Estudio de las soluciones SD-WAN para la protección de las infraestructuras tecnológicas de las organizaciones

Manuela Calvo Barrios

Máster Universitario en Ingeniería de Telecomunicación

Área de Telemática

Nombre Tutor/a de TFM

José López Vicario

Profesor/a responsable de la asignatura

Xavier Vilajosana Guillén

Julio 2023

Universitat Oberta de Catalunya

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Estudio de las soluciones SD-WAN para la protección de las infraestructuras tecnológicas de las organizaciones*
Nombre del autor:	*Manuela Calvo Barrios*
Nombre del consultor/a:	*José López Vicario*
Nombre del PRA:	*Xavi Vilajosana Guillen*
Fecha de entrega (mm/aaaa):	*07/2023*
Titulación o programa:	Máster Universitario en Ingeniería de Telecomunicación
Área del Trabajo Final:	*Area de Telemática*
Idioma del trabajo:	*Castellano*
Palabras clave	*Redes, software, seguridad*

Resumen del Trabajo

El objetivo principal del trabajo es conocer cómo funcionan las soluciones SD-WAN y cómo pueden utilizarse para la protección de las infraestructuras tecnológicas de las organizaciones. En estos tiempos y debido a la transformación digital, las empresas y organizaciones están redefiniendo el despliegue de sus infraestructuras hacia modelos híbridos, entre los que deben convivir los sistemas propios (*on-premise*), las arquitecturas cloud, las infraestructuras de terceros (como proveedores y clientes), los usuarios remotos, entre otros. Las soluciones SD-WAN se diseñaron para apoyar

a las empresas en los cambios motivados por la transformación digital. En estas soluciones se aprovechan las funcionalidades propias de las redes definidas por software, en las que la división del plano de datos y el plano de control de la red permite abordar automáticamente y de forma dinámica incrementos de tráfico, fallos en la red ocasionados por configuraciones incorrectas, indisponibilidad de algún componente de la red o redimensionamiento de esta. El papel que juegan estas soluciones en la securización de las infraestructuras tecnológicas pasa por centralizar en la SD-WAN las comunicaciones entre sistemas, sedes y usuarios, con el fin de conseguir, mediante la aplicación de políticas previamente definidas y aplicadas por el software controlador, una comunicación segura entre entornos. En este trabajo se abordarán los principales aspectos de configuración y funcionamiento de la solución SD-WAN y se estudiará qué papel juegan en el futuro de la protección de infraestructuras tecnológicas.

Abstract

The major aim of this project is to know how SD-WAN solutions work and how they can be used to protect the organizational infrastructures. Nowadays and due to the digital transformation, enterprises and organizations are redefining their infrastructure designs to hybrid models, where legacy architecture, cloud systems, third parties' infrastructure (of providers or clients), remote users and others must coexist. SD-WAN solutions have been designed to support the organizations changes, result of the digital transformation processes. These solutions are based on the software defined networks capabilities, where the division of control and data planes let solve automatic and dynamically traffic growths, network misconfiguration failures, network components unavailability or network resizing. SD-WAN technology plays the role of centralize the communications between systems, headquarters, and users to ensure the technological infrastructure security. For this purpose, policies are defined and applied by the controller software of the network. In this project, key aspects of configuration and functioning of SD-WAN solutions and the role they will play in the future of technological infrastructures protection will be described.

Índice

Lista de tablas

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Las redes definidas por software (SDN) surgieron hace unos años para dar respuesta a los problemas que presentan las redes tradicionales de configuración manual, poca tolerancia a fallos e indisponibilidades y baja escalabilidad. Durante los últimos años han surgido nuevas aplicaciones para las SDN, entre las cuales se encuentran las soluciones de redes de área amplia definidas por software (SD-WAN).

La evolución de las tecnologías cloud dentro de las infraestructuras de las empresas y organizaciones ha puesto en el centro del mercado a las soluciones SD-WAN, que pasa a ser considerada actualmente una tecnología con potencial para revolucionar el uso de los servicios de red de área amplia (WAN). Especialmente interesante es el concepto de las redes basadas en aplicaciones, que ha surgido con las SD-WAN, promete dotar con mayor agilidad las necesidades de las aplicaciones, los servicios y los clientes. [14, pág. 1]

Algunos de los proveedores de seguridad y redes más importantes a nivel mundial, como es Cisco, ya están ofreciendo soluciones SD-WAN basadas en soluciones *Software as a Service* (SaaS), en las que los clientes solo tienen que preocuparse por conectar sus sedes locales y la infraestructura que tienen desplegada en la nube a la SD-WAN. La flexibilidad de estas soluciones permite que los clientes no tengan que preocuparse por la configuración y el mantenimiento de equipos, ya que todo ello queda a cargo del proveedor, y la configuración dinámica de la red y el direccionamiento de paquetes son realizados por el controlador de la red definida por software. [4]

Uno de los principales atractivos de las soluciones SD-WAN para las infraestructuras de las empresas es la centralización de las comunicaciones y la aplicación de políticas de seguridad únicas a toda la infraestructura. la plataforma SD-WAN proporciona protección en el perímetro de la red, interceptando y aislando cada paquete de las comunicaciones, para posteriormente redirigirlo a la zona apropiada o al segmento de la red a la que está destinado, en caso de que la comunicación sea aceptada por las políticas de seguridad definidas. [12, pág. 14 – 15]

Las infraestructuras tecnológicas (IT) de las empresas actuales reciben conexiones de cualquier punto de internet: pueden provenir de sedes *on-premise,* de infraestructura cloud, de dispositivos *Internet of Things* (IoT*)* sin agente, de trabajadores remotos y de otros clientes o terceros con los que su infraestructura está conectada. El rápido éxito que han tenido las arquitecturas cloud, sumado al incremento del ancho de banda y a los servicios 4G y 5G como red de transporte WAN, ha hecho más difícil el establecimiento de medidas de seguridad en las infraestructuras tecnológicas. Las nuevas tendencias del mercado respecto a la seguridad se apoyan en llevar el perímetro de IT a la nube, como es el caso de las soluciones SD-WAN, proporcionando una solución ágil para todos estos problemas y permitiendo apoyar a las empresas en su estrategia de transformación tecnológica. [12, pág. 14 – 15]

La realización de este trabajo se enmarca en esas necesidades que tienen actualmente las empresas. Como se ha mencionado previamente, en estos tiempos las arquitecturas

tecnológicas de las organizaciones están sufriendo procesos de transformación digital en los que se está migrando servicios e infraestructura a sistemas cloud, además de a otras localizaciones geográficas. Con este trabajo se pretende simular la realidad de estas empresas, y estudiar cómo una WAN definida por software es capaz de manejar y centralizar las principales medidas de seguridad para tener en cuenta la supervivencia y buen funcionamiento de los negocios.

1.2. Objetivos del Trabajo

El principal objetivo de este trabajo es demostrar cómo las soluciones SD-WAN son capaces de automáticamente manejar las comunicaciones y las medidas de seguridad que afectan a toda la infraestructura tecnológica de las empresas.

Para conseguir este objetivo se aprovechará una de las principales características de las redes definidas por software. Estas redes generalmente trabajan en el nivel dos del modelo OSI, es decir, los elementos de red que serán configurados por el controlador son switches. Con el trabajo se quiere comprobar cómo el controlador de la red, que hace las funciones del plano de control, es capaz de dirigir el funcionamiento de los switches de capa dos, que se encargan de realizar las funciones del plano de datos. Además, las características propias de las medidas de seguridad también serán realizadas por los switches; pero definidas desde el controlador, que es el que orquesta toda la lógica y el direccionamiento de los paquetes dentro de la SD-WAN.

En la SD-WAN se deberá configurar los balanceadores de carga con algoritmos de planificación de recursos, dado que estarán situados en la primera línea del perímetro de la infraestructura y recibirán todo el tráfico procedente de la red pública o externa. Seguidamente, se deberá filtrar el tráfico que llega a los switches para permitir, denegar o rechazar las comunicaciones, como haría un firewall. Finalmente, deberá configurar las tablas de flujos de los switches de direccionamiento de la red, para que los paquetes aceptados por la red sean dirigidos hacia su correspondiente destino.

Dado el abordaje técnico que tiene el trabajo, a continuación, se detallan los objetivos específicos que se pretenden cubrir:

- Aislamiento de las zonas de la red mediante redes virtuales VLANs para segmentar el tráfico de la red.
- Separación de la infraestructura en tres zonas: red interna, zona desmilitarizada y red externa; divididas por elementos de seguridad robustos, como firewalls, y uso de balanceadores de carga.
- Control de los recursos disponibles y de los flujos de comunicaciones recibidos del exterior mediante el descarte de paquetes, para prevenir que la congestión en la red ocasione indisponibilidades.
- Comparación de los tiempos de respuesta presentes en las comunicaciones realizadas con los diferentes componentes de seguridad de la red.
- Mejora de la eficiencia/rendimiento de la red que introduce la configuración de medidas de seguridad en la SD-WAN. Las SD-WAN, entre sus principales beneficios, cuentan con la mejora de la eficiencia del uso de la red debido a menores congestiones de tráfico, por lo que con los mismos recursos físicos prometen un mayor ancho de banda disponible para los servicios. [14, pág. 1]

Concretamente, se abordará mediante la comparación de la tasa de transmisión máxima disponible en la red sin medidas y en la red securizada, en la que se quiere comprobar si la tasa mejora en al menos un 10%.
- Medición del porcentaje de bloqueo de las comunicaciones no deseadas y del riesgo de ataques exitosos de la red.

Se necesitará la realización de las siguientes tareas para poder alcanzar los objetivos generales y específicos indicados en los párrafos anteriores:

- Conocer los fundamentos teóricos sobre los que se asientan las redes SDN.
- Estudiar cuáles son los componentes que conforman las infraestructuras tecnológicas de las organizaciones y cómo se les dota de medidas de seguridad.
- Definir una topología de red y un script Python que permita la simulación de la infraestructura de prueba en la herramienta Mininet.
- Analizar el comportamiento técnico de la zona WAN definida por software en una red generada en Mininet.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

La motivación que ha llevado a seleccionar este tema ha sido la necesidad de encontrar soluciones de seguridad que se puedan emplear para proteger la información de las personas y los entornos tecnológicos de los diferentes tipos de organizaciones: gobiernos, empresas, instituciones, ONGs, entre otros. A continuación, se incluye el detalle del impacto que se espera del proyecto en las diferentes dimensiones del compromiso ético y global.

El resultado de este trabajo se espera que tenga un impacto en la dimensión de la sostenibilidad, ya que con él se pretenden estudiar cómo las configuraciones más eficientes a nivel de infraestructura y de seguridad de los recursos tecnológicos de las empresas pueden ayudar a reducir el impacto ambiental mediante el menor uso de recursos energéticos. Además de esto, el proyecto se encuentra encuadrado dentro del Objetivo de Desarrollo Sostenible número 9, “Industria, innovación e infraestructura”, y también está parcialmente ligado al objetivo número 12, “Consumo responsable y producción”.

Respecto a la dimensión del comportamiento ético y de responsabilidad social, uno de los objetivos del trabajo es dotar de medidas de ciberseguridad a las infraestructuras de las empresas. Recordemos que en ella se albergan datos de clientes y de otras instituciones y que, además, estas empresas suelen disponer de cadenas de producción de productos y servicios que, por culpa de fallos o ataques informáticos, podrían llegar a situaciones de desaprovechamiento de recursos o incluso acabar con la vida de personas. Para evitar estas situaciones, existe normativa y legislación que obligan a cumplir con ciertas medias de seguridad física e informática, y precisamente es trabajo de la ciberseguridad proteger a las empresas del robo de los datos sensibles de las personas y también evitar errores que puedan llegar a provocar desgracias humanas. Por otro lado, proteger las infraestructuras de las empresas también es necesario para que puedan continuar con su actividad económica. En este sentido, el proyecto encaja con el ODS número 8, “Trabajo decente y crecimiento económico”, ya que las empresas con su actividad contribuyen e impactan en la sociedad generando puestos de trabajo y capital. Pero no solo la seguridad

aplica a las empresas y a la economía, sino también a los gobiernos y organizaciones gubernamentales ya que, en el mundo completamente globalizado y conectado en el que vivimos, es necesario garantizar su buen funcionamiento y la agilidad de los procesos tecnológicos para no colapsar los servicios públicos. Por tanto, el ODS 16, “Paz, justicia e instituciones fuertes”, está también directamente relacionado con los objetivos del trabajo.

Finalmente, respecto a la dimensión de diversidad, género y derechos humanos, como se ha comentado anteriormente, el proyecto está relacionado con la privacidad y los datos y, más concretamente, con el hecho de proteger estos datos en el caso en el que pertenezcan a personas físicas. En este sentido, las empresas muchas veces tratan con información de personas que están en riesgo de exclusión social, son maltratadas, sufren violencia de género, son discriminadas por su procedencia, características físicas u orientación sexual, y es necesario y extremadamente importante que su información no trascienda al conocimiento público para evitar que se vean en peligro. Aunque directamente no esté relacionado con el ODS número 10, “Reducir la desigualdad”, sí que podemos decir que ayuda a evitar que aumenten las desigualdades sociales.

En resumen, con el estudio de las soluciones SD-WAN del proyecto se pretende conseguir proteger a las personas y a los recursos, tanto en el caso de la sostenibilidad, como del comportamiento ético, de la conciencia social, diversidad, género y derechos humanos. El impacto que tiene el trabajo sobre estas dimensiones es inherente a la propia solución, ya que, de ser positiva su evaluación y poderse considerar una buena opción para proteger organizaciones, contaríamos con una opción más para dotar de seguridad a las infraestructuras tecnológicas.

1.4. Enfoque y método seguido

Aunque en el mercado hay fabricantes que permiten probar sus soluciones SD-WAN para poder comprobar cuáles son sus prestaciones en materia de redes y de seguridad, en este trabajo se ha decidido emplear el emulador Mininet para generar una topología de red que permita realizar las pruebas prácticas necesarias para aproximarse a los objetivos definidos para el TFM. Se ha optado por esta herramienta por su naturaleza de código abierto, que permite que cualquier interesada o interesado se familiarice con ella y pueda reproducir el contenido del trabajo.

Dos son las posibilidades que ofrece Mininet para su uso. La primera de ellas consiste en descargar una máquina virtual sin interfaz gráfica que ya se encuentra preconfigurada y lista para su funcionamiento [8]. La segunda, y por la que se ha optado en este trabajo, es instalar una máquina virtual de sistema operativo Ubuntu 22.04, sobre la que luego se ha instalado el emulador Mininet. El motivo por el que se ha seleccionado esta opción es debido a que proporciona una eficiencia mayor y hace más sencilla la creación de un script en Python directamente dentro de la interfaz gráfica que proporciona la máquina virtual de Ubuntu 22.04.

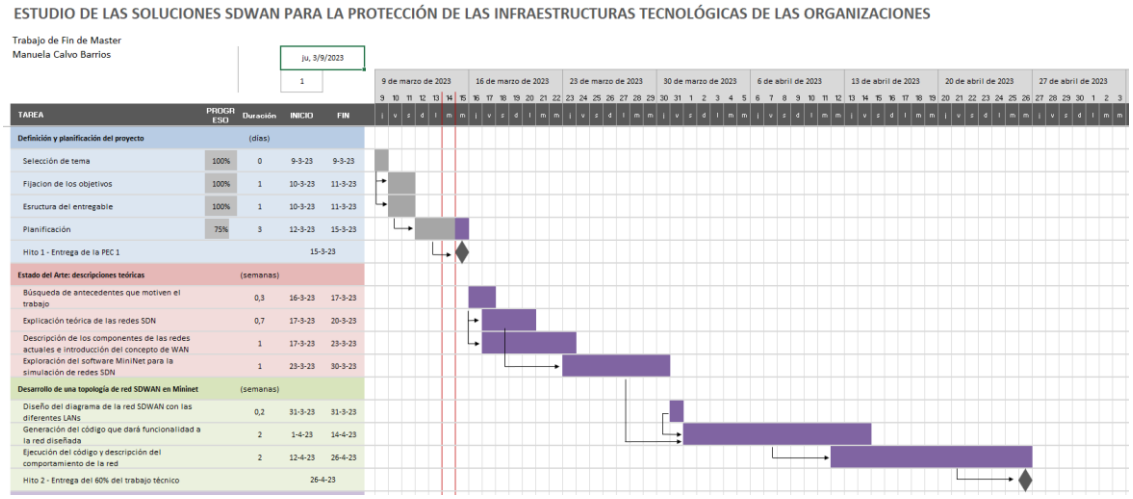
Las necesidades de este trabajo requieren poder customizar una topología de red definida por software. Mininet es una herramienta que permite realizar esta automatización del diagrama de red y aplicar la funcionalidad que sea necesaria de seguridad, sin necesidad de crear un controlador. Este es un punto crítico dentro de la arquitectura definida por software, ya que el controlador como es quien realiza toda la función del plano de control, y no deja de ser un programa informático. Que Mininet proporcione un controlador por

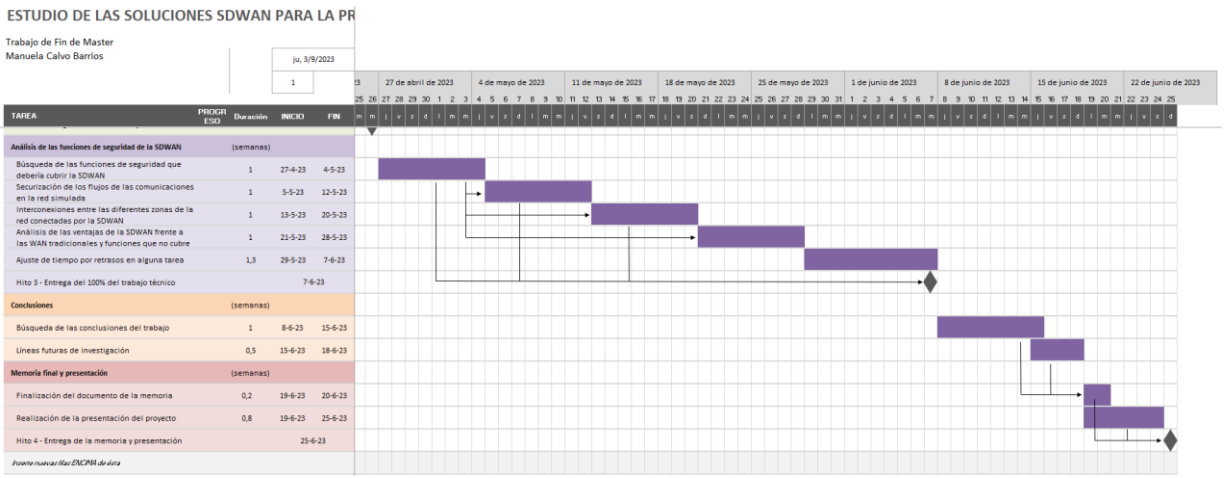
defecto dentro de la solución, o incluso que permita utilizar la topología de red customizada dentro de Mininet con controladores remotos de otros fabricantes, supone una mayor libertad a la hora de probar diferentes topologías de red o funcionalidades.

Dados los objetivos de este trabajo, y teniendo en cuenta lo mencionado en los párrafos previos, se considera que la elección del software Mininet es la adecuada para llevar a cabo un análisis de seguridad de las redes de área amplia definidas por software.

1.5. Planificación del Trabajo

A continuación, se presenta el diagrama de Gantt previsto para este trabajo.





Serán necesarios los siguientes recursos para su realización:

- Software de VirtualBox para instalar y ejecutar máquinas virtuales.
- ISO de Ubuntu 22.04, sobre la que se instalará Mininet.
- Herramienta Mininet de emulación de redes SDN.
- Controlador por defecto proporcionado en Mininet.
- Controlador de SDN.

Durante la realización del trabajo, se encontró un obstáculo que retrasó algunas semanas la finalización del script de configuración de la topología, que debería haber estado bastante avanzado para la entrega del Hito 2. El motivo de este retraso fue que el sistema operativo usado no era compatible con las funcionalidades necesarias para ejecutar una máquina virtual, además de haber otros problemas relativos a los recursos necesarios para la virtualización del software Mininet. Este obstáculo se superó con el cambio del PC de pruebas por otro más moderno. Una vez completado el script, se siguió con el resto de las pruebas y se logró alcanzar la planificación prevista para el Hito 3.

Antes de la entrega de la PEC 4, se añadieron nuevos indicadores de riesgos a las mediciones de resultados, indicadores de bloqueo de las comunicaciones y se definió un último objetivo basado en ellos.

1.6. Breve sumario de productos obtenidos

Como resultado de este trabajo se espera obtener un script que configure la topología de una red formada por varias sedes en puntos geográficos distintos, infraestructura en proveedor cloud y usuarios remotos. Las comunicaciones entre estas entidades mencionadas se realizarán a través de una zona SD-WAN, en la que se centralizará todo el tráfico de la red entre sedes y se configurarán las políticas generales de seguridad. Una vez generada esta topología, se procederá a analizar su comportamiento y a verificar las medidas de seguridad configuradas en la zona SD-WAN. El producto final del proyecto es obtener esta configuración de red completamente securizada definida por software y detallar su funcionamiento.

1.7. Breve descripción de los otros capítulos de la memoria

El proyecto consta de tres capítulos de contenido y un último capítulo de conclusiones y líneas para trabajos futuros.

En el primero de los capítulos, se aborda el estado del arte que enmarca este trabajo, en el que se expone la necesidad de investigación que cubre en relación con la seguridad provista por una solución SD-WAN. En este primer capítulo también se presentan los conceptos teóricos que son necesarios para el estudio posterior de la funcionalidad técnica del trabajo.

En el capítulo de diseño técnico de la solución, se definen los requisitos técnicos y las funcionalidades que deben constar en la solución SD-WAN de prueba, para que permita realizar las pruebas pertinentes que logren dar respuesta a los objetivos definidos del trabajo. También se incluyen en este apartado algunos aspectos como los componentes de la red, listado de comunicaciones permitidas y el detalle de cómo se tienen que realizar las mediciones para establecer si los requisitos técnicos y objetivos del trabajo han sido satisfechos.

El último capítulo de contenido aborda los resultados obtenidos. En él, en primer lugar, se describen los distintos scripts de programación que han sido empleados para simular la red. A continuación, se recogen las diferentes pruebas técnicas que evidencian que los requisitos técnicos definidos durante la fase de diseño e incluidos en el capítulo anterior han sido configurados para la red. Por último, se exponen los resultados obtenidos para

las diferentes pruebas de medición del rendimiento de la red, con las que, posteriormente, se extraerán las conclusiones del trabajo.

Finalmente, se encuentra el capítulo de conclusiones, en el que se hace un repaso de los resultados de las mediciones y pruebas, para discernir si el desarrollo técnico realizado permite evaluar los objetivos que se marcaron para el proyecto. Además, en este apartado se incluye el seguimiento realizado al proyecto, así como posibles líneas futuras de otros trabajos.

2. Materiales y métodos

2.1. Estado del Arte

Las soluciones WAN definidas por software están actualmente en pleno auge. Retomando los objetivos de este trabajo, con la programación y configuración de una topología SD-WAN y el posterior estudio de su comportamiento, se pretende entender el funcionamiento de esta tecnología que está siendo empleada hoy en día en múltiples soluciones cloud.

Haciendo una comparativa del mercado de las soluciones SD-WAN, hay varias opciones que cuentan con características avanzadas de seguridad para entornos empresariales. Ejemplos de este tipo de soluciones son VMWare SASE SD-WAN, Citrix SD-WAN, FortiGate Secure SD-WAN, Cato SASE Cloud o Palo Alto Prisma, que combinan la interconexión de las infraestructuras empresariales *on-premise*, con los sistemas cloud y la instalación de agentes en los puestos de trabajo de las trabajadoras y trabajadores de las empresas para realizar las funciones de seguridad de la SD-WAN. Entre estas funciones podemos encontrar inspección de tráfico SSL, control de navegación web, protección de correo, firewalls de próxima generación para el filtrado de los paquetes, sistemas de prevención de fuga de información y control de acceso.

Los procesos de transformación digital de las empresas son los que han impulsado su apuesta por las soluciones tecnológicas basadas en la nube, entre las que se encuentran las soluciones SD-WAN. De forma alineada con los cambios en el mercado, los equipos técnicos que se encargan de diseñar las arquitecturas de IT están integrando estos productos dentro de las mismas por el menor coste que tienen, ya que no se tienen que responsabilizar de su mantenimiento, así como por otros aspectos como la modernización de sus sistemas, la reducción de tiempos de respuesta, el cumplimiento de estándares y normativas en materia legal y de seguridad, y el alineamiento con las tendencias del mercado.

Otros trabajos que se han realizado en ediciones pasadas de este máster han estudiado el comportamiento de las redes definidas por software empleando el software de Mininet. Algunos de los ejemplos son el trabajo “Análisis de redes definidas por software (SDN) y su aplicación en el entorno sanitario” realizado por Carlos Mediero Martí, o el trabajo “SDN: QoS en redes virtuales multi-tenant con OpenDaylight y Mininet” de Diego Arias Álvarez.

Estos trabajos se enfocaron en estudiar la calidad de servicio de las redes definidas por software o su utilidad para construir las redes hospitalarias y sanitarias, empleando controladores diferentes (ONOS y OpenDaylight) para el análisis del comportamiento de las redes customizadas.

Con este proyecto de fin de máster se pretende continuar con el estudio de las redes definidas por software y, concretamente, analizar la utilidad que se les está dando dentro de las empresas y organizaciones, enfocando este análisis en la centralización de las medidas de seguridad corporativas y la aplicación de políticas en la zona SD-WAN.

2.2. Definición de las redes SDN

En las infraestructuras tecnológicas que tradicionalmente han tenido las empresas, los segmentos de las redes de área local (LANs) se interconectan entre ellas mediante circuitos MPLS (Multiprotocol Label Switching) operados por los proveedores del servicio de Internet. Las redes de las oficinas, por su lado, se conectan con el resto de la infraestructura mediante túneles VPN (Virtual Private Network). Otros protocolos como VoIP son usados por las empleadas y empleados de estas empresas para realizar llamadas de teléfono a través de la red de Internet. Aquí surge la cuestión de si el servicio de Internet podría emplearse por sí solo para interconectar la IT de una empresa.

La respuesta a esta duda parece ser negativa, ya que este tipo de redes no permiten definir niveles de acuerdo de servicio (SLA) para priorizar y organizar el tráfico de la red. Los proveedores de servicio de internet no proporcionan estos niveles de compromiso y para solucionarlo se deberían desplegar routers en las sedes de las empresas que garanticen el envío del tráfico de acuerdo con los niveles de cumplimiento deseados. Las redes definidas por software (SDN), y concretamente la SD-WAN, pueden resolver estos problemas introduciendo un equipo *on-premise* de cliente (CPE) en cada sede en la que la empresa esté presente. [9, pág. 2]

Otro de los problemas con los que se ve afectada la red basada en configuraciones tradicionales es la saturación de los enlaces y equipos por la poca elasticidad y escalabilidad que presenta ante incrementos de la demanda. Como consecuencia, algunos servicios críticos se pueden ver degradados y no cumplir con la política de calidad de servicio (QoS, por sus siglas en inglés), lo que se traduce en una mala experiencia de usuario. Habitualmente, la red tradicional solo es capaz de lidiar con estos problemas mediante la instalación y mantenimiento de nuevos recursos hardware.

Estos sistemas empleados en la red tradicional no fueron diseñados para soportar una escalabilidad, tráfico y movilidad crecientes de las redes Next Generation Network (NGN). La NGN o red de próxima generación integra tráfico de diferentes equipos, entre los que se encuentran routers, switches, redes de 4G y 5G, y puntos de acceso, que deben adaptarse a los servicios emergentes de IoT, como son VoIP, las redes de sensores, Quality of Service (QoS), almacenamiento y cómputo en la nube, y otras aplicaciones, para proveer a los usuarios una red segura, estable, veloz y altamente disponible. [15, pág. 1]

Las limitaciones de las redes tienen su origen, principalmente, en que el procesado de paquetes es realizado en hardware muy específico (plano de datos). Los operadores de red se encuentran también limitados a la hora de administrar su red: la red resultante es muy rígida y ante situaciones cambiantes que surgen en el día a día solo tienen la capacidad de configurar algunos de sus parámetros, encontrándose a merced de la configuración proporcionada por los fabricantes del hardware o por la realizada por el proveedor de servicios. En consecuencia, se complica la evolución de los protocolos de red hacia su versión dinámica y adaptativa a cambios en la red. [15, pág. 2]

El concepto de Software Defined Networking (SDN) junto con el protocolo OpenFlow, están jugando un papel muy importante durante los últimos años en la evolución de los protocolos y las redes, aunque hay que echar la vista atrás a los años 90 para encontrar su origen. [15, pág. 2]

2.2.1. SDN: concepto y arquitectura

SDN surgió con más fuerza hace una década como opción para resolver todos estos problemas a los que se enfrentan las redes tradicionales basadas en sistemas distribuidos. La arquitectura SDN cuenta con la peculiaridad de la separación entre el plano de control, que se encarga de la toma de las decisiones de gestión de red, mientras que el plano de datos es el encargado del envío del tráfico de la red. La red resultante presenta una estructura centralizada, en la que las decisiones de gestión y coordinación de los elementos que conforman la red se enfocan en mayor medida a alcanzar las condiciones operativas óptimas para la citada red de telecomunicación. Además, también permite la evolución de nuevos protocolos del plano de datos sin la necesidad de reemplazar el hardware de los switches de la red. [5, pág. 1]

De forma simplificada, SDN propone una visión de red global en la que el plano de control, el plano de datos y el plano de aplicación están separados. La centralización del plano de control permite obtener una visión general de la red. Además, las soluciones SDN cuentan con una serie de interfaces para que ambos planos, de control y de datos, se comuniquen y puedan intercambiar la información que sea necesaria. En la Figura 2.1 se reflejan los diferentes planos o capas de las redes SDN. [15, pág. 3]

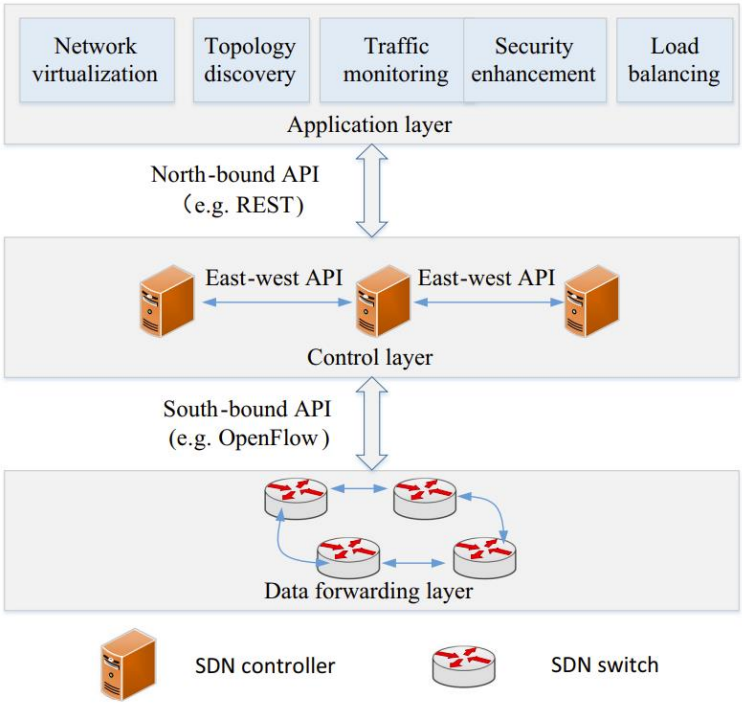


Figura 1. Planos de la red SDN. [15, pág. 3]

En la red tradicional basada en sistemas distribuidos, los switches y routers controlan el encaminamiento del tráfico y el reenvío de paquetes. Esta integración vertical del plano de control y datos incrementa la complejidad de la red, dificultando su control y la toma de decisiones. Sin embargo, en la red SDN los switches y routers ejecutan la funcionalidad del plano de control, pero bajo las órdenes de un nuevo elemento en la red, el controlador. [5, pág. 1]

Se llama controlador al propio software encargado de la gestión de los recursos disponibles en la red. Los controladores de la red son puntos de gestión que recolectan información de la red para decidir, de manera coordinada, la configuración de cada uno de sus recursos y elementos. De esta manera se consigue simplificar el papel de los switches y routers, pues únicamente presentan las funciones del plano de datos dentro de la red SDN. Los controladores se ocupan del plano de control como se ha indicado, definen el comportamiento que debe tener la red y responden a los cambios emergentes con decisiones de control, configuración y gestión para encaminar el tráfico siguiendo el comportamiento predefinido para cada flujo. [5, pág. 1]

A continuación, se detallan los diferentes planos o capas que forman parte de las redes definidas por software, así como las principales funciones que desempeñan.

2.2.2. Controlador

Las redes SDN se diferencian de las redes tradicionales en que existe la figura centralizada del controlador. El controlador se encarga de las funciones del plano de control de la red, es decir, dirige la red con sus decisiones y permite obtener una imagen global de su funcionamiento y su comportamiento. [5, pág. 2 – 3]

Es la inteligencia de la red definida por software, gestiona y controla la red entera. Básicamente es un programa informático que se ejecuta en un nodo de la red conocido como controlador. Este software es capaz de realizar funciones de descubrimiento de red, control del tráfico, diseño de rutas, entre otras, para dotar a la red de la configuración necesaria para garantizar su correcto funcionamiento. Se comunica mediante la API *Southbound* con los switches de la red de la capa física, para indicarles cómo deben de actuar cuando reciben un determinado tipo de tráfico. Normalmente, para esta comunicación se emplea el protocolo OpenFlow, que se ha convertido en un estándar de las redes definidas por software a nivel internacional. [16, pág. 764 – 765]

El controlador se comunica con otros elementos mediante dos interfaces: la interfaz de bajo nivel ya mencionada, *Southbound* API, y la interfaz de alto nivel, *Northbound* API. La *Northbound* API hace de pasarela de las comunicaciones del controlador hacia las aplicaciones de la red ubicadas en el plano de aplicación, y como se ha mencionado, la *Southbound API* se encarga de gestionar las comunicaciones del controlador con los servicios de encaminamiento y tratamiento de paquetes de los switches y el hardware de la red. Entre estas dos interfaces se encuentra el software puro del controlador (plano de control). En la Figura 2.2 muestra cómo es la estructura del controlador y cómo se conecta con las interfaces mencionadas. [15, pág. 5]

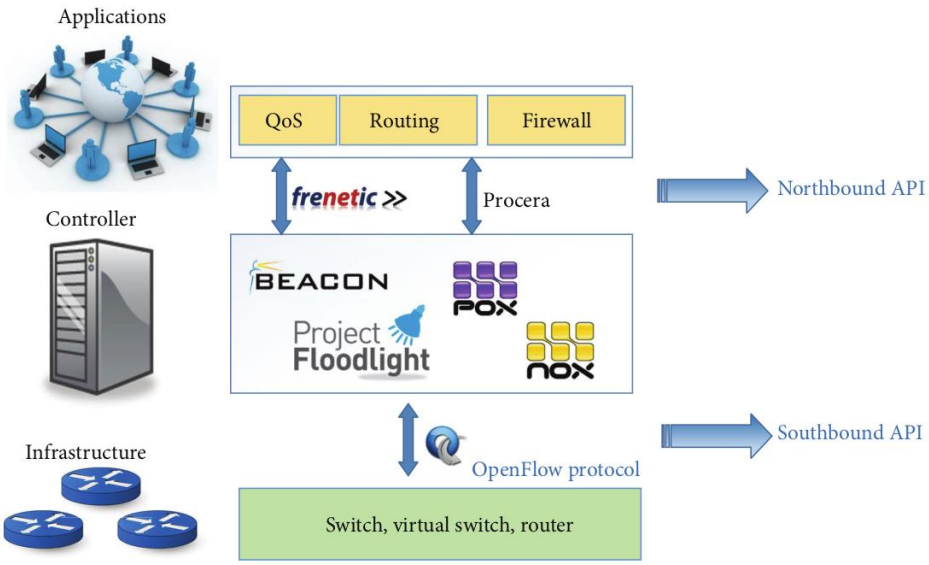


Figura 2. Estructura del controlador.

Generalmente, cuando un paquete llega a los switches en los que se encuentran los servicios de reenvío y encaminamiento, el switch consulta la información que tiene almacenada y que establece cómo ha de comportarse su tabla de flujos cada vez que recibe un paquete. Si no tiene en ella ningún flujo que encaje con el paquete recibido, envía un mensaje al controlador para que le indique qué debe hacer con él. El controlador puede tener información que permita dar instrucciones al switch, o puede informar a una aplicación concreta para que decida qué hacer con ese paquete. En esa estructura, el controlador que se encuentra entre las interfaces hace de traductor para permitir la comunicación entre las aplicaciones de red y el dispositivo que encamina el paquete. [6]

Los servicios de nivel bajo, es decir, los que se encuentran en el plano de datos, pueden ser switches hardware o software. Ambos tipos de switches se comunican con el controlador mediante la interfaz *Southbound API* usando OpenFlow. La información que se intercambia por esta interfaz suele corresponder a eventos como la llegada de un paquete que no se sabe encaminar, notificaciones de la red, como enlaces caídos, y estadísticas. [6]

El controlador de la red consta de los siguientes servicios:

- Servicio de topología: conoce cómo se conectan los switches (servicios de bajo nivel) de la red y construye una topología de ella.

- Servicio de inventariado: descubre los servicios de la red y guarda información de lo que ofrece cada switch.

- Servicio de estadísticas: almacena los contadores que le reportan los nodos y switches, para formar una idea más aproximada de cómo es la red de forma dinámica.

- *Host tracking*: servicio para descubrir las direcciones IP y MAC de los elementos de la red.

Las aplicaciones que se comunican con el controlador a través del *Northbound* API ayudan a gestionar el comportamiento de la red y a implementar nuevas políticas. Se comunican con el controlador SDN mediante la interfaz de alto nivel, a través de la cuál llega al controlador información del core de la red y órdenes de control del comportamiento de la red. [6]

2.2.3. Switches de la red

El plano de datos o plano de envío de datos se compone de switches SDN conectados físicamente mediante redes cableadas o inalámbricas. Un switch es un dispositivo simple que se encarga del reenvío de los paquetes a través de la red, y que emplea tablas de flujos para conocer cómo ha de comportarse cuando un paquete llega a sus interfaces de red. Las tablas de flujos están compuestas de reglas que son las que marcan el comportamiento del switch SDN. [16, pág. 765]

Las reglas de las tablas de flujos tienen 3 componentes: la acción como un contador y un patrón. Cuando un paquete llega al switch, en primer lugar, se comprueba que sus cabeceras coincidan con el patrón de alguna de las reglas de la tabla de flujos de switch. Cuando coinciden con alguna regla, se incrementa en uno su contador y se ejecuta la acción asociada a dicha regla, en función de lo cual el paquete se descartará, se aceptará y enviará, o se denegará. [16, pág. 765]

En el caso en el que las cabeceras del paquete no encajen con ninguna de las reglas disponibles en la tabla de flujos del switch, éste le enviará un mensaje al controlador de la red a través de la *Southbound* API, informándole del tipo de tráfico que ha recibido, para que le indique cómo debe actuar ante él. El controlador responderá al switch con el flujo que debe instalar en su tabla de flujos para atender ese tipo de tráfico y, posteriormente, el switch verificará la acción a ejecutar indicada por el controlador. [16, pág. 765]

Los switches de la red que se comunican con el controlador mediante OpenFlow pueden ser de dos tipos, switches puramente OpenFlow, o switches mixtos en los que están activadas las funciones de un switch OpenFlow y además tienen las funciones de routing y switching de las redes tradicionales. [15, pág. 4]

Los switches híbridos deben tener especificado un mecanismo para clasificar qué tráfico ha de ser tratado por el pipeline de OpenFlow y qué paquetes deberán procesarse usando técnicas "tradicionales". Por ejemplo, podría diferenciarse un tráfico de otro mediante el etiquetado de VLANs para la parte de red tradicional y de puerto de entrada para el tráfico de red Open-Flow. [11]

2.2.4. Plano de aplicación

El plano de aplicación es el que permite a los operadores de red responder rápida y eficientemente a las necesidades de negocio. En los controladores modernos se cuenta con características de virtualización de red como descubriendo descubrimiento de la topología, monitorización del tráfico, gestión y configuración de políticas centralizadas de

seguridad en la red y balanceo de carga, como se puede observar en la Figura 2. El plano de aplicación es el que se comunica con el plano de control a través de la *Northbound* API, proveyendo de un nivel de abstracción a la capa de aplicación de los recursos físicos. [16, pág. 766]

2.2.5. Ventajas de las redes SDN

El principal motivo por el cual las redes SDN han tenido durante estos últimos años un aumento de popularidad es porque, comparándolas con las redes tradicionales, ofrecen una mayor disponibilidad de los servicios ofrecidos. En una red SDN cuando se pierde conectividad con una máquina o conjunto de máquinas, la propia red se encarga de reencaminar el tráfico por otros sistemas e interfaces que se encuentren en funcionamiento, reduciendo así tiempos de desconexión y posibles errores. La red SDN es más escalable que la tradicional porque es capaz de reconfigurarse dinámicamente según las necesidades de un momento determinado. Otra de las peculiaridades que ofrece SDN es que varios controladores pueden coexistir en una misma red, permitiendo que se establezcan diferentes regiones SDN con sus recursos propios que se comunican entre ellas mediante protocolos *east-west* que se establecen entre los controladores. [6]

En la red tradicional, el plano de datos y control están unidos. El plano de datos se encarga del manejo de paquete en función de información almacenada en tablas de enrutamiento. El plano de control es el que dirige la comunicación entre un nodo y otro mediante protocolos de encaminamiento, como BGP, MPLS o OSPF. En este plano se configuran y almacenan las políticas de red, se decide cómo se tratan los paquetes y envían esta información al plano de datos. La arquitectura de las redes tradicionales impide la gestión hábil y a gran escala de cambios en el enrutamiento de la red, ya que para modificar el comportamiento de reenvío de paquetes por parte de los routers es necesario hacerlo mediante las políticas definidas a través de los protocolos de encaminamiento, a diferencia de en las redes definidas por software, donde basta con configurar en el controlador los cambios que sea necesario realizar. [6]

Para acceder al plano de control en la red tradicional hay que hacerlo mediante órdenes y comandos de configuración en el hardware específico de encaminamiento, es decir, de los routers, que a su vez dependen de los fabricantes. Por otro lado, los nodos de la red tradicional son individuales, por lo que cualquier modificación que se quiera hacer en la red deberá hacerse de manera manual sobre los nodos para que surta efecto real, porque no todos los nodos de una red tienen las mismas configuraciones. Aunque con las redes SDN parece que sigue habiendo dependencias con los fabricantes y proveedores respecto al hardware propietario de los switches, sí que se permiten realizar cambios automatizados en todos los switches de la red al mismo tiempo a través del controlador, lo que es útil para gestionar cambios en las políticas de encaminamiento o seguridad mencionadas previamente. [6]

Todos estos inconvenientes de las redes tradicionales son ventajas de la arquitectura SDN. No obstante, a pesar de todas las ventajas que presenta, su complejidad es muy elevada y no es trivial, ya que posee un alto nivel de abstracción y un cambio de paradigma para las ingenieras e ingenieros que diseñan y despliegan las redes. Además, los equipos de la red tradicional emplean un hardware muy específico que no puede ser usado en SDN, por lo que los gastos de CAPEX son elevados. No obstante, con las soluciones actuales basadas en la nube, esto ya no sería un inconveniente.

2.3. OpenFlow

Surgió como una alternativa a los protocolos empleados en los estudios de tráfico y redes que se hacían dentro de las universidades, por la facilidad que estas tienen para validar el adecuado funcionamiento de las nuevas tecnologías. OpenFlow se ha convertido en el protocolo de comunicaciones estándar para intercambiar mensajes entre el controlador SDN y un dispositivo de red. [15, pág. 6]

OpenFlow permite enviar reglas de encaminamiento a los dispositivos de red y definir su comportamiento ante el tráfico que reciben estos dispositivos por sus interfaces. De este modo, cuando un paquete llega al equipo de red, el switch, se lee su cabecera, se consulta la entrada en la tabla de flujos del protocolo OpenFlow y se ejecuta la acción que esté definida para ese tráfico en función de las direcciones origen y destino, el tipo de paquete, su contenido, etc. En la Figura 3 se representa la comunicación entre los dispositivos de red y el controlador empleando el protocolo OpenFlow. [15, pág. 4]

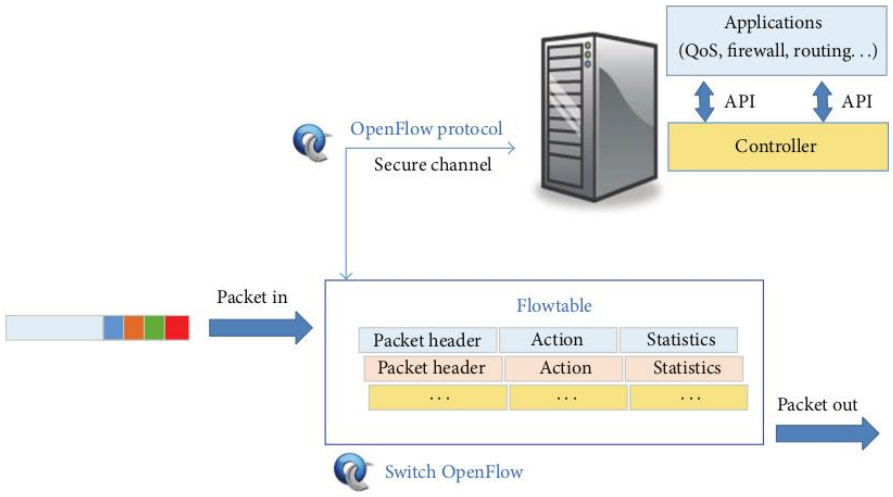


Figura 3. Comunicación entre el controlador y dispositivo SDN a través de OpenFlow.

En OpenFlow los switches se conectan de manera lógica con el controlador mediante puertos OpenFlow. Por tanto, solo se pueden enviar y recibir mensajes OpenFlow por los puertos que hayan sido asignados en el switch o máquina a este cometido. [11, pág. 8]

Los paquetes de datos se reciben en puertos de entrada o recepción (*ingress port)* y se envían por puertos de envío o salida (*output port*). En muchas ocasiones se utilizan los puertos de entrada para consultar la tabla de flujos y decidir qué acción ejecutar sobre el paquete en función de cuál sea su origen. [11, pág. 8]

2.3.1. Tablas de flujos

El formato de OpenFlow está definido para poder establecer reglas de encaminamiento en la llamada tabla de flujos de los switches. Esta tabla tiene principalmente tres secciones importantes:

1. Cabecera del paquete.

- Acción que realizar.

- Algunos datos recopilados y estadísticas para controlar el volumen de tráfico encaminado.

Cabecera

Dependiendo de la versión de OpenFlow soportada en el controlador y en los dispositivos de red, el switch OpenFlow puede procesar un mayor número de campos de paquete. En concreto, en la versión 1.0.0 hay 12 campos, y en la versión 1.3 hay 40 campos. [15, pág. 4]

Estos campos, que se configuran a través de mensajes OpenFlow, se emplean para consultar las tablas de flujos y encontrar la entrada que coincide con el tráfico recibido, lo que permitirá ejecutar la acción asociada al tipo de paquete. Para que un paquete se considere que cumple con los requisitos de una entrada ha de encajar en todos sus campos. En ocasiones, un paquete puede encajar en varias reglas de una tabla de flujos, ya que no es necesario que todos los campos de las reglas estén cubiertos por un valor. Para evitar estas posibles ambigüedades hay definido un campo de prioridad, en el que se indica cuál es la importancia de esa regla sobre el conjunto de todas las reglas. [10, pág. 2 – 4]

Acciones

Una vez que el paquete llega al switch OpenFlow y se comparan sus cabeceras para localizar la entrada que le corresponde en la tabla de flujos, se ejecuta la acción indicada para él. Estas acciones pueden ser de dos tipos, principales y opcionales, y marcan el comportamiento del tráfico dentro de la red. [15, pág. 3]

Las acciones principales son las opciones básicas que cualquier switch OpenFlow debe tener configuradas. En concreto, estas acciones indican si un paquete ha de ser enviado por un puerto determinado, si dicho paquete debe ser encapsulado y enviado al controlador para que se instale en el switch un flujo para ese tipo de tráfico, o si el paquete debe ser descartado por el switch. Las acciones opcionales recogen otros comportamientos menos usuales del switch como encolar paquetes a un puerto determinado y otras especificaciones del estándar 802.1D (incluye técnicas de *bridging*, protocolos y manejo de redes inalámbricas). [15, pág. 4]

Las entradas de las tablas de flujos están asociadas con acciones que debe ejecutar el switch sobre el paquete recibido. Las que no tienen acciones definidas indican que el tráfico que encaje en ellas deberá ser descartado. También puede haber entradas con varias acciones indicadas, en cuyo caso el switch deberá ejecutarlas siguiendo el orden establecido en ellas. Si alguna de las acciones no puede ser procesada por el switch, esa entrada de la tabla de flujos será ignorada por él, ya que los switches no tienen por qué saber manejar todas las acciones, pero sí que informará al controlador que ha habido un error por flujo no soportado. [10, pág. 3]

Cuando se arranca un switch OpenFlow dentro de la red, se debe indicar al controlador qué acciones opcionales puede manejar ese switch, y en función de esa información el controlador establecerá las acciones pertinentes para que no haya errores durante el procesado de paquetes. [10, pág. 3]

Las acciones soportadas por los switches son:

2. Forward o envío de tráfico: deben soportar el envío de tráfico por puertos físicos y por los siguientes puertos virtuales:
    - ALL: envío a todas las interfaces menos la de entrada.
    - CONTROLLER: envío de paquetes por la interfaz del canal seguro de comunicación con el controlador.
    - LOCAL: enviar por la interfaz local del switch.
    - TABLE: poder ejecutar las opciones indicadas en la tabla de flujos.
    - IN_PORT: enviar el paquete por el puerto por el que se recibió en el switch.

- Drop o descarte de tráfico: si no se especifican acciones en una entrada de la tabla de flujos, el tráfico que encaje con ella se descartará.

Por otro lado, hay otras acciones consideradas opcionales que pueden ser soportadas o no en función de cada switch. Son las siguientes:

- Envío de tráfico a través de dos tipos de puertos virtuales:
    - NORMAL: empleando las técnicas de envío de las redes tradicionales (VLAN, L2, L3). El switch habitualmente se guiará por el identificador de VLAN para saber qué debe hacer con ese tipo de tráfico según la funcionalidad de la red tradicional.
    - FLOOD: por inundación, se envía el paquete por todos los puertos del switch.
- Encolar un paquete: enviar el paquete a la cola de alguno de los puertos físicos del switch.
- Modificar un campo: permite incrementar el uso de las funcionalidades del protocolo OpenFlow.

Estadísticas

Las estadísticas o contadores pueden recogerse por tabla, por puerto o por cola. Los switches se encargan de recopilarlas y almacenarlas para que el controlador pueda analizar el comportamiento de la red y modificar los parámetros que sean necesarios. Por ejemplo, si al interpretar las estadísticas el controlador detecta una disminución anormal de tráfico procedente de una interfaz, sabrá reconocer que está sucediendo algún problema en esa interfaz y si no es capaz de identificar con exactitud el origen, podrá examinar en una zona más o menos acotada de la red. Además, también sirven para trazar

el comportamiento habitual de la red en función de horas del día, día de la semana o, incluso, estaciones, ya que no suele ser igual el tráfico que hay en la red a las 23:00h que el que hay a las 03:00h, tampoco el de un martes es igual al del sábado, o el de verano al de invierno, si se trata de una zona de playa, por ejemplo. Todas estas alternativas tienen que poder ser reconocidas por el controlador, para optimizar el funcionamiento de la red en base a la gestión eficiente de los recursos disponibles y del comportamiento que presenta. Las estadísticas sirven para optimizar las decisiones tomadas en el plano de control de la red por el controlador. [15, pág. 4]

2.3.2. Canal OpenFlow

Se llama canal seguro de OpenFlow a la interfaz de comunicación entre el controlador y los switches de la red, es decir, es el protocolo que se emplea para las comunicaciones de la *Southbound* API. Los mensajes intercambiados sirven al controlador de la red para configurar las tablas de flujos de los switches, pudiendo de esta manera definir el comportamiento que debe adoptar la red. No solo el controlador envía mensajes, el switch también se comunica con el controlador mediante el canal. Todos estos mensajes del switch y el controlador siguen el formato del protocolo OpenFlow. [10, pág. 9]

Los paquetes OpenFlow se clasifican en tres tipos: *Controller-to-Switch*, *Asynchronous* y *Symmetric*. A continuación, se explican en mayor detalle cada uno de estos tipos.

Controller-to-Switch

Como su nombre indica, es el controlador el que inicia la comunicación con el switch, quien no tiene en todos los casos que responderle. Los mensajes enviados pueden ser: [10, pág. 10]

- Características: se produce cuando en el establecimiento de la sesión entre controlador y switch, el controlador necesita conocer las características del switch, por lo que le envía este tipo de petición. En este caso el switch debe contestar para que el controlador sepa qué acciones soporta.

- Configuración: el controlador envía peticiones e instrucciones de configuración al switch. El switch solo debe responder a las peticiones.

- Modificación de estado: se utilizan por el controlador para modificar el estado del switch, generalmente mediante la modificación de entradas de la tabla de flujos y establecimiento de las propiedades de los puertos.

- Lectura de estado: el controlador las envía para que el switch le reporte las estadísticas que ha recopilado de sus tablas de flujos, puertos y entradas de las tablas de flujos.

- Envío de paquetes: son útiles cuando el controlador quiere enviar un paquete por un puerto del switch.

- De barrera: se pueden mandar solicitudes y respuestas de este tipo de mensajes, en las que el controlador quiere asegurarse de que los cambios de configuración que debe haber efectuado el switch realmente se han modificado; o para recibir información de las operaciones que han sido completadas.

Asynchronous

Los mensajes asíncronos son aquellos que envía el switch sin petición previa del controlador. El switch suele enviarlos cuando llega un paquete nuevo, cuando cambia su estado o cuando se produce en él un error, por ejemplo, si no puede ejecutar una acción. Principalmente existen cuatro tipos de mensajes asíncronos que envía el switch: [10, pág. 10 – 11]

- *Packet-in*: si llega un paquete al switch que no encaja con ninguna de las entradas de la tabla de flujos, se envía un mensaje al controlador para que instale un flujo nuevo. También se envía este tipo de petición cuando en la acción definida en la tabla de flujos se indica expresamente que se envíe el mensaje al controlador. En el caso en el que el switch tenga un buffer de almacenamiento de los mensajes enviados al controlador, en ese paquete se envía una parte de la cabecera del paquete recibido y un campo ID indicando cuál es el paquete almacenado en el buffer al que se refiere el mensaje "packet-in". Si el switch no dispone de este buffer, todo el paquete se encapsula dentro del "packet-in".

- *Flow-Removed*: para cada uno de los flujos de la tabla del switch hay dos campos que indican la caducidad del flujo. Uno de ellos especifica la caducidad por desuso del flujo, es decir, tras ese tiempo de inactividad en el flujo, el switch debe eliminarlo. El otro indica cuándo, independientemente de que se haya usado o no, debe ser eliminado un flujo de la tabla. Esto es útil para evitar tablas muy extensas con flujos en desuso, y también para que los flujos estén debidamente actualizados en ella. Por ello, cada vez que el switch elimine alguno, se enviará un mensaje asíncrono "flow-removed¨ al controlador, para informarle del flujo eliminado. Así mismo, cuando se recibe en el switch una instrucción del controlador para modificar un flujo que ya ha sido eliminado, también el switch envía el mensaje "flow-removed¨ para que el controlador sepa que está desinstalado.

- *Port-status*: cuando un puerto se cae, es apagado por el usuario o por las especificaciones del estándar 802.1D, entre otros cambios de estado, se envía por parte del switch un mensaje al controlador reportando el cambio en la configuración de la interfaz.

- *Error*: para informar de cualquier error durante el funcionamiento o procesado de paquetes en el switch.

Symmetric

No se necesita solicitud o comunicación previa por ninguna de las dos partes para el envío de este tipo de mensajes. Los tipos que existen son los siguientes: [10, pág. 11]

- *Hello*: fase de establecimiento de la conexión entre switch y controlador.

- *Echo*: en forma de solicitud o respuesta, tanto switch como controlador pueden enviar una solicitud y el otro extremo debe responder. Se usan sobre todo para conocer los valores de latencia, ancho de banda disponible y tiempo de vida restante de la conexión establecida.

- *Vendor*: en ellos los switches pueden emplear características adicionales incompatibles con el protocolo OpenFlow desarrollado hasta el momento.

2.4. Descripción de los elementos de seguridad de las redes modernas

En esta sección se describen los elementos que se emplean para dotar de seguridad las redes de las empresas y organizaciones actualmente. Desde un punto de vista global, estos componentes permiten proteger los accesos a la red interna, la navegación hacia Internet, las peticiones de consumo de recursos ubicados en la red interna y el cifrado de las comunicaciones que transportan toda esta información, tanto en las peticiones como las respuestas.

Estos elementos han ido actualizándose y transformándose durante estos últimos años para poder adaptarse a las necesidades que iban surgiendo dentro de las infraestructuras tecnológicas. Los componentes que se van a detallar son los principales y los que, posteriormente, serán configurados en la red que se desea simular en este trabajo de fin de máster.

2.4.1. Zona desmilitarizada

La zona desmilitarizada o DMZ (De-Millitarized zone) es una aproximación que se utiliza en las infraestructuras de las empresas para generar dos niveles de defensa en la red. La DMZ es la parte de la red de la organización que se encuentra entre Internet y la red interna donde se albergan los servicios de negocio, información sensible o confidencial y activos tecnológicos de la organización. [1, pág. 1]

La defensa en dos niveles de la DMZ consiste en un primer nivel o capa que se expone a la red de internet, externa a la organización, y un segundo nivel que se apantalla la red interna de la organización. Un ciberatacante que intente penetrar hacia los sistemas internos de la empresa necesitaría romper estos dos niveles de defensa, sin ser detectado, para conseguir su objetivo y acceder a los activos internos de la organización y a su información. En consecuencia, una decisión estratégica dentro de los directores de IT de las compañías es decidir qué activos van a estar en la DMZ, puesto que estos recursos deberán ser suficientes para detener posibles ataques perpetrados desde el exterior. [1, pág. 1]

Es importante remarcar que la defensa en dos niveles también aprovecha la necesidad de que en estos niveles existan tecnologías distintas, es decir, no se emplearán los mismos componentes que se exponen a la red pública de internet, y que constituyen la primera fase o capa de defensa, que los de la segunda capa de defensa que está conectada a la red interna. Un atacante que quiera penetrar a la red interna, por tanto, deberá disponer de recursos y conocimientos para vulnerar las tecnologías presentes en estas dos diferentes capas. [1, pág. 1]

La DMZ es un componente que debe emplearse en todas las organizaciones que tengan servicios expuestos en internet, ya que asegura el refinamiento de la seguridad de la capa de aplicación de las comunicaciones. Cualquier servicio ofrecido a la red de internet por parte de las organizaciones debe estar alojado en la DMZ mediante un servidor. Los

sistemas que ejecutan servicios en la DMZ son servidores que son susceptibles a ser atacados por ciberatacantes, sin embargo, como se ha comentado previamente, estos atacantes deben primero vulnerar el primer nivel de seguridad para acceder a ellos. Aunque la transformación digital de las empresas ha llevado a que cada vez haya más recursos alojados en sistemas cloud, las DMZs siguen siendo importantes para proteger el acceso a los servidores internos y a la información de los sistemas *on-premise*. [7, pág. 5 – 6]

2.4.2. Balanceadores de carga

El balanceo de carga es una técnica que se usa para distribuir el tráfico entrante a una red entre diferentes servidores disponibles, de modo que las peticiones puedan ser manejadas y respondidas en un tiempo menor. Por si sola no se trata de una tecnología que provea de mayor seguridad a la red, sino que su función es integrarse con los componentes de seguridad como firewalls, IDS, etc., para proporcionarles un mejor funcionamiento y que operen con mayor eficiencia. No obstante, algunos autores consideran que el balanceo de carga también es una componente de seguridad en las redes definidas por software. [13, pág. 204]

El funcionamiento de los algoritmos de balanceo de carga puede ser estático o dinámico. Los algoritmos estáticos no modifican su comportamiento en función de las métricas de la red, mientras que los algoritmos dinámicos sí. El comportamiento dinámico de los balanceadores consiste en calcular el tamaño de la carga que reciben en cada momento de las peticiones de los clientes y construir una cola donde ir albergando estas peticiones. A continuación, comparan la carga en tiempo real de los servidores que se encuentran en el *pool* de balanceo de carga y emplean una estrategia de balanceo para enviar las peticiones albergadas en la cola al servidor que corresponda. [13, pág. 204]

El trabajo de los balanceadores es complejo, puesto que deben gestionar los recursos de manera eficiente para no sobrecargar ciertos sistemas de la red mientras que otros están ociosos. Además, los balanceadores cuentan con la funcionalidad de proveer alta disponibilidad a los sistemas, debido a que evitan la saturación de los servidores cesando el envío de peticiones cuando se detecta que la carga de estos es elevada, una vez se reduce la carga el servidor volverá a recibir peticiones que atender. [13, pág. 204]

Algunas de las ventajas de aplicar balanceo de carga en las redes son las siguientes:

- El control y la trazabilidad de tráfico.
- Se realiza un balanceo de carga basado en la capacidad de los nodos.
- Mejora la disponibilidad de los recursos e incrementa su utilización.
- Evita el sobreprovisionamiento de recursos de la red.

Como se detallará más tarde en este documento, las soluciones SD-WAN de la actualidad se basan en arquitecturas cloud a las que se conectan las diferentes sedes de las empresas y usuarios remotos. En estos casos, los servidores de balanceo de carga también están provistos por la solución cloud, que es responsable de su mantenimiento y buen funcionamiento para ejecutar los servicios de balanceo de los clientes.

En la tecnología del balanceo de carga existen varias estrategias. A continuación, se exponen las más relevantes: [13, pág. 205]

- Menos conexiones: se trata de una estrategia de balanceo dinámico en la que se selecciona al servidor que tenga menos conexiones activas para reenviarle el tráfico entrante a la red.

- Menos conexiones ponderado: es una extensión de la estrategia anterior, en la que a cada servidor se le asigna un peso. Por defecto el peso de un servidor es 1, y aquellos servidores que tengan peso 0 nunca recibirán conexiones.

- Planificación de balanceo de carga con feedback dinámico: para aquellos servidores que en algunas situaciones tienen sobrecarga y siguen recibiendo peticiones en lugar de ser enviadas a otros con menos carga, la estrategia de feedback dinámico se usa para evitar en estos escenarios la indisponibilidad, en la que se toma la decisión en función del servidor con menor tiempo de respuesta.

- *Round Robin*: estrategia estática mediante la cual los servidores reciben peticiones de servicio de forma circular, es decir, si hay cuatro servidores en el círculo, primero recibirá peticiones el servidor 1, seguidamente el servidor 2, y así sucesivamente hasta llegar al servidor 4. Cuando el último servidor del círculo haya recibido una petición, la siguiente se enviará al servidor 1 para comenzar de nuevo con la asignación en círculo.

- *Round Robin* ponderado o con pesos: es una extensión de la estrategia *Round Robin* en la que a los servidores se le asignan diferentes pesos de acuerdo con su capacidad. Los servidores reciben peticiones empezando por el servidor con mayor peso, hasta llegar al servidor con menor peso, para considerar en la toma de la decisión la capacidad del servidor que contribuya a obtener un mejor rendimiento de esta estrategia.

2.4.3. Firewalls

El concepto tradicional de firewall o cortafuegos se define como un componente que se coloca en la ruta de entrada de los paquetes provenientes de internet hacia la red interna para que puedan ser filtrados y seleccionar si se aceptan, se rechazan o se deniegan. De este modo el firewall protege la red de diferentes tipos de incidencias aceptando únicamente aquel tráfico que tenga configurado en sus reglas. [3, pág. 95 – 99]

Una forma muy básica de configurar un firewall es crear un router con listas de control de acceso (ACLs, por sus siglas en inglés), aunque los firewalls reales pueden realizar acciones de filtrado más avanzadas en los paquetes que llegan a ellos para ser filtrados. Algunas de las acciones permitidas para el filtrado en los cortafuegos son: [3, pág. 95 – 99]

- Por coincidencia de las direcciones IP origen y destino.

- Coincidencia con los puertos TCP y UDP conocidos.

- Descubrir qué puertos TCP y UDP están siendo usados por un flujo y realizar el filtrado sobre ese puerto.

- Hacer que coincida el texto de la URI de una petición HTTP para aceptar o rechazar la descarga de la página web.

- Almacenar el estado de la información de los paquetes para en próximos filtrados de esos paquetes decidir en función del histórico, esto se conoce como inspección con estado (*stateful*).

Existen varios tipos de firewall, por ejemplo, de red, de aplicación, de *host*, etcétera. Sin embargo, un tipo de firewall muy interesante es el *stateful*, que es capaz de registrar la información de los flujos que recibe para aplicar la inteligencia al filtrado de futuros paquetes. Esto puede ser útil, por ejemplo, en un ataque de denegación de servicio: si el atacante intenta perpetrar este tipo de ataque contra un servidor web albergado en la DMZ de la organización, enviará una gran cantidad de peticiones de TCP/UDP al servidor. El firewall se encarga de interceptar y filtrar los paquetes antes de redirigirlos al servidor, de modo que, si detecta un número anormal de peticiones provenientes de la misma dirección IP, las bloquea. [3, pág. 95 – 99]

No obstante, existen otros tipos de usos que se le dan a los firewalls; separar diferentes zonas dentro de una red interna para controlar los accesos y realizar segmentación de la red son algunos ejemplos. Una práctica usual es emplear firewalls para componer los niveles de defensa de la DMZ mencionados previamente en esta sección. [3, pág. 95 – 99]

El flujo de los paquetes entrantes es el siguiente:

- Se coloca un primer firewall en la primera capa de defensa, que se encargará del filtrado de las peticiones realizadas desde cualquier punto de internet y decidirá si estas conexiones pueden ser reenviadas a los servidores web.

- Si estas peticiones son aceptadas y entran a la DMZ, habitualmente los servidores alojados en esta zona deben comunicarse con la red interna para el consumo de los servicios que les permite generar la respuesta a enviar a los usuarios.

- Estas peticiones entre la DMZ y la red interna pasan por la segunda capa de seguridad, es decir, hay otro firewall que filtra este tráfico y decide si las peticiones son legítimas y pueden pasar a la zona interna o no.

Como se ha comentado en secciones anteriores, esta separación de los servidores web y de los usuarios de internet de la red interna de la organización mediante firewall permite prevenir las peticiones ilegítimas y amenazas que puedan vulnerar la seguridad de los activos internos.

Un concepto que ha ido tradicionalmente ligado al de firewall es el de IPS. Los sistemas IPS filtran los paquetes de manera diferente a los firewalls, realizan una comparación de las cabeceras de los paquetes con firmas de *exploits*[1] conocidos que se encuentran

[1] Los *exploits* son programas informáticos que explotan vulnerabilidades conocidas con el fin de conseguir acceso no autorizado a un sistema.

almacenadas en bases de datos para detectar cuando los paquetes son parte de amenazas conocidas. Una vez que se identifica una amenaza, el IPS monitorizará el evento, descartará los paquetes o los redirigirá a otra aplicación para ser examinados con mayor detalle. [3, pág. 100]

2.4.4. VPN

Las redes privadas virtuales o VPNs (por sus siglas en inglés) son componentes de las redes de las empresas que permiten el acceso a la red interna de forma segura a usuarios y a activos ubicados en otra red. Existen dos tipos: [3, pág. 321]

- VPN *user-to-site*: se emplea para que usuarios remotos accedan a la red interna por medio de comunicaciones seguras.

- VPN *site-to-site*: se emplea para conectar dos sedes ubicadas en distintos puntos geográficos de forma segura.

Las VPNs intentan proveer las mismas medidas de seguridad que la red privada de las organizaciones en el acceso a los sistemas de la red interna. Algunas de las principales características de seguridad de las VPN son las siguientes: [3, pág. 321]

- Confidencialidad y privacidad: protegen el contenido de las comunicaciones mediante el cifrado de los datos, de modo que, si estos paquetes son interceptados por un tercero que quiere acceder a su contenido, no podrá conocer la información enviada dado que no se transporta en claro.

- Autenticación: asegura que el extremo de la comunicación, tanto usuario como dispositivo, que se está conectando a la red de la organización es quien dice ser y está autorizado para el acceso. Esta verificación es posible gracias al uso de certificados criptográficos.

- Integridad de la información: se aseguran de que la información contenida en los paquetes no ha sido alterada en tránsito.

- Anti-repetición: previene de que los paquetes sean interceptados, alterados y reenviados por un atacante para burlar las medidas de seguridad de la organización.

Normalmente, las VPN suelen disponer de dos dispositivos conectados a internet que crean un túnel VPN. Cuando un paquete va a ser enviado, los dispositivos añaden en las cabeceras los campos necesarios para que el tráfico se envíe de manera segura y cifran el paquete IP original para enviarlo a través de internet. [3, pág. 322 – 323]

Las VPNs *site-to-site* se construyen ubicando un servidor VPN en cada sede geográfica que se quiera conectar para el envío de la información de manera segura. La diferencia respecto a las VPN *user-to-site* es que, una vez que el túnel *site-to-site* esté levantado, cualquier dispositivo de cualquiera de las dos sedes que necesite comunicarse con algún sistema o activo de la otra sede puede utilizarlo. Frecuentemente se utiliza el protocolo IPSec para construir este tipo de túneles. [3, pág. 323]

2.5. Introducción a la SD-WAN

2.5.1. Necesidades que cubren las soluciones SD-WAN

Las redes de área amplia o WAN son un tipo de redes de telecomunicaciones que conectan nodos distribuidos a lo largo de sedes geográficas distantes. En sus orígenes, estas redes se empleaban para conectar diferentes centros de datos de las compañías de telefonía móviles, o de las grandes empresas que tenían presencia en diferentes países. Sin embargo, gracias a la globalización y a que se han puesto al alcance de las empresas más pequeñas los recursos tecnológicos más avanzados, actualmente las redes WAN se emplean incluso para conectar sedes dentro de una misma ciudad. Debido a ello, es necesario que en esta zona de la red se garantice que las medidas de seguridad están correctamente configuradas. [2, pág. 1]

El éxito que han tenido durante los últimos años los proveedores de servicios cloud ha hecho que las soluciones basadas en SDN den el salto a los mercados, como respuesta a las necesidades de ancho de banda, calidad de servicio, seguridad y confiabilidad. Se ha indicado previamente que las redes tradicionales cuentan con algunas desventajas como son la imposibilidad de separar el software y hardware, la complejidad de algunos protocolos de red y que muchos dispositivos de los que se emplean en las redes son propietarios del fabricante, por lo que el ciclo de vida de su software está a merced de este. Además, las redes SDN permiten obtener una mayor escalabilidad de los sistemas, más difícil de cubrir por las redes tradicionales. [2, pág. 1]

En esta nueva era en la que las empresas y organizaciones están migrando sus servicios *on-premise* a los sistemas cloud, y también debido al surgimiento de soluciones basadas en redes definidas por software, es necesario asegurar que las nuevas infraestructuras modernas cuentan con el mismo nivel de seguridad y las mismas capacidades para proteger los activos que las infraestructuras heredadas. [2, pág. 1]

En lo que respecta a este trabajo, si la seguridad de las redes SDN, y por extensión de la SD-WAN, no está garantizada, no se podrá dar soporte a las empresas en sus transformaciones digitales y habrá resistencia en la sustitución de los componentes tradicionales de las infraestructuras de red por los sistemas modernos. [2, pág. 1]

Es un reto para las ingenieras e ingenieros que diseñan las infraestructuras de las empresas ponerse al día con las nuevas soluciones del mercado y evaluar su seguridad. En la actualidad han surgido bastantes soluciones que advierten de las amenazas de las redes definidas por software, cómo son la replicación de los esquemas del controlador, la autenticación y autorización, esquemas para proteger a los controladores contra ataques de denegación de servicio distribuidos, la monitorización de tráfico y su análisis, o la protección contra el ataque de desbordamiento en las tablas de flujos, entre otros.

En el siguiente apartado se mencionan los aspectos de seguridad que se han de tener en cuenta para el despliegue y la configuración de las redes definidas por software, y que son aplicables también a las soluciones SD-WAN.

2.5.2. Seguridad en la SDN

Las redes definidas por software incluyen algunas ventajas de seguridad respecto a las redes tradicionales, entre las que se encuentra la monitorización continua y detección de tráfico con comportamiento anómalo, mediante la que el controlador tiene constancia en todo momento del tráfico que está generándose la red, por lo que le es sencillo identificar comportamientos anormales o sospechosos propios de un atacante. Además, la gestión y el tratamiento de vulnerabilidades son más eficaces, ya que cuando se detecta la amenaza se puede programar el comportamiento de la red para que automáticamente los switches se actualicen y se parcheen los agujeros de seguridad. [2, pág. 2]

Sin embargo, estas redes también cuentan con una serie de desventajas, entre las que destacan las siguientes: [2, pág. 2 – 4]

- Vulnerabilidades del controlador: dado que la inteligencia de la red es un programa informático, es decir, software que puede tener brechas de seguridad. El acceso al controlador de la red por parte de un atacante le otorgaría acceso a todas las funcionalidades de red que ejecuta este componente, pero, paradójicamente, las arquitecturas SDN están muy concentradas y es difícil para un atacante acceder a ellas. No obstante, es necesario protegerlas porque si el controlador está comprometido, la red y sus servicios también se verán afectados.

- Riesgos de las interfaces programables: aquí se concentran varios aspectos como son vulnerabilidades en el software del controlador o que este proporcione interfaces programables a la capa de aplicación que pueden quedar comprometidas o ser abusadas.

- Puntos de ataque: al estar dividida en tres niveles, la red SDN cuenta con más puntos vulnerables que las redes tradicionales. Estos puntos serían los switches SDN, los enlaces entre ellos, el controlador, los enlaces entre el controlador y los switches, los enlaces entre los diferentes controladores y el software de aplicación.

A continuación, se detallan los principales vectores de ataque de las diferentes capas en las redes definidas por software. Aunque todos ellos no serán probados durante el transcurso del trabajo técnico, es interesante conocerlos para entender el funcionamiento de este tipo de soluciones.

Seguridad del plano de datos

Los switches de una red definida por software cuentan con tres puntos de ataque por los que puede comprometerse su seguridad y, por extensión, la seguridad del resto de elementos de la red. Estos componentes susceptibles de ser atacados son el cliente OpenFlow, el buffer y las tablas de flujos. [16, pág. 768]

Estos tres componentes pueden ser atacados empleando tres métodos conocidos. Uno de ellos es el ataque *Man-in-the-Middle* entre el controlador y el switch, en el que el atacante intercepta los paquetes que el switch envía el controlador de la red, y responde suplantando la identidad del controlador para que el switch instale nuevas reglas de flujos en sus tablas. [16, pág. 768]

Este es uno de los ataques que más éxito tienen en las redes SDN, puesto que únicamente con suplantar la identidad del controlador, el nodo malicioso podría controlar todos los switches de la red, de modo que el comportamiento de estos solo dependería de las indicaciones del atacante a través del nodo malicioso.

Algunas de las contramedidas a este tipo de ataques es la creación de túneles de comunicación entre el controlador y los switches que aseguren la autenticación de los extremos (como hace el protocolo TLS), o el uso de algunas herramientas como VeriFlow, que se interponen en medio de la comunicación para monitorizar y detectar comportamiento anómalo en la red, como, por ejemplo, la creación de reglas no deseadas en las tablas de flujos de los switches. Además, recientemente se ha demostrado que una buena práctica para evitar este tipo de ataques es generar un controlador de respaldo de la red. De esta forma, cuando un switch detecta que el controlador principal está indisponible, puede comunicarse con el controlador secundario para que la red continue funcionando sin cortes. [16, pág. 769]

Otro tipo de ataque ampliamente extendido en las redes tradicionales, y que también afecta a las redes SDN es el de denegación de servicio. Especialmente importantes en las redes definidas por software son los ataques de Denegación de Servicio (DoS) de desbordamiento de tabla de flujos y DoS de desbordamiento de buffer. Estos se producen cuando un nuevo paquete llega a los switches de la red y no encaja con ninguno de los flujos o reglas de sus tablas, entonces los switches envían un mensaje “packet-in” mediante el protocolo OpenFlow, consultando al controlador cómo deben actuar ante ese tipo de tráfico. Este modo de operación puede ser empleado por un atacante para enviar paquetes modificados para que no encajen con ninguna de las reglas de los switches de la red, por ejemplo, de paquetes con direcciones IP que no se encuentran en la red o protocolos y servicios que no se están utilizando. [16, pág. 769]

El controlador enviará un mensaje “packet-out” mediante OpenFlow a todas las peticiones “packet-in” enviadas por los switches de la red, mediante el cual les indica en un nuevo flujo que añadir a su tabla de flujos para saber cómo deben actuar ante ese tipo de tráfico. En consecuencia, si se reciben masivamente en el switch paquetes malformados que no se corresponden con ningún elemento de la red y este solicita al plano de control reglas de comportamiento, el controlador enviará tantas reglas de flujos a instalar en las tablas de flujos del switches como peticiones de tráfico distinto han llegado al switch, por lo que, llegado un cierto momento, la tabla de flujos o el buffer del switch se quedarán sin capacidad para almacenar nuevas reglas, situación que recibe el nombre de desbordamiento. [16, pág. 769]

Las consecuencias que puede tener esto en la red son nefastas, dado que cuando paquetes legítimos lleguen al switch, como el buffer y la tabla de flujos están saturados, si estos paquetes no tienen ninguna regla en ellas que permita redireccionarlos en el plano de datos, no se podrán instalar nuevas reglas en ellos y estos paquetes serán descartados. Como vemos esto afecta negativamente a la disponibilidad de los caminos de la red. [16, pág. 769]

Algunas de las contramedidas que se pueden aplicar para evitar este tipo de ataques de desbordamiento es emplear varios controladores para segmentos distintos de la red. Esto previene de que un ataque exitoso de desbordamiento en un segmento de la red afecte al resto de esta, puesto que como el controlador únicamente decide el comportamiento de los switches que están bajo su responsabilidad, solo estarán indisponibles los switches

correspondientes a su segmento, mientras que el resto de la red continuará funcionando sin problemas. Otra posible solución es emplear de forma combinada control de acceso dinámico y sistemas de detección de intrusiones en los switches de la red. Esta configuración detecta cualquier comportamiento anómalo en los switches del plano de datos de la red gracias a las políticas dinámicas de tráfico, permitiendo descartar todo el tráfico que no provenga de un origen confiable y autenticado en la red, y bloquearlo. [16, pág. 770]

A continuación, en la siguiente tabla se resumen los ataques y contramedidas detallados en este apartado.

Tipo de ataque	Descripción	Contramedidas
Man-in-the-Middle entre el controlador y el switch	El atacante obliga a los switches de la red a que instalen reglas maliciosas en sus tablas de flujos, evadiendo el uso de las reglas determinadas por el controlador de la red.	Creación de túneles de comunicación entre el controlador y los switches.
Denegación de servicio de desbordamiento de tabla de flujos y de desbordamiento de buffer	La instalación masiva de flujos en las tablas de los switches provoca que las reglas necesarias para encaminar el tráfico legítimo no puedan instalarse, al estar la tabla de flujos y el buffer de los switches colapsados por el tráfico ilegítimo.	- Emplear varios controladores para segmentos distintos de la red. - Control de acceso dinámico y sistemas de detección de intrusiones en los switches

Tabla 1. Ataques y contramedidas del plano de datos.

Amenazas del plano de control

Como se ha indicado en anteriores apartados, el plano de control y, en concreto, el controlador de la red definida por el software es uno de los puntos preferidos por los atacantes para tomar el control de toda la red, debido a que es donde se toman las decisiones sobre el comportamiento de los switches del plano de datos. [16, pág. 770]

El principal ataque al que se somete al controlador de las redes SDN son los ataques de denegación de servicio (DoS) y denegación de servicio distribuidos (DDoS), mediante los cuales se envían ingentes peticiones “packet-in” provenientes de switches maliciosos hacia el controlador, que provocan el agotamiento de sus recursos y, como consecuencia, la indisponibilidad de este. Consecuentemente, los switches legítimos de la red no obtendrán respuesta a sus peticiones al controlador y el envío de paquetes quedará bloqueado. [16, pág. 770]

Algunas de las contramedidas propuestas para acabar con los ataques de denegación de servicio y denegación de servicio distribuida sobre el controlador es el análisis de las métricas de los flujos de los switches, para lo que existen algunas herramientas como FloodGuard. Otra opción es construir una arquitectura de red orientada a contenido o CONA (Content-oriented Network Architecture) mediante la colocación de un proxy entre

el cliente y el servidor de contenido, que pueda comunicarse con el controlador. El proxy intercepta todo el tráfico y analiza si pertenece a un ataque de denegación de servicio. Cuando detecta que se alcanza cierto umbral de peticiones o de consumo de recursos en el controlador, bloquea el tráfico para que no se agoten los recursos y quede indisponible. Además, el proxy se encarga de informar al controlador de que está sufriendo un ataque de DoS o DDoS y este envía un mensaje a cada proxy CONA para que eviten redirigir el tráfico procedente del ataque. [16, pág. 771]

También existen amenazas en las redes en las que el plano de control es distribuido y las acciones de la inteligencia de la red recaen sobre múltiples controladores. En ellas es necesario que los controladores sean vistos por la red como un único ente, para que, en caso de fallo de uno de ellos, el resto puedan asumir sus tareas y no existan problemas de seguridad respecto a la autenticación, autorización y la privacidad en la transmisión de los paquetes. Este tipo de esquemas esconden un riesgo de seguridad asociado a una mala configuración de los controladores del plano de control. [16, pág. 771]

Entre las muchas contramedidas que pueden emplearse para resolver estos problemas de configuración en los esquemas con varios controladores se encuentra el balanceo de carga, que se emplea también para mejorar la escalabilidad de las redes. Otra de las posibles opciones pasa por realizar un adecuado dimensionamiento de los controladores de la red y sus conexiones con los switches de los segmentos de red a los que darán respuesta. [16, pág. 771 – 772]

Por último, las aplicaciones de alto nivel de la red definida por software necesitan conocer información de la red como la que le provee el controlador mediante la *Northbound* API. En muchas ocasiones, estas aplicaciones tienen diferentes requerimientos funcionales que necesitan que se les apliquen políticas de seguridad personalizadas. El problema está en que, en muchos casos, estas políticas aún no han sido diseñadas. [16, pág. 772]

En general, una buena aproximación para resolver estos problemas de las políticas de seguridad que aún no están definidas en las redes definidas por software es emplear la autenticación o el control de acceso de las aplicaciones de alto nivel de la red. Aislando los recursos se consigue que el controlador únicamente proporcione información a las aplicaciones cuando sea estrictamente necesario y cuando así el control de acceso lo avale. Para ello, se pueden emplear diferentes soluciones ya creadas de control de acceso en la *Northbound* API, que gestionan los permisos de las aplicaciones, las autentican y autorizan el consumo de recursos de la red. [16, pág. 772]

A continuación, se incluye una tabla con el resumen de este apartado.

Tipo de ataque	Descripción	Contramedidas
DoS y DDoS al controlador de la red	Bloqueo en la distribución del tráfico de la red por la indisponibilidad del controlador.	- Floodguard. - Arquitectura de red orientada a contenido (CONA).

Mala configuración de los controladores distribuidos de la red	Los controladores son vistos de manera independiente para los componentes de la red, por lo que no existen acciones unificadas sobre el tráfico.	- Balanceo de carga entre los controladores. - Dimensionamiento de los controladores y el segmento de red que cubren.
Políticas de seguridad de la capa de aplicación sin diseñar	Las aplicaciones no cuentan con las políticas de seguridad necesarias para garantizar sus requerimientos funcionales.	Aislar las aplicaciones por medio de la gestión de permisos, autenticación y autorización del consumo de recursos de la red.

Tabla 2. Ataques y contramedidas del plano de control.

Riesgos del plano de aplicación

En esta capa de la red definida por software, los atacantes pueden hacerse con la configuración de la red, robar información, dimensionar los recursos de red e instalar software malicioso a las aplicaciones.

Uno de los principales riesgos de este plano es el acceso ilegal de las aplicaciones a los recursos de red que le permitan controlar el comportamiento de esta. Estas aplicaciones generalmente son desarrolladas por terceros y están instaladas en el controlador, por lo que se asume el riesgo de que por defecto en el controlador no existan políticas o mecanismo de seguridad que controlen el comportamiento de las aplicaciones o que verifiquen su legitimidad, a diferencia de lo que sucede con los dispositivos que se conectan a la red. [16, pág. 773]

Para combatir este riesgo, deben definirse perfiles o permisos de grano fino en la aplicación instalada en el controlador, que permitan el aislamiento de los recursos de red y reforzar la priorización de las aplicaciones en el control de acceso. [16, pág. 773]

Otro de los riesgos en los que se incurre en esta capa de aplicación es el provocado por las reglas de seguridad y los conflictos de las configuraciones. Es necesario que existan en el controlador aplicaciones de seguridad que realicen todas estas funciones de seguridad que se han comentado previamente, pero dada su complejidad y sumado a que las redes SDN están en pleno desarrollo, estas funcionalidades pueden no estar debidamente definidas para algunas soluciones del plano de control. [16, pág. 773]

A continuación, se incluye una tabla con el resumen de este apartado.

Tipo de ataque	Descripción	Contramedidas
Acceso ilegal a recursos	Aplicaciones desarrolladas por terceros que pueden comportarse maliciosamente y afectar a los recursos de la red.	Aislar los recursos por medio de la creación de perfiles de grano fino, control de acceso o autorización.

Conflictos de configuración en las políticas de seguridad de la red	Aplicación desigual de las medidas establecidas, fallos en los accesos, indisponibilidad de los recursos o acceso a recursos sin ningún tipo de control.	El controlador debe disponer de aplicaciones de seguridad que realicen la lógica de permisos y aplicación de controles.

Tabla 3. Ataques y contramedidas del plano de datos.

Una vez definidos los aspectos de seguridad de las redes SDN y los componentes de seguridad de las arquitecturas IT, es necesario presentar las herramientas que se van a emplear para reproducir la parte técnica de este proyecto.

2.6. Presentación del software Mininet

Mininet[2] es un software de código abierto que se emplea para la emulación de redes virtuales realistas sobre un kernel real, switches virtuales y código de aplicaciones. Esta herramienta funciona por medio de máquinas virtuales.

Con algunos comandos básicos de Mininet se pueden generar topologías de pruebas que sirven para verificar el funcionamiento de una red definida por software. Sin embargo, esta herramienta también permite la creación de scripts en Python con código para programar la red que se quiere emular. Una vez creado el script, se ejecuta en el entorno de Mininet para construir la topología configurada en él.

El motivo con el que se ha escogido este software para la emulación de redes definidas por software en este trabajo de fin de máster es su simplicidad, la facilidad con la que pueden simularse redes customizadas con la generación de un script, además de la gran documentación que existe en internet y de que se trata de una herramienta completamente gratuita.

Otros proveedores de servicios de redes definidas por software como CISCO, ofrecen emuladores o simuladores para poder probar configuraciones de redes definida por software, sin embargo, estos programas están sujetos a desarrollo y código propietarios de estas empresas, por lo que en ocasiones el estudio del comportamiento del controlador o del plano de datos de la red será el que se aplica concretamente a su propia tecnología, difiriendo de lo que idealmente se considera las redes definidas por software.

Mininet es una potente herramienta que ayuda a desarrollar compartir y experimentar con las redes SDN, especialmente enfocada en el protocolo OpenFlow para la comunicación entre el plano de control y el plano de datos. La herramienta puede ser consultada en la

[2] http://mininet.org/

siguiente página web: Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet[3].

[3] http://mininet.org/

3. Diseño técnico de la arquitectura SD-WAN

Para poder desarrollar los objetivos marcados en el presente trabajo, ha sido necesario estudiar el comportamiento de una arquitectura que cuente con las principales medidas técnicas de seguridad detalladas en los anteriores apartados de este documento.

A continuación, se exponen los requisitos técnicos y los elementos que se han utilizado para implementar la SD-WAN desarrollada durante este proyecto.

3.1. Componentes de la arquitectura de red

A continuación, se detallan los elementos de seguridad y los componentes auxiliares, adicionales a las funciones que se detallarán en los dos siguientes apartados en los dos apartados anteriores, que completan el diseño de la arquitectura de red.

3.1.1. Descripción del diagrama de red y sus componentes

La topología de red diseñada está formada por la SD-WAN, que interconecta varias sedes geográficas de una organización, infraestructura de la misma organización alojada en la nube y usuarios remotos. El esquema de esta red se puede observar en la figura siguiente.

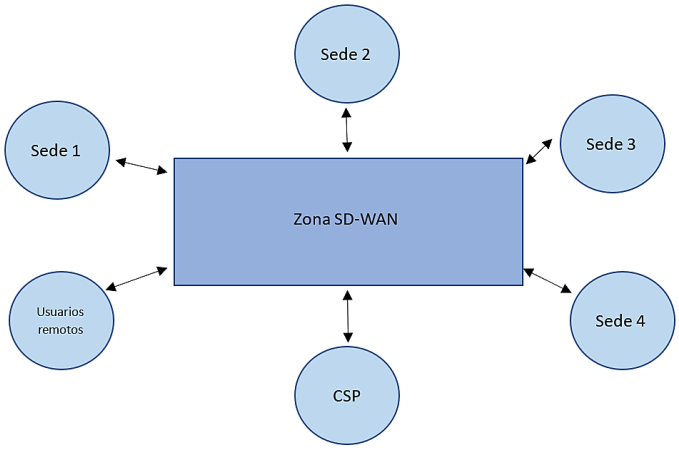


Figura 4. Esquema de la red diseñada.

Para comprobar la funcionalidad descrita en el estado del arte de este trabajo, ha sido necesario incluir en el diagrama de red los siguientes elementos de seguridad:

- Servidores que actúan como balanceadores de carga. Son la capa externa de la SD-WAN, reciben en primera instancia las peticiones remotas o de otras sedes geográficas que van dirigidas a las redes internas de los sistemas de la organización. Son los encargados de redirigir el tráfico a un servidor seleccionado de un conjunto de *pool* de servidores.

- *Pool* de servidores: formado por cinco servidores encargados de resolver las peticiones derivadas por los balanceadores de carga. Estos servidores son escogidos por los balanceadores de carga por medio de los algoritmos mencionados en el apartado anterior.

- Switches con comportamiento de firewalls: cuentan con reglas configuradas por el controlador de la red para redirigir el tráfico y con reglas configuradas de manera estática en el controlador para filtrado de paquetes dentro de la zona SD-WAN.

- Switches en modo aprendizaje: son elementos con la configuración tradicional de un switch, que aprenden su comportamiento a partir de las reglas configuradas en sus tablas de flujos por el controlador.

En la Tabla 1 se puede consultar el listado de elementos de la red, la función que tienen dentro de la misma y la zona en la que está ubicado dentro de ella.

Componente	Descripción	Zona
server11, server12, server13, server14, server15	*Pool* de servidores públicos que están expuestos a la red en la sede 1.	Red interna Sede 1
server21, server22, server23, server24, server25	*Pool* de servidores públicos que están expuestos a la red en la sede 2.	Red interna Sede 2
server31, server32, server33, server34, server35	*Pool* de servidores públicos que están expuestos a la red en la sede 3.	Red interna Sede 3
branch4	*Host* expuesto y accedido públicamente de la sede 4.	Red interna Sede 4
cloud	*Host* expuesto y accedido públicamente de la infraestructura cloud de la organización.	Red interna Sede Cloud
ngfw1	Firewall de capas 2, 3, 4 y 7 encargado de filtrar el tráfico destinado a la sede 1. Elemento de la SD-WAN que se conecta al *pool* de servidores de dicha sede y al switch sb1.	SD-WAN
sb1	Switch con comportamiento de aprendizaje que recibe el tráfico entrante a la SD-WAN y destinado a la sede 1. Se encarga de interceptar y redirigir el tráfico que llega desde el switch s1 hacia el ngfw1.	SD-WAN
lb1	*Host* con comportamiento de balanceador de carga de la sede 1. Tiene el algoritmo aleatorio del fichero *lb.py* configurado para balancear el tráfico.	SD-WAN
fw2	Firewall de capas 3, 4 y 7 encargado del filtrado de paquetes con destino la sede 2. Está conectado al elemento al *pool* de servidores de dicha sede y al elemento sb2.	SD-WAN
sb2	Switch con comportamiento de aprendizaje que recibe el tráfico entrante a la SD-WAN y destinado a la sede 2. Se	SD-WAN

	encarga de interceptar y redirigir el tráfico que llega desde el switch s1 hacia el fw2.	
lb2	*Host* con comportamiento de balanceador de carga de la sede 2. Tiene configurado el algoritmo del fichero *lb_round_robin.py* para balancear el tráfico.	SD-WAN
sb3	Switch con comportamiento de aprendizaje que recibe el tráfico entrante a la SD-WAN y destinado a la sede 3. Se encarga de interceptar y redirigir el tráfico que llega desde el switch s1 hacia el *pool* de servidores de dicha sede.	SD-WAN
lb3	*Host* con comportamiento de balanceador de carga de la sede 3. Tiene configurado el algoritmo del fichero *lb_weighted_round_robin.py* para balancear el tráfico.	SD-WAN
fw4	Firewall de capas 2, 4 y 7 encargado de filtrar el tráfico de destinado a la sede 4. Elemento de la SD-WAN que se conecta al *pool* de servidores de dicha sede y a s1.	SD-WAN
s1	Switch con comportamiento de aprendizaje que recibe el tráfico procedente de la red pública hacia la infraestructura de la organización.	SD-WAN
user	*Host* externo a la organización que envía peticiones hacia los recursos de esta.	Red pública
s2	Switch con comportamiento de aprendizaje que recibe el tráfico con origen o destino el *host user*.	Red pública

Tabla 4. Componentes de la red, función y ubicación.

El objetivo de este trabajo es comprobar el funcionamiento de la SD-WAN, no se ha tenido en cuenta la generación del script el desarrollo de las redes locales (LANs) de cada una de las sedes de la organización. Simplemente se representa en el diagrama de red el *pool* de servidores de cada sede ubicado en su DMZ, que es el encargado de transmitir las peticiones procedentes de los sistemas y recursos alojados en la sede hacia la SD-WAN. De este modo, cada sede tiene un servidor asociado a ella que será el que transmita y reciba el tráfico de la SD-WAN.

La topología de red descrita hasta ahora puede verse representada en la siguiente figura. En ella, los enlaces que conectan los elementos de la red son de capacidad de 1000 Mbps, por lo que se espera que la red acepte como máximo ese flujo de datos por sus enlaces, para evitar que se produzcan situaciones de congestión y descarte de paquetes.

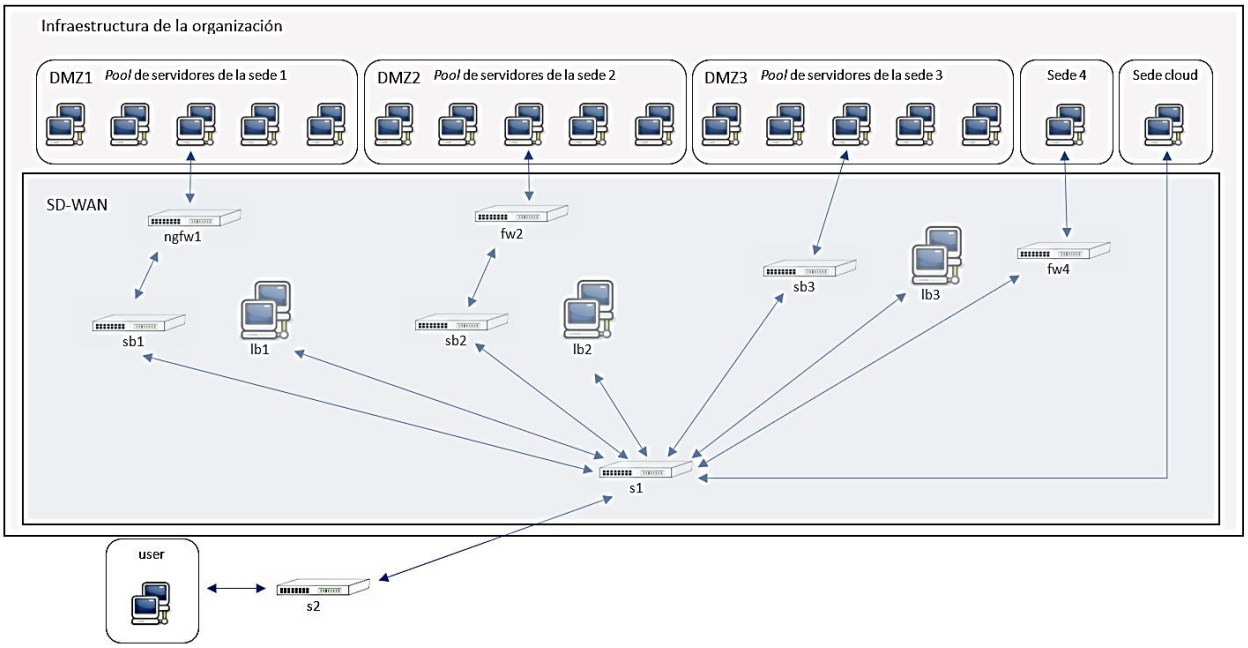


Figura 5. Arquitectura de red de la organización.

3.1.2. Matriz de comunicaciones

Otro aspecto interesante que hay que definir es qué comunicaciones estarán permitidas por la SD-WAN. En la siguiente tabla se establecen limitaciones al tráfico de la red.

Origen \ Destino	Sede 1	Sede 2	Sede 3	Sede 4	Cloud	User
Sede 1	-	OK	KO	OK	OK	OK
Sede 2	OK	-	KO	OK	KO	OK
Sede 3	KO	KO	-	OK	OK	OK
Sede 4	OK	OK	OK	-	OK	OK
Cloud	OK	KO	OK	OK	-	OK
User	OK	OK	OK	OK	OK	-

Tabla 5. Matriz de comunicaciones de la red.

Para interpretar la Tabla 5, es necesario tener en cuenta un OK significa que el tráfico con origen la primera celda de la fila y destino la primera fila de la columna es válido dentro de la red. Un valor KO indica que ese tráfico no está permitido. Por ejemplo, siguiendo lo indicado en la tabla, no está permitido el tráfico de la sede 1 hacia la sede 3. En la tabla anterior, se puede apreciar que el grado de comunicaciones de la red ronda el 60% de las comunicaciones totales.

Para comunicar aquellas sedes que no tienen alcance según la tabla anterior, se propone añadir configuración de segmentación de red basada en las siguientes VLANs.

VLAN	Interfaz 1	Interfaz 2
eth0.100	server11-eth0	server31-eth0
eth0.200	server32-eth0	server21-eth0
eth0.300	server22-eth0	cloud-eth0

Tabla 6. Configuración de VLANs en la red.

Con la configuración de segmentación propuesta, el tráfico viajará etiquetado dentro de la SD-WAN y podrá ser filtrado en los firewalls por la VLAN a la que pertenece para ser permitido. La configuración de las VLANs indicada en la Tabla 6 permitirá que el 11% de los enlaces entre la sede 1 y la sede 3, entre la sede 2 y la sede 3 y entre la sede 2 y la sede cloud, permitan las comunicaciones.

3.1.3. Mediciones del rendimiento de la red y del nivel de seguridad

Finalmente, cuando se disponga al completo de la red configurada, incluidos balanceadores de carga, firewalls, switches y *host*s, se procederá a verificar y comparar cómo mejoran los datos de rendimiento de la red en base a tres variables:

- Ancho de banda disponible.
- Tiempo de respuesta.
- Métricas de seguridad basadas en el riesgo de ataques y en porcentajes de bloqueo.

De forma preliminar, se estima que aquellas comunicaciones que tengan lugar entre orígenes y destinos protegidos por balanceadores de carga y firewalls de varios tipos serás las que presenten un mayor ancho de banda y un tiempo de respuesta menor. Seguidamente, se encontrarán las comunicaciones que solo tengan uno de estos tipos de tecnologías configuradas, y, finalmente, se encontrarán las comunicaciones que no presenten ningún tipo de protección desde origen hasta destino en la red.

Para llevar a cabo estas comprobaciones, se dispone de los caminos de comunicación de la red indicados en la tabla siguiente.

Funciones de seguridad	Origen	Destino
Balanceo de carga, firewall de capas 2 , 3, 4 y 7	User	Sede 1
Balanceador de carga y firewall de capas 3, 4 y 7	User	Sede 2
Balanceador de carga	User	Sede 3
Firewall de capas 2, 4 y 7	User	Sede 4
Sin protección	User	Cloud

Tabla 7. Caminos de comunicación y elementos de seguridad.

Para realizar las mediciones pertinentes y obtener los datos relativos al tiempo de respuesta y al ancho de banda disponible se emplearán herramientas de la consola de Linux que sirven para testear el comportamiento real de las redes en producción.

Para las mediciones relativas a la seguridad de la red, se emplearán métricas de riesgos y porcentaje de bloqueo de la red, que serán descritos en el siguiente apartado.

Una vez definidos los requisitos de la red, dibujado el esquema del diagrama de arquitectura, establecida la matriz de comunicaciones y propuestos los indicadores que se van a emplear para medir y comparar las configuraciones de las funciones de seguridad de la SD-WAN, es necesario programar el script de Python que definirá el comportamiento de la topología de red. Además, también será necesario definir los algoritmos de balanceo de carga que se van a utilizar dentro de la red y configurar los firewalls. Todos estos resultados se abordarán en la siguiente sección de este documento.

3.2. Requisitos técnicos de seguridad en la SD-WAN

A lo largo de este apartado se detallan las necesidades que debe cumplir a nivel funcional y técnico la red SD-WAN para ejecutar las pruebas necesarias en ella.

1. Es necesario implementar reglas de filtrado de paquetes que cuenten con la capacidad de denegar el tráfico no deseado en la red. Estas reglas pueden configurarse por medio de filtros que apliquen a los siguientes campos de los paquetes transmitidos:

    - Direcciones IP de origen del tráfico y destino.
    - Protocolo que se usa para las comunicaciones.
    - Direcciones MAC del origen y del destino.
    - VLAN, en el caso en que la red esté segmentada.

    Estas reglas se deben configurar en los equipos del plano de datos de la red, es decir, en los switches. Estos elementos son los que reciben, a través de sus puertos conectados a la red, el tráfico que deben encaminar correctamente para que llegue a su destino, por lo que se puede aprovechar que todo el tráfico de la red es interceptado y redirigido por ellos para que realicen también labores de filtrado.

    Para medir que se ha realizado adecuadamente el bloqueo de las comunicaciones de la red de acuerdo con lo establecido en el apartado de diseño de la red, los firewalls deberán bloquear al menos el 40% del tráfico generado en la red. Para el tráfico configurado por VLANs, se deberá cumplir que únicamente se aceptan las configuraciones de la tabla 6, que supone un 11% de los enlaces entre las sedes no comunicadas.

2. Implementar mecanismos de mejora de rendimiento de los servidores expuestos a la red pública de las diferentes sedes. Uno de los objetivos del trabajo es medir mediante el tiempo de respuesta y el ancho de banda disponible en la red, la eficacia de las medidas de seguridad implementadas para demostrar que no solo introducen mejoras en la capa de seguridad, sino que también contribuyen al buen funcionamiento y disponibilidad de los elementos de la red.

En este apartado, entra en juego la triada de la disponibilidad, integridad y confidencialidad (conocida por sus siglas en inglés, CIA), con la que las expertas y expertos en ciberseguridad se refieren a que la seguridad de las redes modernas no debe centrarse únicamente en proteger y cifrar las comunicaciones y datos, sino que también debe ocuparse de ofrecer unos buenos niveles de servicio de disponibilidad y respuesta a los usuarios.

Para comprobar que se han aplicado de forma satisfactoria los mecanismos de balanceo de carga y filtrado de paquetes, durante el análisis de resultados se verificará que el ancho de banda disponible y que el tiempo de respuesta de las comunicaciones que se dirigen hacia la *branch1* son mejores que las que van dirigidas hacia la infraestructura *cloud*. En términos generales, el ancho de banda máximo de la red deberá ser superior a 50 Mbps y el tiempo de respuesta máximo deberá ser inferior a 0,15 segundos.

3. La red diseñada debe contar con medidas de protección de comunicaciones, empleando protocolos seguros, que garanticen la autenticación en ambos extremos y que cifren el tráfico en tránsito, con el objetivo de evitar que cualquiera que pueda interceptarlo pueda modificarlo o descubra su contenido.

   Por esta parte, se deberá comprobar en el apartado de resultados que se garantiza un 100% de bloqueo de tráfico no permitido dentro de la red: UDP, HTTP y tráfico sin cifrar.

4. Finalmente, será necesario establecer límites para el tráfico que pueden recibir las sedes de la arquitectura de IT que se va a implementar. Para estos límites hay que determinar qué comunicaciones estarán permitidas en función de: origen, destino, protocolo o segmento de la red.
5. Por otro lado, se emplearán métricas del *National Institute of Standards and Technology* (NIST) [18, pág. 14], para medir el nivel de riesgo de la red en función del esfuerzo que deben hacer los atacantes para vulnerar los caminos hasta llegar a los recursos de negocio. Para ello, se van a emplear los siguientes indicadores:
   a. Número de caminos de ataque (NAP, por sus siglas en inglés): indica los caminos de ataque existentes en una red, respecto a los caminos totales de dicha red.
   b. Esfuerzo medio de los caminos de ataque (ALAP): expresa la media de la dificultad de ataque a la red, en función de la vulnerabilidad de sus caminos.
   c. Camino más corto de ataque (SAP): indica el mínimo esfuerzo que debe hacer un atacante para vulnerar la red, normalmente es a través del camino más desprotegido.

A continuación, en la siguiente tabla se resumen las métricas anteriormente descritas que se van a emplear para medir el rendimiento y los riesgos de ataque dentro de la red.

Métrica	Medición	Umbral
Ancho de banda máximo	En cada uno de los caminos desde *user* hacia las sedes	Mayor a 50 Mbps
Tiempo de respuesta	En cada uno de los caminos desde *user* hacia las sedes	Menor a 0,15 segundos
Porcentaje de bloqueo	En todas las comunicaciones de la red	40% de bloqueo
	Comunicaciones por VLANs	89% de bloqueo
	Comunicaciones UDP	100% de bloqueo
	Comunicaciones HTTP	100% de bloqueo
	Comunicaciones no cifradas	100% de bloqueo
Métrica de riesgos de ataque	Basado en el riesgo de seguridad presente en cada camino de la red.	Comparativa de los diferentes caminos

Tabla 8. Métricas del rendimiento y de la seguridad de la red, cubiertas por la SD-WAN.

Estas son los requisitos técnicos que se han definidos para comprobar la funcionalidad de una red con SD-WAN. En el siguiente apartado, se puede encontrar los elementos de red que se van a utilizar para implementarlos en la SD-WAN simulada.

3.3. Elementos de la red con funciones de seguridad

Como se ha indicado anteriormente, los elementos que se van a describir en el presente apartado son los que van a dotar a la red de las funcionalidades técnicas mencionadas anteriormente.

En primer lugar, para realizar las tareas de filtrado y limitación de las comunicaciones se ha seleccionado el uso de firewalls o cortafuegos, dado que son los elementos tradicionales presentes en las redes de las empresas y también existe la posibilidad de que sean configurados en las soluciones SD-WAN. De hecho, entre las capacidades actuales de las soluciones SD-WAN existentes en el mercado, los fabricantes incluyen la capacidad de filtrado de paquetes por medio de soluciones basadas en infraestructura cloud y conocidas como *firewall as a service*. En concreto, se van a utilizar los siguientes tipos de firewalls:

- Capa 2: para eliminar el tráfico de la capa de enlace de datos que no sea deseado en la red. En concreto, en este nivel se impide el tráfico de peticiones ARP, que emplean la dirección de broadcast y pueden ocasionar la caída de la red por inundación o *flooding*, y que son el modo de efectuar los ataques mencionados en el apartado 2.5.2 (denegación de servicio por desbordamiento de las tablas de flujos o del buffer de los switches, y denegación de servicio al controlador de red).

- Capa 3: para el filtrado de los paquetes en la capa de red, por ejemplo, empleado para restringir el acceso a determinadas direcciones IP de la infraestructura de la empresa, para restringir el tráfico proveniente de ciertas direcciones IP o segmentos de la red.

- Capa 4: impiden el paso de paquetes del protocolo UDP, puesto que es ampliamente empleado para realizar técnicas de *spoofing* o falsificación de la dirección IP origen. Esta técnica es empleada por atacantes para dirigir miles de

peticiones hacia los componentes públicos de las infraestructuras tecnológicas de las organizaciones y ocasionar ataques de denegación de servicio. Con la implementación de estas reglas se dota de protección a los switches de la red y al controlador de estos ataques.

- Impedir el uso de otros servicios inseguros como FTP o HTTP.

En segundo lugar, se van a emplear balanceadores de carga para mejorar el rendimiento de las máquinas expuestas a la red pública. Los balanceadores de carga, como ya se ha mencionado en apartados anteriores, se encargan de redirigir las peticiones destinadas a una sede concreta de la organización a un conjunto de servidores, conocido como *pool* o piscina de servidores dentro de la DMZ de esa sede. En el *pool*, las comunicaciones serán procesadas y derivadas a los servicios internos correspondientes. Para implementar las funciones de balanceo de carga de la red se van a utilizar tres tipos de algoritmos:

- Aleatorio: mediante el cual el balanceador de carga conoce los nombres de los servidores alojados en un *pool* de servidores y selecciona aleatoriamente de entre ellos el servidor que deberá procesar la petición. Para cada petición se selecciona un servidor en concreto. Un ejemplo de configuración de este algoritmo se puede encontrar en el anexo de este documento en el fichero *lb.py*.

- Round Robin: con el que el balanceador de carga selecciona, siguiendo una lista ordenada, al servidor encargado de procesar las peticiones. En este caso, si hay cinco servidores en el *pool* de servidores y empieza a seleccionar por el servidor 1, el orden de selección será: servidor 1, servidor 2, servidor 3, servidor 4, servidor 5, servidor 1, y así sucesivamente. Un ejemplo de implementación de este algoritmo se halla en el fichero *lb_round_robin.py* del anexo de este documento.

- Round Robin con pesos: es el mismo caso que el algoritmo anterior, pero con la configuración de pesos para los distintos servidores. En este tipo de algoritmos, si al servidor 1 se le da un peso 4, tendrá que procesar 4 peticiones antes de que sea seleccionado el servidor 2. Un ejemplo de este tipo de algoritmo puede consultarse en el fichero *lb_round_robin_weighted.py* de anexo de este documento.

En tercer lugar, se hará uso de mecanismos de cifrado de las comunicaciones y de la información en tránsito. La información y los datos de las personas y organizaciones son empleados actualmente como arma de soborno y extorsión por parte de atacantes maliciosos que logran acceder a estos datos, por lo que es necesario garantizar que se mantienen íntegros y confidenciales para, en caso de ser interceptada, no pueda ser utilizada en contra de la organización.

Estas han sido las técnicas implementadas para la configuración de la WAN definida por software que se desea examinar en este proyecto. Se han seleccionado los firewalls para el filtrado de paquetes y los balanceadores de carga en modo *pool*, dado que son soluciones basadas en los productos existentes actualmente en el mercado, que ya están siendo implementadas por otras soluciones SD-WAN. Además, el cifrado de las comunicaciones con el uso de certificados también es ampliamente utilizado en la actualidad. Con la configuración descrita en este apartado, se pretende lograr alcanzar un escenario lo más fiel posible a la realidad de las empresas y organizaciones de estos tiempos. Es destacable mencionar que, los diferentes mecanismos empleados para proteger los recursos se emplearán para, posteriormente, obtener las métricas y resultados de este trabajo.

4. Resultados

A lo largo del apartado anterior se ha realizado el diseño de la arquitectura de red. En él se han definido:

- Los requisitos funcionales que van a ser testeados en el transcurso del proyecto
- Su implementación por medio de los elementos de la red con funciones de seguridad.
- Se ha presentado un diagrama de red con todos los componentes necesario para dar soporte a las funciones de seguridad de la SD-WAN.
- Se ha definido la matriz de comunicaciones, mediante la cual se han establecido los límites del tráfico a aceptar o rechazar por los elementos de seguridad.
- Finalmente, se ha indicado cómo se realizará el procedimiento de chequeo de las variables tiempo de respuesta y ancho de banda disponible para verificar la mejora de rendimiento de la red con las diferentes configuraciones propuestas.

Por tanto, en este capítulo se optará por presentar una descripción de los ficheros Python que se han generado para configurar la topología de la red diseñada, indicar su flujo de comunicaciones y realizar las pruebas que serán el producto final del proyecto.

4.1. Desarrollo de una topología de red basada en SDN con Mininet

En este primer apartado de este capítulo se describe la lógica de los elementos de red y configuraciones provistas por los distintos scripts.

Para el desarrollo de las pruebas se han generado cinco scripts en Python:

- *topo.py*: script principal que genera y configura los componentes del diagrama de red y las comunicaciones entre los elementos.
- *lb.py*: script auxiliar que se emplea para configurar el algoritmo de los balanceadores de carga.
- *lb_round_robin.py*: segundo script auxiliar que usan los balanceadores de carga.
- *lb_weighted_round_robin.py*: tercer script empleado por los balanceadores de carga.
- *https_server.py*: configuración de un servidor HTTPS con certificado SSL para el cifrado del tráfico en tránsito.

A continuación, se describen con detalle el alcance de las configuraciones abordadas en cada uno de ellos.

4.1.1. topo.py

Este script puede ser consultado en el anexo de este documento. Para hacer más sencillo su entendimiento, se van a describir las configuraciones implementadas mediante este script en función de las partes en las que se encuentra dividido.

Parte 1

En esta primera parte del script se añaden a la red los siguientes elementos:

- Sede 1: cinco servidores (*server11, server12, server13, server14, server15*).
- Sede 2: cinco servidores (*server21, server22, server23, server24, server25*).
- Sede 3: cinco servidores (*server31, server32, server33, server34, server35*).
- Sede 4: un servidor (*branch4*).
- Cloud: un servidor (*cloud*).
- Red pública: una máquina (*user*) conectada a un switch (*s2*) que redirige su tráfico.
- SD-WAN:
    - Para el tráfico de la sede 1: un switch (*ngfw1*) que actúa como *Next Generation Firewall*, un switch (*sb1*) con comportamiento de aprendizaje y un servidor balanceador de carga (*lb1*).
    - Para el tráfico de la sede 2: un switch (*fw2*) con comportamiento de firewall, un switch (*sb2*) con comportamiento de aprendizaje y un servidor balanceador de carga (*lb2*).
    - Para el tráfico de la sede 3: un switch (*sb3*) con comportamiento de aprendizaje y un servidor balanceador de carga (*lb3*).
    - Para el tráfico de la sede 4: un switch (*fw4*) con comportamiento de firewall.
    - Un switch (*s1*) que reciba el tráfico procedente de la red pública y el del resto de sedes.

Para crear los servidores y máquinas tipo *host* se ha empleado la siguiente línea de código en el script:

```
server11 = self.addHost('server11', ip='10.0.1.1/16',
                        mac='00:00:00:00:00:11')
```

Para crear los switches, a los que posteriormente se dotará de configuración, se ha empleado la siguiente línea de código:

```
sb1 = self.addSwitch('sb1')
```

Parte 2

En esta zona se generan los enlaces existentes entre los componentes de la red creados en la parte 1 del script. Se ha empleado la siguiente línea de código para crear los links:

```
self.addLink(branch4, fw4, bw=1000, delay='1ms')
```

Para definir el enlace hay que indicar los extremos que quieren conectarse, el ancho de banda disponible en Mbps y el retardo introducido en ms. En este caso, se dispone de 1000 Mbps y 1 ms de retardo en los enlaces de la red.

Para comprobar qué enlaces se han establecido en la topología se puede consultar el diagrama de red del capítulo 3 de este documento. En total, se han generado 28 enlaces.

Parte 3

Última parte del script en la que se define el controlador que se va a emplear en la red, la configuración de los balanceadores de carga, de los servidores dispuestos en las sedes y las reglas instaladas en las tablas de flujos de los switches que han de filtrar los paquetes.

Como controlador se ha seleccionado el tipo *OVSController,* incluido dentro de las funcionalidades provistas por el emulador Mininet.

```
net = Mininet(topo=topo, link=TCLink, controller = OVSController)
```

Respecto a los balanceadores de carga, se han seguido las directrices y algoritmos definidos en el capítulo de diseño técnico de la arquitectura, en el apartado de requisitos técnicos. Para configurar el comportamiento de los servidores balanceadores de carga, se han empleado las siguientes líneas de código:

```
lb1 = net.get('lb1')
lb1.cmd('python3 lb.py &')
```

Con la primera de ellas se devuelve el servidor *lb1* como objeto, y con la segunda se emplea su interfaz de comandos (CLI, por sus siglas en inglés) para cargar la configuración contenida en el script *lb.py*, en el que se encuentra programado el algoritmo que utilizará para el balanceo de carga ese servidor.

Para que los balanceadores de carga trabajen correctamente, deben derivar sus peticiones a los servidores ubicados en la zona perimetral de las sedes. Esto quiere decir que en estos servidores es necesario contar con un servicio de respuesta a las peticiones HTTPS que llegan desde el balanceador de carga, para lo cual es necesario emplear las siguientes líneas de código:

```
server11 = net.get('server11')
server11.cmd('python3 https_server.py &')
```

Con la segunda línea de código, se emplea la CLI del servidor *server11* de la sede 1 para iniciar en esta máquina un servidor HTTPS en el puerto 443.

A continuación, se han creado las interfaces de las diferentes VLANs indicadas en el apartado de los requisitos técnicos del diseño. Para ello, se han utilizado las siguientes líneas de código en el script.

```
server11.cmd('sudo ip link add link server11-eth0 name eth0.100
                    type vlan id 100')
          server11.cmd('sudo ip link set eth0.100 up')
 server11.cmd('sudo ip addr add 192.168.0.11/24 dev eth0.100')
```

Con la primera de ellas, se crea la interfaz eth0.100 en el *server11* y se le asigna el identificador de VLAN100. A continuación, se levanta la interfaz y, finalmente, se configura la IP privada que tendrá asignada.

En último lugar, se han implementado las reglas de las tablas de flujos en cada uno de los switches firewalls de la red. En general, los switches cuentan con un comportamiento de *deny-all* por defecto, al que se han incluido algunas excepciones para permitir el tráfico desde ciertos orígenes y hacia ciertos destinos.

El tráfico permitido se puede consultar en el capítulo anterior, en la sección de la matriz de comunicaciones de la red. En general, los firewalls se han configurado asignando diferentes niveles de prioridad a las reglas:

6. Prioridad 100: para filtrar las comunicaciones etiquetadas con VLANs entre las distintas sedes. Estas conexiones deben ser filtradas con mayor prioridad ante otras debido a que, de no ser así, serían bloqueadas por los firewalls. Un ejemplo para configurar estas reglas es el siguiente:

```
    ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=100,dl_vlan=100,action=NORMAL"')
```

7. Prioridad 20: se ha asignado a las reglas que bloquean los servicios inseguros de la red, tales como UDP o HTTP. La regla configurada ha sido la siguiente:

```
fw4.cmd('ovs-ofctl add-flow fw4 "priority=20,udp,action=DROP"')
```

8. Prioridad 10: se ha asignado a las reglas que indican que el tráfico es aceptado por el firewall y ha de consultarse al controlador qué acción ejecutar. La siguiente línea de código es un ejemplo de una regla configurada en la tabla de flujos del switch *fw4.*

```
    fw4.cmd('ovs-ofctl add-flow fw4
"priority=10,nw_src=10.0.1.1,action=NORMAL"')
```

9. Prioridad 5: se ha asignado esta prioridad a la regla para denegar por defecto el tráfico en los firewalls. La siguiente línea de código indica esta configuración en el script.

```
fw4.cmd('ovs-ofctl add-flow fw4 "priority=5,action=DROP"')
```

Las prioridades indicadas en las reglas anteriores expresan el orden que han de seguir los firewalls para realizar el filtrado de tráfico. A mayor prioridad, antes deberá comprobarse si el paquete inspeccionado encaja con los campos definidos en ella para ejecutar la acción que se indique. De este modo, en primer lugar, se comprobará si los paquetes recibidos se corresponden con tráfico ilegítimo UDP o HTTP. De no ser así, se verificarían las reglas de prioridad 10, que establecen un filtrado en función de las direcciones IP o MAC de origen. En el caso en el que se cumpla que es tráfico permitido, el firewall reenviará el paquete a través de sus interfaces, y, en caso contrario, se pasará a comprobar la regla de prioridad 5. Esta regla denegará por defecto todo el tráfico que no haya sido aceptado por las reglas de prioridad 10.

Las prioridades son necesarias para establecer una jerarquía durante el filtrado del tráfico. A modo de ejemplo, es necesario que se ejecute la regla que bloquea el tráfico UDP y HTTP antes de que se filtre por dirección IP origen, puesto que, si se hiciera al revés, se permitiría tráfico de servicios ilegítimos en la red, que es aceptado porque la regla que filtra el origen es más prioritaria y se ejecuta antes.

4.1.2. lb.py

Con este script se configura el comportamiento del balanceador de carga *lb1* de la red que balancea el tráfico procedente desde la red pública destinado a los servidores de la zona perimetral de la sede 1. En él, se indica el listado de direcciones IP y puertos en los que los servidores del *pool* de balanceo de carga están a la escucha de peticiones HTTPS.

Además, se programa una función que es la encargada de seleccionar de manera aleatoria qué servidor será al que se debe redireccionar al navegador del cliente para que procese su petición.

El comportamiento de este script define que, cuando se selecciona un servidor, se devuelve al cliente una respuesta HTTPS con código 302 y el servidor al que va a ser redirigido.

El contenido del script está disponible en el anexo de este documento.

4.1.3. lb_round_robin.py

En este script se aprovecha las funcionalidades genéricas descritas para *lb.py*, con la diferencia de que se implementa el algoritmo estático Round Robin en el servidor de balanceo de carga *lb2* para gestionar los recursos disponibles en la zona perimetral de la sede 2 y seleccionar a los servidores que resolverán las peticiones de los usuarios en cada momento.

El algoritmo programado sigue las indicaciones descritas en el capítulo de requisitos técnicos, es decir, se selecciona de manera cíclica qué servidor del *pool* de balanceo de carga servirá las peticiones. Para ello, se ha hecho uso del paquete Python *itertools* y de la función *cycle*. El código de este fichero se puede consultar en el anexo de este documento.

4.1.4. lb_weighted_round_robin.py

Empleado para configurar el comportamiento dinámico del servidor de balanceo de carga *lb3*, este script es una extensión de la configuración del fichero *lb_round_robin.py*. El algoritmo basado en pesos define unos pesos iniciales para los servidores en función de los cuales se ejecutará el primer ciclo de selección de servidores, pidiendo a cada servidor del *pool* que ejecute tantas peticiones sucesivas como valor de peso tenga asignado.

Una vez comenzado el siguiente ciclo, se vuelven a calcular los pesos a asignar a los servidores del *pool*, en el caso del script de ejemplo se realiza de manera aleatoria, de modo que para el siguiente ciclo las peticiones realizadas por cada servidor serán diferentes a las del primer ciclo. Cada vez que se inicia un ciclo se vuelven a calcular los pesos, por lo que se tratade un algoritmo de balanceo de carga de asignación dinámica.

El contenido de este fichero de configuración se encuentra disponible en el anexo de este documento.

4.1.5. https_server.py

Finalmente, para configurar que los servidores de los *pools* de las diferentes sedes, así como los servidores balanceadores de carga, acepten únicamente servicios cifrados en tránsito, ha sido necesario añadir la configuración de un servidor HTTPS en el fichero *https_server.py*, que se puede consultar en el anexo.

Para esta configuración se ha generado un certificado autofirmado mediante el siguiente comando:

```
openssl req -x509 -newkey rsa:2048 -nodes -keyout key.pem -out cert.pem
-days 365
```

Aunque se debería generar un certificado para cada servidor de la red, se ha optado por simplificar la configuración de estos empleando el mismo para todos ellos. No obstante, en una red en producción real cada servidor debe ser configurado con su propio certificado expedido por una entidad certificadora de confianza.

4.1.6. Topología de red resultante

La arquitectura diseñada cuenta con los siguientes nodos, interfaces de los componentes, conexiones entre las interfaces y enlaces generados para establecer comunicaciones entre los distintos puntos de la red.

```
mininet> nodes
available nodes are:
branch4 c0 cloud fw2 fw4 lb1 lb2 lb3 ngfw1 s1 s2 sb1 sb2 sb3 server11 server12 server
13 server14 server15 server21 server22 server23 server24 server25 server31 server32 s
erver33 server34 server35 user
```

Figura 6. Nodos disponibles en la red SDN.

```
mininet> dump
<Host branch4: branch4-eth0:10.0.4.1 pid=44999>
<Host cloud: cloud-eth0:10.0.5.1 pid=45001>
<Host lb1: lb1-eth0:10.0.0.1 pid=45003>
<Host lb2: lb2-eth0:10.0.0.2 pid=45005>
<Host lb3: lb3-eth0:10.0.0.3 pid=45007>
<Host server11: server11-eth0:10.0.1.1 pid=45009>
<Host server12: server12-eth0:10.0.1.2 pid=45011>
<Host server13: server13-eth0:10.0.1.3 pid=45013>
<Host server14: server14-eth0:10.0.1.4 pid=45015>
<Host server15: server15-eth0:10.0.1.5 pid=45017>
<Host server21: server21-eth0:10.0.2.1 pid=45019>
<Host server22: server22-eth0:10.0.2.2 pid=45021>
<Host server23: server23-eth0:10.0.2.3 pid=45023>
<Host server24: server24-eth0:10.0.2.4 pid=45025>
<Host server25: server25-eth0:10.0.2.5 pid=45027>
<Host server31: server31-eth0:10.0.3.1 pid=45029>
<Host server32: server32-eth0:10.0.3.2 pid=45031>
<Host server33: server33-eth0:10.0.3.3 pid=45033>
<Host server34: server34-eth0:10.0.3.4 pid=45035>
<Host server35: server35-eth0:10.0.3.5 pid=45037>
<Host user: user-eth0:10.0.10.1 pid=45039>
<OVSSwitch fw2: lo:127.0.0.1,fw2-eth1:None,fw2-eth2:None,fw2-eth3:None,fw2-eth4:None,fw2-eth5:None,fw2-eth6:None pid=45044>
<OVSSwitch fw4: lo:127.0.0.1,fw4-eth1:None,fw4-eth2:None pid=45047>
<OVSSwitch ngfw1: lo:127.0.0.1,ngfw1-eth1:None,ngfw1-eth2:None,ngfw1-eth3:None,ngfw1-eth4:None,ngfw1-eth5:None,ngfw1-eth6:None pid=45050>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None,s1-eth5:None,s1-eth6:None,s1-eth7:None,s1-eth8:None,s1-eth9:None pid=45053>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None pid=45056>
<OVSSwitch sb1: lo:127.0.0.1,sb1-eth1:None,sb1-eth2:None pid=45059>
<OVSSwitch sb2: lo:127.0.0.1,sb2-eth1:None,sb2-eth2:None pid=45062>
<OVSSwitch sb3: lo:127.0.0.1,sb3-eth1:None,sb3-eth2:None,sb3-eth3:None,sb3-eth4:None,sb3-eth5:None,sb3-eth6:None pid=45065>
<OVSController c0: 127.0.0.1:6653 pid=44992>
```

Figura 7. Interfaces y direcciones IP asignadas a los elementos de la red.

```
mininet> net
branch4 branch4-eth0:fw4-eth1
cloud cloud-eth0:s1-eth1
lb1 lb1-eth0:s1-eth5
lb2 lb2-eth0:s1-eth6
lb3 lb3-eth0:s1-eth7
server11 server11-eth0:ngfw1-eth1
server12 server12-eth0:ngfw1-eth2
server13 server13-eth0:ngfw1-eth3
server14 server14-eth0:ngfw1-eth4
server15 server15-eth0:ngfw1-eth5
server21 server21-eth0:fw2-eth1
server22 server22-eth0:fw2-eth2
server23 server23-eth0:fw2-eth3
server24 server24-eth0:fw2-eth4
server25 server25-eth0:fw2-eth5
server31 server31-eth0:sb3-eth1
server32 server32-eth0:sb3-eth2
server33 server33-eth0:sb3-eth3
server34 server34-eth0:sb3-eth4
server35 server35-eth0:sb3-eth5
user user-eth0:s2-eth1
fw2 lo:  fw2-eth1:server21-eth0 fw2-eth2:server22-eth0 fw2-eth3:server23-eth0 fw2-eth4:server24-eth0 fw2-eth5:server25-eth0 fw2-eth6:sb2-eth1
fw4 lo:  fw4-eth1:branch4-eth0 fw4-eth2:s1-eth8
ngfw1 lo:  ngfw1-eth1:server11-eth0 ngfw1-eth2:server12-eth0 ngfw1-eth3:server13-eth0 ngfw1-eth4:server14-eth0 ngfw1-eth5:server15-eth0 ngfw1-eth6:sb1-eth1
s1 lo:  s1-eth1:cloud-eth0 s1-eth2:sb1-eth2 s1-eth3:sb2-eth2 s1-eth4:sb3-eth6 s1-eth5:lb1-eth0 s1-eth6:lb2-eth0 s1-eth7:lb3-eth0 s1-eth8:fw4-eth2 s1-eth9:s2-eth2
s2 lo:  s2-eth1:user-eth0 s2-eth2:s1-eth9
sb1 lo:  sb1-eth1:ngfw1-eth6 sb1-eth2:s1-eth2
sb2 lo:  sb2-eth1:fw2-eth6 sb2-eth2:s1-eth3
sb3 lo:  sb3-eth1:server31-eth0 sb3-eth2:server32-eth0 sb3-eth3:server33-eth0 sb3-eth4:server34-eth0 sb3-eth5:server35-eth0 sb3-eth6:s1-eth4
c0
```

Figura 8. Interfaces conectadas.

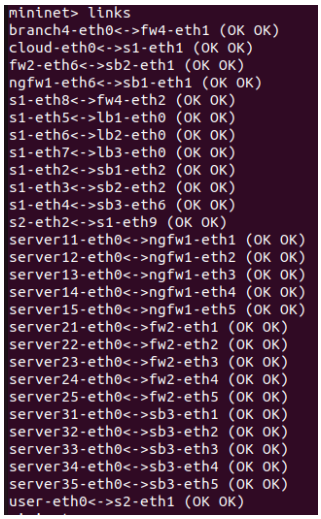


Figura 9. Enlaces generados entre los componentes de la red SDN.

El estado inicial de la red establece la siguiente configuración de los switches de la red. Es notorio destacar que únicamente existen reglas instaladas en los elementos *fw2, fw4* y *ngfw1*, que son los switches que tienen un comportamiento predefinido para filtrar y descartar tráfico. Este comportamiento se muestra en las figuras siguientes.

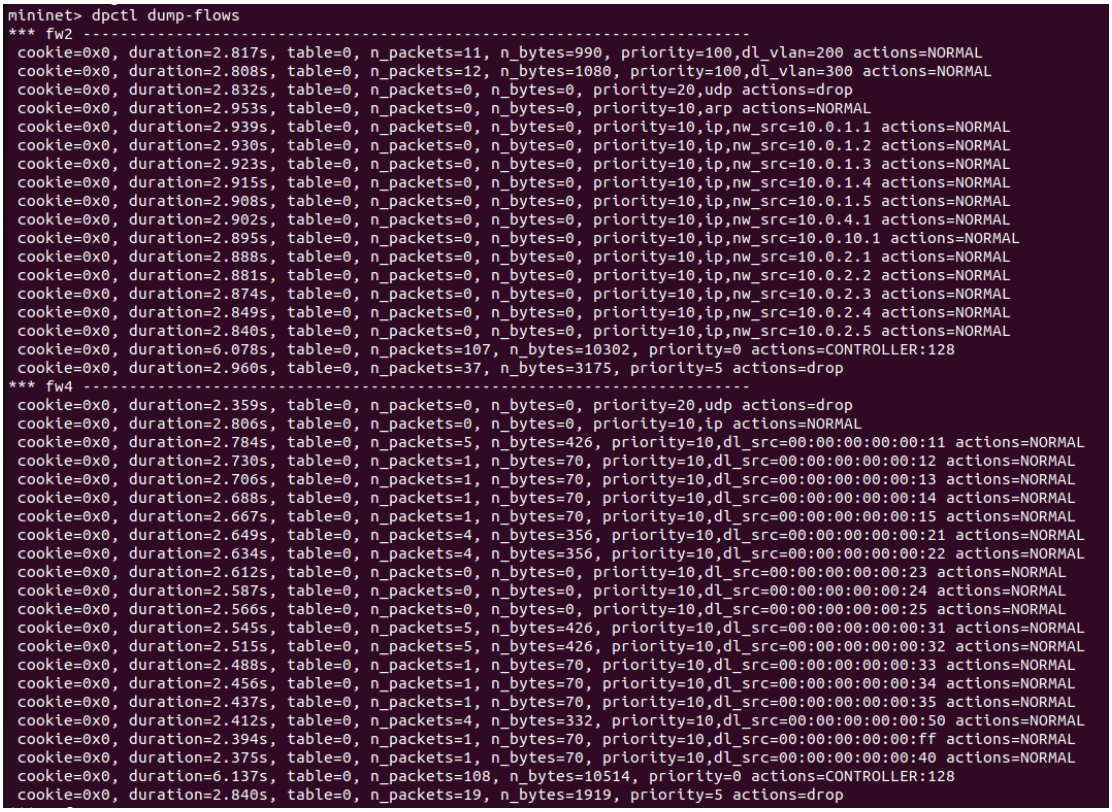


Figura 10. Configuración inicial de los firewalls *fw2* y *fw4*.

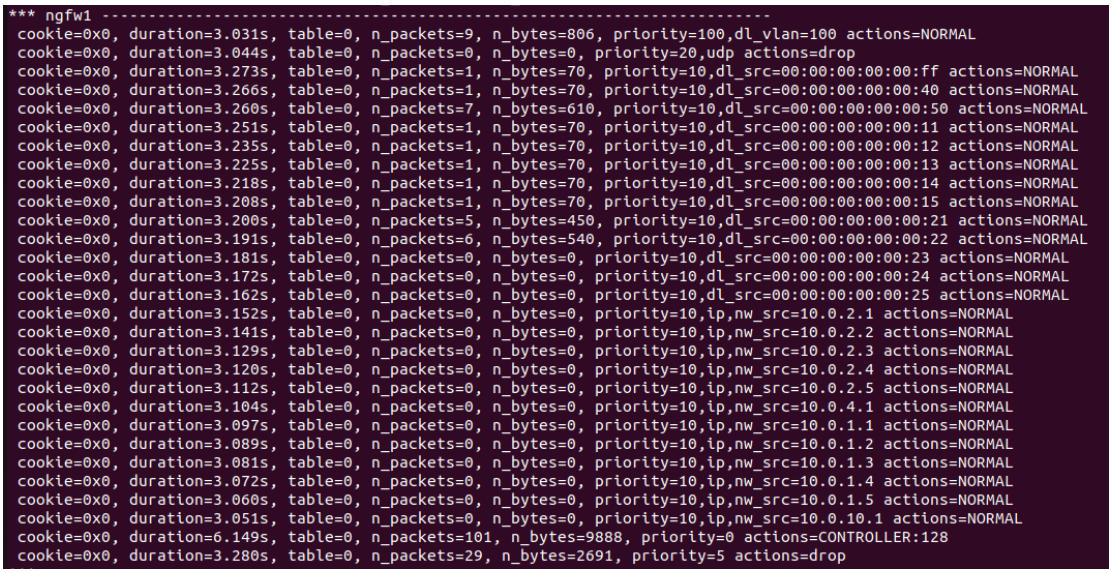


Figura 11. Configuración inicial del firewall *ngfw1*.

El resto de los switches *sb1, sb2, sb3, s1* y *s2* únicamente cuentan con una regla instalada, que es la que determina que todos los paquetes que llegan a ellos deben ser enviados al controlador de la red, para que este determine qué acción debe tomarse sobre cada tipo de tráfico.

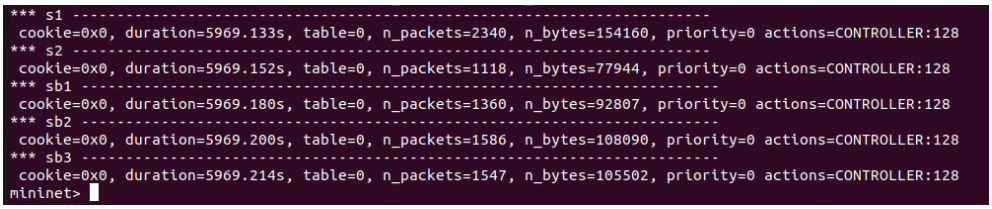


Figura 12. Configuración inicial de los flujos en los switches *s1, s2, sb1, sb2, sb3.*

4.2. Flujos de pruebas técnicas y cálculo del rendimiento de la red

Como se mencionó en el apartado del diseño técnico, una vez configurada la topología de SD-WAN, se ha procedido a realizar las diferentes pruebas que verifican la funcionalidad implementada en la red. Todas las comunicaciones entre sedes pertenecientes a la red interna de la arquitectura de prueba, y entre las sedes y la red pública, pasan por los elementos de la SD-WAN, que es donde se aplican las políticas de seguridad de la red.

El objetivo de estas pruebas es:

- Demostrar que la red es capaz de centralizar las comunicaciones, descartar el tráfico no deseado en la red y aplicar las políticas de seguridad definidas.

- Comprobar el impacto que tiene en el rendimiento de las redes la configuración de elementos de seguridad y control del tráfico.

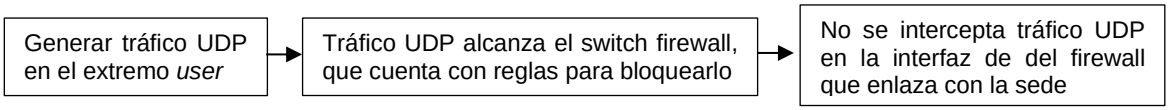
4.2.1. Medición de los indicadores de seguridad de la red

4.2.1.1. Bloqueo del tráfico UDP y HTTP

En esta sección se va a verificar que el tráfico de protocolos no permitidos dentro de la SD-WAN es bloqueado por los diferentes firewalls que hay desplegados en ella. Según la tabla de métricas descrita en el capítulo 3, se deben bloquear el 100% de las conexiones UDP y el 100% de las comunicaciones por el protocolo HTTP.

Bloqueo del tráfico UDP

La prueba seguirá el siguiente diagrama de flujo:



El objetivo es verificar que las reglas de los firewalls de la SD-WAN rechazan adecuadamente el tráfico no permitido en capa 4 de la red. De forma anticipada se puede conocer que el tráfico UDP será bloqueado antes de que llegue a las sedes 1, 2 y 4, quedando las sedes 3 y cloud expuestas al no presentar protección de filtrado de paquetes.

Sedes 1, 2 y 4

```
mininet> user iperf -c 10.0.1.1 -u
------------------------------------------------------------
Client connecting to 10.0.1.1, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 10.0.10.1 port 34660 connected with 10.0.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0153 sec  1.25 MBytes  1.05 Mbits/sec
[  1] Sent 896 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.
mininet>
```

Figura 13. Prueba de tráfico UDP hacia la sede 1.

El componente *ngfw1* bloquea el tráfico UDP que recibe a través de su interfaz e*th6* e impide que se reenvíe por la interfaz *eth1* hacia el *server11.* En la siguiente figura se observa cómo en dicha interfaz no se capturan paquetes UDP.

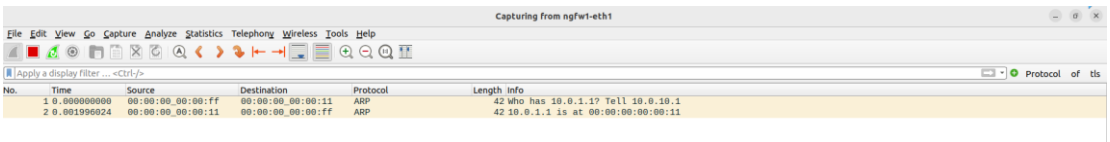


Figura 14. Tráfico capturado en *ngfw1-eth1* durante la prueba.

De igual modo, se verifica que el tráfico UDP es bloqueado antes de que llegue a las sedes 2 y 4 por los componentes *fw2* y *fw4,* respectivamente.

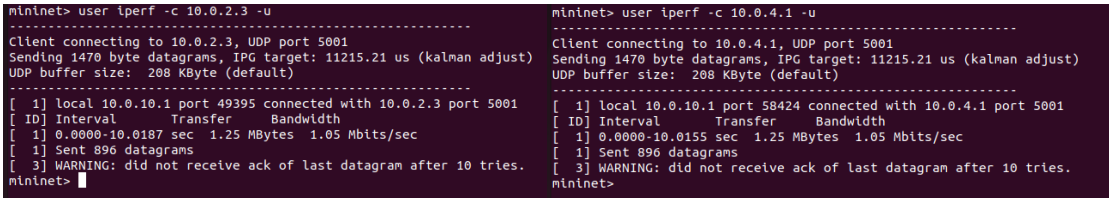


Figura 15. Prueba de tráfico UDP hacia las sedes 2 y 4.

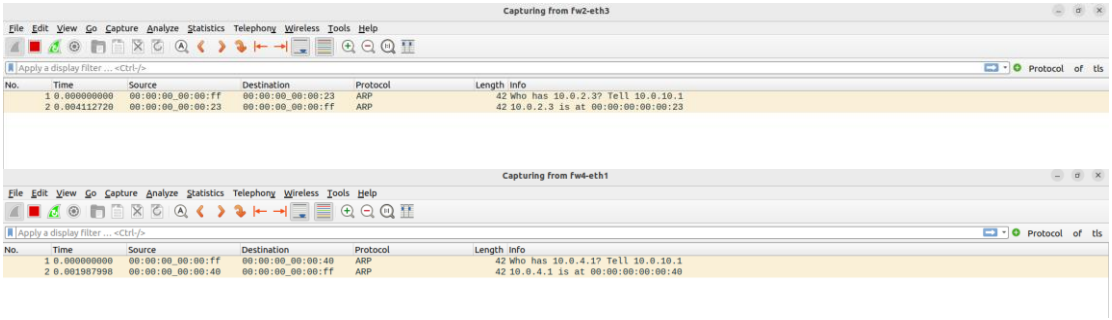


Figura 16. Tráfico capturado en *fw2-eth3* y *fw4-eth1* durante las pruebas.

Sede 3 y sede cloud

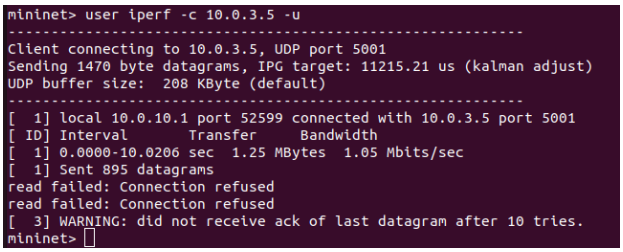


Figura 17. Prueba de tráfico UDP hacia la sede 3.

En el caso de la sede 3, el *host server35* recibe todo el tráfico UDP enviado desde *user* debido a que en este flujo de comunicaciones no existe filtrado de paquetes realizado por ningún firewall. En la siguiente figura se puede apreciar cómo los paquetes UDP llegan al *server35.*

Figura 18. Captura de paquetes UDP en la interfaz *eth0* de *server35*.

Igual sucede en la sede cloud.

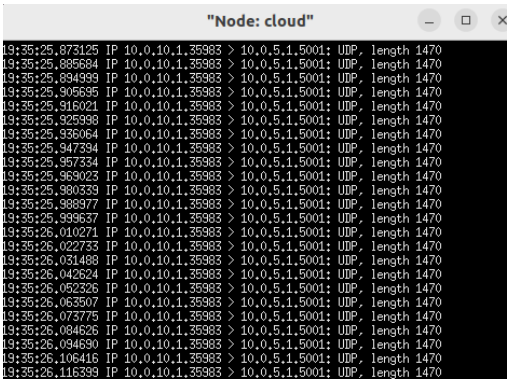
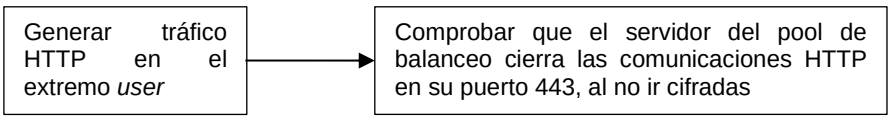


Figura 19. Captura de paquetes UDP en la interfaz *eth0* de *cloud*.

En conclusión, se observa que aquellas sedes que están protegidas mediante la configuración del bloqueo del tráfico UDP en los firewalls de la SD-WAN, no reciben este tipo de tráfico, y presentan el 100% de bloqueo. Sin embargo, las sedes 3 y cloud, al no contar con esta configuración, sus servidores están expuestos a los ataques de denegación de servicio por inundación de paquetes UDP y, por consiguiente, presentan un 0% de bloqueo.

Bloqueo del tráfico HTTP

Para comprobar que los servidores no aceptan conexiones mediante HTTP en su puerto 443 se seguirá el siguiente flujo de pruebas.



En la siguiente figura se observa que el extremo *user* no recibe respuesta a las conexiones HTTP enviadas al servidor.

```
mininet> user curl -L http://10.0.0.1:443/
curl: (56) Recv failure: Connection reset by peer
mininet> xterm lb1
mininet> user curl -L http://10.0.0.1:443/
curl: (56) Recv failure: Connection reset by peer
mininet> user curl -L http://10.0.0.1:443/
curl: (56) Recv failure: Connection reset by peer
mininet>
```

Figura 20. Prueba de petición HTTP hacia el servidor *lb1*.

Cuando el servidor *lb1* recibe la petición GET, detecta que se está empleando el protocolo HTTP en su versión 1.0 en un canal cifrado, por lo que identifica la inconsistencia de la petición y cierra la conexión con el cliente en el *host user* sin enviar respuesta HTTP.

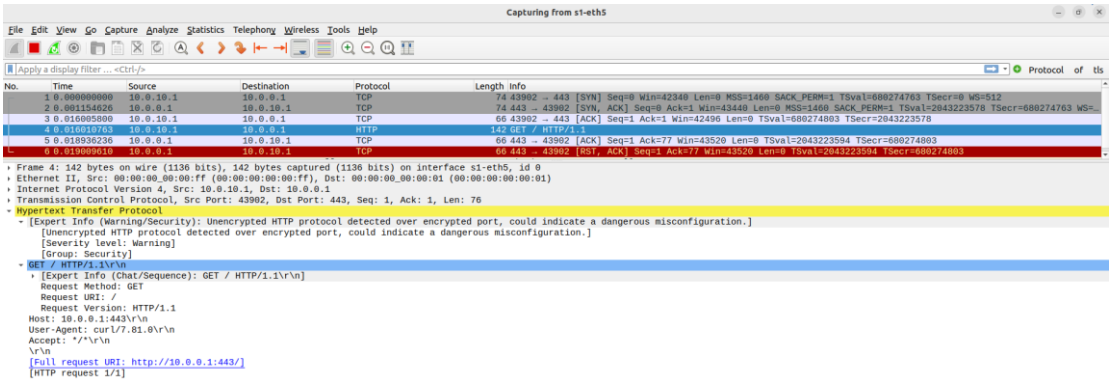


Figura 21. Captura de la petición HTTP GET en *s1-eth5* y cierre de la conexión por parte del servidor *lb1*.

Por tanto, dado que todos los servidores de la red presentan la misma configuración (script *https_server.py*), se concluye con que las peticiones HTTP realizadas sin cifrado del tráfico en tránsito son bloqueadas por los servidores de la red, que están configurados para recibir únicamente tráfico HTTPS. Se alcanza el 100% de bloqueo para el tráfico HTTP de las sedes 1, 2, 3, 4 y cloud.

4.2.1.2. Limitaciones de las comunicaciones no permitidas

Para comprobar la correcta configuración de los firewalls y su grado de alineamiento con las especificaciones técnicas definidas en la matriz de comunicaciones del capítulo 3, se procede a ejecutar la prueba que evidencia las comunicaciones permitidas en la red.

Para comprobar el estado de los flujos de comunicación y los diferentes caminos, se genera tráfico en todos los componentes de la red de tipo *host*, para comprobar cuál es el grado de comunicación y qué conexiones bloquean los firewalls. En la siguiente figura se muestra el resultado obtenido para las comunicaciones de la red.

Figura 22. Comportamiento de la red ante intentos de comunicación.

En la figura anterior, se observa que el 41% del tráfico de la red no ha obtenido respuesta, es decir, ha sido bloqueado por las políticas de filtrado y descarte de paquetes configuradas en los firewalls. Por tanto, se puede concluir con que esta métrica cumple los requisitos indicados en el capítulo 3 de diseño de la red. Además, el comportamiento de la red ante la prueba de bloqueo de las comunicaciones describe que:

- La sede 1 puede comunicarse con las sedes 2, 4, *cloud* y con *user*, pero no con la sede 3.

- La sede 2 puede comunicarse con las sedes 1, 4 y con *user*, y no establece comunicación con las sedes 3 y con *cloud*.

- La sede 3 puede comunicarse con las sedes 4, *cloud* y con *user*, y no puede intercambiar tráfico con las sedes 1 y 2.

- La sede 4 puede establecer comunicación con todas las sedes (1, 2, 3, *cloud*) y con *user*.

- La sede *cloud* establece comunicación con las sedes 1, 3, 4 y con *user*, no puede comunicarse con la sede 2.

- El *host* de la red pública, *user,* puede comunicarse con todos los elementos de la red.

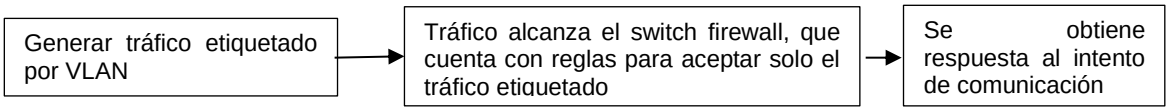
Comparando los resultados obtenidos con la matriz de comunicaciones (mostrada a continuación) diseñada para esta red, se puede observar que los firewalls están configurados correctamente para comportarse como se indica en dicha matriz.

Origen \ Destino	Sede 1	Sede 2	Sede 3	Sede 4	Cloud	User
Sede 1	-	OK	KO	OK	OK	OK
Sede 2	OK	-	KO	OK	KO	OK
Sede 3	KO	KO	-	OK	OK	OK
Sede 4	OK	OK	OK	-	OK	OK
Cloud	OK	KO	OK	OK	-	OK
User	OK	OK	OK	OK	OK	-

Tabla 9. Matriz de comunicaciones de la red.

4.2.1.3. Segmentación de las comunicaciones de la SD-WAN

Para realizar el filtrado de las comunicaciones permitidas entre las diferentes sedes de la arquitectura, también se ha implementado la segmentación de la red basada en VLANs. Para comprobar que se acepta el tráfico etiquetado se ha seguido el siguiente diagrama de flujo en las pruebas.




```
mininet> server11 ping -I eth0.100 192.168.0.31
PING 192.168.0.31 (192.168.0.31) from 192.168.0.11 eth0.100: 56(84) bytes of data.
64 bytes from 192.168.0.31: icmp_seq=1 ttl=64 time=124 ms
64 bytes from 192.168.0.31: icmp_seq=2 ttl=64 time=21.6 ms
64 bytes from 192.168.0.31: icmp_seq=3 ttl=64 time=20.6 ms
64 bytes from 192.168.0.31: icmp_seq=4 ttl=64 time=68.1 ms
^C
--- 192.168.0.31 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 20.592/58.664/124.362/42.510 ms
mininet>
```

Figura 23. Generación de tráfico con VLAN 100.

El tráfico etiquetado con la VLAN es aceptado por el firewall *ngfw1*, que permite que salga hacia el switch s1 para que sea enviado hacia el servidor de la sede 3 destino. Además, el controlador de la red instala flujos en dicho switch s1 para permitir las comunicaciones e indicar a este dispositivo por qué puertos deben salir los diferentes flujos de las comunicaciones, como puede observarse en la siguiente figura.

```
*** s1 ----------------------------------------------------------------------
 cookie=0x0, duration=17.752s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",dl_vlan=100,dl_vlan_pcp=0,dl_src=00:00:00:00:00:31,dl_dst=00:00:00:00:00:11,arp_spa=192.1
68.0.31,arp_tpa=192.168.0.11,arp_op=2 actions=output:"s1-eth2"
 cookie=0x0, duration=12.465s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth4",dl_vlan=100,dl_vlan_pcp=0,dl_src=00:00:00:00:00:31,dl_dst=00:00:00:00:00:11,arp_spa=192.1
68.0.31,arp_tpa=192.168.0.11,arp_op=1 actions=output:"s1-eth2"
 cookie=0x0, duration=12.455s, table=0, n_packets=0, n_bytes=0, idle_timeout=60, priority=1,arp,in_port="s1-eth2",dl_vlan=100,dl_vlan_pcp=0,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:31,arp_spa=192.1
68.0.11,arp_tpa=192.168.0.31,arp_op=2 actions=output:"s1-eth4"
 cookie=0x0, duration=17.729s, table=0, n_packets=3, n_bytes=306, idle_timeout=60, priority=1,icmp,in_port="s1-eth2",dl_vlan=100,dl_vlan_pcp=0,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:31,nw_src=192
.168.0.11,nw_dst=192.168.0.31,nw_tos=0,icmp_type=8,icmp_code=0 actions=output:"s1-eth4"
 cookie=0x0, duration=17.718s, table=0, n_packets=3, n_bytes=306, idle_timeout=60, priority=1,icmp,in_port="s1-eth4",dl_vlan=100,dl_vlan_pcp=0,dl_src=00:00:00:00:00:31,dl_dst=00:00:00:00:00:11,nw_src=192
.168.0.31,nw_dst=192.168.0.11,nw_tos=0,icmp_type=0,icmp_code=0 actions=output:"s1-eth2"
 cookie=0x0, duration=189.649s, table=0, n_packets=319, n_bytes=27933, priority=0 actions=CONTROLLER:128
```

Figura 24. Reglas instaladas en el switch s1 por el controlador.

Se captura el tráfico en la interfaz *ngfw1-eth1*, conectada al *server11*, y se observa que tanto los paquetes ICMP con origen cualquiera de los extremos de la conexión, como los paquetes ARP, son etiquetados con la VLAN100. Estas evidencias se muestran en las siguientes dos figuras.

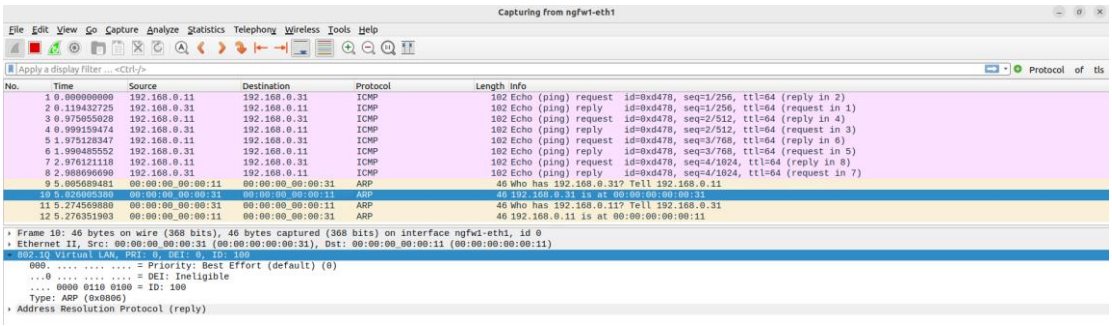


Figura 25. Tráfico ARP con VLAN100.

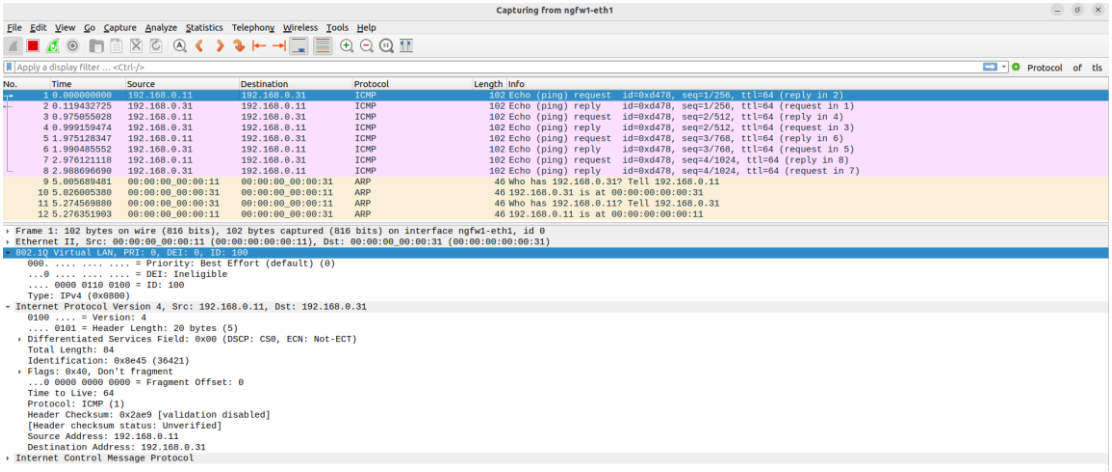


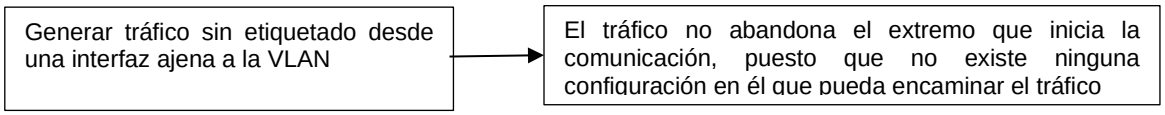
Figura 26. Tráfico ICMP con VLAN100.

Este ejemplo es extensible para la VLAN200 y VLAN300, también configuradas en la red y que se encuentran actualmente en funcionamiento, como se observa en las siguientes figuras.

```
mininet> server32 ping -I eth0.200 192.168.0.21
PING 192.168.0.21 (192.168.0.21) from 192.168.0.32 eth0.200: 56(84) bytes of data.
64 bytes from 192.168.0.21: icmp_seq=1 ttl=64 time=219 ms
64 bytes from 192.168.0.21: icmp_seq=2 ttl=64 time=25.1 ms
64 bytes from 192.168.0.21: icmp_seq=3 ttl=64 time=17.4 ms
64 bytes from 192.168.0.21: icmp_seq=4 ttl=64 time=59.6 ms
^C
--- 192.168.0.21 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 17.358/80.190/218.689/81.526 ms
mininet>
mininet> server22 ping -I eth0.300 192.168.0.51
PING 192.168.0.51 (192.168.0.51) from 192.168.0.22 eth0.300: 56(84) bytes of data.
64 bytes from 192.168.0.51: icmp_seq=1 ttl=64 time=201 ms
64 bytes from 192.168.0.51: icmp_seq=2 ttl=64 time=15.5 ms
64 bytes from 192.168.0.51: icmp_seq=3 ttl=64 time=17.2 ms
64 bytes from 192.168.0.51: icmp_seq=4 ttl=64 time=28.7 ms
^C
--- 192.168.0.51 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 15.531/65.505/200.550/78.132 ms
mininet>
```

Figura 27. Comunicaciones relativas a las VLAN200 y VLAN300.

Un segundo flujo de pruebas se inicia para comprobar qué sucede cuando se intenta establecer una comunicación con una dirección IP destino perteneciente a una interfaz VLAN, sin que el extremo que inicia la comunicación esté incluido en dicha VLAN. Para ello, se ha seguido el siguiente flujo de pruebas.

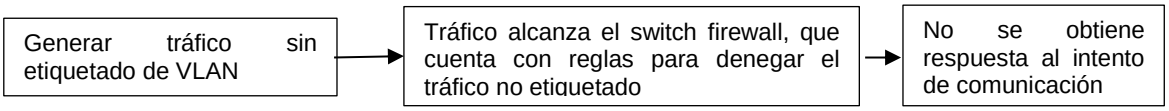



```
mininet> server12 ping 192.168.0.31
ping: connect: Network is unreachable
mininet>
```

Figura 28. No se aceptan comunicaciones hacia *server31*-eth0.100 desde otros servidores de la sede 1.

Se observa que el ping no es exitoso, dado que en ningún momento el tráfico ICMP llega a abandonar el server12 al no estar configurada en él ninguna puerta de enlace que pueda encaminar ese paquete.

Finalmente, se realiza la prueba para comprobar si se aceptan las comunicaciones entre las direcciones IP pertenecientes a una VLAN, pero sin que el tráfico se envíe etiquetado.




```
mininet> server11 ping server31
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data.
From 10.0.1.1 icmp_seq=1 Destination Host Unreachable
From 10.0.1.1 icmp_seq=2 Destination Host Unreachable
From 10.0.1.1 icmp_seq=3 Destination Host Unreachable
From 10.0.1.1 icmp_seq=4 Destination Host Unreachable
From 10.0.1.1 icmp_seq=5 Destination Host Unreachable
From 10.0.1.1 icmp_seq=6 Destination Host Unreachable
^C
--- 10.0.3.1 ping statistics ---
8 packets transmitted, 0 received, +6 errors, 100% packet loss, time 7172ms
pipe 4
mininet>
```

Figura 29. Prueba de comunicación entre *server11* y *server31* sin etiqueta VLAN100.

El tráfico, al no originarse en la interfaz eth0.100 de *server11*, no recibe la etiqueta VLAN100 y, por ende, es descartado por el firewall. En la siguiente figura se observa cómo los paquetes ARP de *server11* no reciben respuesta.

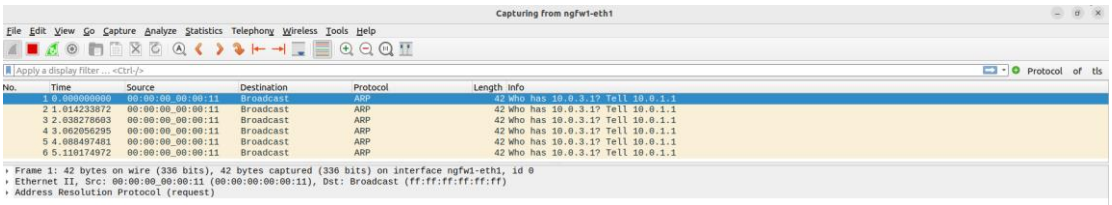


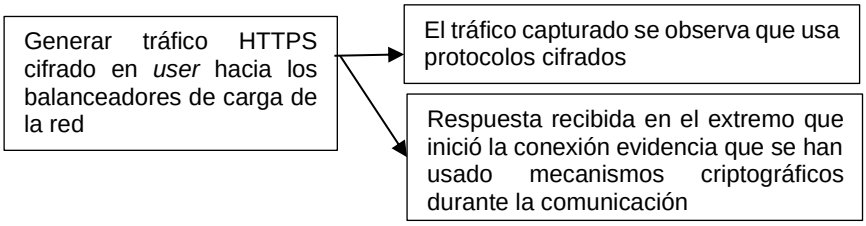
Figura 30. Peticiones ARP en *server11* sin respuesta.

Estos dos últimos flujos de errores son extensibles al resto de componentes de la red, dado que para todos ellos se ha aplicado la misma configuración. Por tanto, únicamente se acepta por parte de los componentes de la SD-WAN el tráfico VLAN de los 6 enlaces configurados. El resto de los 49 enlaces no pueden establecer las comunicaciones pertinentes debido a que son bloqueadas por la red. En conclusión, solamente el 10,9% de las conexiones mediante VLANs son exitosas.

4.2.1.4. Comunicaciones cifradas en la red

Otro de los requisitos establecidos en el capítulo del diseño técnico de la red describe que las comunicaciones de la SD-WAN deben ir cifradas, para preservar la confidencialidad e integridad del tráfico generado en la red. Es por esto por lo que los servidores de la red solo aceptan tráfico HTTPS, cifrado con el certificado que tienen disponible todos ellos.

El flujo de pruebas seguido ha sido el siguiente.




```
mininet> user curl -L -k https://10.0.0.1:443/
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="curl_format.txt">curl_format.txt</a></li>
<li><a href="lb.py">lb.py</a></li>
<li><a href="lb_round_robin.py">lb_round_robin.py</a></li>
<li><a href="lb_weighted_round_robin.py">lb_weighted_round_robin.py</a></li>
<li><a href="out.txt">out.txt</a></li>
<li><a href="topo.py">topo.py</a></li>
</ul>
<hr>
</body>
</html>
```

Figura 31. Petición HTTPS de *user* hacia la sede 1.

```
mininet> user curl -L -k https://10.0.0.2:443/
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="curl_format.txt">curl_format.txt</a></li>
<li><a href="lb.py">lb.py</a></li>
<li><a href="lb_round_robin.py">lb_round_robin.py</a></li>
<li><a href="lb_weighted_round_robin.py">lb_weighted_round_robin.py</a></li>
<li><a href="out.txt">out.txt</a></li>
<li><a href="topo.py">topo.py</a></li>
</ul>
<hr>
</body>
</html>
```

Figura 32. Petición HTTPS de *user* hacia la sede 2.

```
mininet> user curl -L -k https://10.0.0.3:443/
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="curl_format.txt">curl_format.txt</a></li>
<li><a href="lb.py">lb.py</a></li>
<li><a href="lb_round_robin.py">lb_round_robin.py</a></li>
<li><a href="lb_weighted_round_robin.py">lb_weighted_round_robin.py</a></li>
<li><a href="out.txt">out.txt</a></li>
<li><a href="topo.py">topo.py</a></li>
</ul>
<hr>
</body>
</html>
```

Figura 33. Petición HTTPS de *user* hacia la sede 3.

Se puede observar añadiendo la opción '-v' al comando *curl* que todas las comunicaciones van cifradas en tránsito. El servidor balanceador de carga devuelve mediante una comunicación TLS 1.3 a *user* la localización de la redirección dónde deberá redirigir su petición. Además, la comunicación entre el servidor de las sedes que responde las peticiones de *user* también se realiza de con cifrado en tránsito. Una vez que la respuesta HTTPS es recibida por *user*, su contenido es descifrado y mostrado al usuario, tal y como se puede observar en las figuras anteriores.

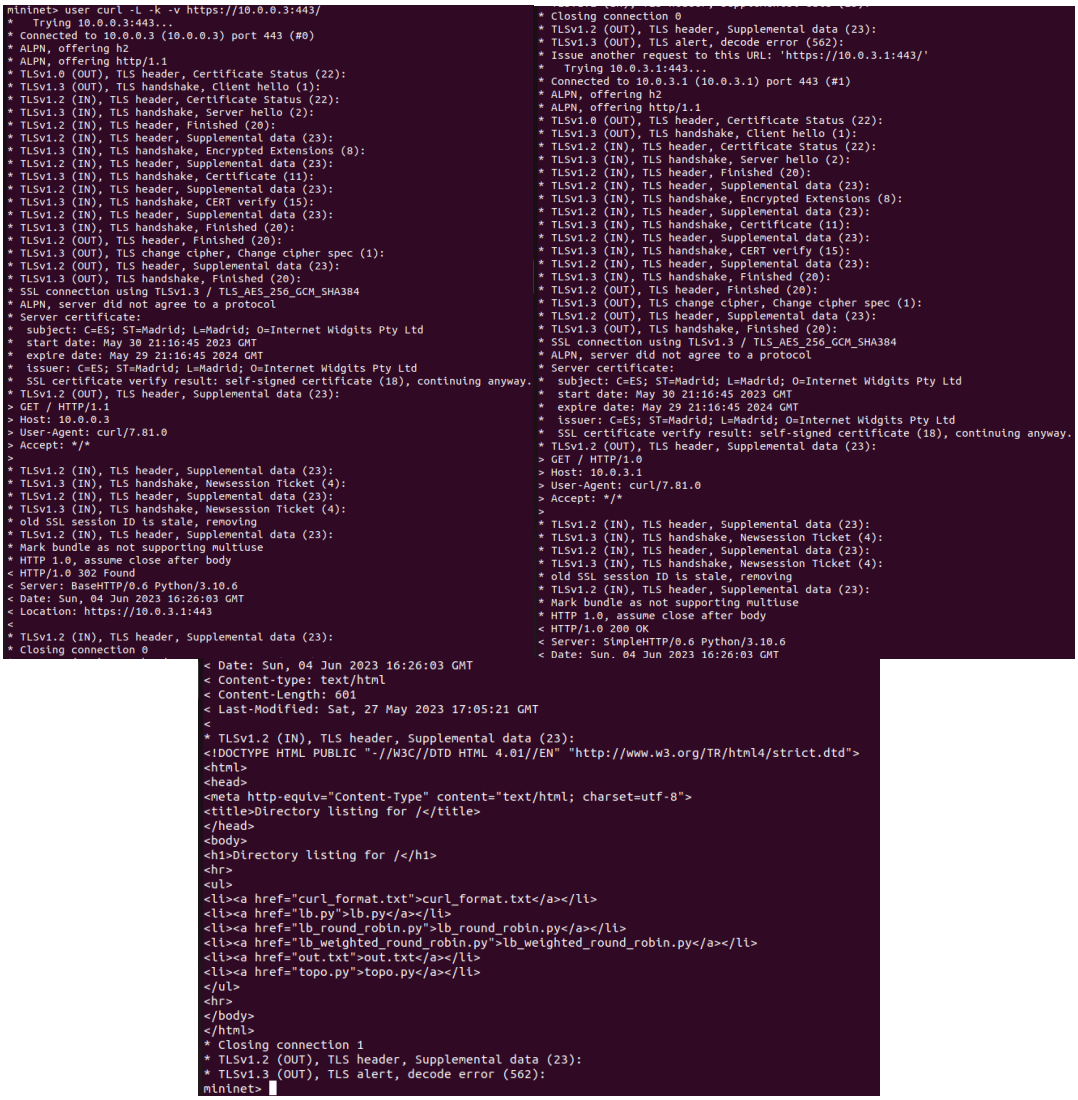


Figura 34. Contenido de las peticiones HTTPS.

Además, en la siguiente figura se comprueba que el tráfico capturado en la interfaz *s1-eth9* del switch s1 (encargado de recibir y reenviar todas las peticiones de la SD-WAN), es parte de comunicaciones cifradas.

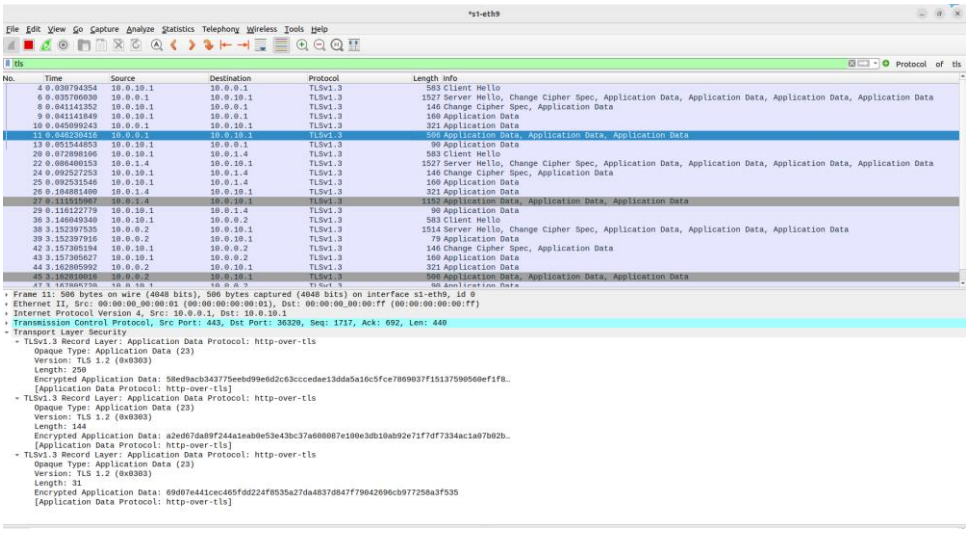


Figura 35. Tráfico TLS 1.3 capturado en *s1-eth9*.

Por consiguiente, por medio de esta prueba se comprueba que, como ya se constató en la prueba 4.2.1 de bloqueo de tráfico HTTP, las comunicaciones de la capa de aplicación que tienen lugar en la red se realizan con el cifrado de los paquetes.

4.2.2. Medición del rendimiento de la red

Medición del tiempo de respuesta

La lógica de la topología de red sería la siguiente, para comprobar que la sede 1 quiere conectarse con la sede 4 de la organización, se deberían llevar a cabo los siguientes pasos:

1. El servidor de la sede 1 envía al balanceador de carga 1 (*lb1*) una petición HTTPS con destino la dirección IP de este balanceador de carga, mediante el siguiente comando.
2. Cuando el balanceador de carga 1 recibe esta petición, aplica el algoritmo de selección aleatoria de balanceo de carga, para seleccionar un servidor del *pool* de servidores para que procese la petición. Seguidamente, envía una respuesta HTTPS con código 302 a *user*, en la que le indica en el campo *Location* la dirección IP del servidor del *pool* de servidores de la sede 1 al que debe redirigirse.
3. Con la dirección de redirección HTTPS, *user* envía nuevamente la petición hacia el servidor del *pool* de la sede 1 que le haya indicado el servidor lb1 de balanceo de carga.
4. Todo el tráfico con destino algún servidor del *pool* de servidores será procesado y analizado por el firewall *ngfw1*, que se encargará de realizar el filtrado del tráfico para estimar si el tráfico es legítimo
5. A continuación, cuando el servidor de la sede 1 recibe la petición HTTPS, la procesará y enviará su respuesta hacia *user*.

Para realizar las pruebas de medición del tiempo de respuesta de las sedes se ejecuta cinco veces la petición indicada previamente con el comando *curl* para el tráfico de cada sede, obteniéndose los tiempos de respuesta indicados en la siguiente tabla.

Destino	Pruebas					Mínimo	Máximo	Media
	1	2	3	4	5			
Sede 1	0,098	0,104	0,105	0,118	0,121	0,098	0,121	0,1092
Sede 2	0,108	0,12	0,116	0,108	0,114	0,108	0,12	0,1132
Sede 3	0,136	0,118	0,131	0,094	0,1	0,094	0,136	0,1158
Sede 4	0,063	0,051	0,066	0,065	0,07	0,051	0,07	0,063
Cloud	0,062	0,042	0,067	0,05	0,084	0,042	0,084	0,061

Tabla 10. Tiempo de respuesta en segundos de las distintas sedes.

La siguiente gráfica representa los tiempos de respuesta indicados en la tabla anterior.

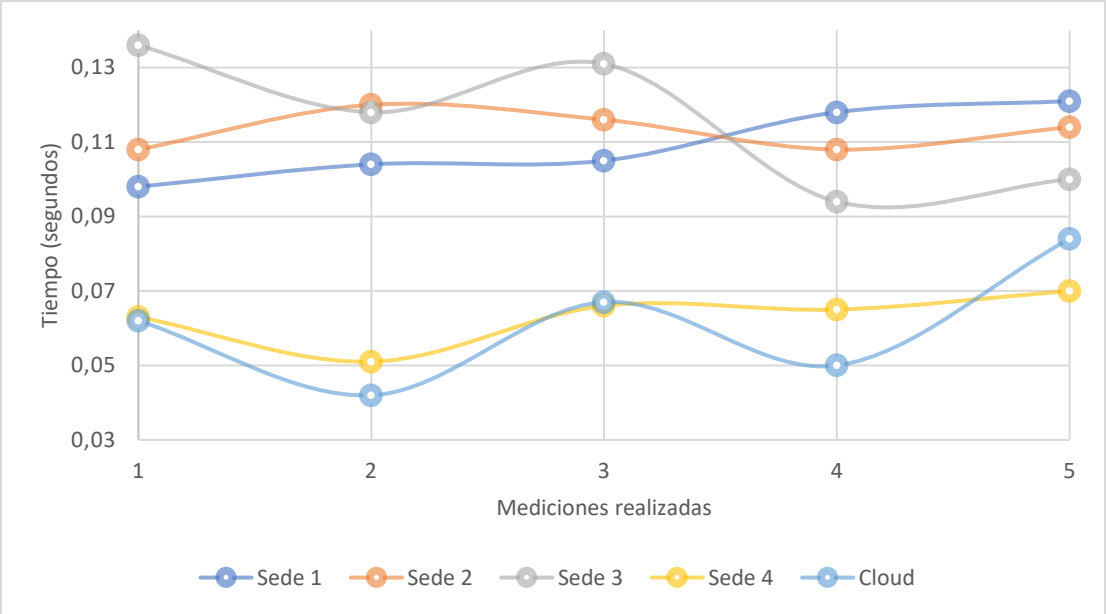


Figura 36. Representación de los tiempos de respuesta de los flujos de comunicación.

Como se puede observar, los flujos de comunicación que presentan menor tempo de respuesta son los que van dirigidos desde *user* hacia la sede 4 y la sede cloud. Por otro lado, el flujo que se dirige desde *user* hacia la sede 3 es el que peor tiempo de respuesta tiene, mínimamente superior al tiempo que se observa desde *user* hacia la sede 2. Finalmente, el flujo de comunicaciones desde *user* hacia la sede 1, se encuentra en una posición intermedia.

Por lo tanto, la mejor opción es seleccionar caminos que solo dispongan de firewalls, en lugar de balanceadores de carga y firewalls. El motivo que da respuesta a este comportamiento es el hecho de que la aplicación de los algoritmos de balanceo de carga introduce retardos en las redes, debido a que implica seleccionar al servidor destino dentro del *pool* además de redireccionar las comunicaciones hacia este, y es un proceso que

conlleva algo de tiempo. Aun así, el flujo de tráfico que mejores resultados ha obtenido de manera general en las mediciones ha sido el que no presenta medidas de seguridad, ya que no se emplea tiempo en la inspección del tráfico ni en garantizar la disponibilidad de la red.

Parece lógico imaginar que, cuanta más funcionalidades se incluyan en la red para interceptar los paquetes, mayor será el retraso que se introduzca en tiempos de respuesta y latencia. Durante los últimos años, se ha introducido en el estudio de los nuevos sistemas de comunicación y tecnologías la variable del tiempo de respuesta, como, por ejemplo, en el caso de la tecnología 6G, donde se prevé que se incremente el nivel de seguridad y se asegure un tiempo de respuesta por debajo de 1 ms, para poder ofrecer la máxima calidad de servicio a comunicaciones de redes vehiculares o médicas. [17, pág. 70 – 71]

Medición del ancho de banda disponible

Para comprobar el ancho de banda disponible para las comunicaciones entre los distintos flujos de comunicación existentes, se utilizará el siguiente flujo de pruebas:

1. Generar solicitudes de descarga de peticiones HTTPS desde el extremo *user* hacia las sedes 1, 2, 3, 4 y cloud, para comprobar cómo afectan las configuraciones realizadas al ancho de banda disponible en cada uno de los caminos. Se ha hecho uso del siguiente comando:

   ```
   wget --no-check-certificate https://<dir_IP>:443/
   ```

2. Repetición de las pruebas en 5 ocasiones y registro de las medidas devueltas.

En la siguiente tabla se recogen los resultados obtenidos para las distintas mediciones de ancho de banda realizadas.

Destino	Pruebas					Mínimo	Máximo	Media
	1	2	3	4	5			
Sede 1	42,9	56,1	26,8	39,2	53,4	26,8	56,1	43,68
Sede 2	36,3	34,6	52,7	51	36,6	34,6	52,7	42,24
Sede 3	25,3	38,4	25,8	7,33	48,9	7,33	48,9	29,146
Sede 4	4,02	12,2	16,8	9,89	5,04	4,02	16,8	9,59
Cloud	15,4	9,33	8,2	14,9	34,3	8,2	34,3	16,426

Tabla 11. Ancho de banda disponible en los flujos de las distintas sedes.

También en este caso se han representado en la figura siguiente los valores del ancho de banda obtenido en las distintas mediciones.

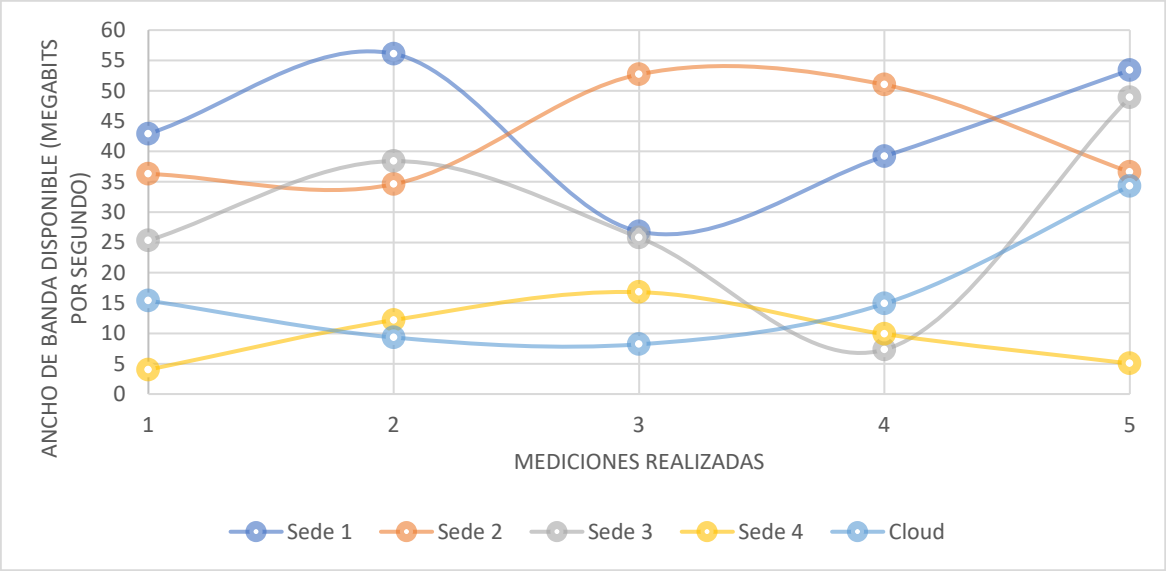


Figura 37. Representación del ancho de banda disponible.

Los resultados obtenidos para las pruebas del ancho de banda arrojan que aquellas sedes que cuentan con balanceo de carga disponen de un mayor ancho de banda que las que no tienen esta configuración. Es decir, las sedes que peores valores en Megabits por segundo son la sede 4 y la sede cloud, y las que obtienen un mejor ancho de banda son la sede 1, la sede 2 y la sede 3.

Aunque hay varios cambios de tendencia en lo que respecta a los resultados obtenidos por las sedes 1, 2 y 3, se pueden ordenar en función al valor medio. De este modo, es la sede 1 la que presenta un mejor resultado, seguida por la sede 2 y, por debajo de esta, la sede 3.

Se observa que el camino que tiene balanceo de carga y firewall de próxima generación es el que presenta un mayor ancho de banda disponible, seguido muy de cerca por el camino con balanceo de carga y firewall en varias capas. Estas dos opciones están bastante alejadas de la opción que presenta solamente balanceo de carga, de la opción sin medidas de seguridad y de la que presenta únicamente un firewall. El motivo de este comportamiento es que, a pesar de que el tiempo de respuesta no sea el mejor de la red, las configuraciones que controlan el tráfico permitido ayudan a mantener los enlaces de la red limpios de tráfico innecesario, como por ejemplo el que va dirigido a la dirección de *broadcast* desde orígenes no permitidos. Respecto al balanceo de carga, por los resultados obtenidos los algoritmos simples, como el algoritmo aleatorio implementado para la sede 1, son más eficientes que los algoritmos dinámicos. Sin embargo, en aquellos casos en los que sea necesario realizar un seguimiento de la carga de los servidores habrá que sacrificar el ancho de banda de la red frente al agotamiento de los recursos.

Finalmente, se puede concluir con que la mejora del ancho de banda en el flujo de la red securizado (desde *user* hacia la sede 1) ha sido superior al 10% en los valores medios, con 43,68 Mbps disponibles, en comparación con los 16,43 Mbps obtenidos en la opción de comunicación desprovista de medidas de seguridad (desde *user* hacia la sede cloud).

4.2.3. Resumen de las mediciones y cálculo del riesgo de ataque de la red

A continuación, se adjunta una tabla resumen con las métricas obtenidas en los anteriores apartados.

Métrica	Umbral		Resultado
Ancho de banda máximo	Mayor a 50 Mbps		Las sedes 1 y 2 son las que superan el umbral
Tiempo de respuesta	Menor a 0,15 segundos		Todas las sedes cumplen el umbral
Porcentaje de bloqueo	Bloqueo de tráfico no deseado en la red	40% de bloqueo	Se ha alcanzado el 41% de bloqueo
	Comunicaciones por VLANs	89% de bloqueo	Se ha obtenido el 89,1% de fallo y el 10,9% de éxito
	Comunicaciones UDP	100% de bloqueo	- Sedes 1, 2, 4: 100% - Sedes 3 y cloud: 0%
	Comunicaciones HTTP	100% de bloqueo	Todas las sedes cuentan con el 100% de bloqueo
	Comunicaciones no cifradas	100% en la capa de aplicación	Todos los caminos de la red cumplen el umbral

Tabla 12. Resumen del resultado de las métricas.

Adicionalmente, para conocer el grado de vulnerabilidad presente en los distintos caminos de la red, se ha realizado el cálculo de la complejidad de ataque a la red.

1. Número de caminos de ataque (NAP): existen cinco caminos (indicados en la tabla 7) presentes en la red, a través de los cuales los atacantes externos pueden alcanzar los recursos internos de las sedes de la organización.

2. Esfuerzo medio de los caminos de ataque (ALAP): para el cálculo de la dificultad de ataque, se tendrán en cuenta la suma de las ponderaciones de la siguiente tabla.

Componente de seguridad de la SD-WAN	Puntuación de seguridad
Firewall de capa 2	2 puntos
Firewall de capa 3	2 puntos
Firewall de capa 4	2 puntos
Bloqueo de UDP	1 punto
Bloqueo de HTTP	1 punto
Uso de comunicaciones cifradas	1 punto
Balanceo de carga	0,5 puntos

Tabla 13. Complejidad de ataque a la red ponderada.

Con base en la tabla anterior, se obtienen los siguientes resultados de complejidad de ataque a los caminos de la red.

Origen – destino	Protección	Puntuación
User – Sede 1	Balanceo de carga, firewall de capas 2, 3, 4, cifrado de comunicaciones y bloqueo UDP, HTTP	10,5
User – Sede 2	Balanceo de carga, firewall de capas 3, 4, cifrado de comunicaciones y bloqueo UDP, HTTP	8,5
User – Sede 3	Balanceo de carga, cifrado de comunicaciones y bloqueo HTTP	2,5
User – Sede 4	Firewall de capas 2, 4, cifrado de comunicaciones y bloqueo UDP, HTTP	7
User – Sede cloud	Cifrado de comunicaciones y bloqueo HTTP	2

Tabla 14. Ponderación de la complejidad de vulnerar los caminos de la red.

La suma del total de puntos obtenidos entre las diferentes sedes, dividida entre el número de caminos de la red, indica la complejidad de ataque de esta. En este caso, se ha obtenido una complejidad de 6,1 puntos por cada camino de la red.

3. Camino más corto de ataque (SAP): el camino con mejor puntuación de complejidad de ataque y, por ende, el más desprotegido, es el camino entre *user* y la sede cloud, que cuenta con medidas de seguridad mínimas.

A continuación, se resume en la siguiente tabla los indicadores de riesgos obtenidos.

Indicador	Valor
NAP	5 caminos de ataque existen en la red
ALAP	6,1 puntos de dificultad por cada camino de la red
SAP	2 puntos, el camino entre *user* y la sede cloud

Tabla 15. Resumen de los indicadores de riesgos de la red.

De la tabla anterior se extrae que, aquellas sedes que cuentan con un mayor despliegue de componentes de la SD-WAN son las que resultan más complejas de vulnerar a los atacantes. Consecuentemente, el riesgo de que un atacante logre alcanzar los recursos internos de una organización está directamente relacionado con el grado de protección con el que cuentan sus caminos. En este caso, la sede cloud de la empresa está copando el primer puesto en grado de riesgo de ataque y vulnerabilidad, y no es de extrañar que sus métricas de seguridad abordadas en la primera parte de este aparado (bloqueo de tráfico UDP, HTTP, uso de firewall, entre otros) fueran las peores de todas las sedes.

En consonancia, la sede 1 es la que mejores resultados de los indicadores de seguridad ha obtenido, y también es la que presenta un mejor riesgo a sufrir ataques, dado que su infraestructura está bien protegida por varios elementos de seguridad de la SD-WAN.

Por tanto, se ha podido comprobar que una buena configuración de las políticas y componentes de la SD-WAN influye directamente en la mejora de la protección de los recursos y en la disminución del riesgo a sufrir ataques exitosos.

5. Conclusiones y trabajos futuros

Durante la realización de este proyecto, se ha configurado la zona SD-WAN de la red de una organización ficticia que contaba con varias sedes *on-premise* y cloud. En el transcurso de las pruebas, se ha verificado que los requisitos de seguridad que garantizan un adecuado comportamiento de los elementos de la red frente a ataques internos o externos están correctamente implementados, mediante la obtención del porcentaje de bloqueo de las comunicaciones que incumplían las políticas de seguridad definidas y aplicadas de manera centralizada en la SD-WAN. Se ha comprobado que las comunicaciones que usan protocolos cifrados en la capa de aplicación y el uso de certificados que autentiquen a los extremos, aseguran la confidencialidad e integridad de los flujos de tráfico de la red. Por otro lado, los firewalls realizan una adecuada segmentación de la red con reglas que filtran el tráfico en varias capas, basadas en protocolos permitidos, direcciones IP origen y destino, VLANs y direcciones MAC, con política de descarte de tráfico configurada por defecto. Por último, los balanceadores de carga que distribuyen las peticiones a través de los servidores ubicados en los *pools* de las DMZs de las diferentes sedes, aseguran la disponibilidad de los recursos de la red.

En lo que respecta a los resultados obtenidos respecto al tiempo de respuesta y al ancho de banda de la red, estos han sido los que se esperaban en el contexto de la red diseñada para este proyecto. Se ha observado que la comparación de tiempos de respuesta evidencia que las soluciones de seguridad introducen retrasos en las comunicaciones. Sin embargo, las investigaciones de las nuevas tecnologías están centrándose en desarrollar alternativas con latencias y tiempos de respuesta reducidos. Durante las pruebas llevadas a cabo, se ha evidenciado que sigue siendo necesario continuar con esta línea de investigación para evitar repercusiones en la latencia de las comunicaciones.

Por su parte, el ancho de banda obtenido en la red securizada es más de un 10% mayor al valor de ancho de banda de la zona de la red sin medidas de seguridad. Además, el camino que mejores resultados ha obtenido en esta prueba es el que más componentes de la SD-WAN y más políticas de seguridad tenía aplicadas. Parece que el motivo es la eliminación de tráfico no deseado en los enlaces de la SD-WAN, que deja disponible una mayor cantidad de recursos para las comunicaciones legítimas.

No obstante, hay que indicar que estos cálculos experimentales se han realizado con datos obtenidos en un entorno de pruebas con recursos limitados y bajo volumen de tráfico, por lo que es posible que el comportamiento de las soluciones SD-WAN en redes reales sea distinto y dependa de otros factores adicionales.

Finalmente, los resultados de riesgos de ataque a las distintas zonas protegidas por la SD-WAN evidencian que, cuantos más componentes de seguridad y políticas se configuren para proteger los recursos internos, menor será el riesgo de sufrir ataques exitosos. En relación con ello, aunque introducir componentes que protejan la red no evita completamente que se produzcan ataques, sí que dificulta que los atacantes puedan acceder a los activos de la empresa, puesto que deben contar con los conocimientos necesarios para evadir las diferentes medidas de seguridad implantadas, que normalmente son varias y basadas en diferentes tecnologías.

Todos los objetivos técnicos marcados al inicio de este trabajo han sido alcanzados: las zonas de la red se han segmentado siguiendo configuraciones de VLAN, reglas en los firewalls y separación de la infraestructura interna y externa de la arquitectura diseñada. También se ha implementado el control de los recursos disponibles y los flujos de tráfico, por medio de balanceadores de carga y firewalls. Los tiempos de respuesta de todas las opciones de comunicación que involucran la SD-WAN han sido comparados y se han extraído las conclusiones que explican los resultados obtenidos. Finalmente, ha sido posible verificar que las medidas de seguridad implementadas introducen mejoras de ancho de banda dentro de la red.

Durante la realización del trabajo, se ha conseguido seguir la planificación marcada. Hubo un desviamiento del plan establecido inicialmente durante el transcurso de la segunda fase y antes de la entrega del hito 2, debido a que, por motivos tecnológicos, los recursos hardware y software no permitían llevar a cabo correctamente la simulación de la red en máquina virtual. Una vez este problema fue subsanado, se consiguió recuperar el ritmo de trabajo y no se han producido más obstáculos reseñables.

En cuanto a la metodología, no se han detectado alteraciones respecto a lo definido incialmente. El único cambio introducido fue el cambio de dispositivo portátil para realizar las pruebas técnicas, lo que supuso un bloqueo durante poco menos de dos semanas. Sin embargo, durante este tiempo se continuó definiendo otros aspectos del trabajo para garantizar el grado de avance del documento. Una vez se materializó el cambio de portátil, fue posible continuar con la configuración técnica y las pruebas.

Respecto a los impactos del trabajo, se puede considerar que todos los impactos positivos previstos se han logrado. Estos hacían referencia a la protección de los datos de personas e instituciones y a la mejora de la eficiencia de la red. El cifrado de las comunicaciones y el uso de controles y componentes de seguridad dentro de la SD-WAN han demostrado que se garantiza la confidencialidad e integridad de la información transmitida y que se consigue mejorar el rendimiento de la red, lo que afecta directamente al menor consumo de recursos y a la sostenibilidad de la solución diseñada.

Finalmente, a lo largo de este trabajo se han identificado futuras necesidades que se indican a continuación:

- Introducir otros elementos de seguridad a la SD-WAN diseñada, tales como VPNs, IDS/IPS, elementos de control de acceso NAC.

- Durante el apartado 2.5.2 del trabajo se indicaron las principales amenazas que sufren las redes SDN en los planos de control, aplicación y datos, y se propusieron contramedidas para mitigarlas. Dado que estos aspectos no se han abordado en detalle con el desarrollo de este trabajo, sería interesarte en un futuro estudiarlos en otros proyectos.

Estas dos necesidades podrían aprovecharse para realizar otros proyectos académicos que avancen con el estudio de las soluciones definidas por software y su aplicación en la seguridad de las infraestructuras tecnológicas de las organizaciones.

6. Glosario

ACL – Lista de control de acceso, por sus siglas en inglés

ALAP – Esfuerzo medio de los caminos de ataque, por sus siglas en inglés

CIA – Confidencialidad, integridad, disponibilidad, por sus siglas en inglés

CLI – *Command Line Interface*

DMZ – Zona desmilitarizada, por sus siglas en inglés

IP – *Internet Protocol*

IPS – *Intrusion Prevention System*

IoT – Internet de las cosas

IT – Infraestructura tecnológica

MPLS – *Multiprotocol Label Switching*

NAP – Número de caminos de ataque, por sus siglas en inglés

NGN – Redes de nueva generación, por sus siglas en inglés

QoS – Calidad de servicio, por sus siglas en inglés

SaaS – *Software as a Service*

SDN – Redes definidas por software, por sus siglas en inglés

SD-WAN – Red de área amplia definida por software, por sus siglas en inglés

SAP – Camino más corto de ataque, por sus siglas en inglés

SLA – Nivel de acuerdo de servicio, por sus siglas en inglés

TCP – *Transport Control Protocol*

UDP – *User Datagram Protocol*

VoIP – *Voice over IP*

VPN – Red privada virtual, por sus siglas en inglés

WAN – Red de área amplia, por sus siglas en inglés

7. Bibliografía

1. ATKINSON, M., KRESS, M., (2023) 'Resource allocation in two-layered cyber-defense', WILEY.
2. BUSTAMANTE, J., AVILA-PENSATEZ, D., (2021) 'Comparative analysis of Cybersecurity mechanisms in SD-WAN architectures: A preliminary results', *Engineering International Research Conference (EIRCON)*.
3. CISCO, 'CCNA 200-301: Official Cert Guide', Volumen 2.
4. CISCO, 'SD-WAN - Software-Defined WAN – Cisco', disponible en: https://www.cisco.com/c/en/us/solutions/enterprise-networks/sd-wan/index.html, accedido el 15/04/2023.
5. HAQUE, I. T., ABU-GHAZALEH, N., (2016) 'Wireless Software Defined Networking: A Survey and Taxonomy', *IEEE Communications Surveys Tutorials 18.4*.
6. MADLER, D., (2015) 'Introduction to SDN', disponible en https://www.youtube.com/watch?v=DiChnu_PAzA, accedido el 10/04/2023
7. MALIL, A. A., ASAD, M., AZEEM, W., (2022) 'Frauds in Banking and Entrepreneurs by Electronic Devices and Combating Using Software and Employment of Demilitrized Zone in the Networks', *Electronic Crime Investigation*, disponible en: http://ijeci.lgu.edu.pk/index.php/ijeci/article/view/118
8. Mininet, Página de descarga de la máquina virtual, disponible en http://mininet.org/download/, accedido el 15/04/2023
9. O'BRIAN, D., (2017), 'Software Defined Networks', *TEL3214 - Computer Communication Networks*.
10. Open Networking Foundation, 'OpenFlow Switch Specification Version 1.0.0'.
11. Open Networking Foundation, (2012) 'OpenFlow Switch Specification Version 1.3.0'.
12. PAMPLIN, S., (2021) 'SD-WAN revolutionises IoT and edge security', *Network Security*.
13. RAHMAN, M., IQBAL, S., GAO, J., (2014) 'Load Balancer as a Service in Cloud Computing, *International Symposium on Service Oriented System Engineering*.
14. SEGEC, P., MORAVCIK, M., URAMOVA, J., PAPAN, J., 'SD-WAN – architecture, functions and benefits', *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)*.
15. VALDIVIESO, A. L., (2014) 'SDN: Evolution and Opportunities in the Development IoT Applications'. *International Journal of Distributed Sensor Networks*.
16. ZHAOGANG SHU, JIAFU WAN, DI LI, JIAXIANG LIN, ATHANASIOS V. VASILAKOS, MUHAMMAD IMRAN, (2016) 'Security in Software-Defined Networking: Threats and Countermeasures', *Springer Science+Business Media*.
17. PING YANG, YUE XIAO, MING XIAO, SHAOQIAN LI, (2019) '6G Wireless Communications: Vision and Potential Techniques', *IEEE Network*.
18. Web: Metrics of Security, disponible en: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=917850, accedido el 14/06/2023.

8. Anexos

8.1. Script de Mininet – topo.py

```
#!/usr/bin/python

from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.net import Mininet
from mininet.topo import Topo
from mininet.link import TCLink, Intf
from mininet.node import OVSController, Host, Node

class MyTopo(Topo):

    def build(self):

            ### PARTE 1 ###

        # Create pool servers in branch 1
        server11 = self.addHost('server11', ip='10.0.1.1/16',
mac='00:00:00:00:00:11')
        server12 = self.addHost('server12', ip='10.0.1.2/16',
mac='00:00:00:00:00:12')
        server13 = self.addHost('server13', ip='10.0.1.3/16',
mac='00:00:00:00:00:13')
        server14 = self.addHost('server14', ip='10.0.1.4/16',
mac='00:00:00:00:00:14')
        server15 = self.addHost('server15', ip='10.0.1.5/16',
mac='00:00:00:00:00:15')

        # Create pool servers in branch 2
        server21 = self.addHost('server21', ip='10.0.2.1/16',
mac='00:00:00:00:00:21')
        server22 = self.addHost('server22', ip='10.0.2.2/16',
mac='00:00:00:00:00:22')
        server23 = self.addHost('server23', ip='10.0.2.3/16',
mac='00:00:00:00:00:23')
        server24 = self.addHost('server24', ip='10.0.2.4/16',
mac='00:00:00:00:00:24')
        server25 = self.addHost('server25', ip='10.0.2.5/16',
mac='00:00:00:00:00:25')

        # Create pool servers in branch 1
        server31 = self.addHost('server31', ip='10.0.3.1/16',
mac='00:00:00:00:00:31')
        server32 = self.addHost('server32', ip='10.0.3.2/16',
mac='00:00:00:00:00:32')
        server33 = self.addHost('server33', ip='10.0.3.3/16',
mac='00:00:00:00:00:33')
        server34 = self.addHost('server34', ip='10.0.3.4/16',
mac='00:00:00:00:00:34')
        server35 = self.addHost('server35', ip='10.0.3.5/16',
mac='00:00:00:00:00:35')

        # Create branches
        branch4 = self.addHost('branch4', ip='10.0.4.1/16',
mac='00:00:00:00:00:40')
        cloud = self.addHost('cloud', ip='10.0.5.1/16', mac='00:00:00:00:00:50')

        # Create user
        user = self.addHost('user', ip='10.0.10.1/16', mac='00:00:00:00:00:ff')
```

```
# Create load balancers
lb1 = self.addHost('lb1', ip='10.0.0.1/16', mac='00:00:00:00:00:01')
lb2 = self.addHost('lb2', ip='10.0.0.2/16', mac='00:00:00:00:00:02')
lb3 = self.addHost('lb3', ip='10.0.0.3/16', mac='00:00:00:00:00:03')

# Create switches for load balancers
sb1 = self.addSwitch('sb1')
sb2 = self.addSwitch('sb2')
sb3 = self.addSwitch('sb3')

# Create firewalls L2, L3, L4
ngfw1 = self.addSwitch('ngfw1')
fw2 = self.addSwitch('fw2')
fw4 = self.addSwitch('fw4')

# Create switch s1 and s2
s1 = self.addSwitch('s1')
s2 = self.addSwitch('s2')


     ### PARTE 2 ###


# Connect servers in branch 1 to SD-WAN
# -- servers connected to ngfw1 firewall
# -- firewall ngfw1 to sb1
self.addLink(server11, ngfw1, bw=1000, delay='1ms')
self.addLink(server12, ngfw1, bw=1000, delay='1ms')
self.addLink(server13, ngfw1, bw=1000, delay='1ms')
self.addLink(server14, ngfw1, bw=1000, delay='1ms')
self.addLink(server15, ngfw1, bw=1000, delay='1ms')

self.addLink(ngfw1, sb1, bw=1000, delay='1ms')


# Connect servers in branch 2 to SD-WAN
# -- servers connected to firewall fw2
# -- firewall fw2 to sb2
self.addLink(server21, fw2, bw=1000, delay='1ms')
self.addLink(server22, fw2, bw=1000, delay='1ms')
self.addLink(server23, fw2, bw=1000, delay='1ms')
self.addLink(server24, fw2, bw=1000, delay='1ms')
self.addLink(server25, fw2, bw=1000, delay='1ms')

self.addLink(fw2, sb2, bw=1000, delay='1ms')


# Connect servers in branch 3 to SD-WAN
# -- servers connected to sb3

self.addLink(server31, sb3, bw=1000, delay='1ms')
self.addLink(server32, sb3, bw=1000, delay='1ms')
self.addLink(server33, sb3, bw=1000, delay='1ms')
self.addLink(server34, sb3, bw=1000, delay='1ms')
self.addLink(server35, sb3, bw=1000, delay='1ms')


# Connect branch 4 to SD-WAN
# -- branch4 server connected to firewall fw4
self.addLink(branch4, fw4, bw=1000, delay='1ms')


# Connect cloud branch to s1 (only forwarding traffic)
self.addLink(cloud, s1, bw=1000, delay='1ms')
```

```
        # Connect user to switch s2
        self.addLink(user, s2, bw=1000, delay='1ms')

        # Connect SD-WAN components
        self.addLink(s1, sb1, bw=1000, delay='1ms')
        self.addLink(s1, sb2, bw=1000, delay='1ms')
        self.addLink(s1, sb3, bw=1000, delay='1ms')
        self.addLink(s1, lb1, bw=1000, delay='1ms')
        self.addLink(s1, lb2, bw=1000, delay='1ms')
        self.addLink(s1, lb3, bw=1000, delay='1ms')
        self.addLink(s1, fw4, bw=1000, delay='1ms')


        # Connect user to SD-WAN s1 switch
        self.addLink(s2, s1, bw=1000, delay='1ms')


if __name__ == '__main__':


       ### PARTE 3 ###

       setLogLevel('info')
       topo = MyTopo()
       net = Mininet(topo=topo, link=TCLink, controller = OVSController)
       net.start()

    # Deploy load balancing algorithms in load balancers (lb1, lb2, lb3)
       lb1 = net.get('lb1')
       lb1.cmd('python3 lb.py &')
       lb2 = net.get('lb2')
       lb2.cmd('python3 lb_round_robin.py &')
       lb3 = net.get('lb3')
       lb3.cmd('python3 lb_weighted_round_robin.py &')


       # Run https service in servers
       server11 = net.get('server11')
       server11.cmd('python3 https_server.py &')
       server12 = net.get('server12')
       server12.cmd('python3 https_server.py &')
       server13 = net.get('server13')
       server13.cmd('python3 https_server.py &')
       server14 = net.get('server14')
       server14.cmd('python3 https_server.py &')
       server15 = net.get('server15')
       server15.cmd('python3 https_server.py &')

       server21 = net.get('server21')
       server21.cmd('python3 https_server.py &')
       server22 = net.get('server22')
       server22.cmd('python3 https_server.py &')
       server23 = net.get('server23')
       server23.cmd('python3 https_server.py &')
       server24 = net.get('server24')
       server24.cmd('python3 https_server.py &')
       server25 = net.get('server25')
       server25.cmd('python3 https_server.py &')

       server31 = net.get('server31')
       server31.cmd('python3 https_server.py &')
       server32 = net.get('server32')
       server32.cmd('python3 https_server.py &')
       server33 = net.get('server33')
       server33.cmd('python3 https_server.py &')
```

```
        server34 = net.get('server34')
        server34.cmd('python3 https_server.py &')
        server35 = net.get('server35')
        server35.cmd('python3 https_server.py &')

        branch4 = net.get('branch4')
        branch4.cmd('python3 https_server.py &')

        cloud = net.get('cloud')
        cloud.cmd('python3 https_server.py &')


    # Configurating VLAN interfaces
    # VLAN 100
        server11.cmd('sudo ip link add link server11-eth0 name eth0.100 type vlan
id 100')
        server11.cmd('sudo ip link set eth0.100 up')
        server11.cmd('sudo ip addr add 192.168.0.11/24 dev eth0.100')
        server31.cmd('sudo ip link add link server31-eth0 name eth0.100 type vlan
id 100')
        server31.cmd('sudo ip link set eth0.100 up')
        server31.cmd('sudo ip addr add 192.168.0.31/24 dev eth0.100')
        # VLAN 200
        server32.cmd('sudo ip link add link server32-eth0 name eth0.200 type vlan
id 200')
        server32.cmd('sudo ip link set eth0.200 up')
        server32.cmd('sudo ip addr add 192.168.0.32/24 dev eth0.200')
        server21.cmd('sudo ip link add link server21-eth0 name eth0.200 type vlan
id 200')
        server21.cmd('sudo ip link set eth0.200 up')
        server21.cmd('sudo ip addr add 192.168.0.21/24 dev eth0.200')
        # VLAN 300
        cloud.cmd('sudo ip link add link cloud-eth0 name eth0.300 type vlan id
300')
        cloud.cmd('sudo ip link set eth0.300 up')
        cloud.cmd('sudo ip addr add 192.168.0.51/24 dev eth0.300')
        server22.cmd('sudo ip link add link server22-eth0 name eth0.300 type vlan
id 300')
        server22.cmd('sudo ip link set eth0.300 up')
        server22.cmd('sudo ip addr add 192.168.0.22/24 dev eth0.300')


        # Configurating firewalls
        # ngfw1 blocks traffic from load balancers, branch 3, cloud branch to
branch 1
        ngfw1 = net.get('ngfw1')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1 "priority=5,action=DROP"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:ff,action=NORMAL"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:40,action=NORMAL"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:50,action=NORMAL"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:11,action=NORMAL"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:12,action=NORMAL"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:13,action=NORMAL"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:14,action=NORMAL"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:15,action=NORMAL"')
        ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:21,action=NORMAL"')
```

```
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:22,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:23,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:24,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,dl_src=00:00:00:00:00:25,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.2.1,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.2.2,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.2.3,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.2.4,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.2.5,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.4.1,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.1.1,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.1.2,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.1.3,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.1.4,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.1.5,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=10,ip,nw_src=10.0.10.1,action=NORMAL"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1 "priority=20,udp,action=DROP"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1 "priority=20,http,action=DROP"')
      ngfw1.cmd('ovs-ofctl add-flow ngfw1
"priority=100,dl_vlan=100,actions=NORMAL"')


      # fw2 blocks layer 3 traffic from load balancers, branch 3, cloud branch
to branch 2
      # all ARP traffic is permitted
      fw2 = net.get('fw2')
      fw2.cmd('ovs-ofctl add-flow fw2 "priority=5,action=DROP"')
      fw2.cmd('ovs-ofctl add-flow fw2 "priority=10,arp,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.1.1,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.1.2,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.1.3,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.1.4,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.1.5,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.4.1,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.10.1,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.2.1,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.2.2,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.2.3,action=NORMAL"')
      fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.2.4,action=NORMAL"')
```

```
        fw2.cmd('ovs-ofctl add-flow fw2
"priority=10,ip,nw_src=10.0.2.5,action=NORMAL"')
        fw2.cmd('ovs-ofctl add-flow fw2 "priority=20,udp,action=DROP"')
        fw2.cmd('ovs-ofctl add-flow fw2 "priority=20,http,action=DROP"')
        fw2.cmd('ovs-ofctl add-flow fw2
"priority=100,dl_vlan=200,actions=NORMAL"')
        fw2.cmd('ovs-ofctl add-flow fw2
"priority=100,dl_vlan=300,actions=NORMAL"')


        # fw4 blocks layer 2 traffic from/to load balancers to/from branch 4
        # all IP traffic is permitted
        fw4 = net.get('fw4')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4 "priority=5,action=DROP"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,ip,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:11,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:12,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:13,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:14,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:15,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:21,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:22,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:23,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:24,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:25,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:31,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:32,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:33,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:34,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:35,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:50,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:ff,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=10,dl_src=00:00:00:00:00:40,action=NORMAL"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=20,udp,action=DROP"')
        fw4.cmd('ovs-ofctl -O OpenFlow13 add-flow fw4
"priority=20,http,action=DROP"')


        # Test connectivity
        #net.pingAll()

        # Start the CLI
        CLI(net)

        # Stop the network
        net.stop()
```

8.2. Configuración de balanceador de carga – lb.py

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import random, ssl

servers = ['https://10.0.1.1:443', 'https://10.0.1.2:443',
'https://10.0.1.3:443', 'https://10.0.1.4:443', 'https://10.0.1.5:443']

class Handler(BaseHTTPRequestHandler):
    def do_GET(self):
        server = random.choice(servers)
        self.send_response(302)
        self.send_header('Location', server)
        self.end_headers()
        self.wfile.write(b'Redirecting...')

# HTTPS server
certfile = 'cert.pem'
keyfile = 'key.pem'
ssl_context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
ssl_context.load_cert_chain(certfile=certfile, keyfile=keyfile)
server_address = ('', 443)
load_balancer = HTTPServer(server_address, Handler)
load_balancer.socket = ssl_context.wrap_socket(load_balancer.socket)
load_balancer.serve_forever()
```

8.3. Configuración de balanceador de carga – lb_round_robin.py

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import itertools, ssl

servers = {
    'https://10.0.2.1:443',
    'https://10.0.2.2:443',
    'https://10.0.2.3:443',
    'https://10.0.2.4:443',
    'https://10.0.2.5:443'
}

server_cycle = itertools.cycle(servers)

class Handler(BaseHTTPRequestHandler):
    def do_GET(self):
        server = next(server_cycle)
        self.send_response(302)
        self.send_header('Location', server)
        self.end_headers()
        self.wfile.write(b'Redirecting...')

# HTTPS server
certfile = 'cert.pem'
keyfile = 'key.pem'
ssl_context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
ssl_context.load_cert_chain(certfile=certfile, keyfile=keyfile)
server_address = ('', 443)
load_balancer = HTTPServer(server_address, Handler)
load_balancer.socket = ssl_context.wrap_socket(load_balancer.socket)
load_balancer.serve_forever()
```

8.4. Configuración de balanceador de carga – lb_weighted_round_robin.py

```
from http.server import BaseHTTPRequestHandler, HTTPServer
import itertools, ssl

servers = {
    'https://10.0.3.1:443': 4,
    'https://10.0.3.2:443': 2,
    'https://10.0.3.3:443': 1,
    'https://10.0.3.4:443': 1,
    'https://10.0.3.5:443': 0
}

class Handler(BaseHTTPRequestHandler):
    def do_GET(self):
        server = weighted_round_robin(servers)
        if server == 'http://10.0.3.5:80':
             for svr in servers:
                     servers[svr] = random.randint(1, 10)
        self.send_response(302)
        self.send_header('Location', server)
        self.end_headers()
        self.wfile.write(b'Redirecting...')

def weighted_round_robin(servers):
    server_list = []
    weights = []
    for server, weight in servers.items():
        server_list.append(server)
        weights.extend([server] * weight)
    server_cycle = itertools.cycle(server_list)
    return next(server_cycle)

# HTTPS server
certfile = 'cert.pem'
keyfile = 'key.pem'
ssl_context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
ssl_context.load_cert_chain(certfile=certfile, keyfile=keyfile)
server_address = ('', 443)
load_balancer = HTTPServer(server_address, Handler)
load_balancer.socket = ssl_context.wrap_socket(load_balancer.socket)
load_balancer.serve_forever()
```

8.5. Configuración de servidor HTTPS con certificado SSL - https_server.py

```
from http.server import HTTPServer, SimpleHTTPRequestHandler
import ssl

server_address = ('', 443)
certfile = 'cert.pem'
keyfile = 'key.pem'
```

```
httpd = HTTPServer(server_address, SimpleHTTPRequestHandler)
httpd.socket = ssl.wrap_socket(httpd.socket, certfile=certfile, keyfile=keyfile,
server_side=True)

print("Server is running at https://localhost:443/")
httpd.serve_forever()
```