instancias virtuales.

1.2.2 Preparación de escenario

El escenario a diseñar presenta una infraestructura considerablemente compleja que no puede ser fácilmente desplegada con mecanismos tradicionales de virtualización en red. Para conseguir un entorno totalmente distribuido, fácilmente reproducible y con gran capacidad de análisis se propone el software GNS3, un emulador gráfico de red de código abierto que permite diseñar topologías de red complejas y poner en marcha escenarios virtuales.

A través de esta herramienta, se construirá el escenario de puntos de acceso virtuales ligeros, pudiéndose controlar en todo momento de forma centralizada el tráfico de red y el estado de los mismos.

1.2.3 Implementación de la tecnología SD-WAN

Una vez establecida la estructura SDWN de puntos de acceso ligeros mediante el emulador de gráfico de red, ésta se integrará en una arquitectura de SDN (Software-Defined Network) con capacidad de controlar y redistribuir el tráfico para tomar decisiones de encaminamiento a través de redes WAN (SD-WAN), consiguiendo balancear la carga y garantizar la calidad de servicio.

1.3 Estado del Arte

1.3.1 Virtualización

La virtualización es una tecnología que permite ejecutar una instancia virtual de un sistema informático en una capa de abstracción sobre el hardware real. Por tanto, se convierte en un escenario apropiado para crear el entorno controlado de laboratorio deseado.

La virtualización surgió en los esfuerzos por reducir costes a través de una mayor utilización de la capacidad de los equipos y recursos; y se ha convertido en un componente fundamental en la industria tecnológica. La virtualización ha

supuesto una revolución transformando los procesos de negocio, cambiando la dinámica del mercado y creando mercados completamente nuevos. [2]

Esta tecnología es utilizada tanto por usuarios de escritorio como por grandes empresas que alojan toda su infraestructura operativa en la nube o en centros de datos.

Los usuarios de escritorio suelen buscar ejecutar aplicaciones destinadas a un sistema operativo diferente del sistema anfitrión sin tener que reiniciar o cambiar de equipo.

En las empresas, los administradores de sistemas o arquitectos de nube hacen uso de la virtualización, para fragmentar un sistema o aplicación con grandes requisitos en componentes más pequeños que se pueden virtualizar, lo que permite administrar los recursos de manera eficiente. [3]

También permite el aislamiento, manteniendo los programas que se ejecutan dentro de una máquina virtual a salvo de los procesos que tienen lugar en otra similar y que se ejecuta en el mismo servidor físico.

Entre las ventajas que ofrece esta tecnología se destacan las siguientes [4]:

- Reducción de costes: Cuando se virtualiza un entorno, un único servidor físico se convierte en múltiples máquinas virtuales, con las que es posible ejecutar sistemas y aplicaciones diferentes e independientes entre sí. Sin embargo, un único servidor físico sin virtualización, sólo podría ejecutar un sistema operativo al mismo tiempo, así como aplicaciones con dependencia de ese sistema.

- Recuperación ante desastres: Mediante un entorno virtualizado, la recuperación de las instancias virtuales en caso de fallo de operación se realiza de una manera fácil y rápida, gracias a las herramientas de replicación, clonado y aprovisionamiento que ofrecen dichos entornos.

- Mantenimiento: Las tareas de administración de los sistemas virtualizados a la hora de aplicar actualizaciones, instalar o mantener software nuevo y realizar configuraciones replicadas se reducen.

- Eficiencia energética: La reducción del número de servidores físicos a utilizar, supone una gran reducción en consumo energético, incluyendo la refrigeración necesaria para el correcto funcionamiento de los mismos. Esto deriva en una reducción de los gastos del negocio, y de la huella de carbono generada por la empresa o el centro de datos.

1.3.2 Adaptadores de red inalámbricos

Hace años que las redes Wi-Fi se encuentran implantadas en todo lo que nos rodea. Desde 2018, más de 2.97 billones de dispositivos habilitados para utilizar tecnología Wi-Fi se fabrican a nivel mundial cada año. [5]

La primera versión del protocolo 802.11 se lanzó en 1997 y proporcionaba velocidades de enlace de hasta 2 Mbit/s, se actualizó en 1999 con el estándar 802.11b para permitir velocidades de enlace de hasta 11 Mbit/s y resultó ser muy popular. A partir de entonces, múltiples estándares han ido llegando para incrementar la velocidad y estabilidad de las conexiones inalámbricas hasta el actual 802.11ax o también conocido como Wi-Fi 6, con velocidades teóricas de enlace de entre 600 hasta 9608 Mbit/s. [6]

Para que un dispositivo pueda hacer uso de esta tecnología, necesita un adaptador o tarjeta de red inalámbrica, un módulo (con una o varias antenas) que actúa como conversor de medios entre el espectro radioeléctrico y la señal cableada.

Estos dispositivos requieren un software controlador para funcionar y dependiendo del fabricante, podrán implementar algunas funciones opcionales y características propias. Los controladores inalámbricos integrados en el núcleo de Linux desarrollados por los principales fabricantes implementan la mayoría de las funciones, mientras que el resto se encuentran todavía en desarrollo.

Los modos de funcionamiento más comunes soportados por las tarjetas de red inalámbricas son ad-hoc, infraestructura, AP y *Monitor*. Las conexiones entre pares se realizan en modo *ad-hoc*. En este modo, los dispositivos se comunican directamente entre sí sin ningún agente central.

En el modo de infraestructura, un punto de acceso (AP) es el coordinador de cada cliente. Éste es el modo de operación más habitual, puesto que es él utilizado cuando un usuario se conecta a una red inalámbrica para acceder a Internet a través de su dispositivo. En este modo genera se un punto de acceso con un solo dispositivo hardware que actúa como un punto de conexión central, al que múltiples clientes inalámbricos pueden conectarse. [7]

En el modo *Monitor*, la interfaz no se une a ninguna red, este modo se utiliza generalmente para analizar el tráfico de forma pasiva, la interfaz recibe todos los paquetes de su canal de escucha, aunque no estén destinados a ella. También puede ser utilizada para inyectar paquetes, (uno de los aspectos más interesantes para este trabajo).

La compatibilidad de estos dispositivos se ve muy limitada cuando los fabricantes no liberan el código de los controladores, y es la comunidad, la que tiene que recurrir a técnicas de ingeniería inversa para desarrollar un controlador funcional, que posteriormente pondrá a disposición de todo el mundo con una licencia de software libre. Tan sólo algunos fabricantes como *Atheros* liberan el código de sus controladores, siendo el fabricante de referencia para la incorporación de la tecnología inalámbrica en sistemas y proyectos que dependan totalmente de software libre, en un plano en el que se quedan fuera muchos fabricantes. [8]

En la actualidad sólo un reducido número de tarjetas inalámbricas soportan todos los modos de funcionamiento anteriormente descritos para el estándar 801.11ac La tarjeta más común para funcionar en este modo es la RTL8812AU ensamblada por *Realtek*, con controladores libres funcionales gracias al equipo que desarrolla *Aircrack-ng*[3], un conjunto de herramientas para evaluar la seguridad de las redes Wi-Fi. En el portal Linux Wireless[4] se documenta toda la información existente acerca de los controladores 802.11 que son desarrollados e integrados en el kernel de Linux.

1.3.3 Redes definidas por software

Las redes definidas por software o SDN representan un enfoque de arquitectura de red que permite que la red sea controlada de forma inteligente y centralizada, o gestionada usando aplicaciones de software. Esto ayuda a los administradores a gestionar toda la red de forma coherente y holística, independientemente de la tecnología de red subyacente.

Las arquitecturas de red tradicionales y sus instrumentos de gestión no se diseñaron para hacer frente a una demanda tan elástica, esto limita gravemente la capacidad para responder de manera efectiva a los requisitos de escala, rendimiento y experiencia del usuario de los entornos dinámicos actuales, o para desplegar servicios diferenciados. Las SDN han surgido como la respuesta de la industria para hacer frente a estos desafíos.

Las SDN permiten que las redes reaccionen de forma dinámica a los cambios en los patrones de uso y en la disponibilidad de los recursos de la red, consiguiendo un tiempo de respuesta menor a las solicitudes de las aplicaciones, servicios y usuarios. Además, las SDN proporcionan una separación entre las funciones del

[3] https://github.com/aircrack-ng/rtl8812au

[4] https://wireless.wiki.kernel.org/en/users/drivers

plano de control (controlador) y del plano de datos (*switch*), lo que permite optimizar las redes de forma continua, y responder rápidamente a los cambios en el uso de estas sin necesidad de reconfigurar manualmente la infraestructura existente, o de adquirir nuevo hardware.

OpenFlow es considerado uno de los primeros estándares de redes definidas por software; definido por la Open Networking Foundation (ONF) para implementar SDN en equipamientos de red. Está diseñado para proporcionar consistencia en la gestión e ingeniería del tráfico, haciendo que su administración sea independiente del hardware que se pretende controlar. [9] Este estándar puede venir incluido en *switch* hardware que lo integren, o ser parte de una solución software; en cualquier caso, el método de funcionamiento es el mismo. El conmutador se comunica a través de un canal seguro a un controlador externo, el cual se encargará de definir la tabla o tablas de flujo y la tabla de grupo que el *switch* utilizará para realizar la búsqueda y reenvío de paquetes. [10]

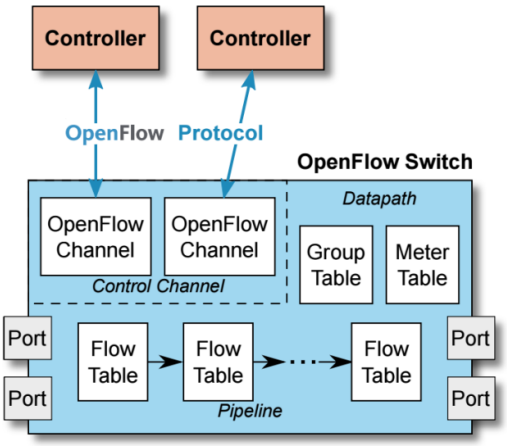


Figura 1 Diagrama de funcionamiento de un *switch* OpenFlow

2.Virtualización y soporte

Puesto que este proyecto depende casi en su totalidad de la tecnología de virtualización, se ha realizado un estudio de los componentes que mejor encajan para el desarrollo del escenario final, cuáles son sus capacidades y ventajas, y por qué, se ha seleccionado una determinada configuración.

A la hora de seleccionar una tecnología, software o sistema para cada una de las necesidades, se ha valorado profundamente y se ha otorgado prioridad al hecho de que la solución sea de código abierto, puesto que carece de restricciones de licencia y distribución, suelen contar con una amplia comunidad, aportan más flexibilidad y contribuyen al desarrollo del software libre.

2.1 Emulador de red

La emulación de red es una técnica para evaluar el rendimiento de aplicaciones reales en una red virtual. El objetivo es medir con precisión la capacidad de respuesta, el rendimiento y la calidad de la experiencia del usuario final de una aplicación, antes de aplicar cambios o añadidos a un sistema final que vaya a ser desplegado en producción. [11]

Existe una gama de emuladores de red, sin embargo, entre los más utilizados se encuentran GNS3, Eve-NG y VIRL. [12]

2.1.1 GNS3

Una interfaz gratuita cliente/servidor de código abierto destinada a la virtualización y a la emulación de redes. Está basado en Python y es compatible con una gran variedad de *router* de diferentes fabricantes.

2.1.2 VIRL

Un emulador de red virtual patentado por Cisco, muy apreciado por particulares e instituciones educativas. Lamentablemente no se ha podido evaluar el funcionamiento de este software dadas sus restricciones de licencia y distribución.

2.1.3 EVE-NG

Un emulador de red virtual propietario desarrollado para particulares y pequeñas empresas. Es una solución mucho menos flexible ya que no se presenta como software individual, sino que se ejecuta sobre una distribución de GNU/Linux Ubuntu modificada.

2.1.4 Elección

Mientras que VIRL y EVE-NG son soluciones propietarias de pago menos reconocidas en el sector y menos flexibles, GNS3 está totalmente soportado por la comunidad que lo desarrolla y cuenta con gran popularidad y documentación.

Por tanto, a partir de este punto ha quedado establecido como requisito fundamental que el marco de trabajo se realizará con el emulador GNS3 y las posteriores decisiones estarán ligadas al correcto funcionamiento de este emulador, ya que será la herramienta de diseño y control sobre la que se proyectará el escenario final.

2.2 Host

El sistema operativo anfitrión o huésped, es el software instalado en un equipo que interactuará con el hardware subyacente. El término *host* suele utilizarse para describir el sistema operativo principal utilizado en un servidor, para diferenciarlo del sistema o sistemas operativos invitados. Este sistema operativo ejecutará un hipervisor para delegar las funciones de virtualización.

El equipo utilizado para ejecutar el sistema anfitrión ha sido un Intel NUC 5i3RYH con las siguientes especificaciones:

- CPU: Intel i3-5010U
- GPU: Intel HD Graphics 5500
- SSD: KINGSTON SV300S37A240G
- RAM: 8GB PC3L-12800S

Si bien este no es un equipo de altas prestaciones tanto por sus especificaciones como por su formato de bajo perfil, es más que suficiente para albergar al completo el escenario a implementar, teniendo siempre en cuenta que no sería suficiente para un despliegue en producción y es únicamente válido para una prueba de concepto.

Los sistemas operativos candidatos para albergar este equipo han sido los siguientes:

2.2.1 VMware ESXi 6.0

Un hipervisor sin sistema operativo, diseñado específicamente para proporcionar un entorno de virtualización fiable y seguro. Es un software más

empresarial que particular, y eso se refleja claramente en su diseño, una de las principales características de VMware ESXi es su arquitectura que simplifica notablemente el proceso de mantener una infraestructura virtual consistente. Su modelo "bare-metal" con características muy cerradas no ofrece muchas opciones de configuración, pero lo hacen fácilmente mantenible, seguro y robusto. [13]

Sin embargo, no resulta un sistema efectivo para el objetivo a conseguir en este proyecto, puesto que una instancia virtual de ESXi tendría que actuar como Host para ejecutar la base del escenario mediante el emulador GNS3, que a su vez tendría que correr instancias virtualizadas. Esta situación de meta-virtualización reduce la flexibilidad y facilidad de administración para solventar posibles errores.

2.2.2 OpenSUSE Leap 15.1

Una conocida distribución de GNU/Linux en el área de la administración de sistemas que cuenta con el respaldo y soporte de actualizaciones periódicas del producto de pago profesional SUSE Enterprise Linux; proporciona una solución estable y menos experimental que la mayoría de las otras distribuciones GNU/Linux, lo que la hace muy adecuada para entornos críticos en los que la estabilidad resulta fundamental, como es el área de servidores y el ámbito de este proyecto.

Ha dado buenos resultados durante las pruebas realizadas con entornos virtuales, sin embargo, carece de un repositorio de paquetes tan completo como sus distribuciones competidoras, lo que provocó que no fuese ejecución de una versión reciente y estable del servidor de GNS3.

2.2.3 Ubuntu Server 20.04

Ubuntu es posiblemente la distribución GNU/Linux más conocida y utilizada en todo el mundo, cuenta con apoyo por parte de grandes empresas y su desarrollo está soportado económicamente. La popularidad de esta distribución se debe en parte a la facilidad de instalación y uso (fundamental para usuarios que quieren iniciarse en el ecosistema GNU/Linux), y eso ha conseguido que sea una de las distribuciones con más software disponible y soporte comunitario.

La variante para servidores evaluada es muy similar a la opción de escritorio, salvo que no dispone de entorno gráfico, la instalación está más simplificada y no incluye aplicaciones por defecto por lo que la hacen algo más liviana para actuar únicamente como servidor.

Además, la instalación y configuración tanto del sistema como del servidor GNS3 pueden ser llevadas a cabo de forma rápida y sin inconvenientes (Ver Anexo

I), lo cual es muy importante a la hora de ofrecer una solución completa fácilmente reproducible.

Por todo lo expuesto anteriormente el sistema operativo elegido finalmente es Ubuntu Server.

2.3 Hipervisor

Un hipervisor, es un software, firmware o hardware que crea y ejecuta máquinas virtuales; esto permite que una máquina anfitriona o *host* pueda coexistir con varias máquinas invitadas al compartir virtualmente sus recursos, como la memoria y el procesamiento.

Existen dos tipos principales de hipervisores, denominados "Tipo 1" (o "*bare-metal*") y "Tipo 2" (o "*hosted*"). Un hipervisor de Tipo 1 actúa como un sistema operativo ligero y se ejecuta directamente en el hardware del host, mientras que uno de Tipo 2 se ejecuta como una capa de software en un sistema operativo, como otras aplicaciones informáticas.

El Tipo 1 es el más comúnmente utilizado, en el que el software de virtualización se instala directamente en el hardware donde normalmente se instala el sistema operativo. Debido a que los hipervisores de Tipo 1 están aislados de un sistema operativo que puede ser propenso a ataques, son extremadamente seguros. Además, generalmente proporcionan un mejor rendimiento que los de Tipo 2 debido a una gestión más eficiente de los recursos. [14]

Entre los Hipervisores evaluados, eliminando la solución VMware ESXi (considerada tanto sistema Host como hipervisor) se encuentran KVM y Xen, soluciones totalmente libres y ampliamente respaldadas por la comunidad.

Es importante destacar que se han descartado populares hipervisores de Tipo 2 como Oracle VirtualBox y VMware Workstation que si bien podrían haber encajado en el escenario de la solución con GNS3, no resultan lo suficientemente flexibles ya que dependen de un software de terceros que únicamente incrementarían la complejidad de la solución, eso, añadido a que su rendimiento podría verse mermado en contraposición con una solución en kernel nativa de Tipo 1, han sido factores determinantes para quedarse fuera de los análisis.

2.3.1 KVM

Un hipervisor de código abierto integrado en el kernel de Linux desde su versión 2.6.20. El tipo de este hipervisor no está claramente definido y podría ser

indistintamente categorizado, ya que el módulo para ejecutar KVM convierte el núcleo de Linux en un hipervisor Tipo 1, mientras que el sistema en su conjunto puede ser categorizado como Tipo 2 puesto que, el *host* es totalmente funcional y no exclusivo para la virtualización.

Su rendimiento y características se han evaluado con los sistemas operativos Ubuntu Server 20.04 y openSUSE 15.1.

2.3.2 Xen

Un hipervisor Tipo 1 de código abierto reconocido por su excelente rendimiento y que se ejecuta directamente en el hardware del host. Para ejecutar un host con Xen, al contrario que con KVM, es necesario disponer de un kernel que cuente con un soporte específico para Xen, ya que no viene integrado directamente en el núcleo de Linux; por tanto, es necesario escoger una distribución de GNU/Linux que tenga soporte para Xen de forma nativa o construir un kernel personalizado que agregue dicho soporte.

La distribución utilizada para realizar las pruebas con este hipervisor ha sido OpenSUSE 15.1 (Ver Anexo II) ya que cuenta con un buen soporte y documentación para Xen.[5]

Este hipervisor soporta dos modos de operación: paravirtualización (PV) y virtualización completa (HVM); sin embargo, por cuestiones de viabilidad y complejidad se ha decidido evaluar únicamente el modo de virtualización completa puesto que la paravirtualización requiere una serie de características específicas en la instancia virtualizada y se aleja del propósito de fácil replicación y despliegue.

2.3.3 Evaluación

Las pruebas realizadas con ambos hipervisores han consistido por un lado, en la instalación completa de un sistema operativo que soporte una gran variedad de adaptadores inalámbricos, (*Kali GNU/Linux* en este caso) sobre el que se han realizado pruebas de rendimiento y estabilidad para verificar el correcto funcionamiento del hipervisor, y, por otro lado, transferir el control del hardware USB conectado al sistema anfitrión, a las instancias virtuales.

La herramienta utilizada para la prueba de rendimiento se denomina “stress”, una simple utilidad de línea de comando para realizar pruebas de estrés de entrada

[5] https://doc.opensuse.org/documentation/leap/virtualization/html/book-virt/part-virt-xen.html

y salida de CPU, memoria y disco. La prueba consiste en elevar la carga de la CPU virtual hasta el 100% de su capacidad durante 20 minutos y verificar en el M*onitor* de máquinas virtuales, que la carga corresponde con la de la instancia.

La siguiente prueba realizada es de vital importancia para el desarrollo del proyecto, ya que permite corroborar si es posible, la utilización de adaptadores de red inalámbricos USB en las instancias virtualizadas. Se conoce como "*USB passthrough*" al mecanismo que incorporan los hipervisores, para delegar el control y uso de un dispositivo USB conectado al servidor físico, y gestionado por el sistema operativo huésped, a la instancia virtual. Para la solución planteada es necesario desplegar puntos de acceso inalámbricos en las instancias virtuales mediante adaptadores de red inalámbricos USB; por ello, esta prueba es de especial relevancia.

Por último, se realizó una prueba de integración de cada uno de los hipervisores con el emulador GNS3, en la cual se determinó que lamentablemente, éste no permite ejecutar instancias virtuales con Xen por defecto. GNS3 utiliza QEMU para administrar la emulación de instancias con el hipervisor KVM, no obstante, es posible desactivar la aceleración que KVM proporciona, y en las opciones avanzadas de la instancia especificar los parámetros necesarios para que QEMU pueda trabajar de forma acelerada en un entorno con Xen. Esto es bastante más complejo, puesto que requiere especificar de forma manual cada uno de los parámetros de la emulación, perdiendo la flexibilidad deseada, por lo que se ha descartado el uso de Xen pese a los buenos resultados obtenidos en las pruebas de rendimiento y compatibilidad.

2.3.4 Resumen

Tras evaluar los distintos componentes que formarán parte de la solución de virtualización y concluyendo para cada uno de ellos una elección final, se resume el *stack* de tecnologías utilizado que será la base sobre la que se desplegará el escenario final.

Sobre el servidor físico se instalará un sistema operativo basado en GNU/Linux que incorporará mecanismos para habilitar técnicas de virtualización. El sistema escogido ha sido Ubuntu Server 20.04 que integra en el kernel el hipervisor KVM. En este sistema se instalará el servidor del emulador de redes GNS3 con el que de forma gráfica a través de un cliente se aprovisionarán y gestionarán las instancias virtuales, así como se monitorizará la infraestructura de red diseñada.

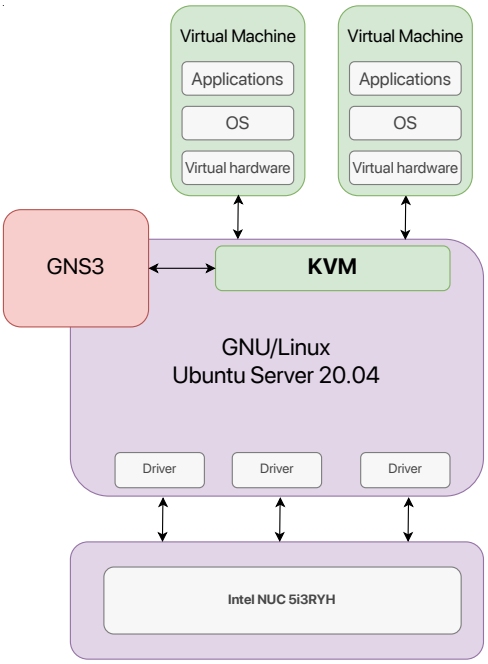


Figura 2 Arquitectura simplificada del stack seleccionado

Como se puede ver de forma esquematizada en la Figura 2, las instancias virtuales se ejecutan de forma independiente y aislada gracias al hipervisor integrado en el sistema operativo, que a su vez es controlador por GNS3 para aprovisionar, arrancar y detener las instancias virtuales.

3.Estudio de sistemas operativos

En este apartado se desarrolla el estudio realizado de los sistemas operativos que ocuparán las instancias virtualizadas y serán parte del escenario sobre el que se desplegará la solución final de LVAP.

Entre los sistemas candidatos se han escogido Debian y OpenWrt de la familia GNU/Linux, así como OpenBSD y FreeBSD de la familia BSD. Pese a que ambas familias de sistemas se basan en la filosofía UNIX, son sistemas muy diferentes entre sí (incluso entre sistemas BSD) ya que los desarrollos de sus núcleos siguen caminos alternativos y parten de bases distintas, lo que conlleva comportamientos muy dispares tanto en el hardware y el software que manejan.

Cabe mencionar que no se han evaluado otras de las múltiples y populares distribuciones GNU/Linux existentes, ya que todas ellas comparten el mismo núcleo y por lo tanto no se incorporan mejoras en la compatibilidad de los controladores y el hardware entre éstas. Específicamente, las diferencias más notables entre distribuciones son los entornos de escritorio disponibles, el gestor de paquetes, el servidor de pantalla, así como sus metas, objetivos y filosofía; características que no resultan de especial relevancia para este estudio.

Las razones de la elección de estos sistemas operativos junto con las respectivas pruebas realizadas con cada uno de ellos se encuentran recogidas en los anexos III, IV, V y VI.

3.1 Requisitos

Los sistemas operativos invitados seleccionados para su evaluación han sido escogidos en base a algunos de los criterios que se exponen a continuación y con los cuales se ha tomado una decisión final. Estos criterios se consideran fundamentales para garantizar el éxito de la solución propuesta tanto en términos de eficacia como eficiencia.

- **Compatible**

Es de vital importancia que el sistema operativo invitado sea compatible de forma nativa con un número significativo de dispositivos inalámbricos, o en su defecto esté habilitado para compilar e instalar controladores existentes que agreguen la compatibilidad de los dispositivos.

Dependiendo del controlador, es posible que el sistema operativo pueda utilizar el adaptador inalámbrico para conectarse a una red inalámbrica, pero no

sea capaz de hacer uso de funciones más avanzadas como los modos *AP* o *Monitor con inyección de paquetes,* que en este proyecto se requieren para desplegar la solución LVAP (Ver Anexo VIII). Para verificar las capacidades de los adaptadores se han realizado pruebas específicas con cada sistema operativo.

- **Liviano**

Debido a que el sistema va a ejecutarse como una instancia virtualizada en un entorno en el que coexistirá con otras múltiples instancias, éste deberá ser de naturaleza ligera ya que los recursos compartidos de los que dispondrá como el disco, RAM y CPU son limitados. Esto significa que deberá consumir pocos recursos durante y tras el arranque, permitiendo minimizar la asignación inicial de dichos recursos, que en consecuencia, se traduce en un mayor número de instancias virtuales disponibles.

También resulta interesante que el tiempo de arranque sea el menor posible ya que facilitará tareas de configuración, actualización y despliegue en las que se requieran reinicios o cambios en el hardware como reasignación de recursos, cambios en los adaptadores de red virtuales o adaptadores inalámbricos.

- **Robusto**

Los sistemas deberán estar preparados para operar de manera estable durante todo su ciclo de funcionamiento, por ello, se han evitado todos aquellos sistemas "*rolling release*" o de actualización continua, apostando por plataformas más estables y con componentes verificados y avalados por la comunidad. Errores intermitentes aparentemente despreciables que podrían afectar al funcionamiento del hardware, como las tarjetas de red, provocarían una gran cantidad de ruido y distracción en el funcionamiento y evaluación de la solución a implementar.

Además, todos los sistemas candidatos están soportados por una amplia comunidad que los valida a diario y que puede ayudar a superar problemas o dificultades con la instalación y configuración, tanto del propio sistema, como con aplicaciones o hardware de terceros.

3.2 Evaluación

En el aspecto de compatibilidad, los sistemas operativos más capacitados han resultado ser Debian y OpenWrt. Esto no supone ninguna sorpresa puesto que ambos comparten el núcleo Linux, el cual dispone en sus últimas versiones de una considerable cantidad de controladores que cubren la mayoría de las tarjetas de red inalámbrica disponibles. Para poder utilizar un adaptador determinado no

soportado de forma nativa, ha sido necesario en Debian, compilar los controladores a partir de código fuente proporcionado por la comunidad, mientras que en OpenWrt estos controladores se podrían descargar previamente compilados desde su repositorio de paquetes.

Los sistemas basados en BSD, han demostrado no contar con el soporte de adaptadores inalámbricos que se requieren en este proyecto, ya que si bien son compatibles con la mayoría de modos de operación en tarjetas que operan con los estándares 802.11a, 802.11b y 802.11g, el soporte para el estándar 802.11n se encuentra relegado a muy pocos dispositivos, siendo éste soporte prácticamente inexistente para el estándar 802.11ac; uno de los más relevantes hoy en día, también conocido como Wi-Fi 5G y cuyo funcionamiento debe estar garantizado para aplicar la solución LVAP.

En los apartados de rendimiento, ninguno ha presentado defectos notables, sin embargo, en cuanto a estabilidad, un dispositivo determinado provocaba la caída generalizada del sistema anfitrión al ser desconectado en caliente. Se desconocen las causas de este comportamiento que podría ser derivado de un fallo en el hardware del dispositivo. Se han cronometrado los tiempos de arranque desde el estado de parada de cada uno de los sistemas obteniendo los siguientes resultados:

Sistema Operativo	**Tiempo de arranque**
OpenWrt 19.07.3	10 segundos
Debian 10.2	12 segundos
FreeBSD 12.1	25 segundos
OpenBSD 6.7	32 segundos

Tabla 1 Comparativa de tiempos de arranque

Se puede observar que los menores tiempos de arranque se obtienen en los sistemas de la familia GNU/Linux con una diferencia de apenas dos segundos entre ambos, mientras que OpenBSD tarda en arrancar más del triple que su competidor OpenWrt.

Finalmente se ha optado por elegir las distribuciones GNU/Linux Debian y OpenWrt como las opciones adecuadas para albergar el rol de puntos de acceso, por su condición de sistemas ligeros, fiables y compatibles con un amplio número de adaptadores inalámbricos. Se considera adecuada la presencia de ambas

distribuciones en la solución final, con el objetivo de no depender de una única opción, además de proveer un escenario más heterogéneo.

No obstante, cabe destacar que se ha seleccionado el sistema operativo OpenBSD para las instancias de la solución dedicadas al enrutamiento y filtrado de la red, por sus excelentes resultados a la hora de realizar configuraciones avanzadas de forma sencilla, por su reconocida estabilidad en el ámbito profesional y por el potente Filtro de Paquetes (PF) que ostenta, que permite aplicar reglas de control y firewall, así como habilitar servicios como NAT[6] utilizando una sintaxis sencilla y legible.

[6] **N**etwork **A**ddress **T**ranslation

4.Escenario e infraestructura de control

En este apartado se pone finalmente de manifiesto el escenario que incluye todos los elementos objeto de estudio, que se han ido contrastando a lo largo de la memoria. Se disponen los sistemas operativos seleccionados como instancias virtuales, que actuarán como enrutadores y puntos de acceso mediante los métodos de aprovisionamiento de GNS3, (Ver Anexo VII) y con el que se administrará y evaluará toda la infraestructura.

En primer lugar, se procederá a explicar qué se desea conseguir con este escenario y el porqué de la disposición elegida.

4.1 Arquitectura propuesta

Los principales propósitos de este primer diseño de escenario, consisten en llevar a cabo el despliegue de los puntos de acceso que formarán parte de la solución LVAP en instancias virtuales, así como crear una red en el plano de control que será utilizada para coordinar y gestionar modificaciones.

Esta red virtual de control es necesaria para administrar las instancias virtuales una vez desplegadas y en funcionamiento para realizar tareas de configuración, monitorización e instalación de software de forma local, mediante la consola de la instancia o de forma remota vía SSH[7].

La red propuesta estará compuesta por las instancias virtuales que actúan como punto de acceso, más una, adicional, que ejercerá de *router,* y estarán todas conectadas entre sí mediante conmutadores virtuales no administrados. Esta red permite la comunicación entre diferentes puntos de acceso, la gestión remota de los sistemas desde el exterior, así como la descarga de la aplicación LVAP y los paquetes dependientes necesarios en el interior de las instancias.

Puesto que esta red se establece únicamente para propósitos de control, no se han establecido políticas de nombrado, las cuales serán especificadas posteriormente en la red de datos.

[7] **SSH:** Secure Shell

4.2 Descripción del escenario

A continuación, se presenta el diseño y despliegue de la arquitectura anterior como un escenario final de GNS3 con todos los componentes implicados:

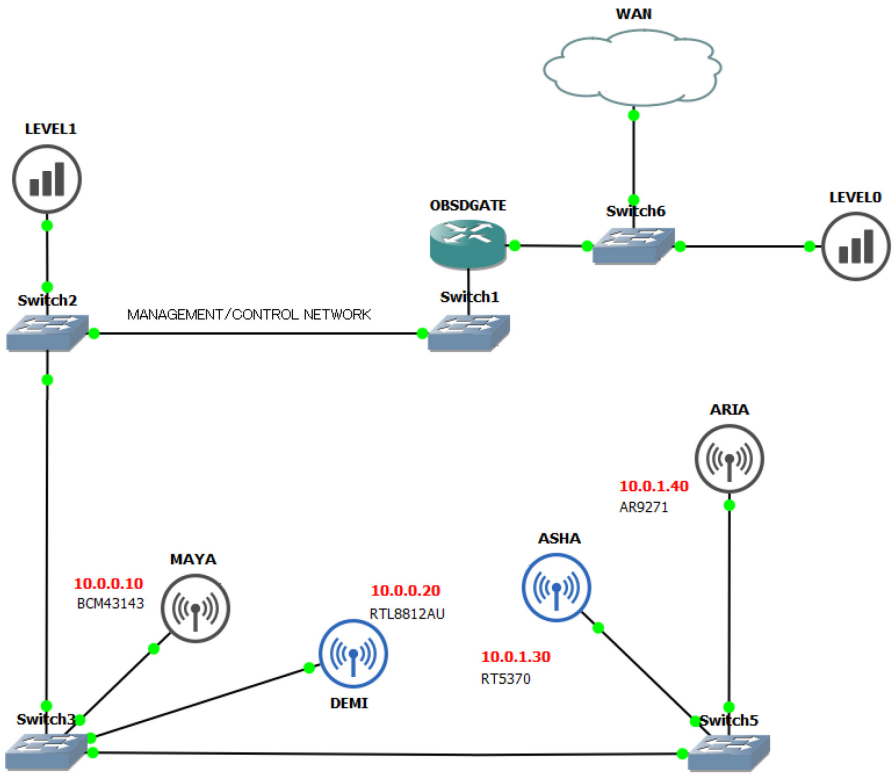


Figura 3 Escenario para la red de control en GNS3

DEMI y **ASHA**, instancias virtuales operadas con QEMU para albergar los puntos de acceso bajo el sistema operativo GNU/Linux Debian 10.2.

MAYA y **ARIA**, instancias virtuales operadas con QEMU para albergar los puntos de acceso bajo el sistema operativo GNU/Linux OpenWrt 19.07

WAN, se trata de una instancia "Cloud", dispositivo que actúa de puente entre la tarjeta de red física del equipo anfitrión y un puerto virtual en GNS3, permitiendo establecer una comunicación bidireccional entre la red virtual diseñada y la red física real a la que está el host conectado.

OBSDGATE, una instancia virtual operada por OpenBSD que actúa de enrutador y firewall para la red control, proporcionando conexión a internet a todos los puntos de acceso, así como configuración remota.

LEVEL, dos instancias ultraligeras preparadas para detectar y corregir fallos en la red rápidamente, que ejecutan Alpine GNU/Linux v3.12 a través de

Docker, la primera de ellas se encuentra conectada en un *switch* previo al router para comprobar que la instancia WAN está operativa, la segunda se ha conectado tras el *router* OBSDGATE para analizar tanto la red interna de control como el correcto funcionamiento del enrutador.

Todas las instancias están conectadas mediante conmutadores virtuales no gestionados.

5.Red de datos

En este apartado se expone la estructura de la red en el plano de datos, que será la encargada de proporcionar conexión a Internet a los usuarios conectados a los puntos de acceso, e interconectará a todos ellos independientemente del punto del acceso al que estén conectados, consiguiendo una topología unificada y balanceada con los beneficios de la gestión centralizada, pero a su vez muy flexible gracias a la incorporación de SDN a la solución.

5.1 Arquitectura propuesta

El principal objetivo de este diseño es el de proveer a los usuarios que estén conectados a los puntos de acceso una conexión de calidad, balanceando el tráfico en función de las necesidades de la red y del tipo de contenido consumido, garantizando siempre que sea posible una línea para aplicaciones críticas determinadas, atendiendo a diferentes criterios.

En este caso se propone una solución basada en dos líneas independientes:

- **Línea crítica**: La comunicación será transmitida por este enlace únicamente para aplicaciones que se consideren fundamentales y a las que se desee proporcionar un caudal exclusivo para evitar cortes o altas latencias derivadas de la saturación de la red. Estas aplicaciones pueden englobar servicios como los de comunicación entre personas (correo electrónico, mensajería instantánea, VoIP), seguridad (alarmas, cámaras de vigilancia) o dispositivos IoT.
- **Línea por defecto**: Este enlace será utilizado para todas aquellas conexiones consideradas no esenciales y que no hayan sido derivadas hacia la línea crítica. La carga que deberá soportar este enlace será mucho mayor puesto que generalmente transportará la mayoría del tráfico generado por el usuario.

Este modelo de separación ha sido el implementado en el escenario para la prueba de concepto, sin embargo, las herramientas y técnicas utilizadas son compatibles con cualquier otra configuración deseada.

Un segundo objetivo a alcanzar, es el de proporcionar transparencia absoluta en las comunicaciones realizadas entre usuarios que puedan encontrarse en el mismo o diferentes puntos de acceso, permitiendo unificar la red eliminando componentes intermediarios y reduciendo la complejidad en la gestión. A su vez, esto permite el

uso de aplicaciones entre usuarios conectados basadas en P2P, sin necesidad de recurrir a mecanismos de NAT transversal, puesto que todos forman parte del mismo grupo y se encuentran al mismo nivel.

En la siguiente figura se muestra de forma esquematizada la arquitectura propuesta con la línea crítica (L1), la línea por defecto (L2) y la línea entre puntos de acceso (TNL):

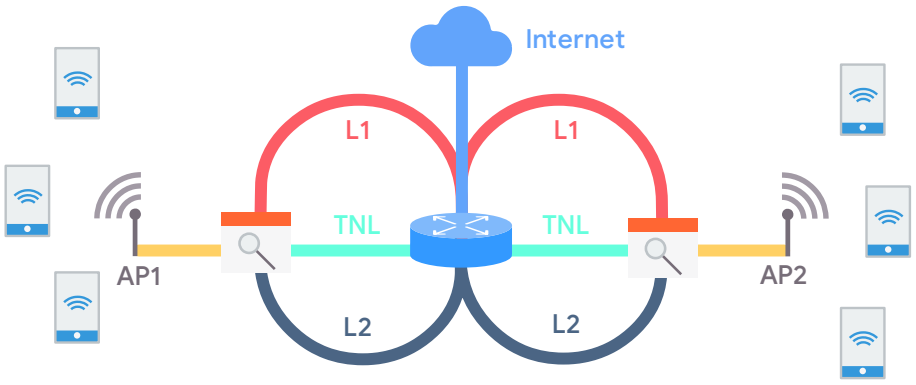


Figura 4 Esquema la arquitectura en la red de datos

Para lograr que el enrutamiento y la tunelización sea totalmente transparente al usuario y a la configuración de los puntos de acceso, es necesario disponer de elementos que analicen el tráfico de la red y tomen decisiones en base a reglas definidas por el administrador; es en este punto, donde entra el potencial de las redes definidas por software.

5.2 SD-WAN con Open vSwitch

Puesto que el objetivo es aislar la gestión de los puntos de acceso de la infraestructura y requisitos de la red de datos, es necesario introducir conmutadores o *switch* en la red que sean capaces de reenviar paquetes entre diferentes interfaces de red, en función a una lógica previamente establecida y basada en los criterios de la arquitectura planteada.

Como ocurre generalmente en las redes definidas por software (SDN), los conmutadores no operan únicamente en L2, sino que suelen trabajar en múltiples capas. Un conmutador OpenFlow puede decidir cómo reenviar tramas y paquetes basándose en atributos como las direcciones MAC de origen/destino de la trama, los campos de cabecera del paquete IP, los indicadores ICMP, el número de puerto UDP/TCP y otros parámetros. Los flujos de reenvío y los flujos de red del conmutador pueden definirse basándose en cualquier conjunto de atributos. [15]

Gracias a estas características, OpenFlow se presenta como la solución ideal en el plano de datos para gestionar y resolver el enrutamiento de los paquetes de los usuarios, en base a los dos enlaces independientes disponibles.

Existen implementaciones software de *switch* compatibles con OpenFlow, es decir que siga el estándar y sea capaz de recibir e interpretar comandos utilizando el protocolo OpenFlow. Uno de estos *switch* es Open vSwitch (OvS), una plataforma de conmutación virtual multicapa de código abierto, a la altura de alternativas propietarias como VMware vSwitch o Cisco Nexus 1000V [16], y que se encuentra disponible directamente como instancia recomendada de GNS3.

Open vSwitch no solo ofrece capacidades de conmutación virtual a los entornos de servidores virtuales, también permite la creación de túneles con diferentes protocolos como GRE, VXLAN, STT y Geneve con soporte IPSec; una característica muy interesante para abordar el objetivo de proporcionar una conexión transparente a usuarios entre diferentes puntos de acceso.

De entre todas las opciones disponibles se ha escogido la tunelización GRE (encapsulamiento genérico de ruta), puesto que es una de las más utilizadas y sencillas de implementar. GRE es un protocolo de comunicación utilizado para establecer una conexión directa, (punto a punto) entre los nodos de la red. Se trata de un método simple y eficaz de transportar datos a través de una red pública, como es Internet, y que permite que dos pares compartan datos como si fueran miembros de la misma red.

Mediante Open vSwitch es posible hacer uso de la potencia de OpenFlow para definir políticas de encaminamiento, así como introducir mecanismos de tunelización para crear caminos directos e independientes, todo ello en un mismo dispositivo materializado como instancia de GNS3.

Cuando un paquete sea emitido por un punto de acceso, éste será capturado por el conmutador OvS que actuará en base a lo siguiente:

- Si la dirección IP de destino se corresponde con la de un **servicio designado fundamental**, el paquete es reenviado por la ruta crítica.
- Si la dirección IP de destino se corresponde con la de un **dispositivo vecino** ubicado en otro punto de acceso, el paquete es reenviado por el túnel GRE.
- En cualquier **otro caso** el paquete es reenviado por la ruta por defecto.

En la siguiente figura se describe la casuística como un diagrama de flujo.

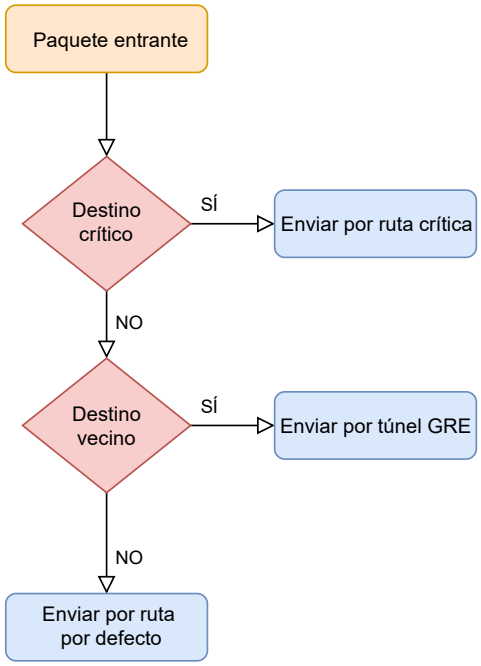


Figura 5 Evaluación de reglas como un diagrama de flujo

Los paquetes dirigidos a destinos críticos se evalúan en primer lugar para ser reexpedidos lo antes posible. Cuando un paquete entrante se dispone a ser inspeccionado, las reglas o flujos son evaluados en orden secuencial; esto provoca que una lista con un gran número de reglas, pueda degradar la experiencia de usuario sino se cuenta con un hardware potente capaz de evaluar las reglas y reenviar los paquetes de forma eficiente.

5.3 Metodología de filtrado

A la hora de determinar el carácter de un servicio o aplicación para designarlo crítico, es necesario conocer la dirección IP de éste, puesto que va a ser el único identificador que las reglas o flujos podrán evaluar para reexpedir el paquete por una u otra ruta.

Cuando se conoce a la perfección la infraestructura de red de una aplicación o servicios públicos disponibles en Internet, es sencillo determinar cuál será la dirección o direcciones IP que éstos utilizarán para ofrecer servicio. No obstante, esto no es lo habitual puesto que la mayoría de las empresas se reservan el rango de direcciones IP que utilizan para sus aplicaciones.

Una primera opción, es resolver el nombre de dominio de la aplicación que se desea filtrar para obtener su dirección IP, sin embargo, esta técnica no es actualmente válida ya que muchos de los servidores de nombres (DNS) emplean

técnicas de rotación de direcciones para aplicar sistemas de balanceo de carga o de DNS geográfico, para redirigir al cliente al servidor más próximo en función del origen de su petición, esto implica que dependiendo de la ubicación del usuario, la dirección IP resuelta para un mismo dominio, sea diferente incluso dentro del mismo país. Estas técnicas son ampliamente utilizadas gracias a la popularización y accesibilidad que han experimentado las redes de distribución de contenido o CDN las cuales han revolucionado la forma en la que se sirven las aplicaciones y los servicios. A pesar de ello, dificultan la tarea de identificar una aplicación o servicio por su dirección IP.

Métodos como la captura de la respuesta de una petición DNS, o la vinculación de la dirección IP con un paquete que contenga el Server Name Indication (SNI) de una petición HTTPS, requerirían implementar técnicas de inspección profunda de paquetes en OpenFlow, lo cual incrementaría considerablemente la complejidad del escenario, y no resultarían técnicas efectivas con los actuales estándares de encriptación de peticiones DNS sobre HTTPS, (DoH[8]) o TLS (DoT[9]) que han ido ganando popularidad en los últimos años, así como la extensión ECH[10] o "*Encrypted Client Hello*" del estándar TLS 1.3 (aún en desarrollo pero ya implementada por alguna CDN), la cual cifra el nombre del dominio haciendo imposible relacionar el nombre de dominio con su dirección IP.

La única forma inequívoca de determinar si un cliente está accediendo a un determinado servicio o utilizando determinada aplicación, es conocer la dirección destino del paquete, un campo que no se puede enmascarar para que pueda ser enrutado correctamente, y solo puede ser camuflado mediante el uso de redes y técnicas de tunelización VPN, alternativas con las cuales dejaría de tener sentido la bifurcación de rutas para ofrecer una mayor calidad de servicio, puesto que una VPN implica una conexión retransmitida y no directa.

Finalmente, la técnica empleada para determinar las posibles direcciones IP de una aplicación, consiste en un proceso de investigación manual, en el cual se lleva a cabo una captura del tráfico de una comunicación entre un cliente y una aplicación a filtrar; este proceso se efectúa para determinar qué direcciones IP hay implicadas en la comunicación. Esto es necesario puesto que no siempre es posible recuperar la dirección IP de las aplicaciones o servicios (en los que en ocasiones

[8] https://tools.ietf.org/html/rfc8484

[9] https://tools.ietf.org/html/rfc7858

[10] https://tools.ietf.org/html/draft-ietf-tls-esni-07

incluso el dominio es desconocido), como ocurre en las aplicaciones móviles o de escritorio, y en entornos web con *JavaScript* en los que no es fácil determinar qué peticiones se realizan.

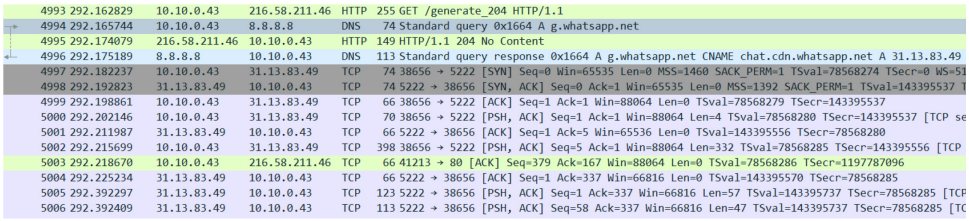


Figura 6 Captura de comunicación de WhatsApp con Wireshark

Una vez capturada la comunicación y anotadas las direcciones implicadas, hay que eliminar todas aquellas que no correspondan a la aplicación como pueden ser las peticiones a servidores DNS, comunicaciones del sistema en segundo plano y otras que no sean relevantes. Con la lista de direcciones filtrada, se hace uso de una herramienta online para recuperar el número de sistema autónomo (AS) que corresponde a una determinada IP. Un sistema autónomo representa el grupo de redes IP que poseen una política de rutas propia e independiente y generalmente, solo empresas proveedoras de servicios a gran escala cuentan con más de uno.

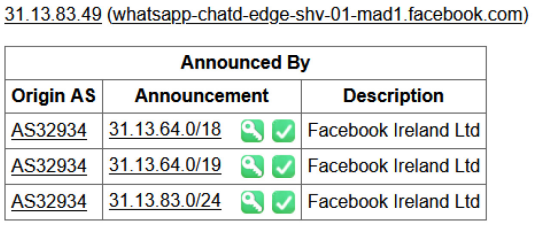


Figura 7 Correspondencia dirección IP con ASN para WhatsApp

Conocido el número de sistema autónomo, es posible obtener todos los prefijos IP que administra, utilizando herramientas de monitorización BGP[11] online, como BGP Toolkit[12] de *Hurricane Electric*. Una vez recuperados los bloques de direcciones relevantes, son procesados para generar entradas de reglas o flujos que se puedan introducir directamente en Open vSwitch. (Ver Anexo X)

Este método también plantea cuestiones de viabilidad, puesto que la aplicación de reglas de filtrado tan agresivas, puede provocar que se incluyan otras muchas direcciones que no tengan relación directa o indirecta con la aplicación o servicio que se desea filtrar, sin embargo, de esta manera es posible garantizar que

[11] **B**order **G**ateway **P**rotocol

[12] https://bgp.he.net/

todo el tráfico derivado del uso de la aplicación o servicio filtrado será siempre transmitido por la ruta crítica.

No ha resultado compleja la aplicación de este método, puesto que el objetivo buscado es el de priorizar aplicaciones y servicios de comunicación, los cuales son llevados generalmente por grandes compañías que engloban más de una aplicación bajo un mismo sistema autónomo como es el caso de *Facebook Inc* con aplicaciones sociales como *WhatsApp*, *Instagram* y *Facebook*. No obstante, este proceso está sujeto a actualizaciones y revisiones periódicas ya que las aplicaciones pueden cambiar de dominio, de direcciones o incluso de propietario.

5.4 Descripción del escenario

A continuación, se presenta la red de datos como escenario de GNS3:

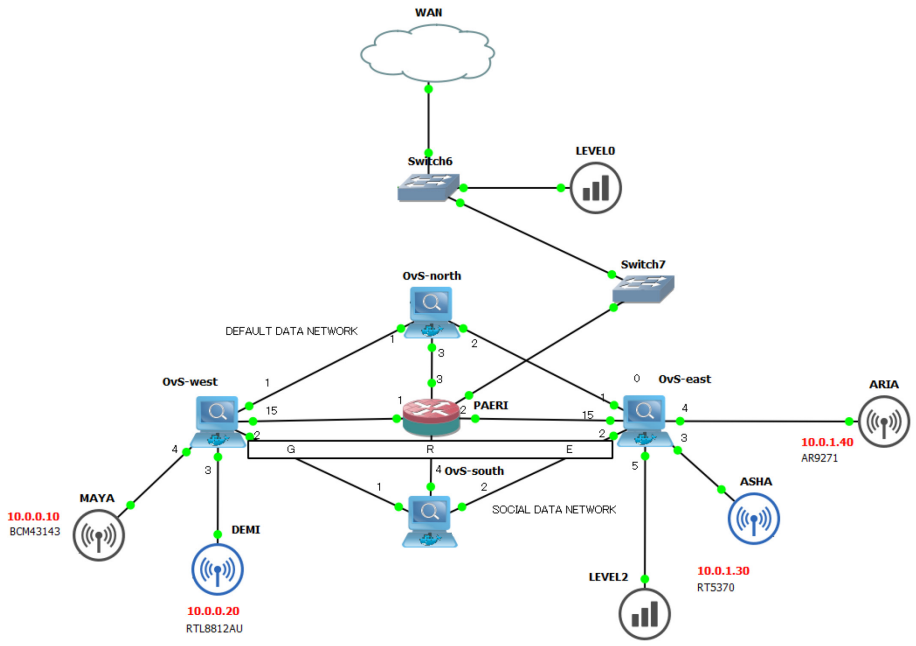


Figura 8 Escenario para la red de datos en GNS3

OvS-west y **OvS-east** son instancias operadas por Open vSwitch que ejecutan toda la lógica de filtrado y establecen una tunelización GRE entre ellos como se puede apreciar en la figura. Ambos representan ubicaciones distintas.

OvS-north y **OvS-south** instancias operadas por Open vSwitch con el único propósito de reexpedir paquetes y aislar la red.

P/AERI se trata de un enrutador operado con OpenBSD que conecta tanto la ruta crítica como la ruta por defecto con Internet, además de ser el punto de

interconexión entre todos los equipos Open vSwitch, es decir, el puente para establecer una conexión directa tunelizada. Se ha utilizado un único enrutador para simplificar el escenario de la prueba de concepto, sin embargo, un entorno real debería contar con equipos de enrutamiento diferenciados para cada una de las líneas.

Se mantienen las instancias **WAN** y **LEVEL** presentadas en la red de control.

Para organizar la estructura de red e identificar fácilmente las instancias, se ha escogido una máscara de red lo suficientemente grande como para asignar direcciones fácilmente distinguibles y reconocibles a todas ellas sin la necesidad de ampliar o interconectar subredes, por ello, se ha escogido el bloque IPv4 más amplio disponible para el ámbito privado: "10.0.0.0/8". Para que la identificación de una determinada instancia sea rápida y sencilla, la asignación de las direcciones se realizará de forma estática y se aplicarán las siguientes reglas de nombrado:

- Se reserva el segundo bloque de 8 bits.
- El tercer bloque de 8 bits identificará el grupo de estaciones o puntos de acceso. (De 0 a 255)
- El último bloque de 8 bits identificará el número de estación que en este caso se ha construido con múltiplos de 10. (De 10 a 250)

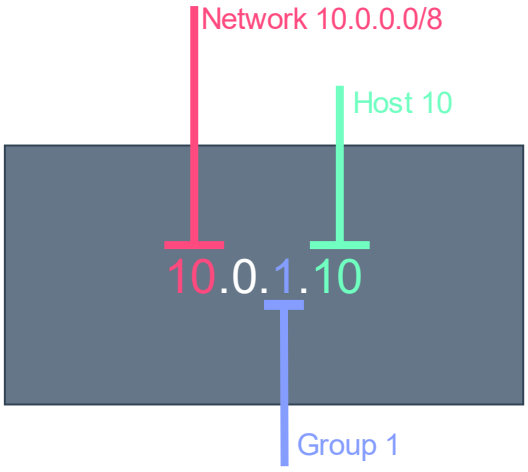


Figura 9 Agrupación de instancias y grupos por bloques en IPv4

6. Escenario final

Una vez realizado el recorrido por el plano de gestión, en el que se construye una infraestructura que otorga un control directo sobre cada una de las instancias, y el plano de datos, que mediante técnicas de SDN, se desarrolla una red dinámica y balanceada, son ambos combinados en un único escenario en el que coexisten y simulan un escenario completo real y final.

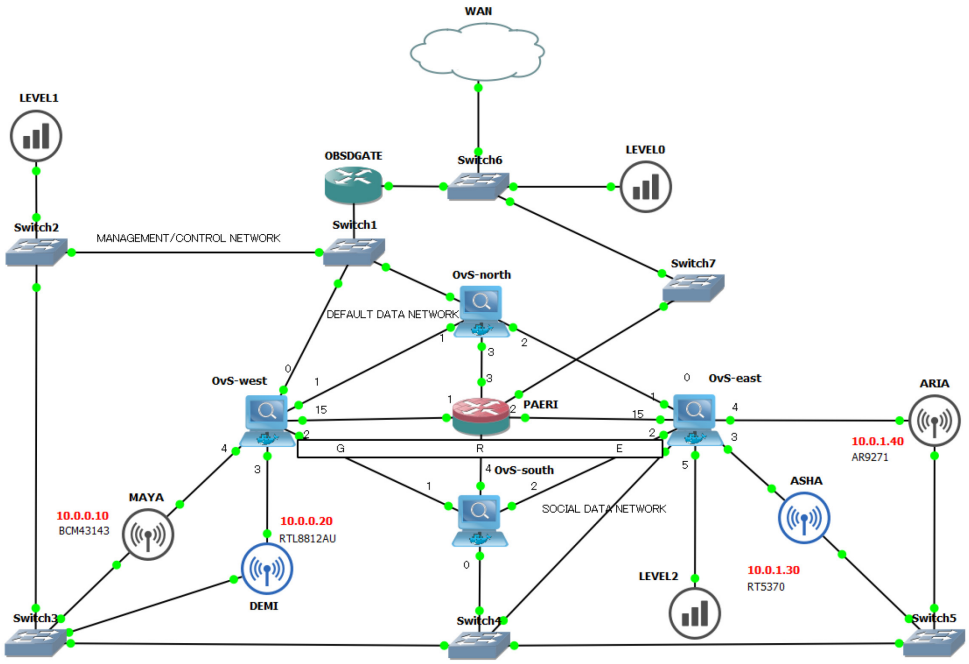


Figura 10 Escenario final en GNS3

Como se puede observar, todas las instancias se encuentran conectadas a la red de control para ser administradas remotamente, se aprecia el enlace crítico y por defecto definidos como "**SOCIAL DATA NETWORK**" y "**DEFAULT DATA NETWORK**" respectivamente, así como el enlace GRE que vincula los dos extremos de los conmutadores Open vSwitch principales.

6.1 Evaluación

Se han realizado múltiples pruebas que han resultado exitosas para evaluar el correcto funcionamiento de la infraestructura planteada, con reglas de filtrado para populares aplicaciones de mensajería como *WhatsApp* y *Telegram*, o redes sociales como *Facebook* e *Instagram*, sin embargo, se ha optado por utilizar una configuración más clara e ilustrativa para evidenciar el funcionamiento de la operativa.

En esta prueba se va a utilizar el escenario completo dispuesto en la Figura 10, pero únicamente se realizarán operaciones sobre dos dispositivos cliente conectados a dos puntos de acceso diferentes (DEMI y ARIA). El objetivo de esta prueba es retransmitir todo el tráfico destinado a los **servicios web que ofrece la universidad** por la ruta crítica, y descartar el resto del tráfico por la ruta por defecto.

La Universidad de Zaragoza cuenta para toda su infraestructura de red con el bloque de direcciones IPv4 *155.210.0.0/16,* el cual se encuentra administrado por el sistema autónomo AS766 perteneciente a **Red.es**, una entidad pública empresarial española que a su vez gestiona **RedIRIS**, la red académica y de investigación que utiliza la universidad para conectarse con Internet. [17]

Announced By		
Origin AS	**Announcement**	**Description**
AS766	155.210.0.0/16	Universidad de Zaragoza

Figura 11 ASN origen y bloque de direcciones de Universidad de Zaragoza

En este caso, el sistema autónomo AS766 administra otros muchos bloques que pertenecen a otras instituciones y que no se desean filtrar. En el Anexo IX se incluyen las reglas de flujo y configuración aplicadas en los dispositivos Open vSwitch para simular este escenario.

Todas las instancias que actúan como puntos de acceso han sido configuradas para operar de tal modo, como se describe en los anexos para Debian y OpenWrt GNU/Linux.

Como se puede apreciar en la Figura 12, se han establecido puntos de captura de tráfico en los enlaces de los conmutadores perimetrales *OvS-north* y *OvS-south* para poder discernir en todo momento si los paquetes son redirigidos por la ruta crítica o por la ruta por defecto, así como en el enlace dedicado al túnel GRE para la analizar la comunicación directa entre usuarios.

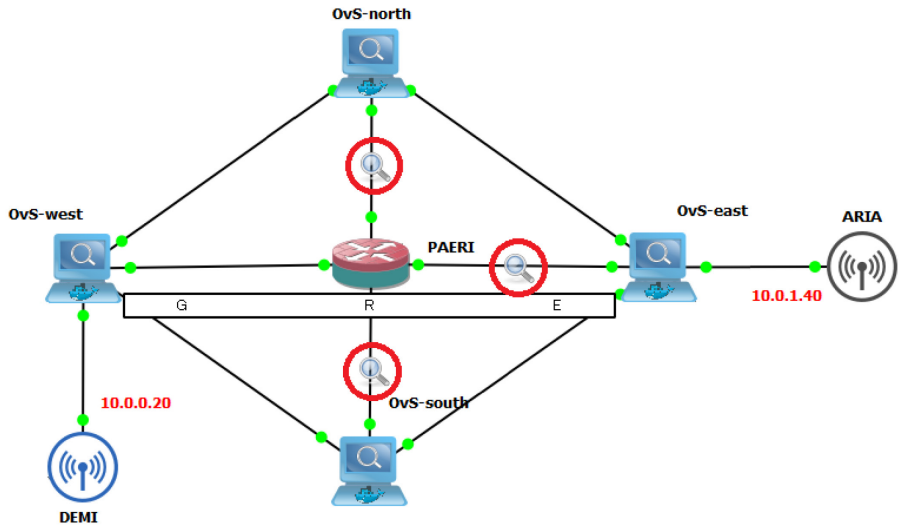


Figura 12 Captura de tráfico desde GNS3

Las pruebas que se han realizado consisten en verificar si desde cada uno de los puntos de acceso, se obtiene el resultado deseado. Los clientes conectados a **DEMI** y a **ARIA** dispondrán de las direcciones IPv4 10.0.0.21 y 10.0.1.41 respectivamente:

- **Tunelización GRE**

En esta prueba se verifica que el enlace entre puntos de acceso funciona correctamente. Se ha realizado una prueba de *ping* cruzada entre los dispositivos conectados a los puntos de acceso para atestiguar que la comunicación se envía a través del túnel y funciona correctamente:

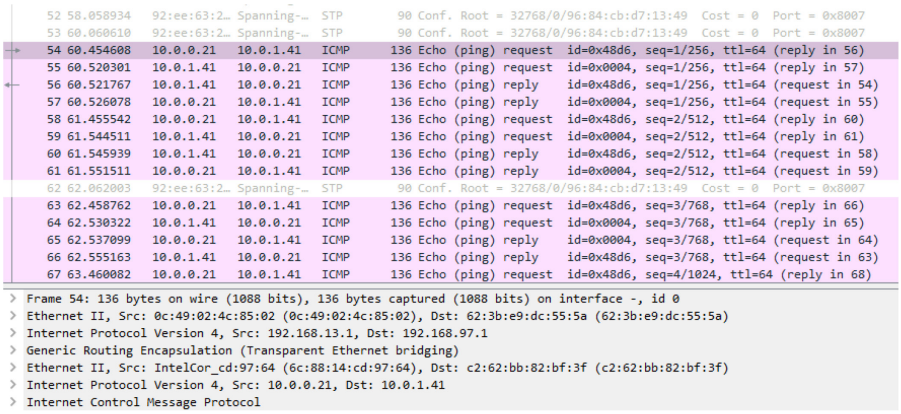


Figura 13 Captura de tráfico en enlace GRE con Wireshark

Como se puede apreciar, ambos dispositivos envían paquetes ICMP[13] de solicitud (*ICMP Echo Request*) que son devueltos de inmediato en forma de respuesta (*ICMP Echo Reply*). Todo este tráfico generado con la herramienta *ping* se encuentra encapsulado mediante un túnel GRE que a su vez es transmitido por IPv4.

- **Ruta crítica**

Para evaluar el correcto funcionamiento del enrutamiento condicional, se hace uso de nuevo de la herramienta *ping*, que se utilizará para hacer peticiones ICMP a el dominio cabecera de la universidad (*unizar.es*). Esta prueba resulta interesante, puesto que el servidor de nombres utilizado para resolver la dirección IP del dominio, no se ha establecido como servicio crítico.

En la siguiente figura se muestra el tráfico recuperado en el enlace de la ruta por defecto, donde se aprecia que la única comunicación capturada, es para resolver la dirección del nombre de dominio contra el servidor DNS con dirección *1.1.1.1*, la cual no ha sido establecida como un destino crítico:

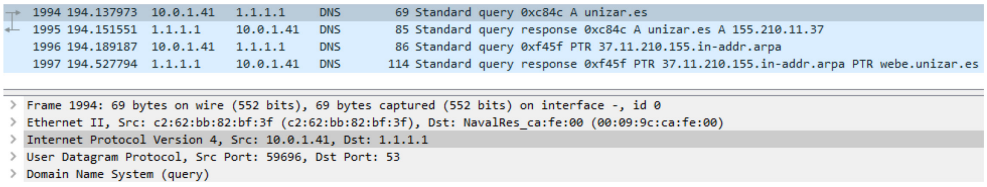


Figura 14 Captura de tráfico en enlace por defecto con Wireshark

Es en la ruta crítica, donde se puede ver toda la traza del tráfico ICMP generado con la herramienta *ping*, y en la que no se aprecia ninguna comunicación previa:

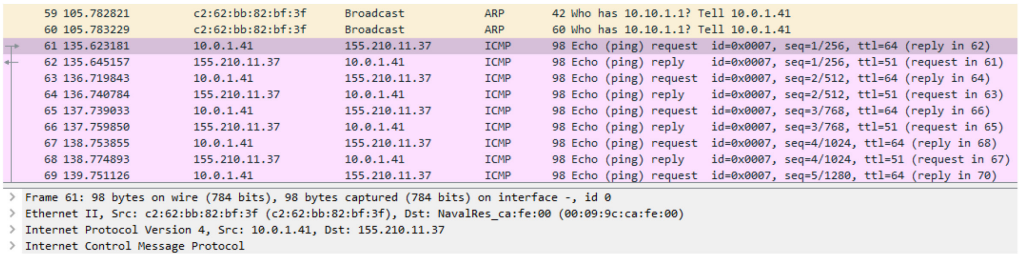


Figura 15 Captura de tráfico en enlace crítico con Wireshark

[13] **I**nternet **C**ontrol **M**essage **P**rotocol

7.Continuación

Como se ha comentado previamente, los conmutadores compatibles con OpenFlow permiten definir el comportamiento de la red de forma automática mediante un controlador central, operado por un software que se comunica remotamente con los *switch* compatibles. Este comportamiento no ha sido implementado en esta prueba de concepto puesto que no resulta fundamental para demostrar la arquitectura presentada, sin embargo, una corriente alternativa iniciada a partir de este trabajo puede abarcar la incorporación de este controlador para hacer un mejor aprovechamiento de las técnicas SDN que ofrece OpenFlow y Open vSwitch.

El sistema de filtrado podría especializarse permitiendo discriminar los paquetes por tipo de protocolo de comunicación, en base a los criterios deseados para aplicaciones críticas, para lo que habría que incorporar mecanismos de inspección de paquetes, que trabajen en capas más profundas que en las que opera habitualmente la solución OpenFlow estándar.

Sería necesario automatizar el sistema de filtrado empleado basado en ASN, puesto que actualmente requiere de constante revisión y mantenimiento. El objetivo sería elaborar una plataforma software integrada en una o varias instancias virtuales, que sea capaz de realizar la captura de paquetes, analizar los dominios implicados, y si coinciden con los criterios de filtrado, obtener sus direcciones, consultar el sistema autónomo al que corresponden esas direcciones y generar reglas para dirigir el tráfico de acuerdo con el destino de todos los paquetes entrantes.

Se ha utilizado el protocolo IPv4 para toda la configuración del escenario por cuestiones de simplicidad a la hora de elaborar la prueba de concepto, no obstante, se considera interesante la integración de mecanismos en la solución que permitan llevar a cabo la tan acusada transición a IPv6. En un entorno de puntos de acceso en el que se pueden conectar múltiples usuarios, resultaría idóneo incorporar el protocolo de comunicación IP más reciente para comunicaciones punto a punto entre equipos de fuera y dentro de la red.

8.Conclusiones

El recorrido realizado desde el comienzo de este proyecto para que haya logrado ser un éxito y se hayan cumplido todos los objetivos propuestos, ha necesitado de un exhaustivo y profundo estudio sobre las tecnologías de virtualización, alternativas disponibles y la realización de numerosas pruebas de compatibilidad y rendimiento, pasando por el diseño una arquitectura de red, que pudiera estar a la altura de los modelos que hoy en día las empresas requieren, y finalizando con un despliegue en el que coexisten todas las tecnologías analizadas y se pone de manifiesto el valor de la infraestructura diseñada.

Mediante este proyecto no solo es posible dar soporte a la solución de puntos de acceso virtuales ligeros (LVAP), sino que se presenta como un escenario estudiado y preparado para implementar sobre él, cualquier aplicación que implique el "**control y gestión de redes inalámbricas en entornos virtuales distribuidos**".

Queda mucho trabajo por hacer por parte de los grandes fabricantes de adaptadores inalámbricos, para que la comunidad pueda disfrutar de controladores oficiales libres y no tenga que perder el tiempo en rehacerlos mediante complejas técnicas de ingeniería inversa, que han lastrado enormemente el desarrollo y soporte de la tecnología de red inalámbrica en todos los ámbitos del esquema *Open Source*. Esta comunidad que desinteresadamente se ha involucrado en un desarrollo tan complejo, que genera documentación de calidad y que resuelve cualquier duda de implementación, ha contribuido enormemente a que este proyecto se haya convertido en una realidad.

9.Referencias

[1] CORDIS | European Commission, "What to do With the Wi-Fi Wild West | Wi-5 Project | H2020 | CORDIS | European Commission," 4 9 2017. [Online]. Available: https://cordis.europa.eu/project/id/644262.

[2] K. MacIver, "The coming impact of the cloud revolution | I-CIO," 6 2014. [Online]. Available: https://www.i-cio.com/strategy/cloud/item/the-coming-impact-of-the-cloud-revolution.

[3] Red Hat, Inc., "What is virtualization? | Opensource.com," [Online]. Available: https://opensource.com/resources/virtualization.

[4] J. Shamir, "5 Benefits of Virtualization | IBM," 8 4 2020. [Online]. Available: https://www.ibm.com/cloud/blog/5-benefits-of-virtualization.

[5] Research and Markets, "Global Wi-Fi Enabled Devices Shipment Forecast, 2019 - 2023," 2 8 2020. [Online]. Available: https://www.researchandmarkets.com/reports/4826074/global-wi-fi-enabled-devices-shipment-forecast.

[6] Intel Corporation, «What Is Wi-Fi 6? - Intel,» [En línea]. Available: https://www.intel.la/content/www/xl/es/gaming/resources/wifi-6.html.

[7] V. Madhavanunni and S. Srikanth, "Analysis of open source drivers for IEEE 802.11 WLANs," 4 2 2010. [Online]. Available: https://www.researchgate.net/publication/224116216_Analysis_of_open_source_drivers_for_IEEE_80211_WLANs.

[8] Wikipedia®, "Comparison of open-source wireless drivers - Wikipedia," 11 8 2020. [Online]. Available: https://en.wikipedia.org/wiki/Comparison_of_open-source_wireless_drivers.

[9] NoviFlow inc., "The basics of SDN and the OpenFlow Network Architecture | NoviFlow," [Online]. Available: https://noviflow.com/the-basics-of-sdn-and-the-openflow-network-architecture/.

[10] A. Zhu, "OpenFlow Switch: What Is It and How Does it Work? | by Aria Zhu | Medium," 30 07 2018. [Online]. Available: https://medium.com/@AriaZhu/openflow-switch-what-is-it-and-how-does-it-work-7589ea7ea29c.

[11] Z. Cole, "Network simulation or emulation? | Network World," 22 9 2017. [Online]. Available:

https://www.networkworld.com/article/3227076/network-simulation-or-emulation.html.

[12] R. Bhardwaj, "GNS3 vs Eve-NG vs VIRL - IP With Ease," 4 6 2020. [Online]. Available: https://ipwithease.com/gns3-vs-eve-ng-vs-virl/.

[13] The Access Group, "VMware ESXi – What is ESXi and what are its features?," 22 6 2017. [Online]. Available: https://www.theaccessgroup.com/blog/chs-vmware-esxi-what-is-esxi-and-what-are-its-features/.

[14] VMware, Inc, "What is a Hypervisor? | VMware Glossary," [Online]. Available: https://www.vmware.com/topics/glossary/content/hypervisor.

[15] I. Akbari, "SDN part 2: Building an SDN playground on the cloud using Open vSwitch and OpenDaylight | by Iman Akbari | Medium," 13 6 2019. [Online]. Available: https://medium.com/@blackvvine/sdn-part-2-building-an-sdn-playground-on-the-cloud-using-open-vswitch-and-opendaylight-a0e2de029ce1.

[16] R. Sheldon, "A comprehensive guide to Open vSwitch and its capabilities," 11 5 2020. [Online]. Available: https://searchservervirtualization.techtarget.com/tip/A-comprehensive-coverage-of-Open-vSwitch-and-its-capabilities.

[17] Red.es, «Quiénes somos | Red.es,» [En línea]. Available: https://www.red.es/redes/es/quienes-somos/redes.

[18] Kody, "How to Check if Your Wireless Network Adapter Supports Monitor Mode & Packet Injection « Null Byte :: WonderHowTo," 12 11 2018. [Online]. Available: https://null-byte.wonderhowto.com/how-to/check-if-your-wireless-network-adapter-supports-monitor-mode-packet-injection-0191221/.

[19] Aircrack-ng, "injection_test [Aircrack-ng]," [Online]. Available: https://www.aircrack-ng.org/doku.php?id=injection_test.

[20] OpenBSD Project, "OpenBSD," [Online]. Available: https://www.openbsd.org/.

[21] R. Cerrato, «How to Build your Own Wireless Router (Part 3) | by Renaud Cerrato | Medium,» 22 10 2018. [En línea]. Available: https://medium.com/@renaudcerrato/how-to-build-your-own-wireless-router-from-scratch-part-3-d54eecce157f.

[22] D. Okoi, "What is FreeBSD? Why Should You Choose It Over Linux?," 6 7 2018. [Online]. Available: https://www.fossmint.com/what-is-freebsd-why-should-you-choose-it-over-linux/.

Anexo I. Preparación del Host

La instalación del servidor GNS3, requiere una secuencia de pasos más compleja que la de un servicio estándar ya que utiliza muchos componentes para funcionar. La distribución GNU/Linux escogida para alojar este servidor ha sido Ubuntu Server 20.04.1, la última disponible en el momento de la instalación. Esta distribución tiene un excelente gestor de paquetes, así como una comunidad muy activa que puede asistir en caso de problemas durante la instalación o la ejecución del servidor; además, la documentación de GNS3 provee unos pasos a seguir para que la instalación sobre Ubuntu sea de forma guiada.

Configuración de la red

A partir de Ubuntu 18.04 la configuración de la red deja de seguir el funcionamiento de Debian y pasa a realizarse a través de **NETPLAN**, una utilidad que actúa como capa de abstracción, y genera archivos de configuración específicos para entregar el control de los dispositivos a un *demonio* de red particular.

Los ficheros de configuración se encuentran en `/etc/netplan/`, para establecer un direccionamiento estático basta con crear un archivo de extensión "*.yaml*" y agregar en él lo siguiente:

```
network:
  version: 2
  ethernets:
    enp0s25:
      addresses:
        - 192.168.1.254/24
      gateway4: 192.168.1.1
      nameservers:
          addresses: [1.1.1.1, 1.0.0.1]
```

En el apartado "**ethernets**" se incluyen las interfaces de red del sistema que se desean configurar. Seguidamente se reflejan las direcciones para la interfaz, puerta de enlace y servidores DNS.

Instalación del servidor GNS3

En primer lugar, se añade el repositorio de GNS3 desde el servicio *PPA*[14], que permite a los usuarios subir los paquetes de fuentes de Ubuntu que se construyen y publican como un repositorio apto.

[14] **P**ersonal **P**ackage **A**rchive

Se actualiza la base de datos de paquetes y se instala en este caso solo el servidor, ya que no se va a necesitar el cliente.

```
sudo add-apt-repository ppa:gns3/ppa
sudo apt update
sudo apt install gns3-server
```

En una instalación limpia por defecto de Ubuntu, no viene ninguna versión de Docker instalada, sin embargo, si se ha utilizado Docker con anterioridad es recomendable reemplazarlo por la última versión de *docker-ce* de los repositorios oficiales.

```
sudo apt remove docker docker-engine docker.io
```

A continuación, se instalan los siguientes paquetes para poder descargar e instalar la clave GPG del sitio oficial de Docker.

```
sudo apt-get install apt-transport-https ca-certificates curl \ software-
properties-common
```

Se añade el repositorio y se actualiza la base local de paquetes:

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable"
sudo apt update
```

Ahora se puede instalar la versión *docker-ce* oficial:

```
sudo apt install docker-ce
```

Finalmente, es importante agregar el usuario que vaya a ejecutar el servidor GNS3 a los siguientes grupos *(ubridge, libvirt, kvm, docker),* para que pueda ejecutar máquinas virtuales y contenedores:

```
sudo usermod -aG ubridge gns3
sudo usermod -aG libvirt gns3
sudo usermod -aG kvm gns3
sudo usermod -aG docker gns3
```

En este caso se ha utilizado el usuario *gns3* que va a ser creado en siguiente apartado.

Ejecutar el servidor GNS3 como un demonio

Por defecto, al instalar el servidor GNS3 no se crea ningún *demonio* en sistema que ejecute el servidor y hay que crearlo de forma manual, GNS3 proporciona una serie de explicaciones para hacerlo, pero hay que ajustar algunos detalles.

En primer lugar, se crea el usuario que va a ejecutar el *demonio*:

```
sudo adduser gns3
```

La documentación prevé la utilización de diferentes versiones, para lo que propone la clonación del repositorio del servidor, y la posterior selección de la rama de *Git* dependiendo de la versión del servidor instalada; sin embargo, por si solo el fichero de iniciación del *demonio* no funciona y hay que ajustarlo, para ello, se propone la siguiente configuración para Ubuntu 20.04 y la versión 2.2.11 de GNS3.

Se crea con un editor de texto el fichero del *demonio* en `/lib/systemd/system/`

```
sudo nano /lib/systemd/system/gns3.service
```

Y se introduce el siguiente contenido:

```
[Unit]
Description=GNS3 server
Wants=network-online.target
After=network.target network-online.target
[Service]
Type=forking
User=gns3
Group=gns3
PermissionsStartOnly=true
ExecStartPre=/bin/mkdir -p /var/log/gns3 /run/gns3
ExecStartPre=/bin/chown -R gns3:gns3 /var/log/gns3 /run/gns3
ExecStart=/usr/bin/gns3server --log /var/log/gns3/gns3.log \
--pid /run/gns3/gns3.pid --daemon
ExecReload=/bin/kill -s HUP $MAINPID
Restart=on-abort
PIDFile=/run/gns3/gns3.pid
[Install]
WantedBy=multi-user.target
```

En el fichero figura que usuario va a ejecutar el servicio, dónde se van a guardar los registros y la ruta absoluta del servicio a ejecutar (`/usr/bin/gns3server`). Para asegurar que el nuevo fichero se ha incluido en la biblioteca de demonios se procede a reiniciar el servicio:

```
sudo systemctl daemon-reload
```

A continuación, hay que proceder a habilitarlo y a iniciarlo para comprobar que todo ha ido correctamente:

```
sudo systemctl enable gns3
sudo systemctl start gns3
```

Finalmente se puede verificar el funcionamiento del sistema:

```
sudo systemctl status gns3
```

Permitir la redirección de dispositivos USB para máquinas virtuales

Los dispositivos siguen unas reglas que el administrador del sistema puede alterar, éstas se encuentran en el directorio `/etc/udev/rules.d/`

Por defecto, las reglas que regulan los dispositivos USB/PCI son estrictas en cuanto al uso que pueden hacer diversas aplicaciones y servicios de los estos, para poder transferir el control de un dispositivo USB o PCI a una de estas aplicaciones o servicios hay que crear una nueva regla.

Para especificar esta nueva regla se crea un nuevo fichero en el directorio de reglas:

```
sudo nano /etc/udev/rules.d/usb.rules
```

Y se copia el siguiente contenido:

```
SUBSYSTEM="usb", MODE="0666"
```

Esta regla es demasiado permisiva, pero asegura que no hay ningún problema a la hora de transferir el control a otros servicios siempre que se aplique en un entorno asegurado y controlado.

Anexo II. Test de Hipervisores

Reporte de pruebas de virtualización y transferencia de control USB para adaptadores inalámbricos sobre QEMU y openSUSE Leap 15.1

Para las pruebas se ha utilizado "*virt-manager*" como gestor gráfico de máquinas virtuales. Mediante este gestor gráfico se pueden especificar todos los parámetros de la máquina QEMU a ejecutar, abstrayéndose el usuario de todos los ficheros de configuración.

En todos los casos se ha realizado la prueba con el sistema operativo kali-*linux-2020.2-live-amd64* y el adaptador *TL-WN722N* con chipset *Atheros AR9271*; se ha utilizado esta configuración para garantizar una mayor compatibilidad entre el sistema operativo invitado y el adaptador inalámbrico.

KVM

La configuración a través de este hipervisor resulta muy sencilla, puesto que "*virt-manager*" administra adecuadamente todos los recursos. Para la configuración se ha conectado el administrador al servidor KVM local, y se ha creado una nueva máquina virtual con disco duro mínimo y arranque mediante ISO, en la que se establece como dispositivo USB anfitrión *0cf3:9271 Qualcomm Atheros Communications AR9271 802.11n.*

Al iniciar la máquina virtual, el dispositivo se reconoce inmediatamente.

Xen (Virtualización completa)

En este caso es necesario arrancar un kernel especial que incluya el hipervisor Xen. Para arrancar este kernel se elige al iniciar el host en el GRUB. Se ha iniciado la conexión local al servidor Xen con "*virt-manager*", se ha creado una máquina virtual de tipo virtualización completa y se ha aplicado una configuración equivalente a la utilizada con KVM; sin embargo, la gestión de los dispositivos USB con "*virt-manager*" parece no funcionar adecuadamente con el Hipervisor Xen porque pese a que establece la configuración, el dispositivo no es reconocido en la máquina invitada. Para una correcta configuración, se ha buscado información sobre el hipervisor y se ha llegado a la siguiente solución:

1. Establecer la máquina virtual sin ningún tipo de dispositivo USB.
2. Crear una controladora USB V2.0 de 6 puertos:

```
xl usbctrl-attach generic version=2 ports=6
```

3. Especificar el controlador y puerto del adaptador USB, verificado con *lsusb*:

```
xl usbctrl-attach generic version=2 ports=6
```

La máquina virtual reconoce el dispositivo y se puede operar con él.

Anexo III. Test Debian

Una distribución GNU/Linux independiente, confiable, estable y escalable, popular entre los usuarios avanzados debido a su excelencia técnica y su profundo compromiso con las necesidades y expectativas de la comunidad Linux.

Se ha evaluado esta distribución por su amplio gestor de paquetes, y por el gran potencial de la comunidad que se encuentra utilizando, promoviendo y desarrollando esta distribución. Debian puede operar con diferentes núcleos como el de FreeBSD o Hurd de GNU, sin embargo, donde destaca esta distribución es en combinación con el núcleo Linux, gracias al cual puede obtener la compatibilidad necesaria. La simplicidad y estabilidad son dos grandes pilares de esta distribución, además de aportar la ligereza necesaria para no demandar muchos recursos hardware.

Durante las pruebas realizadas, Debian se comportó realmente bien tanto en aspectos de compatibilidad con adaptadores inalámbricos como con instalación de controladores de terceros.

A continuación, se presenta la configuración aplicada y las pruebas de compatibilidad realizadas para evaluar el funcionamiento de este sistema. La versión de Debian seleccionada se trata de Debian Minimal 10.2.0, recomendada por las instancias de GNS3. Se trata de una versión modificada, que cuenta con redimensionamiento automático de disco y *ssh*/*nmap* integrados.

Preparación

La carga se realiza utilizando el método de aprovisionamiento automático, manteniendo los ajustes recomendados, e iniciando la instancia con el sistema ya instalado. Es necesario cambiar la distribución de teclado que por defecto viene en inglés americano (EN-US), para realizar los primeros ajustes de red con comodidad. Llevar a cabo este cambio en la configuración resulta muy sencillo puesto que Debian provee herramientas pseudo-gráficas que facilitan la gestión.

En primer lugar, se introduce el siguiente comando:

```
dpkg-reconfigure keyboard-configuration
```

Mediante las flechas del teclado se establece una nueva distribución para éste, y se procede a reiniciar el servicio y el sistema para aplicar la configuración:

```
service keyboard-setup restart
reboot
```

Configuración de red

En Debian los parámetros de red se establecen en un único fichero cuya ruta es `/etc/network/interfaces` y se establecen agrupados por interfaces de red.

Para establecer una configuración estática como es el caso se debe incluir al fichero lo siguiente:

```
    auto ens3
    iface eth3 inet static
        address 192.0.2.7
        netmask 255.255.255.0
        gateway 192.0.2.254
```

Los servidores DNS se establecen en `/etc/resolv.conf` como en la mayoría de las distribuciones.

Configuración de punto de acceso

Debian no dispone en la instalación por defecto de la herramienta para generar puntos de acceso *hostapd*, y es necesario descargarla del repositorio de paquetes oficial:

```
sudo apt-get install hostapd
```

Por defecto este servicio se encuentra protegido para prevenir su arranque por lo que hay que solicitar al sistema su habilitación:

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
```

Para utilizar controladores no libres de diferentes tipos de dispositivos como adaptadores de red inalámbricos, es necesario descargarlos del repositorio oficial de recursos no libres ya incluido en el sistema:

```
sudo apt-get install firmware-misc-nonfree
```

Una vez instalados los controladores, se procede a editar el fichero de configuración de *hostapd* para levantar una red inalámbrica con el adaptador:

En este fichero se reflejan, entre otras cosas, el código de país de operación, la interfaz, el nombre del punto de acceso, el canal de transmisión, el cifrado y otras opciones avanzadas:

```
country_code=ES
interface=wlan0
bridge=br0
ssid=DEBIAN_AP_TEST
```

```
hw_mode=g
channel=7
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Anexo IV. Test OpenBSD

Considerado uno de los sistemas operativos más seguros de tipo UNIX, resultado de una profunda auditoría de código fuente e integración de tecnología de seguridad vanguardista, como su filtro de paquetes avanzado y servicios de redes privadas en entornos distribuidos, puede presumir de haber sido víctima únicamente de dos vulnerabilidades de seguridad remotas en la instalación por defecto. [15]

Este sistema se ha evaluado por su simplicidad a la hora de aplicar complejas configuraciones de red, por su robustez y estabilidad y por su excelente documentación que ha simplificado enormemente las tareas de administración.

Si bien OpenBSD no es especialmente conocido por su soporte hardware, ha demostrado en las pruebas realizadas ser compatible con múltiples adaptadores de red en diferentes modos de operación. No obstante, este sistema no cuenta con el soporte necesario que este proyecto requiere, puesto que no hay apenas controladores disponibles para dispositivos no soportados de forma nativa. El soporte para el estándar 802.11ac se encuentra en momento de redacción de este proyecto en un estado muy precario, y no es admitido por la mayoría de los controladores disponibles.

A continuación, se recogen las configuración aplicada para el despliegue de un punto de acceso inalámbrico con acceso a Internet desde OpenBSD, un sistema de la familia BSD seleccionado como un buen candidato a albergar los puntos de acceso finales por ser un sistema robusto, ágil y seguro.

Todas las pruebas se han realizado bajo los hipervisores KVM y Xen, bajo OpenSUSE 15.1 y el emulador QEMU.

Preparación

La instalación del sistema no presenta dificultad alguna, se ha descargado la ISO de 448MB del sitio web oficial que incluye el sistema base sin ningún elemento adicional.

Se crea una nueva máquina virtual con el gestor "*virt-manager*" en la que se establecen los parámetros propuestos para la ejecución del sistema. Se comienza la ejecución de la instalación y se siguen los pasos del asistente guiado, estableciendo la distribución de teclado, la estructura de disco, la ubicación de los paquetes, la contraseña de root, etc... Una vez realizada la instalación se procede a actualizar el firmware de los dispositivos del sistema con *fw_update*.

1. Configuración de la interfaz de red LAN

La configuración de las interfaces de red en OpenBSD resulta muy sencilla y ordenada, puesto que para cada una de ellas hay simplemente que crear un archivo en `/etc` que presenta la siguiente sintaxis: `hostname.`<*interface*>. Esta estructura permite identificar errores rápidamente, así como la eliminación de configuración duplicada.

En cada fichero basta con especificar la siguiente entrada donde se especifica la versión de IP, dirección, máscara de red y broadcast:

```
inet 192.168.1.10 255.255.255.0 192.168.7.255
```

La dirección de la puerta de enlace se establece en el fichero `/etc/mygate` y los servidores DNS en el fichero `/etc/resolv.conf`

2. Configuración de la interfaz de red WLAN

Para establecer que una interfaz debe trabajar en modo Host AP, basta con crear el archivo `hostname.wlan0` en `/etc` y especificar la siguiente configuración:

```
media autoselect mode 11n mediaopt hostap chan 1
nwid OBSD_WIFI wpakey p!ssword
inet 10.0.3.1 255.255.255.0
```

En esta configuración se ha establecido que el adaptador funcionará en modo 802.11n como punto de acceso en el canal 1, el nombre de la red o SSID y la contraseña. Además, se establece una dirección estática a la interfaz.

3. Enrutamiento y NAT

Se ha deseado evaluar el funcionamiento del filtro de paquetes que ofrece OpenBSD, para aplicar reglas de cortafuegos y habilitar el servicio de traducción de direcciones. En primer lugar, es necesario habilitar enrutamiento de paquetes en el sistema, para ello, es necesario añadir al fichero de configuración `/etc/sysctl.conf` la siguiente entrada:

```
net.inet.ip.forwarding=1
```

Posteriormente, hay que habilitar el servicio NAT en PF, el firewall de OpenBSD. Para ello, se establecen las siguientes reglas en el fichero de configuración `/etc/pf.conf`

```
wired = "<wan_interface>"
wireless = "<wireless_interface>"
icmp_types = "{echoreq, unreach}"
```

```
set block-policy return
set loginterface $wireless
set skip on lo0

# NAT for wireless clients
match out on egress inet from ! (egress:network) \
to any nat-to (egress:0)

# block everything by default
block in all

# Let traffic out
pass out quick

# Let traffic in
pass in quick inet proto { tcp udp } from any to any
pass in quick inet proto icmp all icmp-type $icmp_types keep state
```

Tras esta configuración se procede a reiniciar el sistema para que se apliquen todos los cambios. Si todo se ha realizado correctamente, cualquier dispositivo con una interfaz inalámbrica deberá conectarse a Internet a través del punto de acceso creado, estableciendo las direcciones IP estáticas pertinentes. La ejecución de esta prueba permite comprobar que el punto de acceso está totalmente operativo y funciona correctamente.

Tras las pruebas realizadas, no se ha considerado un sistema adecuado para albergar la infraestructura de puntos de acceso, puesto que este sistema no garantiza la compatibilidad con adaptadores inalámbricos que se requiere y por ello, ha sido descartado.

Estos son los inconvenientes detectados:

- Soporte limitado de dispositivos
- Soporte de IEEE 802.11n limitado
- Soporte de IEEE 802.11ac limitado

No obstante, se ha considerado que éste es un sistema idóneo para ser virtualizado, debido a que presenta ciertas ventajas con respecto a otras alternativas basadas en GNU/Linux por su reducido tiempo de instalación y configuración; permitiendo la realización de numerosas pruebas en un tiempo relativamente corto. Por ello, ha quedado establecido el uso de este sistema en instancias que tengan funciones alternativas a las de puntos de acceso, en las que se aconseja la estabilidad y especialización que OpenBSD proporciona.

Anexo V. Test OpenWrt

El proyecto OpenWrt es un sistema operativo GNU/Linux dirigido a equipamientos integrados o embebidos. Generalmente los firmwares orientados a este tipo de dispositivos se caracterizan por ser totalmente estáticos, ya que no permiten utilizar el dispositivo ni realizar configuraciones más allá de las opciones que proporciona el fabricante, OpenWrt proporciona un gestor de paquetes y un sistema de archivos totalmente escribible, lo que permite personalizar el dispositivo mediante la inclusión o eliminación de componentes para adaptarse a cualquier entorno o aplicación. Ofrece imágenes precompiladas del sistema para múltiples plataformas y arquitecturas que suelen tener los *router*, como *MIPS* o *ARM*, pero también tiene soporte para *x86-64*. GNS3, ofrece una instancia preparada con el sistema ya instalado con lo que se ha desplegado este sistema.

Este sistema cuenta con una gran popularidad en el área de los *router* y puntos de acceso inalámbricos, ya que es muy superior a los firmwares que ofrece el fabricante, y se encuentra especialmente diseñado para ello. Se ha evaluado porque resulta un sistema realmente atractivo sobre el que desplegar el escenario, además de ser extremadamente ligero, lo cual resulta muy adecuado para ser virtualizado.

Las pruebas realizadas con este sistema han resultado muy satisfactorias pues, al estar orientado a enrutadores, puntos de acceso y otros equipamientos de red, está equipado con un gran catálogo de controladores para múltiples adaptadores de red, algunos modificados para ofrecer incluso más rendimiento o características añadidas [17]. Además, el rendimiento resulta excelente al ser un sistema específico para hardware con recursos muy limitados, por lo únicamente integra únicamente los componentes software necesarios.

Configuración de la red

La sintaxis a la hora de establecer los parámetros de las interfaces de red, es similar a la de los sistemas basados en Debian con alguna diferencia de nombrado. Todos los ficheros de configuración de OpenWrt se encuentran en `/etc/config`. En este caso se edita el fichero "`./network`" donde se encuentra la configuración de las interfaces de red.

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

```
config interface 'lan'
        option ifname 'eth0'
        option proto 'static'
        option ipaddr '192.168.7.10'
        option netmask '255.255.255.0'
        option gateway '192.168.7.1'
        option ip6assign '60'

config interface 'wan'
        option ifname 'eth1'
        option proto 'static'
        option ipaddr '10.0.0.10'
        option netmask '255.255.255.0'
        option ip6assign '60'
```

Para aplicar los cambios, se debe reiniciar el servicio de red con la siguiente entrada:

```
service network reload
```

Es necesario cambiar también la dirección del servidor de nombres en /etc/resolv.conf

Soporte USB

La instancia recomendada por GNS3 de OpenWrt está construida pensando en un consumo de recursos muy bajo, y por defecto no cuenta ni con soporte inalámbrico ni para dispositivos USB que hay que añadir a posteriori. Una vez la conexión a la red está operativa, es posible actualizar la lista local de paquetes del repositorio oficial de OpenWrt y proceder a instalar los controladores necesarios para añadir el soporte USB y Wi-Fi.

```
opkg update && opkg install kmod-usb-core kmod-usb-storage kmod-usb-uhci
kmod-usb-ohci kmod-usb2 kmod-usb3 usbutils kmod-mac80211 kmod-cfg80211 iw
```

Tras la instalación de estos paquetes, el soporte USB estará activo, sin embargo, esta distribución no incluye ningún controlador Wi-Fi por lo que habrá que instalarlo atendiendo al tipo de tarjeta y chipset. En la web de OpenWrt[15] se pueden encontrar los controladores necesarios para cada dispositivo con el nombre identificativo del paquete.

En este caso se realiza la instalación para adaptadores de red de la marca Broadcom:

[15] https://openwrt.org/packages/table/

```
opkg install kmod-brcmfmac
```

La habilitación del punto de acceso Wi-Fi se puede realizar mediante *hostapd* o mediante el mecanismo que incorpora internamente OpenWrt.

Anexo VI. Test FreeBSD

FreeBSD es un derivado libre y de código abierto de BSD, con un enfoque en la velocidad, estabilidad, seguridad y consistencia. Es conocido por actuar como un excelente servidor para aplicaciones y servicios, gracias a sus robustos servicios de red que le permiten maximizar la memoria y trabajar con cargas pesadas, para entregar y mantener buenos tiempos de respuesta para miles de procesos de usuarios simultáneos. [16]

Se ha evaluado este sistema por su popular uso en aplicaciones críticas de red, así como para ofrecer una alternativa con respecto a su homólogo OpenBSD, sin embargo, durante las pruebas realizadas se han detectado algunos problemas de compatibilidad siendo nulo el soporte para 802.11ac y muy limitado en 802.11n, otorgando resultados muy por debajo de sus competidores.

FreeBSD presenta características muy particulares, junto con una sintaxis y entorno de configuración exclusivos, que implican grandes diferencias con los sistemas GNU/Linux o de la propia familia BSD, sin embargo, es ampliamente conocido y utilizado en entornos de red complejos inclusive sustituyendo a enrutadores y otro equipamiento profesional dedicado, donde se requiere un sistema robusto y potente al mismo tiempo. Estas características hacen que sea posiblemente, la alternativa libre BSD con mayor documentación y comunidad.

Preparación

Para su instalación, se ha utilizado la instancia proporcionada y recomendada por GNS3, qué corresponde con la versión 12.1 de FreeBSD.

La imagen instalada por defecto, viene con la distribución de teclado **EN-US**; para cambiarla por otro idioma, es necesario especificar en el fichero `/etc/rc.conf` el parámetro *keymap*. En este caso se muestra cómo se define para la distribución de teclado en español:

```
keymap="es.acc.kbd"
```

También se puede cambiar el nombre de la máquina especificando en el mismo fichero el parámetro hostname:

```
hostname="fbsdtest"
```

Por defecto, la imagen de disco incluida con la plantilla de FreeBSD viene con un espacio muy ajustado al sistema recién instalado, provocando que no se puedan instalar controladores o paquetes de terceros. Para resolver este

inconveniente, hay que redimensionar la imagen de disco y aplicar los cambios en el sistema base:

1. Redimensionamiento

Llevar a cabo el redimensionamiento de la imagen de disco es posible mediante el menú de propiedades de la máquina, en el apartado HDD utilizando la opción "*Resize*" es posible incrementar el tamaño de la imagen en MB.

Una vez modificado el tamaño de la imagen de disco hay que comprobar su estado. Por lo general, después de un redimensionamiento, el sistema lo interpreta como corrupto por lo que hay que corregirlo:

```
root@asha:~ # gpart show
=>       3  10436725  vtbd0  GPT  (5.0G) [CORRUPT]
         3       116      1  freebsd-boot  (58K)
       119   2097152      2  freebsd-swap  (1.0G)
   2097271   8339457      3  freebsd-ufs  (3.0G)
```

Para recuperar la partición corrupta hay que aplicar el comando `gpart recover`.

```
# gpart recover vtbd0
vtbd0 recovered

# gpart show
=>       3  10436725  vtbd0  GPT  (5.0G)
         3       116      1  freebsd-boot  (58K)
       119   2097152      2  freebsd-swap  (1.0G)
   2097271   8339457      3  freebsd-ufs  (3.0G)
                              - free -  (1.0G)
```

En este momento se puede apreciar que aparece una nueva partición que no tiene espacio asignado.

Ahora es posible redimensionar el espacio de la partición "*freebsd-ufs*", en este caso con el identificador 2, para ello, es necesario ejecutar `gpart resize`. Un redimensionamiento siempre conlleva un riesgo y es posible perder toda la información del disco, por ello, ha de hacerse únicamente tras el primer inicio de la instancia o tras salvar datos y ficheros de configuración importantes.

```
# gpart resize -i 2 vtbd0
# gpart show
>        3  10436725  vtbd0  GPT  (5.0G)
         3       116      1  freebsd-boot  (58K)
       119   2097152      2  freebsd-swap  (1.0G)
   2097271   8339457      3  freebsd-ufs  (4.0G)
```

Como se puede ver, la partición "*freebsd-ufs*" ha sido correctamente redimensionada y solo queda ejecutar la utilidad "*growfs*", la cual permite expandir un sistema de archivos UFS y hacer efectivo el cambio:

```
# service growfs onestart
Growing root partition to fill device
vtbd0 recovering is not needed
vtbd0p2 resized
```

De esta forma se expande el sistema de archivos de manera automática según la detección de la utilidad, y con esto queda incrementado el espacio de la unidad principal para poder instalar utilidades y controladores adicionales.

2. Configuración de red

Los parámetros de la red se especifican de forma habitual en un único fichero de configuración `/etc/rc.conf`, este archivo contiene información descriptiva sobre el nombre del host local, detalles de configuración de cualquier posible interfaz de red y qué servicios deben iniciarse en el momento del arranque inicial del sistema.

En esta configuración se han especificado dos tarjetas de red Ethernet y una tarjeta de red inalámbrica USB soportada por el sistema:

```
ifconfig_vtnet0="inet 192.168.7.30 netmask 255.255.255.0"
ifconfig_vtnet1="inet 192.168.97.30 netmask 255.255.255.0"

# Ralink Technology USB IEEE 802.11a/g/n wireless network device
wlans_run0="wlan0"

create_args_wlan0="wlanmode hostap country ES regdomain ETSI"
ifconfig_wlan0="up"

defaultrouter="192.168.97.254"
```

Como se aprecia, se ha especificado la dirección IP de cada una de las interfaces Ethernet, para la tarjeta Wi-Fi se ha detallado cual va a ser la regla de nombrado de la interfaz, además se específica que va a operar como punto de acceso y en qué región.

Finalmente se especifica cuál es el *router* por defecto, también es posible cambiar la BSSID de la tarjeta de red inalámbrica para obtener una red de puntos de acceso con BSSID único:

```
ifconfig wlan0 ether 00:09:9c:11:22:33
```

3. Configuración de punto de acceso

La configuración del punto de acceso se refleja en el fichero `/etc/hostapd.conf` como muchas distribuciones de GNU/Linux siguiendo una sintaxis muy similar. En el especificaremos los siguientes parámetros agrupados por bloques:

- Interfaz de red
- Interfaz de control separado
- SSID
- Seguridad
- Canal de transmisión
- Parámetros avanzados de transmisión

Este es el fichero resultante:

```
interface=wlan0
ctrl_interface=/var/run/hostapd
ctrl_interface_group=wheel

ssid=ASHA1

wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP

channel=6/40
hw_mode=g
wmm_enabled=1

beacon_int=100
dtim_period=2
max_num_sta=255
rts_threshold=2347
fragm_threshold=2346
macaddr_acl=0
auth_algs=3
```

Para que el servicio inicie al arrancar el sistema, hay que añadir la siguiente línea en el fichero `/etc/rc.conf`:

```
hostapd_enable="YES"
```

Tras las numerosas pruebas realizadas, no se ha considerado un sistema adecuado para albergar los puntos de acceso, puesto que el manejo de redes inalámbricas ha resultado muy deficiente con respecto a la alternativa BSD analizada (OpenBSD), así como cualquier alternativa GNU/Linux que integra una gran cantidad de drivers en el kernel. Por todo esto, se rechaza su viabilidad como una opción adecuada a utilizar.

A continuación, se listan los inconvenientes detectados:

- Soporte limitado de dispositivos
- Soporte de IEEE 802.11n limitado
- Soporte de IEEE 802.11ac nulo
- Tamaño de disco: El tamaño de la imagen base es de 3.0GB, muy superior a los competidores, además necesita redimensionamiento, esto no es adecuado para un entorno virtual.

Anexo VII. Entorno GNS3

El servidor GNS3, necesita ser administrado mediante un cliente gráfico cuyas versiones tienen que ser idénticas para que la conexión entre ambos sea posible. Para la realización de este proyecto, el cliente se ha ejecutado sobre un equipo con el sistema operativo Windows 10; sin embargo, este aspecto carece de relevancia ya que cualquier sistema podría actuar de cliente y no repercutiría a la infraestructura de servicio principal.

Una vez definidos todos los sistemas operativos a utilizar, el siguiente paso es preparar el emulador GNS3 para desplegar dichos sistemas, establecer el reparto de recursos y aplicar configuraciones avanzadas. El aprovisionamiento o creación de las instancias virtuales que formarán parte del escenario final y sobre las que se desplegará la solución pueden realizarse de diferentes maneras:

- **Instancias aprobadas por GNS3:** Son instancias de componentes y sistemas que utilizan la configuración recomendada por GNS3, y han sido probadas exhaustivamente. Se descargan a través del *GNS3 Marketplace* y prometen una experiencia de usuario mayor y la aparición de menos errores; que si se intentan configurar los ajustes de manera manual y utilizar imágenes no verificadas.

  El problema de optar por esta vía es que no se encuentran disponibles todos los sistemas y los disponibles no suelen estar actualizados a la última versión por cuestiones de estabilidad. Esto resulta un inconveniente si se desean utilizar controladores para dispositivos que únicamente están soportados en las versiones más actualizadas de los sistemas operativos.

- **Instancias manuales:** Permiten crear una instancia virtual a partir de una imagen de disco existente con sistema instalado previamente en ella. También es posible la creación de una imagen en blanco con un tamaño de almacenamiento definido, para instalar posteriormente en ella, el sistema operativo deseado a través de una imagen de disco óptico. En ambos casos, se crea una instancia virtual de cero, y es necesario especificar todos sus recursos como la memoria RAM, la cantidad de CPUs virtuales, adaptadores de red y otras opciones avanzadas.

Se han definido las siguientes etapas de operación de las instancias virtuales con las cuales se va a trabajar:

- **PROVISIONING** o aprovisionamiento: Representa el proceso manual comentado previamente de creación y configuración inicial de la instancia virtual que será posteriormente ejecutada.

- **STAGING** o puesta en marcha: Una vez se le indica a GNS3 que se desea arrancar una instancia virtual, éste recoge toda la configuración definida por el usuario referente a memoria, CPUs virtuales, disco, adaptadores de red, tipo de consola, periféricos externos y otras opciones avanzadas. Esta fase es de vital importancia ya que en ella leerá la información necesaria para que se realice el traspaso del control de los dispositivos USB del host a la instancia virtual, a continuación, se invocará al emulador QEMU que se encargará de abastecer los recursos necesarios según la configuración establecida y procederá a arrancar la instancia.

- **RUNNING** o en ejecución: Una vez la instancia ha sido correctamente abastecida e iniciada, el estado se mantiene estable a no ser que se produzca un fallo en el hipervisor o se le indique a GNS3 la voluntad de detenerla.

- **STOPPING** o fase de detención: Si la instancia en GNS3 está configurada para realizar un apagado mediante señal ACPI[16], el sistema operativo procederá a su secuencia de apagado y habrá que esperar unos segundos a que se complete el proceso.

- **TERMINATED** o fase de resolución: En este momento la instancia virtual está totalmente parada sin ocupar recursos del host y lista para ser iniciada de nuevo. En esta fase se pueden realizar algunas configuraciones que no es posible realizar en funcionamiento o en la etapa RUNNING como la gestión de memoria, disco o adaptadores de red.

En la siguiente figura se puede ver de forma esquematizada las fases definidas para las instancias virtuales y el ciclo de vida de éstas:

[16] **ACPI**: Advanced Configuration and Power Interface

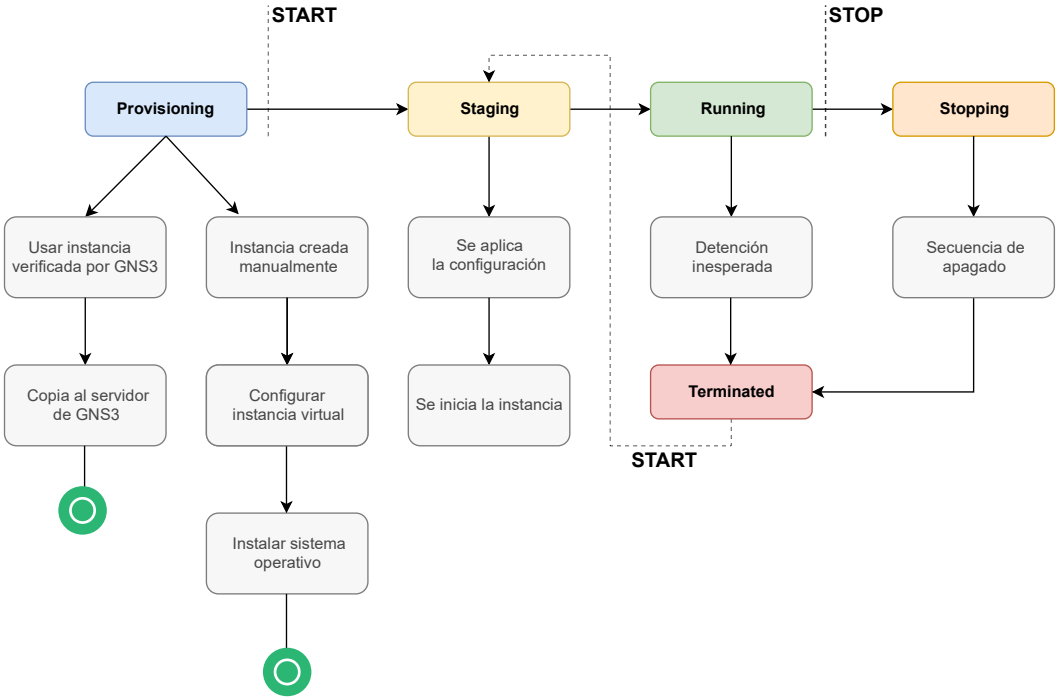


Figura 16 Ciclo de vida de las instancias virtuales de GNS3

USB Passthrough

Para transferir el control de un dispositivo USB que administra el Host, a una instancia virtualizada a través de GNS3, es necesario aplicar a ésta una configuración avanzada que será leída por el hipervisor al momento de ser invocada.

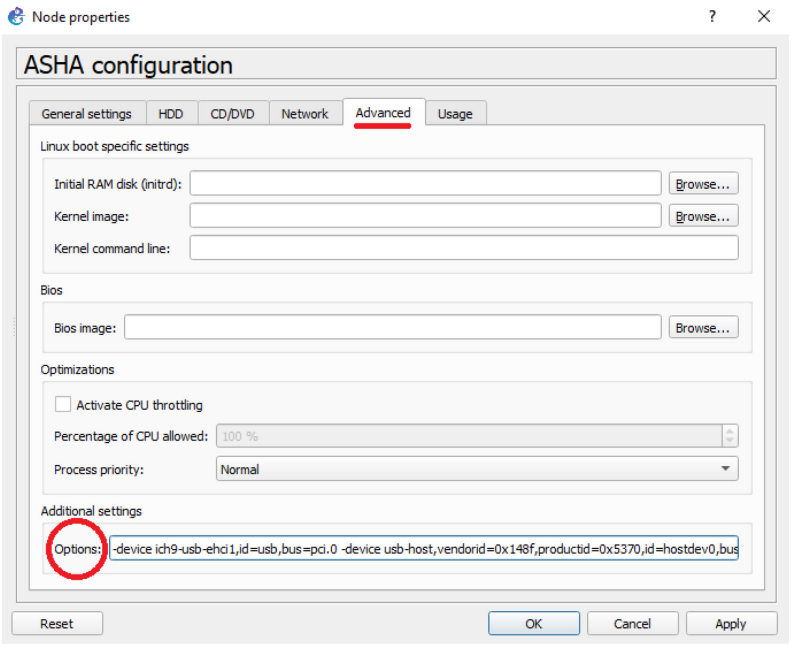


Figura 17 Configuración avanzada en una instancia de GNS3

En el apartado de opciones adicionales se han de insertar dos entradas, una para el controlador y otra para el host, que será diferente para cada dispositivo.

En primer lugar, se define un tipo de controlador por defecto que se desea utilizar, este este caso "*ich9-usb-ehci1*" conectado al bus PCI:

```
-device ich9-usb-ehci1,id=usb,bus=pci.0
```

En segundo lugar, se define el tipo de dispositivo USB a conectar al controlador atendiendo a su identificador de producto y de proveedor. Esta información se puede obtener ejecutando el comando "*lsusb*" en sistema anfitrión con el dispositivo conectando y anotando los valores separados por ":".

En este caso se está utilizando un adaptador de red *Atheros AR9271*:

```
-device usb-host,vendorid=0x0cf3,productid=0x9271,id=hostdev0,bus=usb.0
```

Con esta configuración los dispositivos ya pueden funcionar en las instancias virtuales.

Apagado ACPI

Las instancias virtuales pueden sufrir daños en su sistema de archivos, al igual que en los equipos físicos si el apagado de éstas no se realiza correctamente. Por defecto GNS3 tiene establecido el apagado forzado en todas sus máquinas, por lo que se recomienda cambiarlo para prevenir estos problemas.

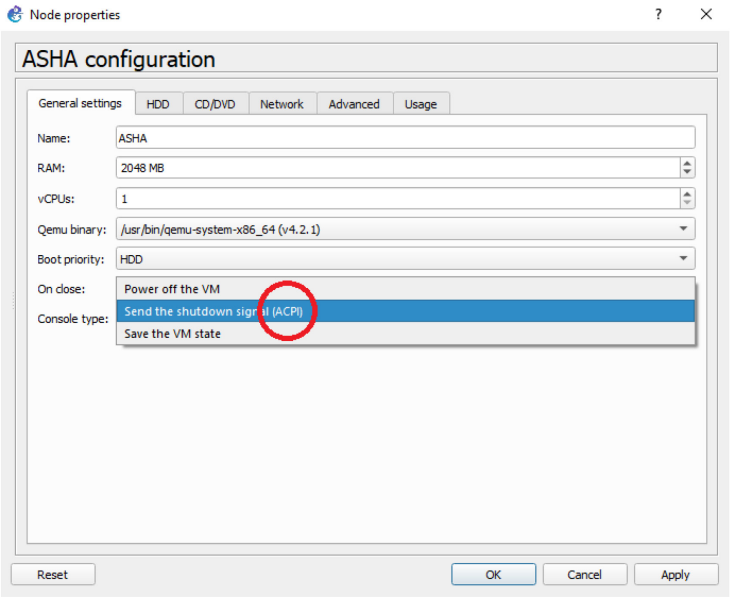


Figura 18 Configuración general en una instancia de GNS3

Anexo VIII. Inyección de paquetes

Uno de los requisitos de la solución LVAP, es la compatibilidad del adaptador inalámbrico con la capacidad de inyección de paquetes en modo *Monitor,* principalmente en el modo de 5 GHz o 802.11ac.

La mayoría de los adaptadores del mercado (con su correspondiente controlador para el kernel de Linux), soportan operar en modo *Monitor* para escuchar cualquier comunicación en el aire; sin embargo, este modo de funcionamiento pasivo no permite de forma estándar enviar datos. Para ello, se necesita la inyección de paquetes.

El envío y la recepción de tramas de gestión y control, es necesario para suplantar las estaciones base y los clientes, y para escuchar las tramas que están destinadas a adaptadores concretos. En el marco de auditoría inalámbrica, es necesario para realizar ataques de desautenticación; puede ser usado para capturar el *handshake*[17] de 4 vías de WPA, para forzar a un usuario a un AP malicioso, para recuperar un SSID oculto, e incluso generar marcos ARP para un ataque de repetición de WEP.

Por lo tanto, la inyección de paquetes y el modo *Monitor* son dos funcionalidades que proporcionan un control de muy bajo nivel, necesarias y utilizadas en la aplicación del proyecto LVAP que se desea desplegar y por lo que deben estar soportadas.

El software utilizado para evaluar esta funcionalidad es *Aireplay-ng*, que forma parte de la suite de auditoría inalámbrica de código abierto A*ircrack-ng* y permite realizar pruebas de inyección desde la versión 0.9 en adelante.

El hardware utilizado para realizar las pruebas se trata del **Alfa AWUS036ACH** con el chipset **Realtek RTL8812AU**, un potente adaptador Wi-Fi soportado desde 2017. Dispone de antenas duales y soporta los estándares 802.11ac, a, b, g y n compatibilidad con 300 Mbps a 2,4 GHz y 867 Mbps a 5 GHz. [18] Reconocido por ser uno de los primeros adaptadores en soportar inyección de paquetes en 802.11ac, su uso se encuentra ampliamente extendido y es muy utilizado por la comunidad para realizar labores de auditoría, junto con la distribución Kali GNU/Linux orientada a seguridad.

[17] **Handshake**: Un intercambio de señales entre dos dispositivos cuando se inician las comunicaciones para asegurar la sincronización.

La prueba de inyección determina si la tarjeta puede inyectar con éxito, y determina los tiempos de respuesta de *ping* al Punto de Acceso (AP).

El programa inicialmente envía peticiones *broadcast*[18] a todos los AP que se encuentren al alcance y estén escuchando para que respondan con una descripción de sí mismos. No todos responderán a este tipo de petición, se elabora una lista de AP que responden para ser utilizada en los pasos siguientes, si algún AP responde, se muestra un mensaje por pantalla indicando que la tarjeta puede inyectar con éxito.

Al mismo tiempo, cualquier AP identificado a través de un paquete *beacon*[19] también se añade a la lista que se procesará en los pasos siguientes.

A continuación, por cada AP de la lista, se envían 30 paquetes dirigidos a un AP especifico. El recuento de las respuestas recibidas más el porcentaje se muestra por pantalla. Esto indica si puede comunicarse con el AP y con qué calidad de enlace. [19]

A continuación, se muestra el procedimiento llevado a cabo con la utilidad para evaluar el funcionamiento de la inyección. En primer lugar, se inhabilita el tráfico IP de la interfaz para cambiar su modo de operación:

```
sudo ifconfig wlx00c0caa11a7f down
```

Se especifica con *iwconfig* el modo de operación de la tarjeta, se rehabilita el tráfico IP de la interfaz y se selecciona el canal en el que deberá escuchar el adaptador:

```
sudo iwconfig wlx00c0caa11a7f mode monitor
sudo ifconfig wlx00c0caa11a7f up
sudo iwconfig wlx00c0caa11a7f channel 100
```

El dispositivo ya está listo para realizar la prueba de inyección que se llevará a cabo con *aireplay-ng*, especificando como parámetro que se va a realizar una prueba de inyección, y en qué interfaz:

```
sudo aireplay-ng -9 wlx00c0caa11a7f
19:16:38  Trying broadcast probe requests...
19:16:38  Injection is working!
```

[18] **Broadcast**: Transmisión de datos que serán recibidos por todos los dispositivos en una red.

[19] **Beacon**: Paquetes que contienen toda la información sobre el punto de acceso y son transmitidos periódicamente para anunciar la presencia de la red inalámbrica.

```
19:16:40  Found 5 APs

19:16:40  Trying directed probe requests...
19:16:40  72:51:63:D3:91:1C - channel: 100 - 'Invitado-D0AB'
19:16:43  Ping (min/avg/max): 5.693ms/91.151ms/185.281ms Power: -74.55
19:16:43  20/30:  66%

19:16:43  72:51:63:D3:91:1D - channel: 100 - 'MiFibra-D0AB-5G'
19:16:48  Ping (min/avg/max): 10.186ms/99.387ms/122.692ms Power: -74.60
19:16:48  15/30:  50%

19:16:48  30:93:BC:99:7D:45 - channel: 100 - ''
19:16:52  Ping (min/avg/max): 4.088ms/46.057ms/184.778ms Power: -66.93
19:16:52  14/30:  46%

19:16:52  30:93:BC:99:7D:46 - channel: 100 - 'MiFibra-7FAA'
19:16:56  Ping (min/avg/max): 4.638ms/65.142ms/195.729ms Power: -66.79
19:16:56  14/30:  46%

19:16:56  72:51:63:D3:91:1F - channel: 100 - 'MiFibra-D0AB'
19:17:00  Ping (min/avg/max): 5.521ms/73.773ms/119.833ms Power: -74.58
19:17:00  12/30:  40%
```

Como se puede apreciar, se muestra el mensaje de que la inyección de paquetes ha resultado satisfactoria, además, se visualiza la lista de AP con los que se ha realizado la prueba, junto con su calidad de enlace y número de paquetes inyectados con éxito.

Adicionalmente, se han realizado pruebas con herramientas como Fluxion[20] bajo Kali Linux para verificar las características de la inyección.

[20] https://github.com/FluxionNetwork/fluxion

Anexo IX. Reglas de flujos OpenFlow

Para cada uno de los *switch* Open vSwitch, se habilita el protocolo STP[21] con el objetivo de gestionar automáticamente la presencia de bucles en la red, y se establece el modo de operación como "*secure*" para que no se autoconfiguren los flujos cuando no exista una conexión con un controlador.

```
ovs-vsctl set br br0 stp_enable=true
ovs-vsctl set-fail-mode br0 secure
```

A continuación, se reflejan los flujos creados para redistribuir el tráfico según el destino de los paquetes.

- **Conmutador oeste**

```
# Critical route
ovs-ofctl add-flow br0 dl_type=0x0800,nw_dst=155.210.0.0/16,actions=output:2
# To P/AERI
ovs-ofctl add-flow br0 arp,nw_dst=10.10.10.10,actions=output:1
ovs-ofctl add-flow br0 dl_dst=00:09:9c:ca:fe:00,actions=output:1
# To BROADCAST
ovs-ofctl add-flow br0
dl_type=0x0806,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:all
# To DEMI
ovs-ofctl add-flow br0 dl_dst=00:09:9c:11:11:11,actions=output:3
# To MAYA
ovs-ofctl add-flow br0 dl_dst=0c:84:b9:9e:90:01,actions=output:4
# To ASHA
ovs-ofctl add-flow br0 dl_dst=0c:84:b9:76:78:01,actions=output:7
# To ARIA
ovs-ofctl add-flow br0 dl_dst=0c:49:02:79:fa:01,actions=output:7
# To DEV1
ovs-ofctl add-flow br0 dl_dst=c2:62:bb:82:bf:3f,actions=output:7
# To DEV2
ovs-ofctl add-flow br0 dl_dst=f4:f2:6d:19:2e:39,actions=output:3
```

- **Conmutador este**

```
# Reset flows
ovs-ofctl del-flows br0
# Critical route
ovs-ofctl add-flow br0 dl_type=0x0800,nw_dst=155.210.0.0/16,actions=output:2
# To P/AERI
ovs-ofctl add-flow br0 arp,nw_dst=10.10.10.10,actions=output:1
ovs-ofctl add-flow br0 dl_dst=00:09:9c:ca:fe:00,actions=output:1
```

[21] **S**panning **T**ree **P**rotocol

```
# To ARIA
ovs-ofctl add-flow br0 dl_dst=0c:49:02:79:fa:01,actions=output:4
# To DEMI
ovs-ofctl add-flow br0 dl_dst=00:09:9c:11:11:11,actions=output:7
# To FBSDAP
ovs-ofctl add-flow br0 dl_dst=0c:84:b9:76:78:01,actions=output:3
# To MAYA
ovs-ofctl add-flow br0 dl_dst=0c:84:b9:9e:90:01,actions=output:7
# To BROADCAST
ovs-ofctl add-flow br0
dl_type=0x0806,dl_dst=ff:ff:ff:ff:ff:ff,actions=output:all
# To DEV1
ovs-ofctl add-flow br0 dl_dst=c2:62:bb:82:bf:3f,actions=output:4
# To DEV2
ovs-ofctl add-flow br0 dl_dst=f4:f2:6d:19:2e:39,actions=output:7
```

Para configurar la tunelización hay que especificar en cada uno de los extremos la siguiente configuración cambiando la dirección IP remota:

```
ovs-vsctl add-port br0 eth15 -- set Interface eth15 type=gre
options:remote_ip=192.168.97.1
ifconfig eth15 192.168.13.1 netmask 255.255.255.0 up
route add default gw 192.168.13.254
```

- **Conmutadores perimetrales (norte y sur)**

```
# Reset all flows
ovs-ofctl del-flows br0

# To BROADCAST
ovs-ofctl add-flow br0 in_port=1,dl_type=0x0806,actions=output:3
ovs-ofctl add-flow br0 in_port=2,dl_type=0x0806,actions=output:3
ovs-ofctl add-flow br0 dl_dst=00:09:9c:ca:fe:00,actions=output:3

# To DEBIAP
ovs-ofctl add-flow br0 dl_dst=0c:49:02:6b:2c:01,actions=output:1
# To FBDSAP
ovs-ofctl add-flow br0 dl_dst=0c:49:02:ef:21:01,actions=output:2
# To OBSDAP
ovs-ofctl add-flow br0 dl_dst=0c:49:02:f3:9c:01,actions=output:1
# To OWRTAP
ovs-ofctl add-flow br0 dl_dst=0c:49:02:79:fa:01,actions=output:2
# To DEV1
ovs-ofctl add-flow br0 dl_dst=c2:62:bb:82:bf:3f,actions=output:2
# To DEV2
ovs-ofctl add-flow br0 dl_dst=f4:f2:6d:19:2e:39,actions=output:1
```

Anexo X. Script generador de reglas

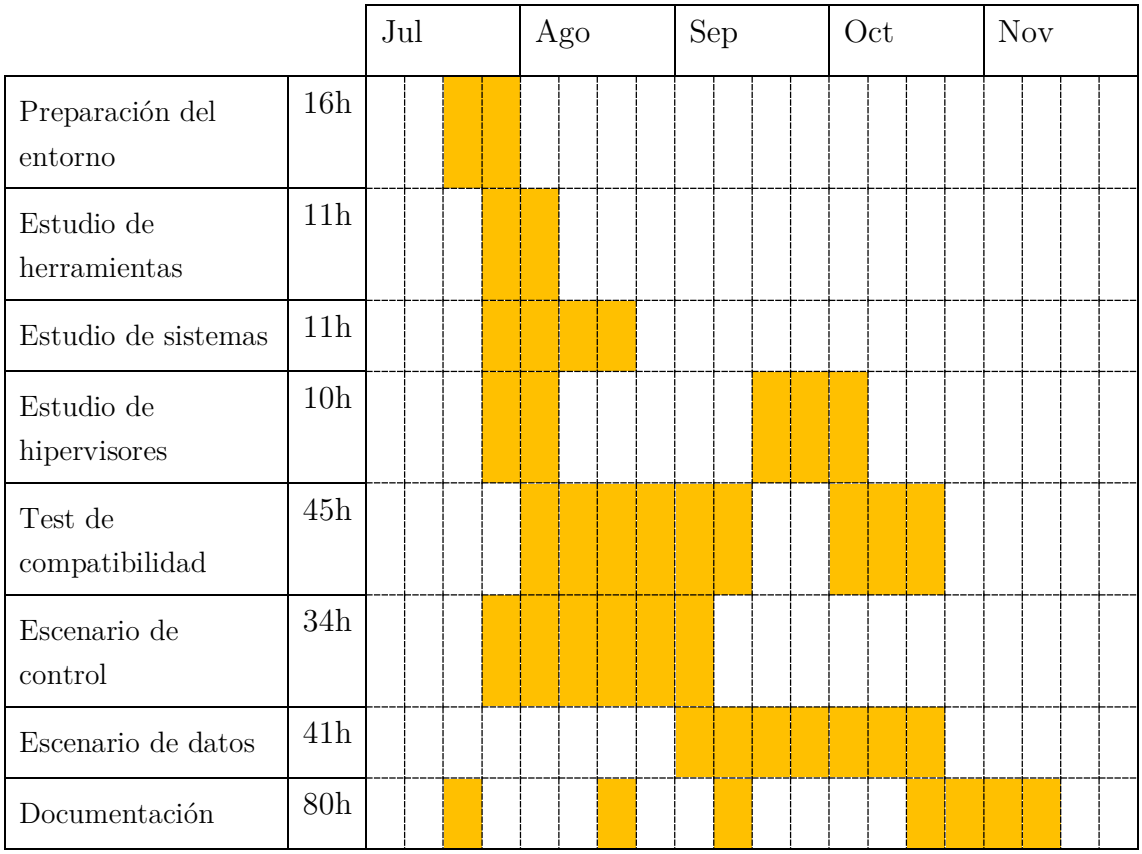
Se ha desarrollado un pequeño script en lenguaje **Ruby**, con el que a partir de una lista de bloques de direcciones IP, se generan las reglas de filtrado como comandos OpenFlow para ser directamente introducidos en Open vSwitch.

Este lenguaje permite desarrollar una solución en tan solo 7 líneas de código:

```
1 File.open("filter.txt", "w+") do |f|
2       File.open("blocks.txt", "r").each_line do |line|
3             unless line.to_s.strip.empty?
4                   f.puts ("ovs-ofctl add-flow br0
  dl_type=0x0800,nw_dst=" + line.strip + ",actions=output:2")
5             end
6       end
7 end
```

Anexo XI. Gestión del proyecto

La labor de realización y documentación de este proyecto se le estima una duración real de 248 horas, distribuidas según el diagrama que se presenta a continuación:



Cabe destacar que tanto las pruebas de compatibilidad como diseño e implementación de la red de datos abarcaron más tiempo de desarrollo del previsto, resultando ser las labores más complejas del proyecto. No obstante, las tareas de configuración e implementación de la red y los sistemas en el escenario de control se llevaron a cabo más rápido de lo pronosticado, permitiendo adelantar mucho trabajo en poco tiempo.