TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la
upf. Universitat
Pompeu Fabra
Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

Análisis de soluciones open-source routing

Memoria

Jaume Reverte Andrés

Tutor: Pere Barberan Agut

Curs 2021/2022

TecnoCampus
Mataró-Maresme

Abstract

The main objective is to analyse different open-source software alternatives for the simulation of networks, carrying out a set of simulations on each platform to determine which is best platform to determine which one is best suited for teaching networks.

Several examples and network structures will be created and shown that map as closely as possible to reality.

These analyses and the results obtained will allow the different alternatives used to be positioned and compared in order to generate a ranking.

Resum

L'objectiu principal consisteix a analitzar diferents alternatives de programes open source per a la simulació de xarxes, realitzant un conjunt de simulacions a cada plataforma per determinar quina s'adapta millor per a l'ensenyament de xarxes.

Es crearan i mostraran diversos exemples i estructures de xarxa que simulin el màxim possible a la realitat.

Aquestes anàlisis i resultats obtinguts permetran situar i comparar les diferents alternatives utilitzades per acabar generant un rànquing.

Resumen

El objetivo principal consiste en analizar diferentes alternativas de programas open source para la simulación de redes, realizando un conjunto de simulaciones en cada plataforma para determinar cuál se adapta mejor para la enseñanza de redes.

Se crearán y mostrarán diversos ejemplos y estructuras de red que mapeen lo máximo posible a la realidad.

Estos análisis y resultados obtenidos permitirán situar y comparar las diferentes alternativas utilizadas para terminar generando un ranking.

Índice

Índice de figuras

Índice de tablas

Glosario de términos

VM	Virtual Machine
MV	Máquina Virtual
OS	Operating system
SO	Sistema Operativo
GPL	GNU General Public License
GUI	Graphical User Interface
CLI	Command Line Interface
CORE	Common Open Research Emulator
EMANE	Extendable Mobile Ad-hoc Network Emulator
IMUNES	Integrated Multiprotocol Network Emulator/Simulator
NS-3	Network Simulator
OSPF	Open Shortest Path First
OSS	Open-Source Software
ARP	Address Resolution Protocol
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarized Zone
DNS	Domain Name Server
WEB	World Wide Web
RIP	Routing Information Protocol
OVA	Open Virtual Appliance
OVF	Open Virtualization Format

SSH	Secure Shell
RHEL	Red Hat Enterprise Linux
KVM	Kernel-based Virtual Machine
PC	Personal Computer

1. Introducción

La simulación de redes informáticas en la enseñanza puede ser una herramienta bienvenida y útil para muchas escuelas, sin embargo, proporcionar a los estudiantes un aula especializada que este suficientemente equipada para los propósitos de enseñanza es un asunto costoso y ciertamente no posible para todos los centros. Además, a menudo la enseñanza de las redes informáticas es sólo teórica y carece de sentido práctico.

Una alternativa para obtener la vertiente práctica es el uso de aplicaciones para la simulación / emulación de redes informáticas. En estas aplicaciones es fácil presentar a los estudiantes los principios básicos de las operaciones de redes, la configuración de dispositivos de red, la explicación del flujo de datos de la red, entre otros. También es útil para enseñar los principios del funcionamiento de la red, como es el enrutamiento, los protocolos, la cooperación de capas individuales en la transmisión de datos o la resolución de problemas en la red.

Una vez la escuela ha decidido empezar con la utilización de una aplicación de este tipo, se llega a la siguiente cuestión, ¿qué aplicación debe implementarse?

Se obtendrá varias respuestas y soluciones disponibles en el mercado, las cuales serán presentadas y comparadas a lo largo de este proyecto.

Uno de los requisitos para su correcta implementación es encontrar una solución que sea gratuita, debido a las circunstancias que rodean el ámbito académico como son la gran rotación de usuarios, la disparidad en los sistemas utilizados y el poco poder adquisitivo de los centros.

Pero las aplicaciones gratuitas se encuentran habitualmente con el problema de falta de soporte a diferencia de las alternativas comerciales, por lo que existe la posibilidad de que aparezcan más errores.

Este problema no ocurre en aplicaciones gratuitas con una comunidad grande, ya que existen más posibilidades de que ellos solucionen dichos errores y se haya elaborado una documentación suficientemente detallada como para utilizarlas sin ningún tipo de problema. Por lo tanto, es importante encontrar una aplicación que cuente con una comunidad activa que participe en ella.

Por regla general, el usuario asume que se encuentra con una calidad más pobre al utilizar una aplicación gratuita frente a una solución comercial, pero se estaría hablando de un concepto totalmente erróneo y, de hecho, puede ser todo lo contrario. Tales aplicaciones disponibles gratuitamente generalmente son desarrolladas por personas que las utilizan directamente para su propio uso, y, además, se han creado para una cuestión específica obteniendo así un producto a medida para cierta necesidad del desarrollador.

Para que sea realmente seguro utilizar la aplicación ha de estar bajo alguno de los siguientes estándares legales: Licencia Pública General de GNU / GPL u *Open Source*.

2. Marco teórico

2.1. Diferencias entre *Network Simulator* y *Network Emulator*

En primer lugar, para llegar a entender plenamente el proyecto, se debe tener en cuenta la diferencia entre un simulador y un emulador. Estos términos generalmente son confundidos, y aunque tengan ciertas similitudes, estos contienen varias diferencias significativas.

Tanto los emuladores como los simuladores permiten realizar pruebas en entornos flexibles y definidos por software. De este modo, permiten ejecutar pruebas con mayor rapidez y facilidad que si se tuviera que configurar un dispositivo de hardware real.

Pero el hecho de que tanto los simuladores como los emuladores sirvan para fines similares no significa que funcionen de forma idéntica, ya que se encuentran diferencias esenciales entre ellos.

Un simulador está diseñado para crear un entorno que contenga todas las variables y configuraciones de *software* que existirán en el entorno de producción real de una aplicación, sin embargo, los simuladores no intentan emular el *hardware* real que albergará la aplicación en producción. Teniendo en cuenta que los simuladores sólo crean entornos de software, estos pueden implementarse utilizando lenguajes de programación de alto nivel, en cambio, un emulador sí intenta imitar todas las características de hardware de un entorno de producción, así como las de *software*. Para conseguirlo, normalmente se debe escribir un emulador en lenguaje ensamblador.

En cierto sentido, los emuladores ocupan un lugar intermedio entre los simuladores y los dispositivos reales. Los simuladores sólo imitan las características del entorno que pueden configurarse o definirse mediante *software*, mientras que los emuladores imitan tanto las características del hardware como las del software.

Dado que los emuladores pueden no hacer un trabajo perfecto al emular el hardware y el software de un entorno de producción, no son un sustituto de las pruebas con dispositivos reales, sólo permiten configurar un entorno más parecido al que se obtendría en un dispositivo real.

Por lo general, los simuladores son la mejor herramienta para los escenarios de pruebas de software centrados en asegurar que una aplicación funciona como se espera cuando interactúa con aplicaciones o entornos externos.

Por ejemplo, es posible que desee probar la capacidad de una aplicación para enviar datos a otra aplicación, para ello, suele ser suficiente un entorno simulado, ya que es poco probable que la configuración de hardware subyacente tenga un gran impacto en las transacciones de datos de la aplicación. Del mismo modo, cuando se pretende asegurar si la interfaz de una aplicación se muestra correctamente bajo diferentes resoluciones de pantalla, los entornos de prueba simulados son apropiados.

Por otro lado, los emuladores son más útiles cuando se necesita probar la interacción del software con el *hardware* subyacente, o una combinación de *hardware* y *software*.

Pero ¿Cómo se puede averiguar si una aplicación causara problemas una vez desplegada? Con la ayuda de un emulador se podrá obtener la respuesta, quizás además también se necesite saber cómo se comporta la aplicación utilizando diferentes tipos de CPU o distintas asignaciones de memoria.

No obstante, esta diferencia no está del todo clara ya que, mayoritariamente, en internet suelen referirse, incorrectamente, a todos estos programas como simuladores.

2.2. Técnicas de experimentación de redes

Actualmente existen diferentes soluciones para experimentar y aprender sobre redes, en este subapartado se van a enumerar las técnicas más importantes, a la vez que se explicarán sus ventajas e inconvenientes.

***Testbeds*:** Se trata de la más rudimentaria y la única que necesita el uso de *hardware* específico. Se basa en el despliegue de *hardware* de computación y redes que busca replicar las condiciones en las que se va a desplegar en la realidad, por lo tanto, se busca replicar al máximo detalle las condiciones que va a tener en el mundo real.

La mayor ventaja de este tipo de prueba es que, al hacerse con el *hardware* que se usará en la realidad, la fidelidad es prácticamente perfecta.

El mayor inconveniente es que debido al uso de muchos componentes de hardware, los costes para la realización de prueba de diferentes componentes aumentan exponencialmente.

Virtualización: Se trata de convertir aquellos equipos, que en *testbed* son físicos, en equipos virtualizados. Este permite tener el mismo nivel de fiabilidad que se encontraría en *testbed* pero con un ahorro de costes al poder reducir el *hardware* necesario.

Simulación: Otra alternativa que existe a *testbeds*, que puede ser virtualizado o no, es la simulación. Permite la creación de cientos y miles de nodos simulados en un solo servidor, pero el mayor problema que se encuentra es la pérdida de fiabilidad, delimitando mucho el alcance del proyecto y la comunicación de los componentes, alegándose así de un escenario más realista.

Emulación: Gracias a la existencia de SO virtualizados, las herramientas de emulación de Linux y FreeBSD[1] permiten la creación de redes experimentales utilizando componentes del sistema operativo.

Más adelante se analizarán las diferencias entre componentes virtualizados y componentes contenerizados, por ahora únicamente se concluye en que la principal diferencia encontrada es la mayor ocupación de espacio en los virtualizados, y, por lo tanto, que en una misma máquina no es posible contener tantos a la vez.

Teniendo en cuenta y una vez conocidas todas las diferentes soluciones que existen para la experimentación de redes, en el alcance del proyecto se van a escoger aquellas que utilicen la tecnología de virtualización y simulación, ya que son las más teóricas y permiten alcanzar el objetivo del proyecto de una forma más sencilla.

[1] FreeBSD se trata de un OS principalmente usado para servidores, escritorios y plataformas embedidas. Tiene dos características que lo hacen idóneo para las tareas de emulación. La primera es módulos de compatibilidad, se refiere a la capacidad que tiene de ejecutar programas de otros sistemas operativos (incluyendo OS como Linux, SCO, NetBSD y BSDI). La segunda es la carga dinámica de los módulos de kernel, que permite tener acceso a nuevos protocolos de red o emuladores binarios en tiempo de ejecución sin necesidad de generar un nuevo kernel.

2.3. Diferentes tecnologías de virtualización

Existen dos grandes tecnologías diferenciadas, los contenedores y la virtualización.

Ambas tienen como objetivo la virtualización de una aplicación dentro de un *host*. La principal diferencia es la interacción que tienen con el *host* que las soporta, además de las necesidades de capas intermedias para esa comunicación.

Cuando se habla de virtualización se deben tener en cuenta dos elementos, el hipervisor y el sistema operativo. El sistema operativo es necesario para el correcto funcionamiento de la máquina virtualizada, y el hipervisor es el encargado de comunicar el sistema operativo de la máquina virtualizada con el del *host*.

En el caso de los contenedores se elimina del sistema operativo y el hipervisor siendo sustituido ambas cosas por el gestor de contenedores.

En la imagen siguiente podemos ver de forma esquemática las dos opciones, se puede observar como en el caso de las VM cada aplicación tiene consigo su propio SO, en cambio en Contenedores el OS es el de la máquina *host* ya que las aplicaciones utilizan el *Container Engine* para comunicarse con el sin necesidad de un OS propio.

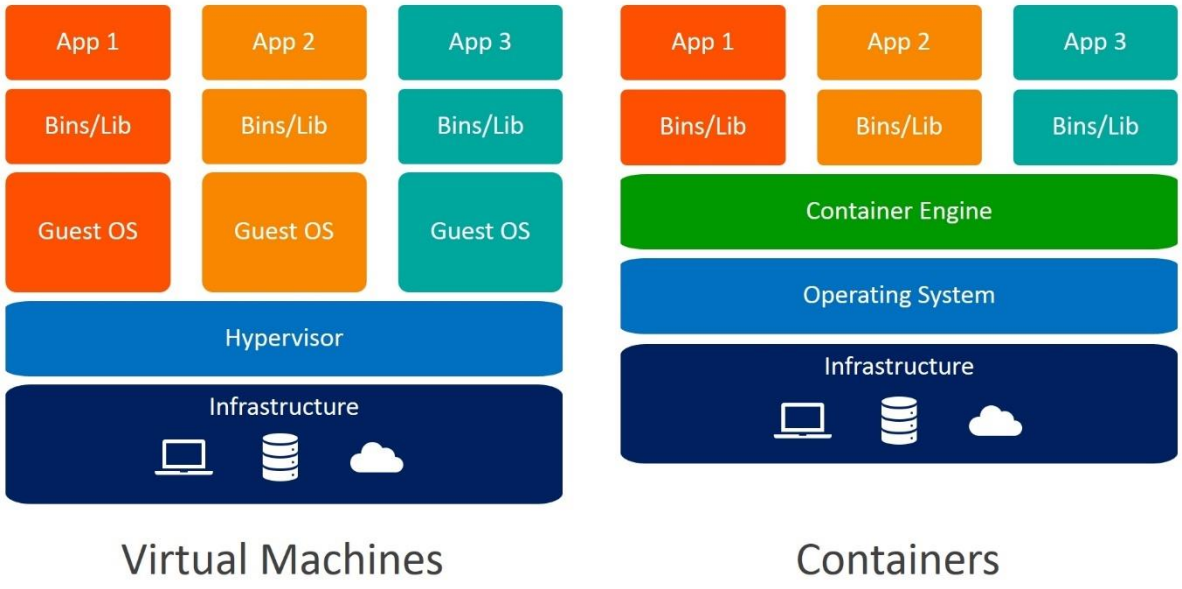

Ilustración 1 Comparación Máquina Virtual - Contenedor

En la siguiente tabla se pueden observar las principales diferencias que existen entre ambas tecnologías:

	Máquina virtual	Contenedores
Sistema Operativo Invitado	Cada VM utiliza su propio OS en un espacio de memoria propio.	Todas las máquinas invitadas comparten el mismo OS y *Kernel*.
Comunicación	A través de dispositivos de *ethernet*.	Mecanismos estándares IPC como Señales, *pipes*, *sockets*, etc.
Seguridad	Depende de la implementación del hipervisor.	*Access control*.
Rendimiento	Fallas de rendimiento ya que se ha de traducir cada instrucción del OS invitado al OS del *host*.	Rendimiento cercano al que se conseguiría nativamente.
Aislamiento	Se comparten librerías, documentos, etc, entre invitados.	Subdirectorios pueden ser transparentemente montados y se pueden compartir.
Tiempo de inicio	Tardan aproximadamente algunos minutos antes de inicializarse.	Se pueden crear y encender en pocos segundos.
Almacenamiento	Utilizan mucho espacio ya que requiere la instalación de un OS para cada máquina.	Al compartir el OS del *host* no necesitan tanto espacio de disco.

Tabla 1 Diferencias entre VM y contenedores

De la comparativa se concluye que encontrar una aplicación que trabaje con contenedores es mejor, para nuestro caso de estudio, que una aplicación que trabaje con VM, ya que las primeras pueden ser utilizadas en equipos menos potentes, adaptándose más a las necesidades de un estudiante.

2.4. Imunes

2.4.1. Introducción

Es un proyecto que surge de la Universidad de Zagreb, en concreto de la Facultad de Ingeniería Eléctrica y de Computación, han desarrollado el emulador/simulador integrado de redes multiprotocolo (IMUNES o *Integrated Multiprotocol Network Emulator / Simulator*) para utilizarlo como herramienta de investigación y educación de redes. IMUNES funciona con los sistemas operativos FreeBSD y Linux. Utiliza la tecnología de virtualización de la pila de red a nivel del kernel proporcionada por FreeBSD y los contenedores Docker y Open vSwitch en Linux.

La descripción que ofrecen en su página web sobre el proyecto es la siguiente:

"Hemos desarrollado un marco de emulación/simulación de topología de red realista basado en el núcleo de los sistemas operativos FreeBSD y Linux particionado en múltiples nodos virtuales ligeros, que pueden interconectarse mediante enlaces a nivel de núcleo para formar topologías de red arbitrariamente complejas. "

En su página web detallan las siguientes características:

- Utiliza las tecnologías de Jails[2], Netgraph (FreeBSD), Docker[3] y Open vSwitch (Linux).
- Emulación / simulación en tiempo real de topologías de redes IP.
- 100s a 1000s de nodos virtuales en una única maquina física.
- Arquitectura escalable para experimentos de gran escala.
- La GUI permite crear y configurar los diferentes nodos.
- Ligero, no ocupa demasiado.
- *Open source* y gratis.

En su página de Github muestran la siguiente tabla de prueba que funcionan y en que OS se ha de realizar:

[2] Jails se puede considerar un tipo de virtualización a nivel de sistema operativo ya que mejora considerablemente el entorno generado por chroot. Los procesos generados solo pueden acceder al sistema de archivos, el conjunto de usuarios y el subsistema de red, separando a estos procesos del resto del sistema.

[3] Docker es una de las principales tecnologías de contenedores que existe en la actualidad. Para no hacer repetitivo, durante el proyecto cuando aparezca algún programa que utiliza contenedores para funcionar utilizara esta tecnología.

	Linux	FreeBSD
Benchamrk	YES	YES
DHCP	YES	YES
DNS+Mail+Web	YES	YES
Functional_tests	NO	YES
Gif	NO	YES
OSPF	YES	YES
Ping	YES	YES
RIP	YES	YES
Services	YES	YES
Traceroute	YES	YES

Tabla 2 Capacidades Imunes según OS host

2.4.2. Descarga e instalación

Para poder ser utilizado Imunes requiere FreeBSD 8 (o mayor) y necesita un kernel compilado con la VIMAGE opción incluida, también se puede instalar en una maquina Linux, pero la instalación con Ova resulta más sencilla. Por lo que se procede a instalar la máquina virtual que ofrecen con todas las características ya preconfiguradas. Se descarga la Ova de la máquina:

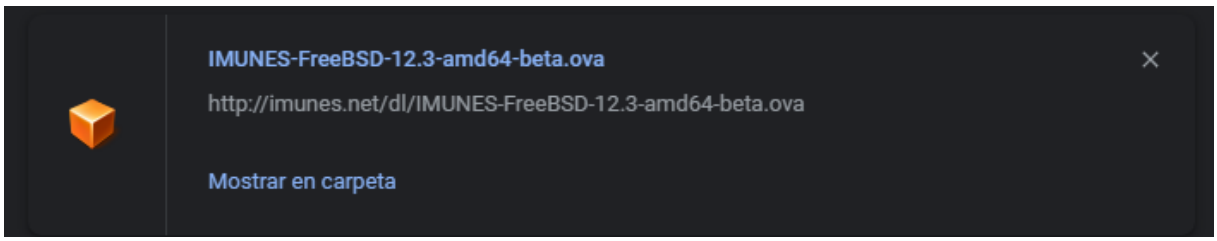

Ilustración 2 Descarga OVA Imunes

Y después se procede a importarla dentro de un gestor de máquinas virtuales, en este caso VirtualBox.

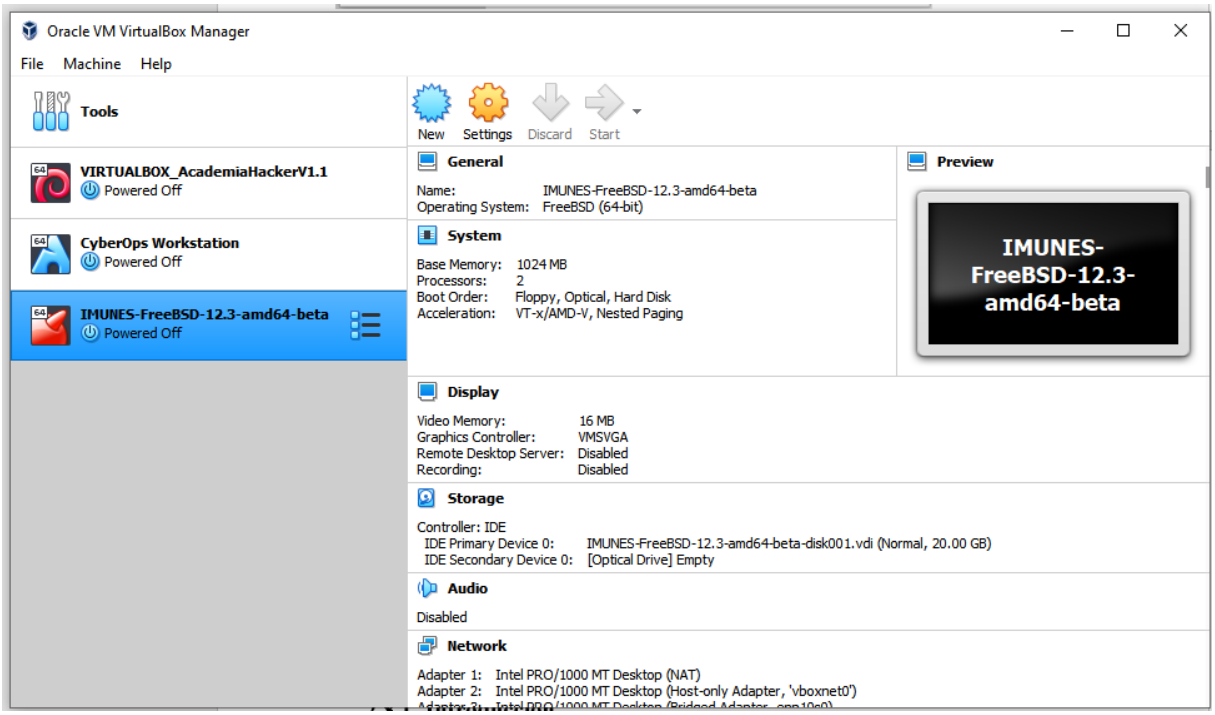

Ilustración 3 VM Imunes importada

Se inicia la máquina virtual pulsando *Start*. Es muy probable que salga un error de adaptador de red, simplemente se ha de modificar el adaptador de red preconfigurado por el que tenga la máquina *host* que se esté utilizando para conectarse a internet.

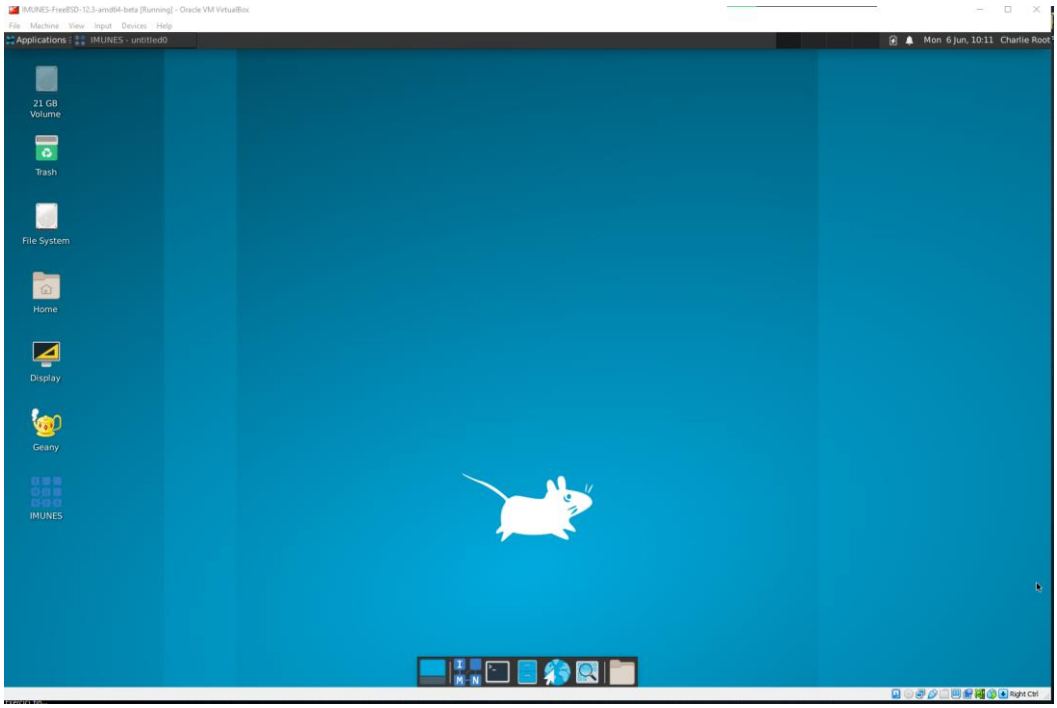

Ilustración 4 Imunes VM escritorio

Se puede ver en la imagen del escritorio, en el menú inferior, que Imunes ya está preinstalado en esta máquina virtual y por lo tanto únicamente hace falta pulsar en el icono para que se abra el programa.

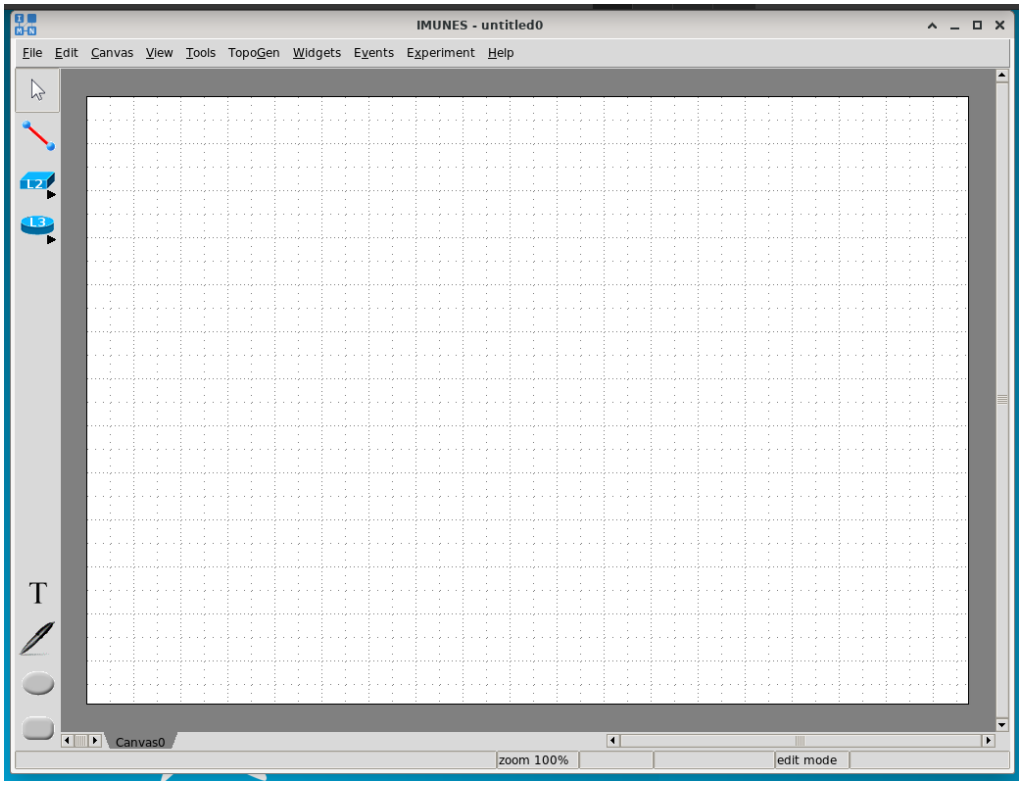

Ilustración 5 Imunes página de inicio

2.4.3. Usabilidad

Al contar con una interfaz gráfica, el uso se hace bastante ameno y sencillo. Se trata de *drag and drop* para ir añadiendo los diferentes elementos que van a conformar la topología.

A modo de ejemplo de uso se ha creado la siguiente topología, dos ordenadores conectados a un *switch* que está conectado a la vez a un *router*. A continuación, se muestra la topología implementada dentro de la herramienta:

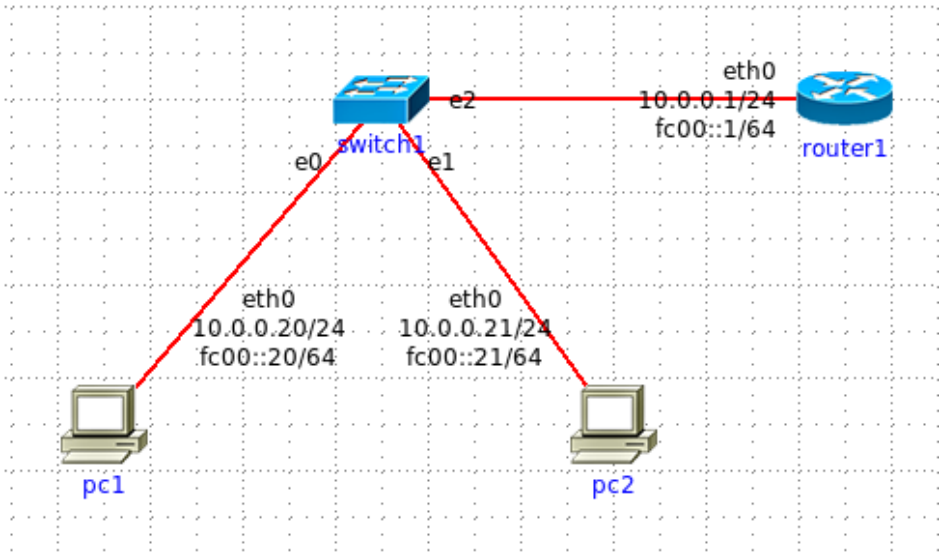

Ilustración 6 Imunes configuración básica

Entrando en detalle, para colocar uno de los diferentes elementos, simplemente se ha de hacer clic en él y luego pulsar en la pantalla de la topología. El cable entre dispositivos es único para todas las conexiones y no se puede modificar.

Las direcciones IPv4 y IPv6 se asignan de forma automática a los diferentes equipos.

Existen dos formas diferentes de configurar los dispositivos:

- La forma sencilla, pulsando clic derecho y accediendo a configuración, se abre un entorno grafico donde el usuario puede seleccionar que servicios del dispositivo quiere activar.
- La forma compleja, utilizando la consola, permitiendo modificar equipos en ejecución y generar conexiones remotas a los dispositivos (mediante *telnet* o SSH).

Una vez generada la topología se ha de ejecutar para que se levanten los contenedores de los diferentes equipos.

2.4.4. Elementos disponibles

Permite crear:

- *Hub*
- LAN switch
- Clic switch
- *External interface*

- RSTP *switch*
- *Filter Node*
- *Packet generator*
- *External connection*
- *Router*
- *Click router*
- *Host / Server*
- PC
- NAT64

2.4.5. Comprobaciones

Interfaz Gráfica	Sí	-
Instalación sencilla	Sí	-
Funcionalidades básicas	Sí	-
No necesita VM	Sí	-

Tabla 3 Comprobaciones Imunes

2.5. GNS3

2.5.1. Introducción

GNS, se trata de una aplicación de simulación / emulación de redes informáticas centrada en el *software* de Cisco. Se basa en el lenguaje de programación Python 3.

Al levantar una máquina virtual independiente para cada dispositivo a desplegar, es importante conocer los requisitos técnicos recomendados:

Sistema operativo: Windows 7 (64 bits) y posteriores, Mavericks (10.9) y posteriores o cualquier distro de Linux.

Procesador: 4 o más núcleos lógicos.

Memoria RAM: 8 GB.

Espacio libre en el disco: 35GB.

2.5.2. Descarga e instalación

La aplicación se puede descargar a través del sitio web http://www.gns3.com/.

Una cosa interesante de este programa es que permite durante el mismo proceso de instalación, instalar programas útiles para el proceso de aprendizaje como son WinPcap, Npcap, Putty, Dynamips, Qemu y Wireshark.

Para la descarga simplemente hay que ir a la página web oficial y descargar el ejecutable que allí se encuentra. El usuario ha de tener una cuenta activa y se ha de completar el proceso de registro antes de la descarga.

Ilustración 7 Descarga GNS3

Una vez tenemos el ejecutable se han de seguir los pasos del instalador. Para poder gestionar y utilizar las máquinas virtuales correctamente es necesario disponer de la VM GNS instalada y configurada en el equipo.

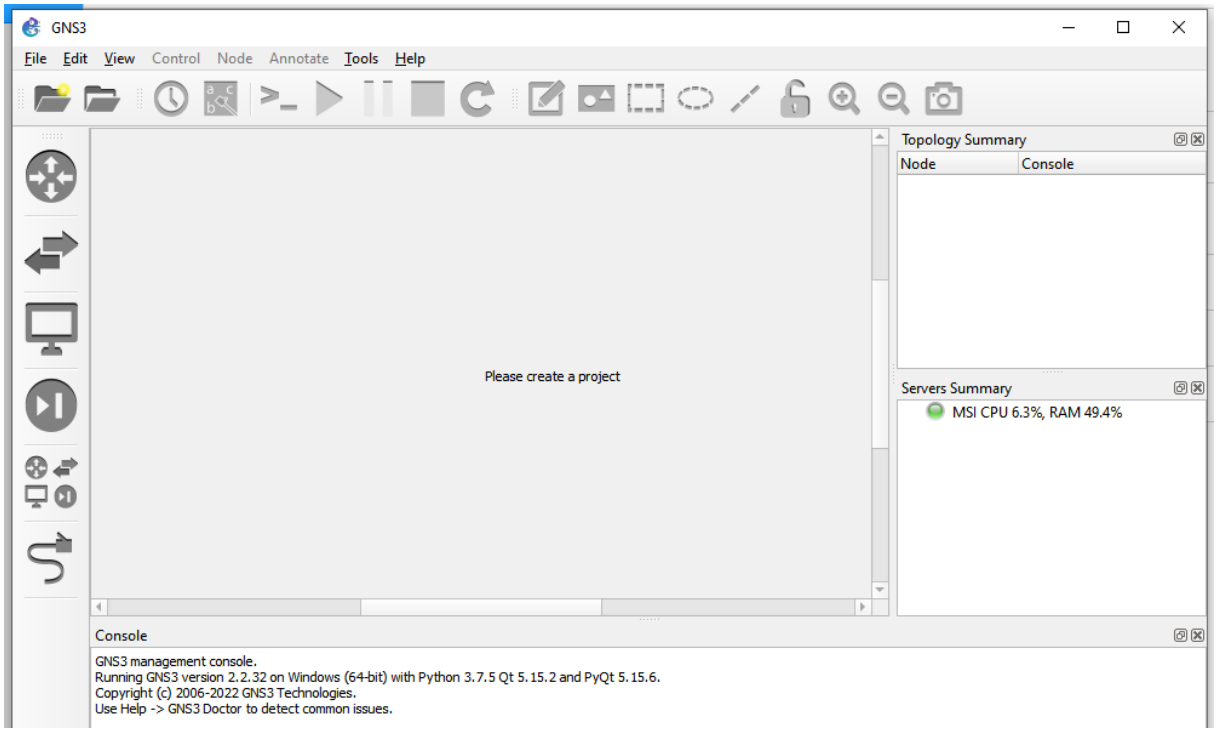

Ilustración 8 GNS3 página de inicio

2.5.3. Comprobaciones

Interfaz Gráfica	Sí	-
Instalación sencilla	No	Requiere de varios componentes a parte del propio programa para funcionar correctamente.
Funcionalidades básicas	Sí	-
No necesita VM	No	Para la versión de escritorio es necesario tener una máquina virtual que de soporte al sistema.

Tabla 4 Comporbaciones GNS3

Esta opción ha sido eliminada antes de las pruebas ya que al trabajar únicamente con VM hace imprescindible el uso de ordenadores de última generación, lo cual no se alinea con la realidad académica.

2.6. ContainerLab

2.6.1. Introducción

Se trata de un emulador de redes *open source* de reciente creación que permite la construcción de redes en un ambiente similar al de *devops*[4].

[4] Se puede hacer esta afirmación ya que los *scripts* necesarios para levantar las máquinas virtuales beben directamente de la forma de desplegar redes y aplicaciones en la nube.

Proporciona una interfaz de línea de comandos para orquestar y gestionar laboratorios de redes basados en contenedores y es compatible con imágenes de componentes comerciales y libres.

Lo más interesante es que soporta cualquier sistema operativo *open source* que esté publicado como una imagen de contenedor. También soporta redes basadas en dispositivos VM, permitiendo así a los usuarios utilizar imágenes de *routers* comerciales.

Inicialmente desarrollado por los ingenieros de Nokia, actualmente permite contribuciones de toda la comunidad.

2.6.2. Descarga e instalación

Es mejor utilizarlo en Linux. Funciona tanto en distribuciones basadas en Debian como en RHEL, e incluso puede ejecutarse en Windows Subsystem para Linux (WSL2). Su principal dependencia es Docker, así que primero se debe instalar Docker. Las pruebas se hacen en un sistema Ubuntu 20.04.

```
sudo apt install apt-transport-https ca-certificates
sudo apt install -y curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
focal stable"
sudo apt update
apt-cache policy docker-ce
sudo apt install -y docker-ce
```

Una vez se ha completado lo de arriba se procede a instalar Containerlab.

```
bash -c "$(curl -sL https://get-clab.srlinux.dev)"
```

Esta es la manera en la que se ha instalado el programa dentro del banco de pruebas. No obstante, a continuación, se detalla los pasos que se indican en la documentación oficial del programa. Hay varias formas de descarga oficiales, pero en este proyecto únicamente se van a valorar la instalación mediante *script*, la instalación manual y la instalación mediante gestores de paquetes.

2.6.2.1. Mediante *script*

Esta solución es bastante sencilla, no obstante, requiere de tener privilegios de administrador en el sistema, sistema operativo Linux o con una VM Linux.

Este *script* es muy útil ya que detecta automáticamente el OS del *host* y descarga en consecuencia.

```
# download and install the latest release (may require sudo)
bash -c "$(curl -sL https://get.containerlab.dev)"

# download a specific version - 0.10.3 (may require sudo)
bash -c "$(curl -sL https://get.containerlab.dev)" -- -v 0.10.3

# with wget
bash -c "$(wget -qO - https://get.containerlab.dev)"
```

2.6.2.2. Mediante gestor de paquetes

Esta solución depende de que el host tenga un gestor de paquetes instalado, sirven los repositorios de APT, YUM y APK.

```
# mediante gestor de APT
echo "deb [trusted=yes] https://apt.fury.io/netdevops/ /" | \
sudo tee -a /etc/apt/sources.list.d/netdevops.list
apt update && apt install containerlab

# mediante gestor de YUM
yum-config-manager --add-repo=https://yum.fury.io/netdevops/ && \
echo "gpgcheck=0" | sudo tee -a /etc/yum.repos.d/yum.fury.io_netdevops_.repo
yum install containerlab

# mediante gestor de APK
Se puede descargar el paquete .apk mediante Github releases.
```

2.6.2.3. Manual

En el caso de que sea imposible instalarlo de otra manera se puede recurrir a la instalación manual, aunque queda desaconsejado hacerlo.

```
# get the latest available tag
LATEST=$(curl -s https://github.com/srl-labs/containerlab/releases/latest | \
    sed -e 's/.*tag\/v\(.*\)\".*/\1/')

# download tar.gz archive
curl -L -o /tmp/clab.tar.gz "https://github.com/srl-
labs/containerlab/releases/download/v${LATEST}/containerlab_${LATEST}_Linux_a
md64.tar.gz"

# create containerlab directory
mkdir -p /etc/containerlab

# extract downloaded archive into the containerlab directory
tar -zxvf /tmp/clab.tar.gz -C /etc/containerlab

# (optional) move containerlab binary somewhere in the $PATH
mv /etc/containerlab/containerlab /usr/bin && chmod a+x /usr/bin/containerlab
```

2.6.3. Usabilidad

2.6.3.1. Un nodo SR Linux

Se trata del ejemplo más sencillo que se puede ejecutar en el entorno Containerlab.

Consiste en un solo nodo SR Linux.

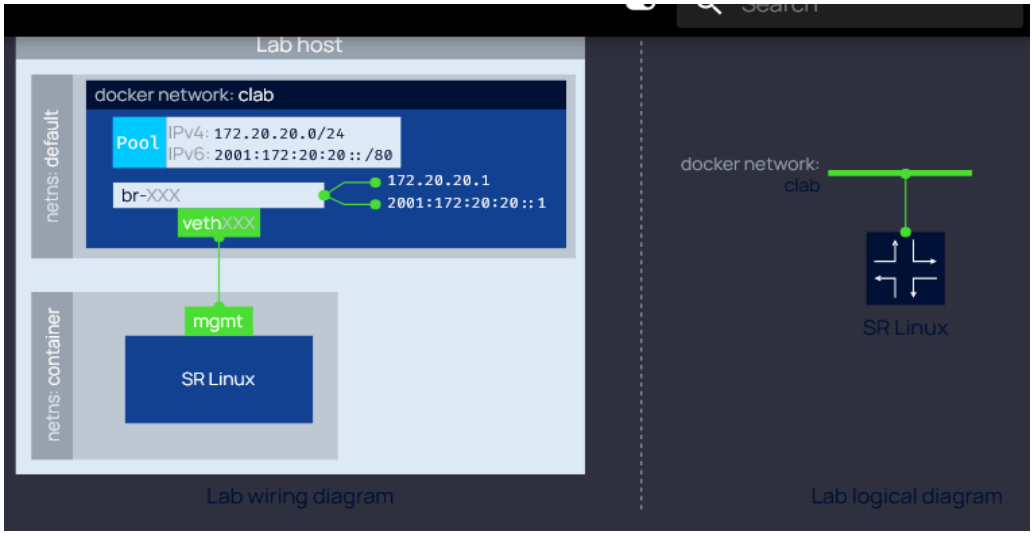

Ilustración 9 Topología Un Sr Linux Nodes Containerlab

2.6.3.2. Dos nodos SR Linux

El segundo ejemplo consiste en dos nodos SR Linux conectados entre sí mediante las interfaces e1-1.

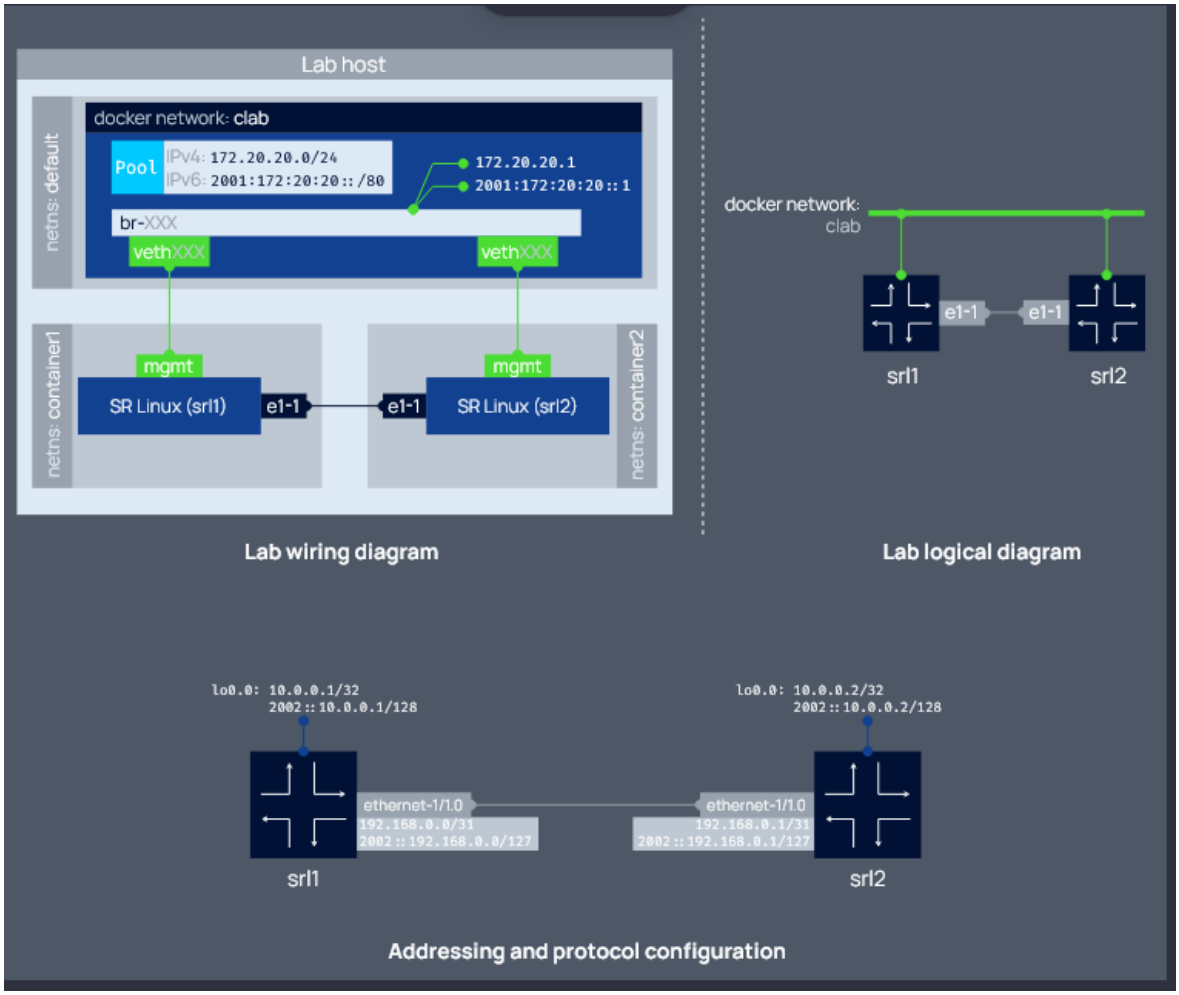

Ilustración 10 Topología dos Sr Linux Nodes Containerlab

El *yaml* para desplegar esta tipología es el siguiente:

```
name: srl02

topology:
  kinds:
    srl:
      type: ixr6 # See https://www.nokia.com/networks/products/7250-interconnect-router/
      image: ghcr.io/nokia/srlinux
  nodes:
    srl1:
      kind: srl
      startup-config: srl1.cfg.json
    srl2:
      kind: srl
      startup-config: srl2.cfg.json

  links:
    - endpoints: ["srl1:e1-1", "srl2:e1-1"]
```

Se observa gracias al yaml mostrado como el despliegue de estos contenedores tienen el mismo formato que cualquier despliegue de aplicaciones en entornos *devops.*

2.6.4. Elementos disponibles

Se centra en OS de redes utilizados típicamente para la prueba de diseños y utilidades de redes, como:

- Nokia SR-Linux
- Arista cEOS
- Azure SONiC
- Juniper cRPD
- Cumulus VX

Aparte, ofrece la posibilidad de lanzar VM utilizando la integración con Vrnetlab. Vrnetlab se trata de una VM regular dentro de un contenedor que permite ejecutarse como si se tratara de una imagen. A continuación, una imagen donde se muestra el funcionamiento de Vrnetlab:

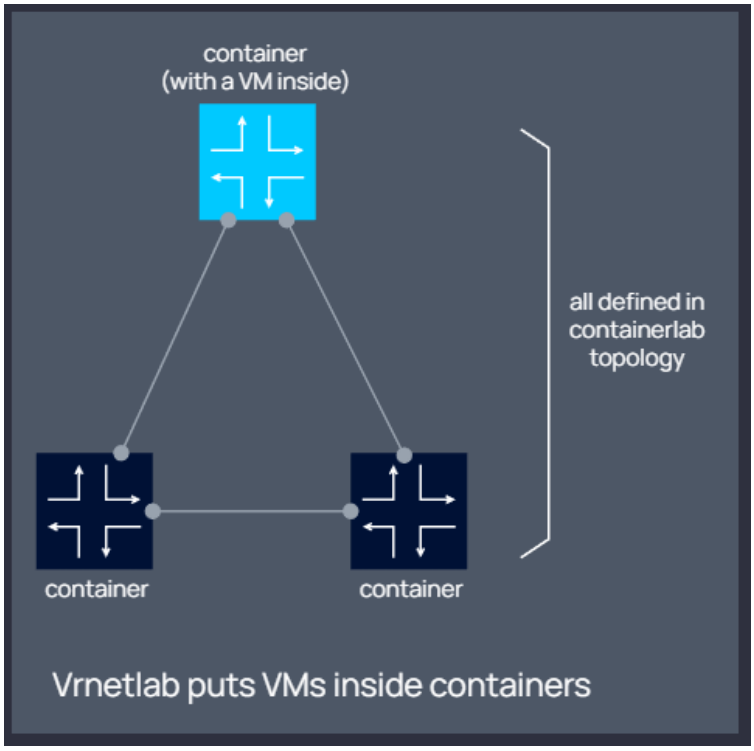

Ilustración 11 VM en Containerlab diagrama

Con esta tecnología se disponen de los siguientes componentes:

- Nokia virtual SR OS (vSim/VSR)
- Juniper vMX
- Juniper vQFX
- Cisco IOS XRv9k
- Cisco Nexus 9000v
- Dell FTOS10v
- Cisco CSR 1000v
- Arista vEOS
- Palo Alto PAN
- IPInfusion OcNOS

2.6.5. Comprobaciones

Interfaz Gráfica	No	Se han de programar los *scripts* para la creación de los diferentes elementos de la topología.
Instalación sencilla	Sí	-
Funcionalidades básicas	Sí	-
No necesita VM	No	Hay maquinas disponibles que necesitan ser generadas con su propia VM, pero se trata de casos muy concretos.

Tabla 5 Comprobaciones ContainerLab

Se trata de la aplicación que más capacidades de alto nivel presenta. Permite de forma semi automatizada levantar y destruir diferentes topologías.

No obstante, ha sido eliminada durante las pruebas porque presenta demasiadas opciones al estudiante y junto con su tremenda complejidad hace que no sea una herramienta buena para la educación de redes.

Se trata de una herramienta más enfocada a un nivel profesional y no educativo.

Además, no permite de forma sencilla la creación de las topologías que se buscan poder generar.

2.7. Educational Network Simulator

2.7.1. Introducción

Se trata del simulador más liviano de los presentes. Trabaja con archivos html. Es una forma muy diferente de estudiar las redes y a un alto nivel. Busca proporcionar a escueles secundarias un simulador para que practiquen y entiendan conceptos como: Elementos básicos de una red (*hosts*, *routers* y *switchs*), direcciones IP, *Unicasting* / *Broadcasting*, protocolos, *ping*, *traceroute*, NAT, DNS, HTTP y DHCP.

Se trata por lo tanto de un simulador que busca la simplicidad para explicar conceptos básicos de redes.

2.7.2. Descarga e instalación

Para instalar y utilizarlo simplemente se ha de ir a la página web oficial y descargar el archivo zip, descomprimirlo y entonces ejecutar el archivo html que se encuentra dentro llamado “simulator”.

Una vez abierto aparecerá la siguiente página de inicio:

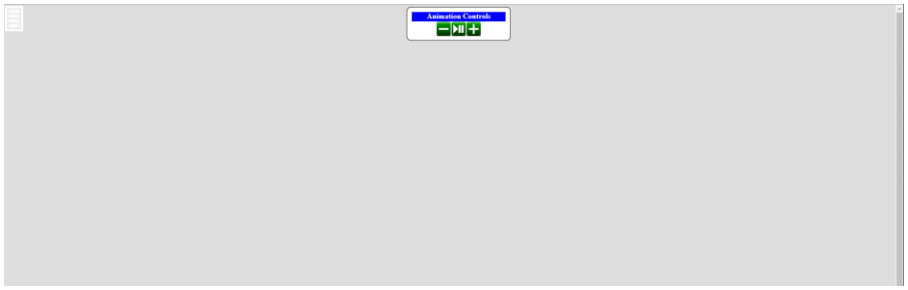

Ilustración 12 ENS página de inicio

2.7.3. Usabilidad

Se trata de un programa muy sencillo, hay un menú arriba a la izquierda que permite al usuario seleccionar y desplegar los diferentes elementos disponibles en la plataforma. Estos elementos son: *Host, DHCP server, DNS server, web server, switch y router*.

La topología probada es la siguiente:

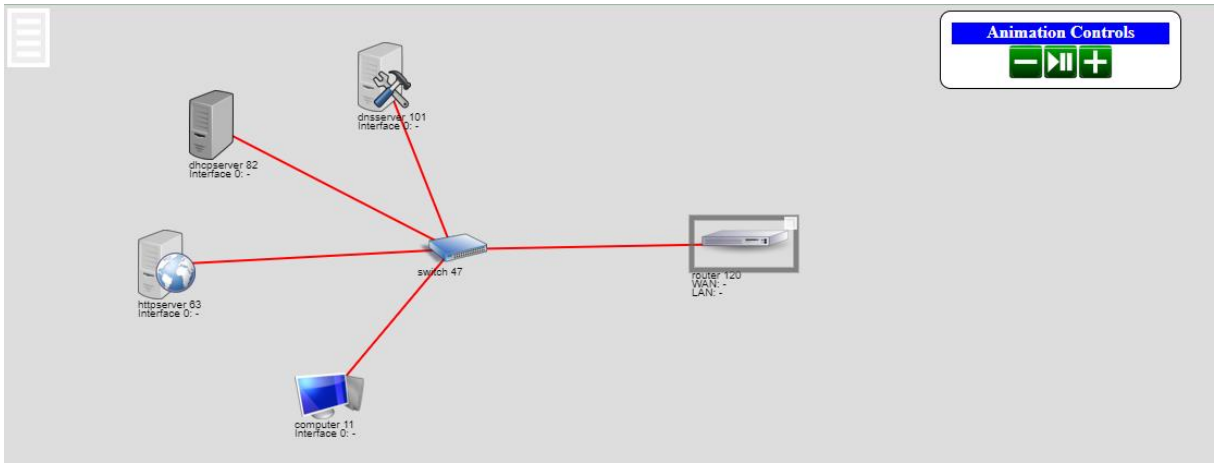

Ilustración 13 topología básica ENS

Haciendo clic arriba a la derecha en un cuadrado blanco que aparece al pasar el cursor por encima de cada elemento, permite hacer pequeñas configuraciones en los equipos,

2.7.4. Comprobaciones

Interfaz Gráfica	Sí	-
Instalación sencilla	Sí	-
Funcionalidades básicas	Sí	-
No necesita VM	Sí	-

Tabla 6 Comprobaciones ENS

2.8. Labtainers

2.8.1. Introducción

Se trata de una herramienta de virtualización cuyo principal objetivo es la ciberseguridad. Para ello tiene un catálogo de 50 laboratorios diferentes en donde el estudiante podrá encontrar retos que le proporcionarán conocimiento sobre las tecnologías que envuelven las redes.

Para el despliegue de los elementos de la topología se utilizan contenedores.

Es una herramienta un tanto extraña y diferente ya que en si el objetivo principal no es el de mostrar topologías de red. No obstante, es una buena alternativa ya que permite reforzar conocimientos de redes mediante los laboratorios montados, permitiendo al alumno de forma interactiva entender las diferentes tecnologías que hay.

2.8.2. Descarga e instalación

En la página web oficial se encuentra un archivo OVA el cual se ha de instalar y configurar dentro del entorno de VirtualBox.

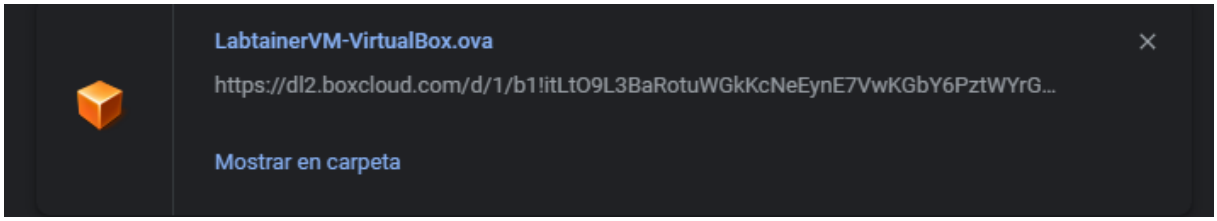

Ilustración 14 Decsrga OVA Labtainers

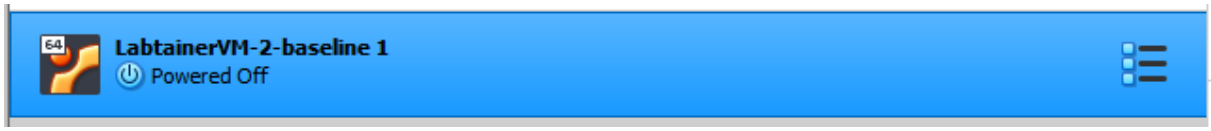

Ilustración 15 VM Labtainers

Al iniciar la VM ha de aparecer un escritorio de un sistema Ubuntu y una consola de comandos con el siguiente texto:

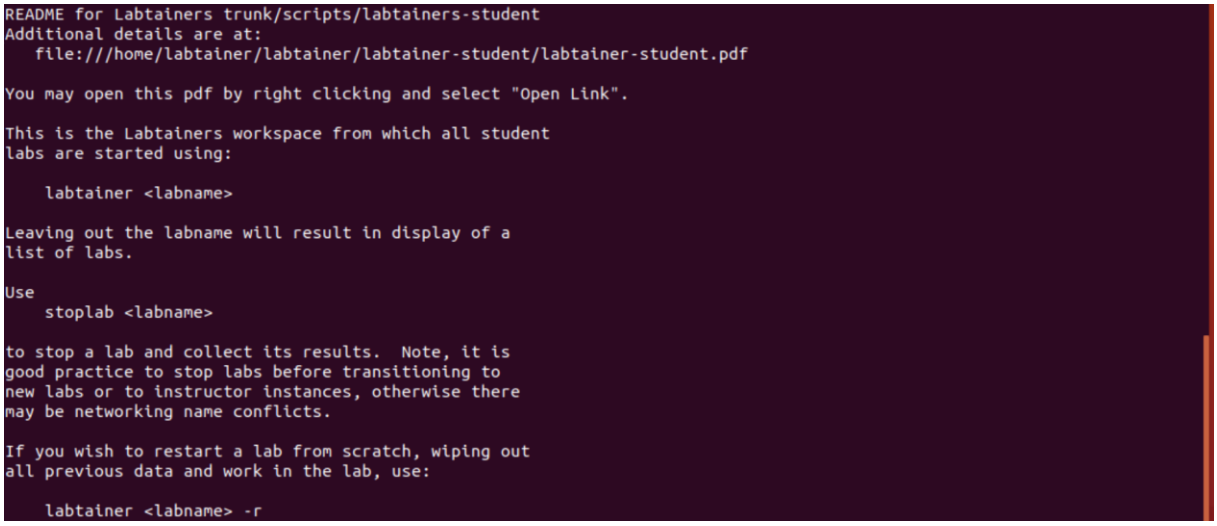

Ilustración 16 Texto bienvenida Labtainers

Por defecto la máquina presupone que el usuario entrante es un estudiante. Esto es importante ya que es una herramienta que ofrece tanto soluciones para los estudiantes como para los profesores. Se recomienda distribuir una máquina virtual propia, eliminando o protegiendo la carpeta profesor para que los alumnos no puedan acceder fácilmente a ella.

2.8.3. Usabilidad

Es una herramienta pensada sobre todo para ser utilizada en los entornos de laboratorios que ofrece. Para ello primero se ha de buscar que laboratorio queremos ejecutar, utilizando el comando labtainer en la consola nos sale la lista de todos los laboratorios ya implementados.

Para iniciar un laboratorio se hace con el comando labtainer <nombreLaboratorio>. Una vez se ejecuta el comando descargará el laboratorio, con las máquinas necesarias y únicamente se tiene que indicar el correo electrónico del estudiante, y dar a *enter* para iniciar el laboratorio.

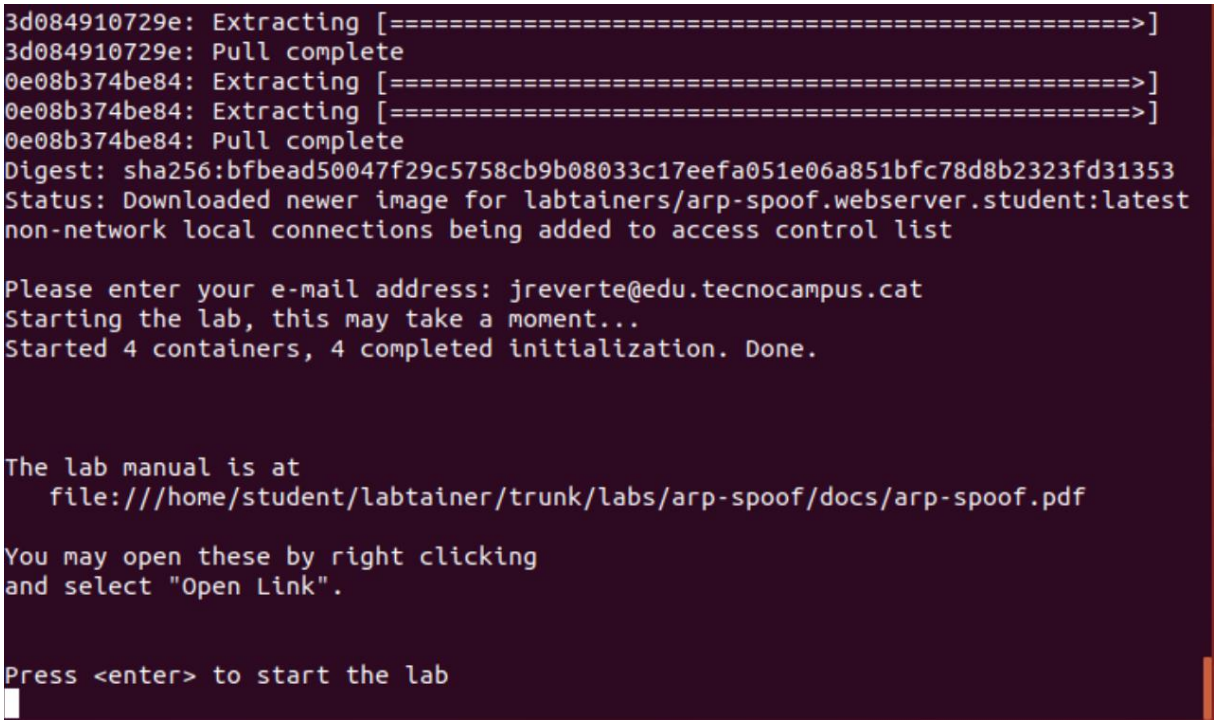

Ilustración 17 Comandos inicio Labtainers

Cada laboratorio incluye las máquinas necesarias para hacer el laboratorio y un pdf explicándolo.

En el caso del laboratorio probado para la usabilidad básica la topología levantada es la siguiente:

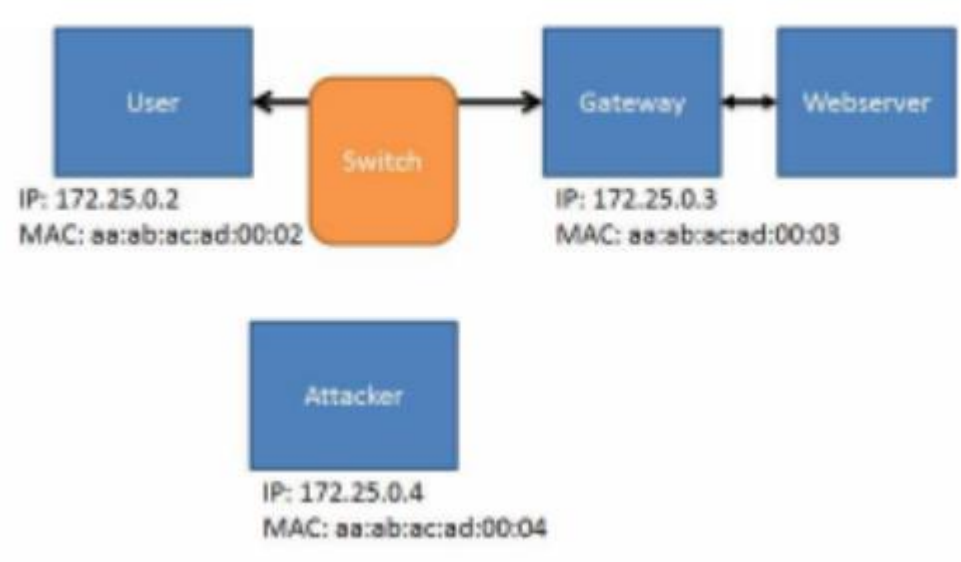

Ilustración 18 Topología básica Labtainers

2.8.4. Comprobaciones

Interfaz Gráfica	No	El alumno puede ver visualmente la topología desplegada pero no tiene interfaz gráfica más allá de los comandos.
Instalación sencilla	Sí	-
Funcionalidades básicas	Sí	-
No necesita VM	Sí	-

Tabla 7 Comprobaciones Labtainer

2.9. Cloonix

2.9.1. Introducción

Se definen como una herramienta que ayuda en la emulación de redes hecha mediante máquinas virtuales y conexiones emuladas. Tiene dos objetivos:

- El primero es representar gráficamente ítems complejos como qemu-kvm y openswitch mediante ítems visuales llamados guitems.
- El segundo es generar una comunidad que comparta sus creaciones dentro del ecosistema Cloonix.

Se trata de un set de elementos de *software* en C, como: qemu-kvm, openswitch, dpdk, spice, wireshark y openssh.

Cloonix gestiona *clusters* de máquinas virtuales, el servidor es controlado por clientes que actúan a distancia o en el mismo *host*. Todos los clientes cloonix se llaman cloonix_xxx xxx puede ser uno de los siguientes: gui, cli, ssh, scp, dta, ice, mon, osh, ocp, ovs, xwy.

El servidor cloonix se lanza en el servidor y se denomina cloonix_net, crea y controla guitems que se ejecutan en el servidor y se visualizan en la gui lanzado con cloonix_gui.

El guitem principal es la máquina virtual kvm, alrededor de este guitem hay otros 6: lan, nat, tap, phy, d2d, a2b.

Las máquinas virtuales kvm pueden conectarse mediante comandos cloonix, las conexiones emuladas entre máquinas virtuales se basan en openvswitch y dpdk y son imaginadas por el guitem lan.

2.9.2. Descarga e instalación

Durante el desarrollo de este proyecto la versión de Cloonix ha cambiado a la versión 33. Por lo que en este apartado se mostrara como instalar el programa y como gestionar para aumentar la versión a la más moderna.

Para obtener Cloonix se ha de instalarlo desde el git del proyecto con los siguientes comandos:

```
cd ~
git clone https://github.com/clownix/cloonix.git
cd cloonix
sudo ./install_depends build
./doitall
```

En caso de tener una versión anterior disponible ya instalada se ha de utilizar los siguientes comandos:

cd ~/cloonix

git pull

sudo ./allclean

./doitall

Es importante activar KVM, esto se puede hacer mediante dos comandos, aunque se van a necesitar permisos de administrador para ejecutarlos:

Sudo modprobe kvm-intel nested=1

Sudo chmod 666 /dev/kvm

2.9.3. Usabilidad

Una vez instalada, se puede arrancar una nueva topología ejecutando: cloonix_startnet <nombreTopologia>, y para abrir la interfaz gráfica: cloonix_graph <nombreTopologia>.

Se abre entonces la interfaz gráfica donde pulsando clic derecho se puede ir añadiendo diferentes elementos, diferentes guitems.

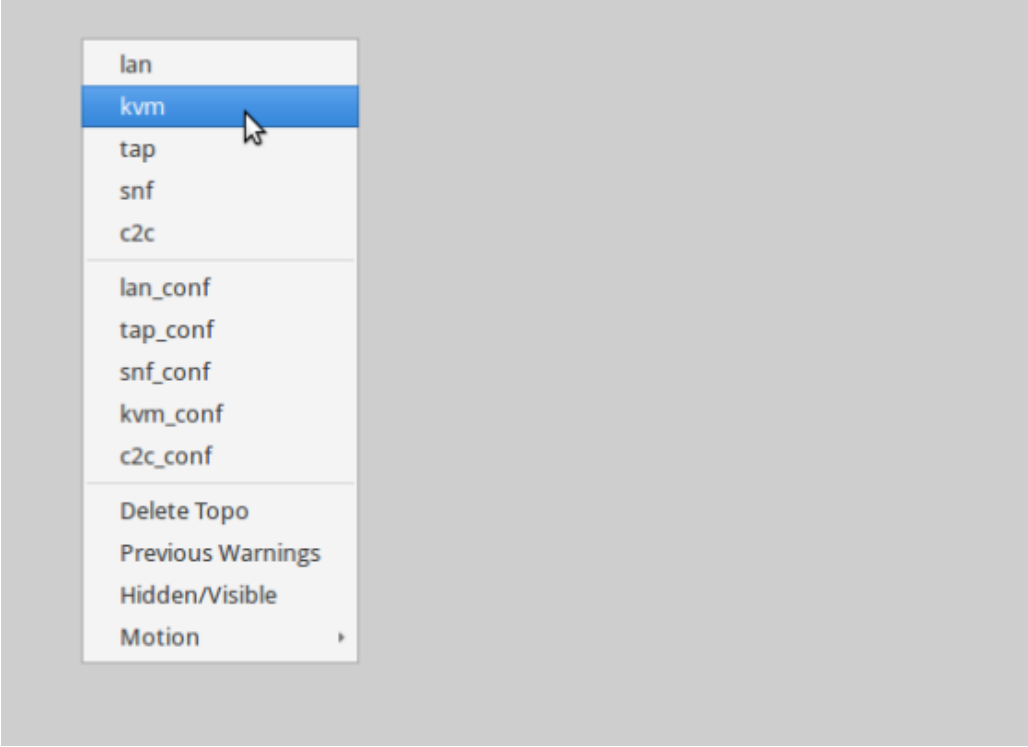

Ilustración 19 Cloonix Página de inicio

Se ha probado una topología sencilla con tres nodos y un *sniffer* en medio.

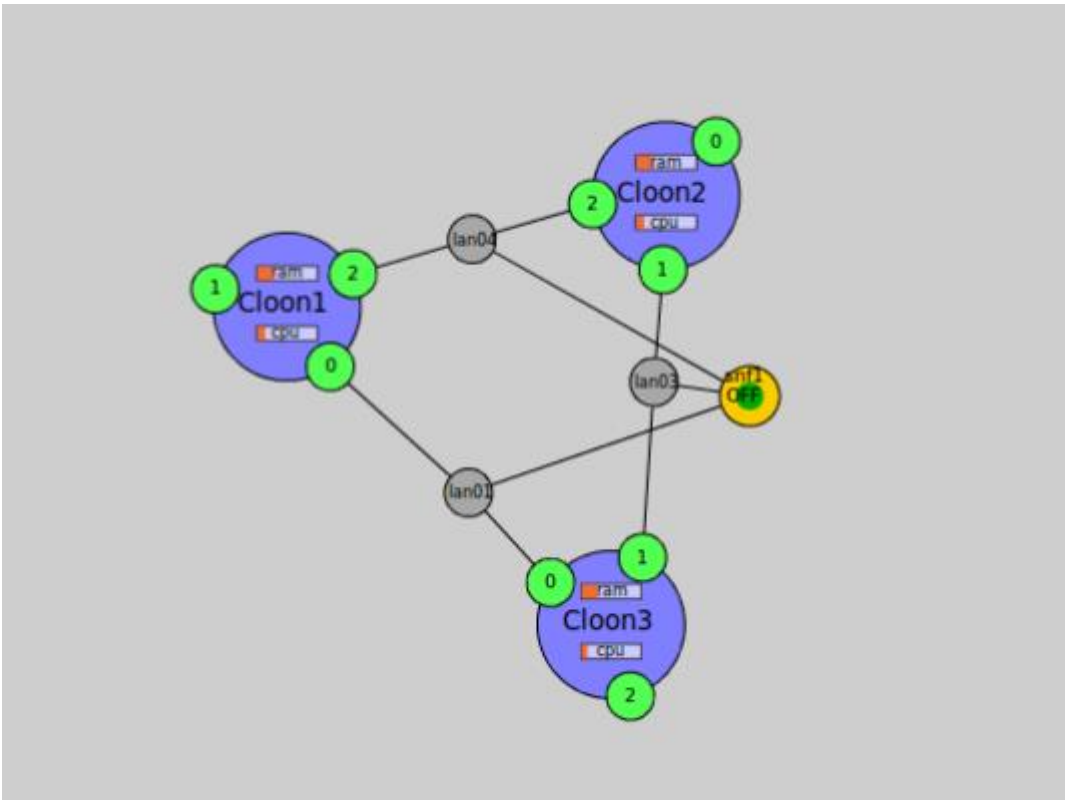

Ilustración 20 Cloonix topología básica

A parte de la creación mediante la interfaz gráfica, se pueden crear las mismas topologías mediante comandos, a continuación, se muestran los comandos necesarios para crear la topología mostrada arriba:

cloonix_ctrl nemo add kvm Cloon1 1000 1 classic,classic,classic jessie.qcow2 --balloon

cloonix_ctrl nemo add lan eth Cloon1 0 lan01

cloonix_ctrl nemo add lan eth Cloon1 2 lan04

cloonix_ctrl nemo add kvm Cloon2 1000 1 classic,classic,classic jessie.qcow2 --balloon

cloonix_ctrl nemo add lan eth Cloon2 1 lan03

cloonix_ctrl nemo add lan eth Cloon2 2 lan04

cloonix_ctrl nemo add kvm Cloon3 1000 1 classic,classic,classic jessie.qcow2 --balloon

cloonix_ctrl nemo add lan eth Cloon3 0 lan01

cloonix_ctrl nemo add lan eth Cloon3 1 lan03

cloonix_ctrl nemo add snf snf1 classic

cloonix_ctrl nemo add lan sat snf1 lan03

cloonix_ctrl nemo add lan sat snf1 lan01

cloonix_ctrl nemo add lan sat snf1 lan04

cloonix_ctrl nemo cnf lay stop

cloonix_ctrl nemo cnf lay width_height 539 403

cloonix_ctrl nemo cnf lay scale 150 113 539 403

cloonix_ctrl nemo cnf lay abs_xy_kvm Cloon3 188 225

cloonix_ctrl nemo cnf lay abs_xy_kvm Cloon1 25 66

cloonix_ctrl nemo cnf lay abs_xy_kvm Cloon2 215 11

cloonix_ctrl nemo cnf lay abs_xy_lan lan03 209 104

cloonix_ctrl nemo cnf lay abs_xy_lan lan01 116 159

cloonix_ctrl nemo cnf lay abs_xy_lan lan04 118 33

2.9.4. Comprobaciones

Interfaz Gráfica	Sí	-
Instalación sencilla	Sí	-
Funcionalidades básicas	No	No se pueden realizar la mayoría de las topologías seleccionadas.
No necesita VM	No	En este caso sí que necesita levantar VM por cada elemento creado.

Tabla 8 Comprobaciones Cloonix

2.10. Eve-ng

2.10.1. Introducción

Se ha eliminado directamente ya que en la actualidad ha dejado de ser una herramienta *open source* como tal, y por lo tanto no cumple con uno de los requisitos básicos.

Existe una versión *community* gratis que no cumple los requisitos para ser una herramienta *open source*.

2.11. Kathara

2.11.1. Introducción

Busca probar soluciones de red, con escenarios realistas mediante el lenguaje P4 para implementar NFV.

Está basado en Docker, permite la implementación de imágenes de *SDN*, *switches*, *routers*, DNS, servidores web y mucho más.

En su página web enseñan la arquitectura que hace funcionar a este programa:

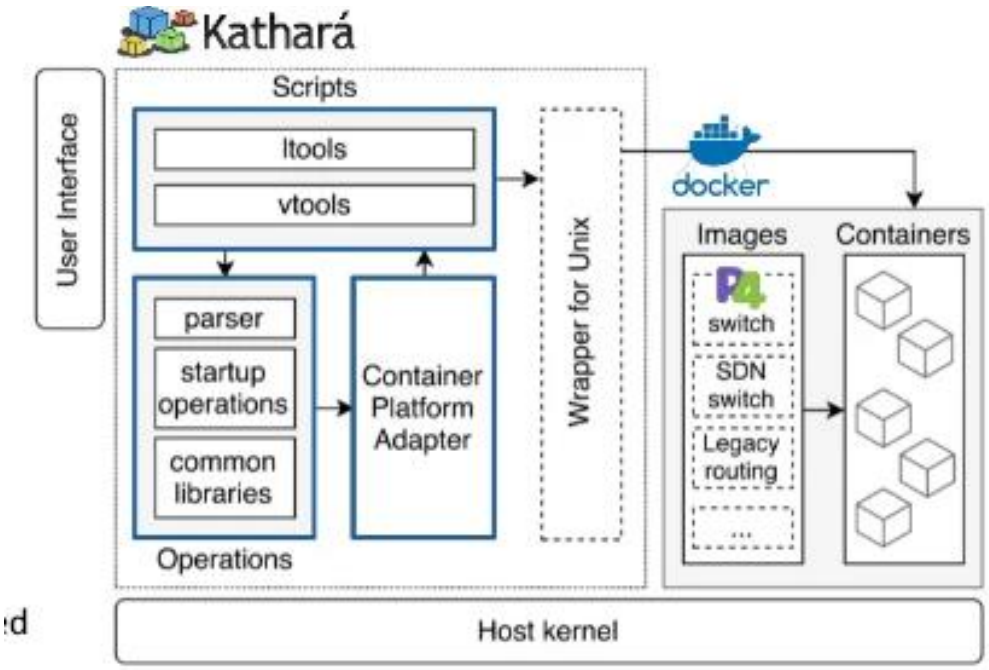

Ilustración Arquitectura oficial Kathara

Vemos conceptos ya tratados como kernel para hablar con el *host* y Docker para gestionar los contenedores. El concepto más nuevo que implementa es el uso del lenguaje P4 para la generación de las imágenes de los diferentes elementos de la red simulada / emulada.

2.11.2. Descarga e instalación

Kathara se puede utilizar directamente desde Windows. Para instalar la aplicación se va a la página web https://www.kathara.org/download.html y se pulsa en la descarga para Windows.

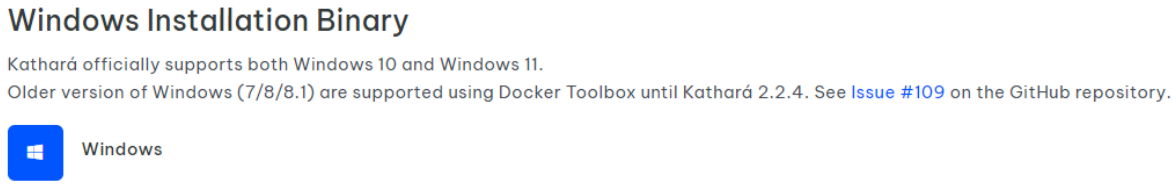

Ilustración 21 Instalación del binary de windows

Se ejecuta el archivo instalado y se siguen todos los pasos con tal de instalarlo en el sistema.

Para el correcto funcionamiento en Windows es imprescindible instalar Docker, se puede hacer a través del siguiente enlace: https://docs.docker.com/desktop/windows/install/.

2.11.3. Usabilidad

Antes de continuar, es mejor comprobar que se haya instalado correctamente Kathara en el sistema, para comprobarlo se utiliza:

```
C:\Users\jaume>kathara check
INFO - Installing Kathara Network Plugin...
INFO - Kathara Network Plugin installed successfully!
*       Current Manager is: Docker (Kathara)
*       Manager version is: 20.10.16
*       Python version is: 3.9.5 (tags/v3.9.5:0a7dcbd, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)]
*       Kathara version is: 3.5.0
*       Operating System version is: Windows-10-10.0.19044-SP0
*       Trying to run `Hello World` container...
INFO - Pulling image `kathara/quagga:latest`... This may take a while.
Deploying devices...|##########################################################################################| 1/1
*       Container run successfully.
Deleting devices...|##########################################################################################| 1/1
```

Ilustración 22 Comprobar instalación kathara

Para la muestra de configuración básica se ha elegido levantar un simple ordenador, debajo se muestra los comandos necesarios para crearlo y el *output* que se obtiene:

```
C:\Users\jaume>kathara vstart -n pc1 --eth 0:A
INFO - ====================== Starting Device ======================
Deploying collision domains...|##########################################################################################| 1/1
Deploying devices...|#####################################################################################################| 1/1

C:\Users\jaume>kathara vclean -n pc1
Deleting devices...|######################################################################################################| 1/1
Deleting collision domains...|############################################################################################| 1/1
INFO - Device `pc1` deleted successfully!
```

Ilustración 23 Creación topología básica kathara

Al crear cualquier máquina, se abre un Shell donde se pueden introducir comandos para gestionar y configurar el dispositivo.

```
root@pc1: /
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 26:c1:6f:b3:42:2f  txqueuelen 1000  (Ethernet)
        RX packets 12  bytes 976 (976.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Ilustración 24 Consola elemento desplegado kathara

2.11.4. Elementos disponibles

Utiliza imágenes propias cargadas dentro de contenedores, de los siguientes elementos: *Routers*, PC, *Switch* y servidores. A parte permite el uso de imágenes comerciales de productos *legacy*.

2.11.5. Comprobaciones

Interfaz Gráfica	No	La creación y configuración es mediante código. La interfaz únicamente permite ver la topología creada.
Instalación sencilla	Sí	-
Funcionalidades básicas	Sí	-
No necesita VM	Sí	-

Tabla 9 Comprobaciones Kathara

2.12. Mininet

2.12.1. Introducción

Tiene una particularidad, está ideada para desplegar aplicaciones en ella. Genera una red virtual realista que permite el despliegue de aplicaciones dentro suyo. No es el único programa que las permite, pero sí es el único ideado para ello.

2.12.2. Descarga e instalación

Dentro del siguiente enlace: http://mininet.org/download/, se puede instalar una máquina virtual con Mininet preconfigurado. Se ha de importar dentro de VirtualBox y ya se puede empezar a utilizar.

2.12.3. Usabilidad

Para utilizar la herramienta de una forma más sencilla es recomendable conectarse mediante SSH a la máquina.

Para este ejemplo, se ha utilizado el mismo ejemplo de la documentación oficial:

```
#!/usr/bin/python


from mininet.topo import Topo

from mininet.net import Mininet

from mininet.util import dumpNodeConnections

from mininet.log import setLogLevel


class SingleSwitchTopo(Topo):

    "Single switch connected to n hosts."

    def build(self, n=2):

        switch = self.addSwitch('s1')

        # Python's range(N) generates 0..N-1
```

```
        for h in range(n):
            host = self.addHost('h%s' % (h + 1))
            self.addLink(host, switch)


def simpleTest():
    "Create and test a simple network"
    topo = SingleSwitchTopo(n=4)
    net = Mininet(topo)
    net.start()
    print( "Dumping host connections" )
    dumpNodeConnections(net.hosts)
    print( "Testing network connectivity" )
    net.pingAll()
    net.stop()


if __name__ == '__main__':
    # Tell mininet to print useful information
    setLogLevel('info')
    simpleTest()
```

Para levantar la red únicamente se tiene que ejecutar el *script* de arriba.

2.12.4. Comprobaciones

Interfaz Gráfica	No	La creación y configuración es mediante código. La interfaz únicamente permite ver la topología creada.
Instalación sencilla	Sí	-

Funcionalidades básicas	Sí	-
No necesita VM	Sí	-

Tabla 10 Comprobaciones Mininet

Durante la fase final del análisis esta solución ha sido eliminada ya que se considera poco relevante para el estudiante tener que aprender un lenguaje de programación, Python, para poder aprender sobre redes.

2.13. Core

2.13.1. Introducción

Se trata de una herramienta para construir redes virtuales, parte de un emulador y por tanto genera una representación virtual real de los componentes de la red.

Permite conectar estos dispositivos virtualizados con dispositivos reales.

2.13.2. Descarga e instalación

En el proyecto se ha utilizado la instalación automática ya que es la más sencilla, pero a continuación se expondrán y explicarán las diferentes alternativas.

2.13.2.1. Método automático

Es el método más sencillo y únicamente se ha de clonar el repositorio oficial de Github y luego utilizar inv install, este último paso va a variar según el OS del host.

```
# clone CORE repo
git clone https://github.com/coreemu/core.git
cd core

# install dependencies to run installation task
./setup.sh

# run the following or open a new terminal
source ~/.bashrc

# Ubuntu
inv install

# CentOS
inv install -p /usr
```

2.13.2.2. Mediante DockerFile

Se puede instalar y lanzar como contenedor de Docker. Esto es especialmente útil para no depender del OS del host donde se descarga el programa.

```
# clone core
git clone https://github.com/coreemu/core.git
cd core

# build image
sudo docker build -t core .

# start container
sudo docker run -itd --name core -e DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix:rw --
privileged core

# enable xhost access to the root user
xhost +local:root

# launch core-gui
sudo docker exec -it core core-gui
```

2.13.3. Usabilidad

Para la prueba de usabilidad se ha creado una topología simple utilizando dos *hosts* y dos *routers* interconectados.

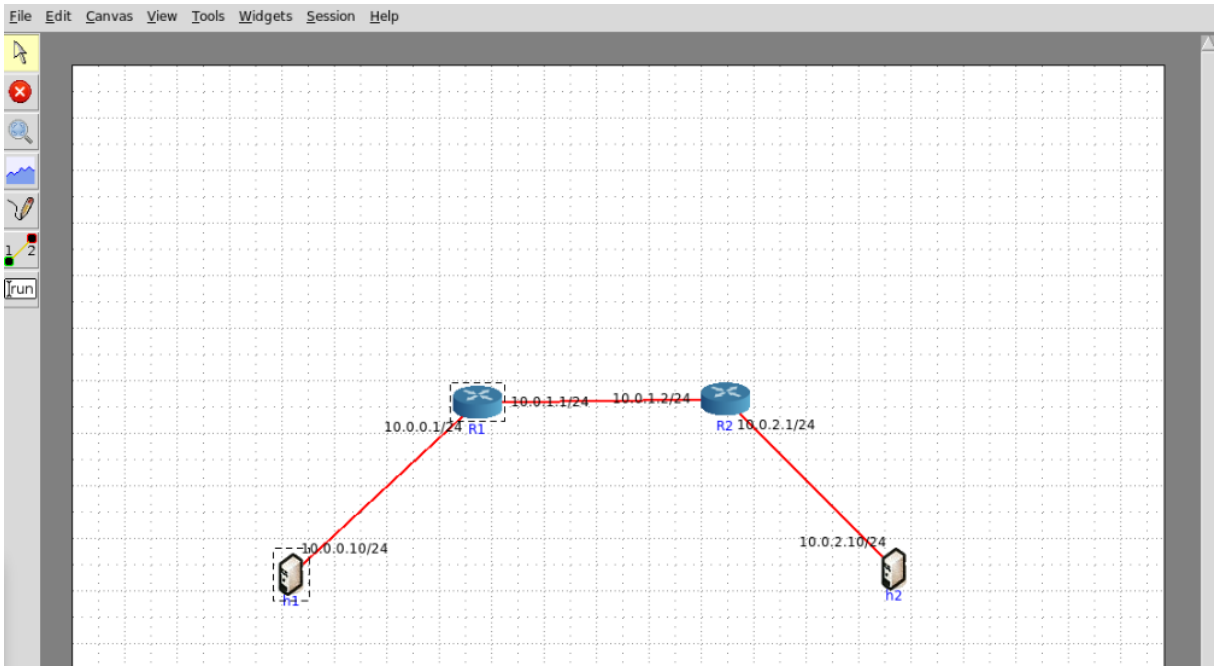

Ilustración 27 Usabilidad básica Core

Es fácil de utilizar, el único inconveniente es que hay muchos servicios base que no vienen preinstalados y es el usuario que ha de gestionar su correcta instalación y configuración.

2.13.4. Elementos disponibles

Existen una gran variedad de opciones diferentes de dispositivos para emular redes dentro del entorno de Core.

Core Nodes: son los dispositivos estándares de core, estos ocupan muy poco espacio de almacenamiento y están optimizados para la emulación de redes.

Docker Nodes: Los nodos Docker proporcionan una comodidad para ejecutar nodos utilizando imágenes y sistemas de archivos predefinidos que los nodos core no proporcionan. Se añade de esta manera nuevos dispositivos para emular, con el coste de ser menos optimizados que los propios de la plataforma core.

LXC Nodes: Funciona similar a Docker Nodes. Los nodos LXC proporcionan una comodidad para ejecutar nodos utilizando imágenes y sistemas de archivos predefinidos que los nodos core no proporcionan.

2.13.5. Comprobaciones

Interfaz Gráfica	Sí	-
Instalación sencilla	No	Aunque la instalación mostrada parezca sencilla, en el equipo en el que se ha probado la herramienta se tuvieron que realizar muchos más pasos que los expuestos con tal de descargarla.
Funcionalidades básicas	Sí	-
No necesita VM	Sí	-

Tabla 11 Comprobaciones Core

2.14. Netsim

2.14.1. Introducción

Se trata de una herramienta militar y con un marcado carácter comercial. Por motivos no técnicos se ha decidió no continuar con el análisis.

3. Objetivos y alcance

El principal objetivo del proyecto es la búsqueda y selección de una solución en la simulación de redes para el ámbito educacional, y que esta se adapte a las características de este ámbito.

Por lo tanto, debemos:

- Buscar diferentes alternativas *open source* para la simulación y/o emulación de redes en un entorno académico.

Los objetivos secundarios del proyecto son:

- Analizar funcional y técnicamente las diferentes alternativas.
- Estudiar y determinar varios modelos de redes que permitan hacer los diferentes análisis.
- Generar lista de KPI con los datos extraídos en los diferentes análisis.
- Crear un *ranking* basado en las pruebas realizadas.
- Generar un manual de instalación y uso sencillo para las plataformas utilizadas.

Se trabaja con tres clientes: el centro educativo, el profesorado y el alumnado.

El centro educativo busca ofrecer un excelente servicio a sus alumnos ahorrando el máximo de inversión posible. En base a esto, vemos que la solución resultante del ranking favorece enormemente a estos dos objetivos.

El profesor necesita una herramienta que le permita de forma didáctica y sencilla explicar el funcionamiento de las redes a sus alumnos, pero a la vez debe tener también como opción, a partir de la herramienta, de que los alumnos puedan exponer los conocimientos de redes de forma práctica y no solo teórica.

El alumno busca tener un instrumento que facilite su acceso al conocimiento, así que, el uso de una herramienta de simulación le ayudará a ejecutar y aprovechar esos conocimientos teóricos adquiridos de redes reforzándose con un apartado más práctico. A parte ha de ser una solución con una curva de aprendizaje sencilla ya que el objetivo es que aprendan a gestionar redes, no que se especialicen en una herramienta compleja.

4. Análisis de referentes

Se han encontrado 3 fuentes diferentes que tratan este tema, aunque con un enfoque diferente al que se toma en este proyecto.

- **Comparison of Containerization and Virtualization in Cloud Architectures**
- **Performance Evaluation of MPLS in a Virtualized Service Provider Core**
- **Assessing modelling and visualisation capabilities of modelling tools: limitations and gaps of open-source modelling tools**

5. Metodología

Para elegir la metodología del proyecto es imprescindible valorar entre la utilización de un método agile o waterfall. Agile es la mejor alternativa, ya que permite hacer mejoras incrementales en el proyecto a medida que transcurre el tiempo.

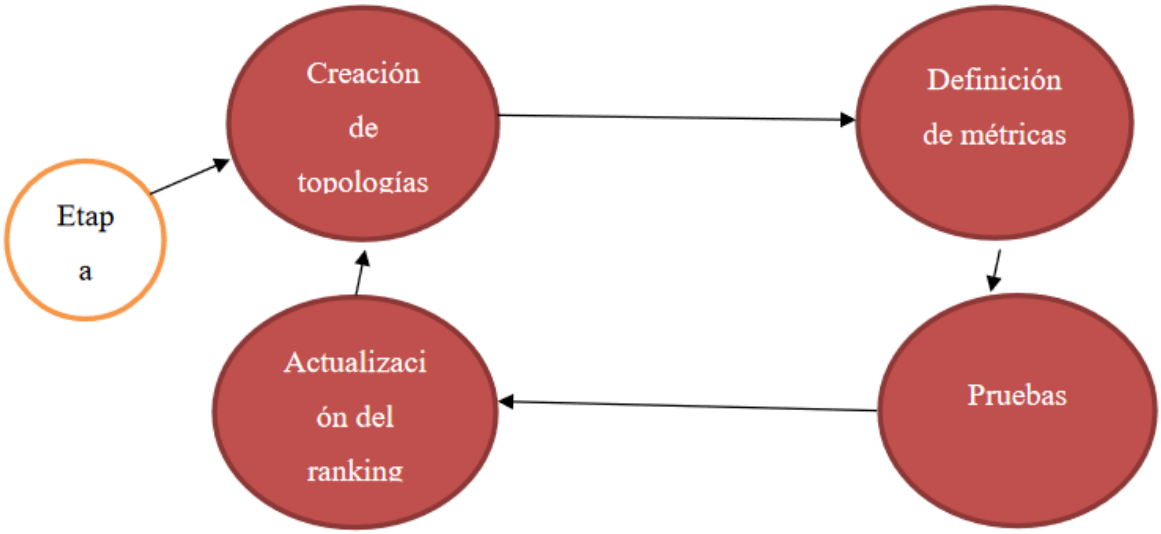

Ilustración 25Proceso de iteración inicial

En la etapa inicial se pretende hacer un análisis de mercado de las diferentes propuestas open source que haya existentes. Este análisis tiene como objetivo observar el background no técnico, una visión más histórica de las diferentes soluciones que más adelante se van a analizar técnicamente. Esta etapa rompe con la metodología agile a utilizar, pero marca el inicio del proyecto y es importante para situar al lector y presentar los diversos programas.

Una vez se ha hecho el análisis de las diferentes alternativas que existen, se va a crear la primera topología sencilla de red para probar con cada uno de los programas, a la par que se deciden las métricas a analizar en esta ronda de pruebas. Después de cada una de las pruebas, se anotarán los resultados de las métricas obtenidas con tal de comenzar a generar los datos para el ranking final.

Cuando finaliza el análisis, se itera, creando un nuevo análisis con una topología diferente y añadiendo o quitando métricas a analizar. Se pretende en cada iteración los datos que se extraen con tal de tener al final del proyecto un ranking lo más certero posible.

Esta metodología es la mejor ya que permitirá hacer pruebas hasta el último momento ya que en cada iteración se pretende tener preparado un entregable, es decir, en cada iteración se va a ir generando un ranking actualizado con la nueva información extraída.

5.1. Proceso de selección de plataforma

Con tal de elegir las herramientas que se han implementado en la fase de desarrollo se ha creado un árbol de decisiones para ilustrar el proceso de decisión:

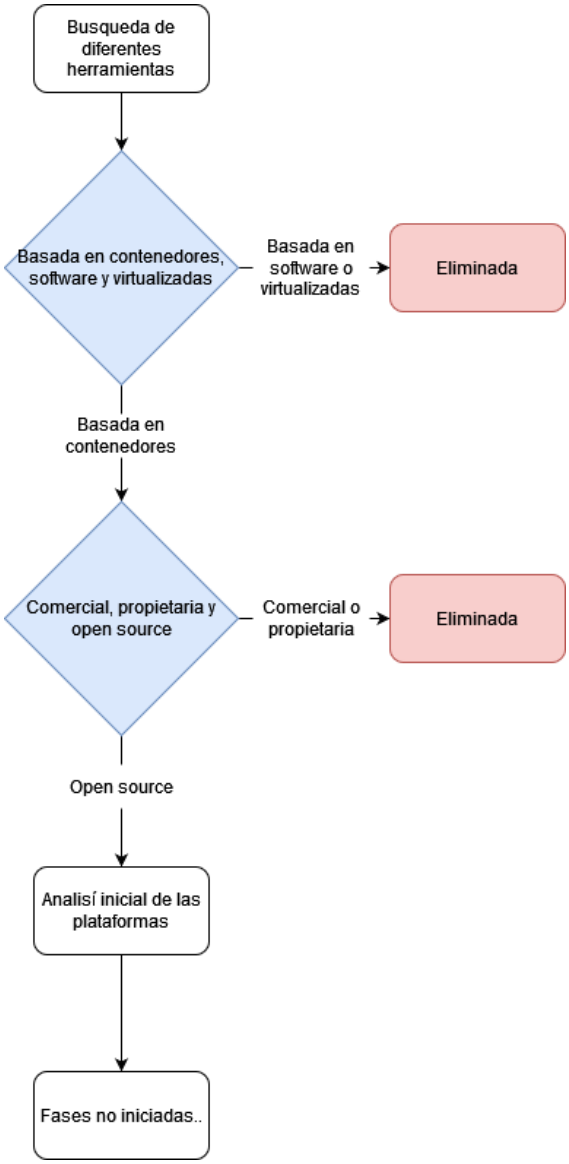

Ilustración 26 Proceso de selección de programas

Se observa por lo tanto que los requerimientos iniciales para añadir una plataforma al estudio es que esté basada en contenedores, por las ventajas ya expuestas que presenta este tipo de tecnología y por el dominio / uso público sin restricciones.

Este último filtro / requerimiento no es arbitrario, corresponde a que debe ser una plataforma que sea fácil de utilizar dentro de un entorno lectivo, además, recurrir a plataformas comerciales, en muchos casos, puede ser algo complejo.

Utilizando estos filtros en las diferentes plataformas descritas en el subapartado 2.3 se analizarán durante el proyecto aquellas plataformas que hayan pasado la criba.

La lista resultante de esta preselección se puede encontrar a partir del apartado 2.7 Imunes.

Para acabar seleccionando únicamente aquellas que se adaptan realmente al entorno académico donde se van a utilizar, se ha realizado una serie de comprobaciones que han de cumplir (estas comprobaciones están definidos en los apartados de cada herramienta analizada):

Interfaz gráfica: en un entorno educativo, es importante disponer de una interfaz gráfica sencilla para el usuario. Ya que permite tanto a alumno como a profesor determinar de forma visual que la tarea ha sido realizada, a la vez que se hace más ameno para el estudiante a la hora de realizar la tarea. Es algo importante, aunque no se presenta como algo imprescindible ya que no imposibilita el uso de la aplicación si no tiene interfaz.

Instalación sencilla: cuando una herramienta se la han de descargar todos los alumnos de un aula, pero únicamente hay un profesor, es un requisito necesario que la herramienta sea sencilla de instalar por parte del estudiante. Ha de requerir la mínima intervención del profesor.

Cumple las funcionalidades básicas: No se entrará en detalle en este apartado, pero un requisito fundamental es que sea capaz de mostrar a los alumnos no solo como funciona un entorno de red, sino que permita crear diferentes servicios de red básicos.

No levanta máquinas virtuales por cada elemento de la red: Es importante para el rendimiento del equipo que la herramienta seleccionada no levante una máquina virtual, ya que esto acarrea altos costes de rendimiento. Esto haría que muchos estudiantes no pudieran utilizar la herramienta al ser demasiado exigente con los requerimientos de los equipos.

5.2. Servicios para analizar

Una vez se hayan probado todas las diferentes plataformas y hayan pasado todas las comprobaciones, todas las herramientas que hayan pasado a la siguiente fase serán probadas y se comprobará que puedan simular los siguientes servicios: Static Routing, ARP, Web Server (Aquí entran los diferentes servicios de un servidor web), Load Balancer, DNS, RIP y OSPF.

6. Desarrollo

Se empieza la selección de plataforma con la siguiente lista de programas:

Antidote	Boson Network Simulator	Cisco Modelling Lab	Cloonix
dCloud	Dynamips	GNS3	Imunes
Kathara	Mininet	Omnet++	Packet Tracer
Virt	Riverbed Modeller	QualNet	Nab
Yans	Marionnet	Line	Labtainers
Netsim	Lstacker	Core	Psimulator
DockEmu	Containerlab	J-Sim	Educational Network Simulator

Tabla 12 Lista inicial plataformas

El primer filtro ha sido eliminar aquellas que tienen licencias comerciales. Se han hecho las correspondientes búsquedas a las páginas webs oficiales de cada herramienta para determinar si se tratan de herramientas *open source* o no. La lista resultante es la siguiente:

Antidote	Boson Network Simulator	Cisco Modelling Lab	Cloonix
dCloud	Dynamips	GNS3	Imunes
Kathara	Mininet	Omnet++	Packet Tracer
Virt	Riverbed Modeller	QualNet	Nab
Yans	Marionnet	Line	Labtainers
Netsim	Lstacker	Core	Psimulator
DockEmu	Containerlab	J-Sim	Educational Network Simulator

Tabla 13 Lista después filtro comerciales

Después se ha eliminado aquellas que no se hayan actualizado últimamente. Esto se ha hecho ya que no se quiere utilizar una herramienta que este *deprecated* y no tenga ningún tipo de soporte ni tenga actualizaciones.

Antidote			Cloonix
dCloud	Dynamips	GNS3	Imunes

Kathara	Mininet	Omnet++	
			Nab
Yans	Marionnet	Line	Labtainers
Netsim	Lstacker	Core	Psimulator
DockEmu	Containerlab	J-Sim	Educational Network Simulator

Tabla 14 Lista después filtro deprecated

Después de eliminar aquellas que no pasan el filtro del tiempo, en la lista únicamente aparecen aquellas herramientas que se han instalado y probado durante la duración de este proyecto.

La única herramienta que aparece a continuación y no se ha llegado a probar es la de GNS3 debido a que durante el estudio se obtuvo la idea errónea de que solo aceptaba máquinas virtuales de cisco, comerciales.

Containerlab	Labtainers	GNS3	Imunes
Kathara	Mininet	Omnet++	Cloonix
Netsim		Core	Educational Network Simulator

Tabla 15 Lista herramientas probadas

Una vez se han testeado todas y cada una de las plataformas de arriba, se ha ido eliminando aquellas que analizando su funcionamiento y/o capacidades se ha comprobado que no cumplían las condiciones necesarias para ser seleccionada como la herramienta para la educación.

Omnet++, Netsim y Eve-ng han sido eliminadas antes de realizar las pruebas, en la fase de introducción ya que al analizar la información de sus respectivas documentaciones se ha determinado que no tenía sentido continuar con su análisis.

De las pruebas se ha extraído la siguiente información clave:

	Interfaz Gráfica	Instalación sencilla	Funcionalidades básicas	No necesita VM
Imunes	✓	✓	✓	✓
GNS-3	✓		✓	
ContainerLab		✓	✓	✓
ENS	✓	✓	✓	✓
Labtainers		✓	✓	✓
Cloonix	✓	✓		
Kathara		✓	✓	✓

Mininet		✓	✓	✓
Core	✓		✓	✓

Tabla 16 Comprobaciones de todas las plataformas

El primer filtro que se ha pasado a la lista anterior ha sido si la herramienta tenía 2 o más elementos faltantes de las comprobaciones. Eliminando así a Cloonix y GNS3.

De todas las restantes se han realizado las pruebas de Static Routing, ARP, DHCP, Load Balancer, DMZ, DNS, Mail, Web, RIP, OSPF, Ping y Wireshark. De estas pruebas y teniendo en consideración el objetivo educacional del proyecto se han escogido como ganadoras a:

- Imunes
- Educational Network Simulator
- Labtainers

A continuación, se muestran las pruebas hechas en estas tres plataformas, se ha decidido no añadir todas las pruebas realizadas y solo las finalistas ya que no aportarían nada al objetivo del proyecto de seleccionar la herramienta más idónea para la educación en redes. Es más interesante presentar aquellas que se han decidido como ganadoras.

Las pruebas realizadas en otras plataformas se podrán ver en la carpeta adjunta en la nube si se quieren más referencias.

6.1. Static Routing / ARP

6.1.1. Imunes

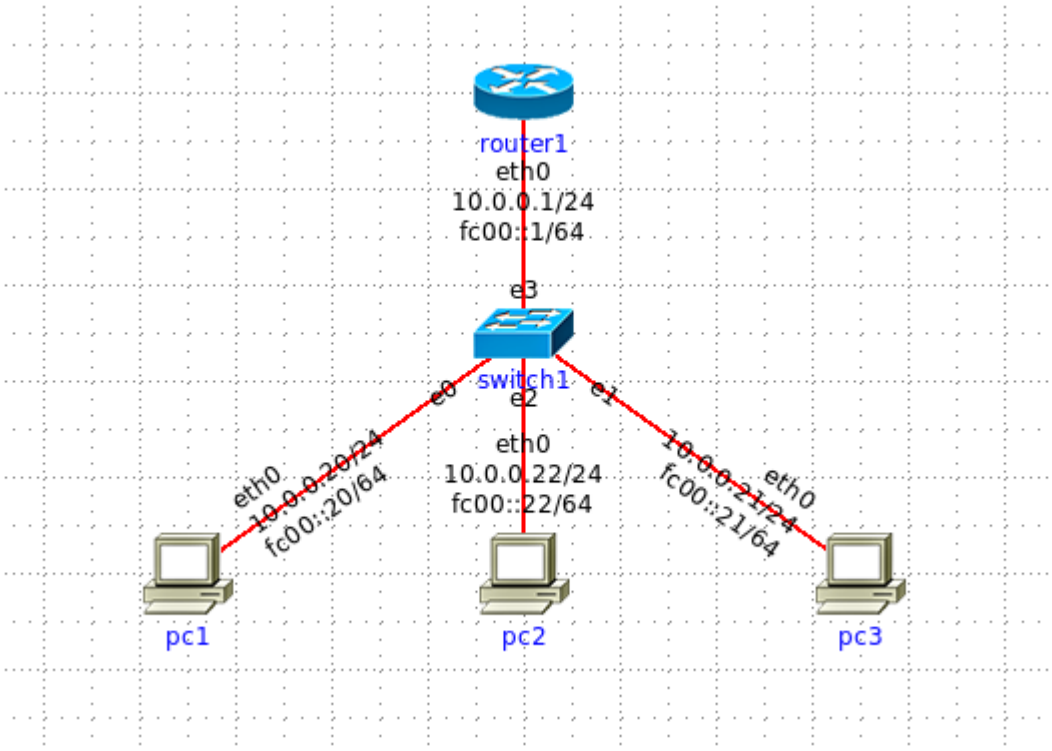

Ilustración 27 Static Routing Imunes

6.1.2. Educational Network Simulator

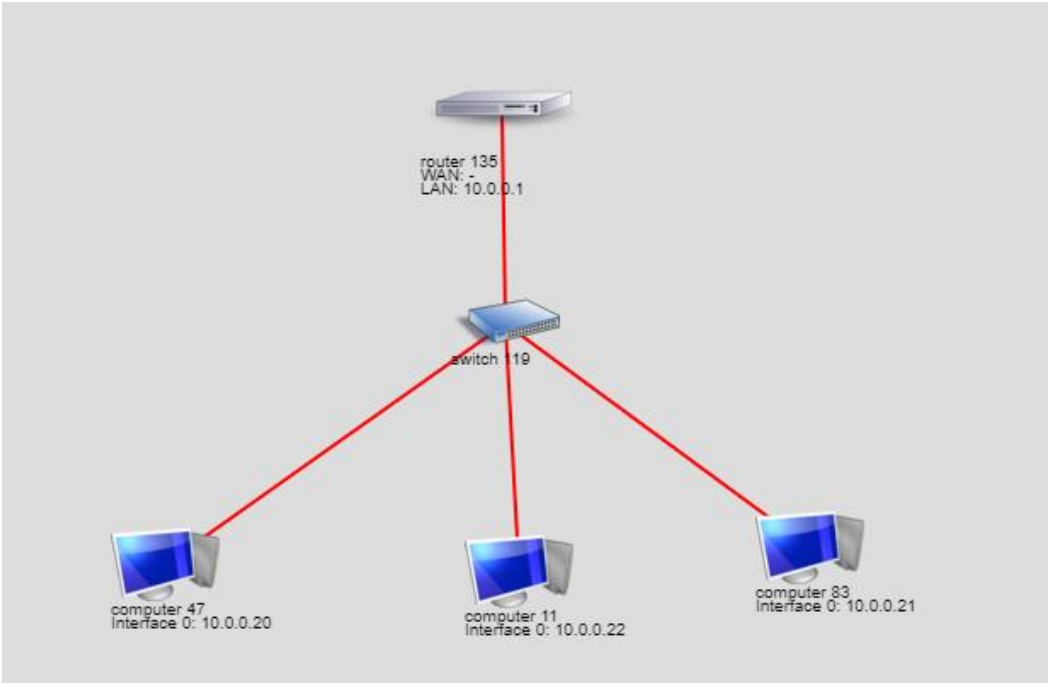

Ilustración 28 Static Routing ENS

6.2. DHCP

6.2.1. Imunes

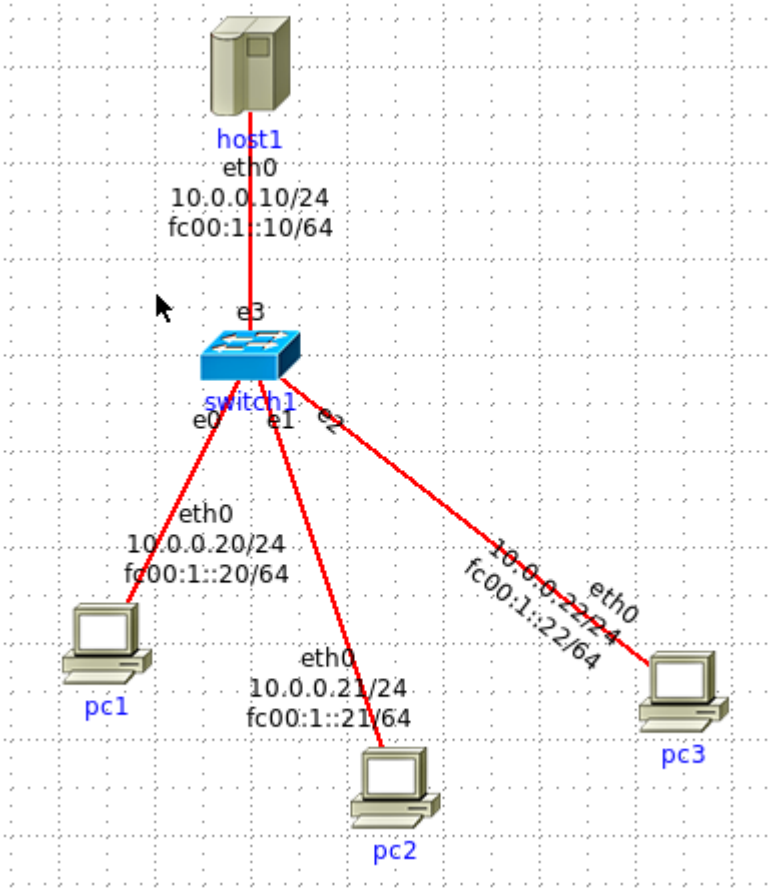

Ilustración 29 DHCP Imunes

6.2.2. Educational Network Simulator

No tiene las capacidades técnicas para este servicio.

6.2.3. Labtainer

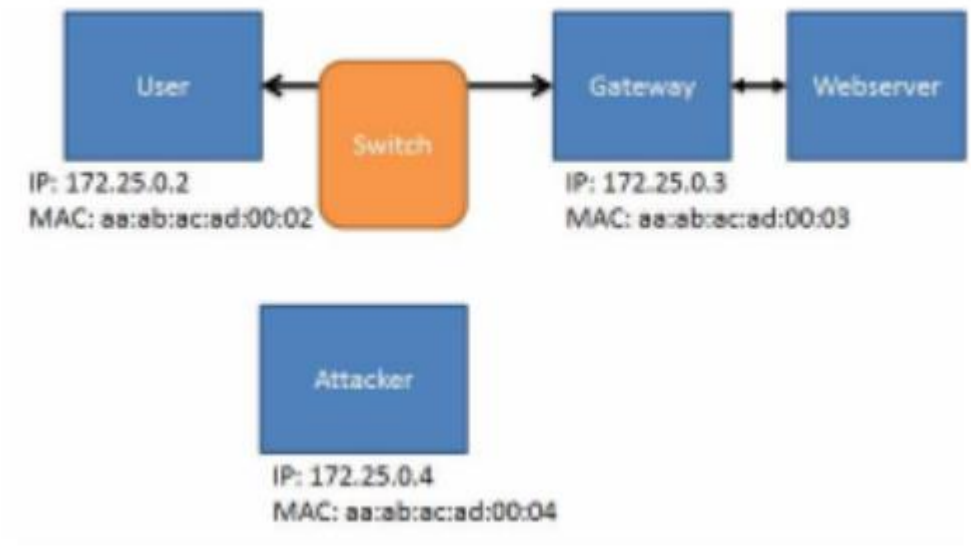

Ilustración 30 DHCP Labtainer

6.3. Load Balancer

6.3.1. Imunes

No tiene las capacidades técnicas para este servicio.

6.3.2. Educational Network Simulator

No tiene las capacidades técnicas para este servicio.

6.3.3. Labtainer

No tiene las capacidades técnicas para este servicio.

6.4. DMZ + DNS + Mail + WEB

6.4.1. Imunes

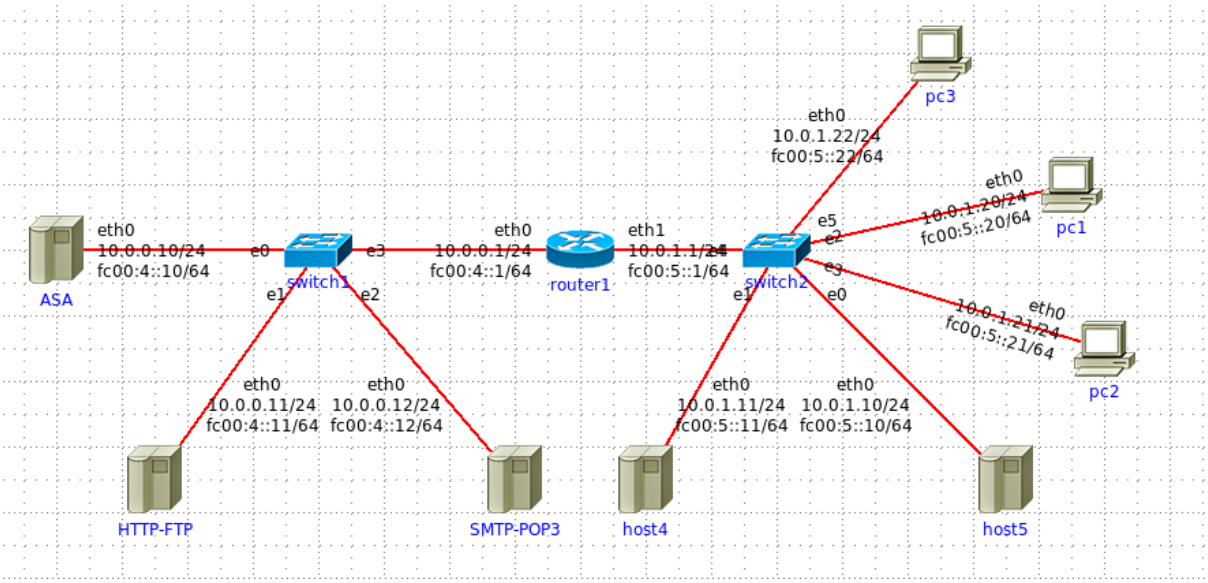

Ilustración 31 DMZ + DNS + Mail + WEB Imunes

6.4.2. Educational Network Simulator

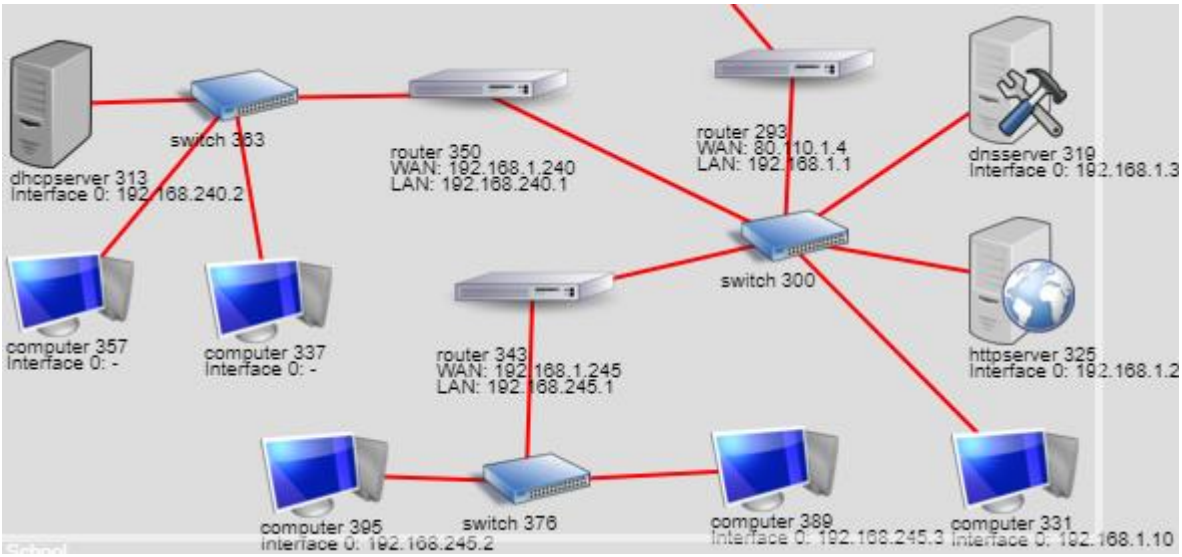

Ilustración 32 DMZ + DNS + Mail + WEB ENS

6.4.3. Labtainer

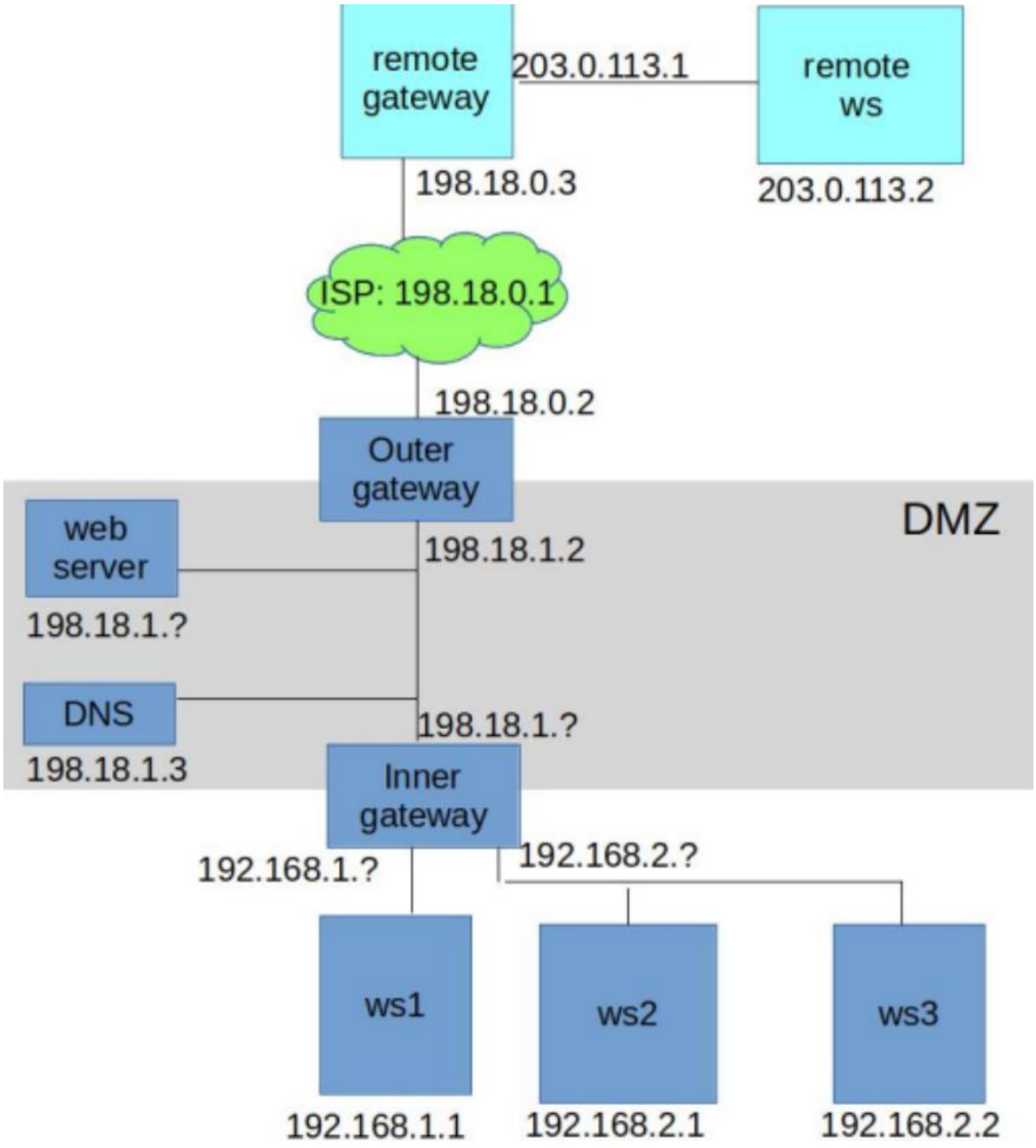

Ilustración 33 DMZ + DNS + Mail + WEB Labtainer

6.5. RIP

6.5.1. Imunes

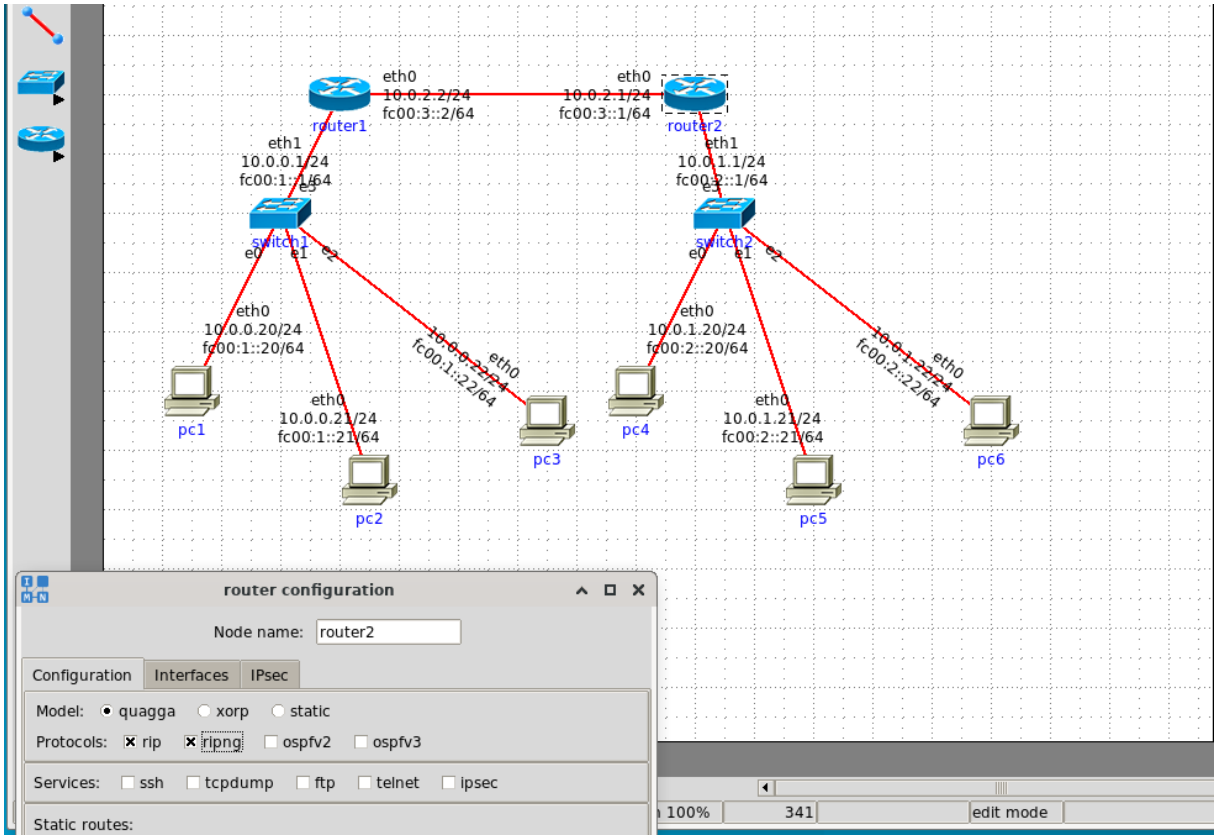

Ilustración 34 RIP Imunes

6.5.2. Educational Network Simulator

No tiene las capacidades técnicas para este servicio.

6.5.3. Labtainer

No tiene las capacidades técnicas para este servicio.

6.6. OSPF

6.6.1. Imunes

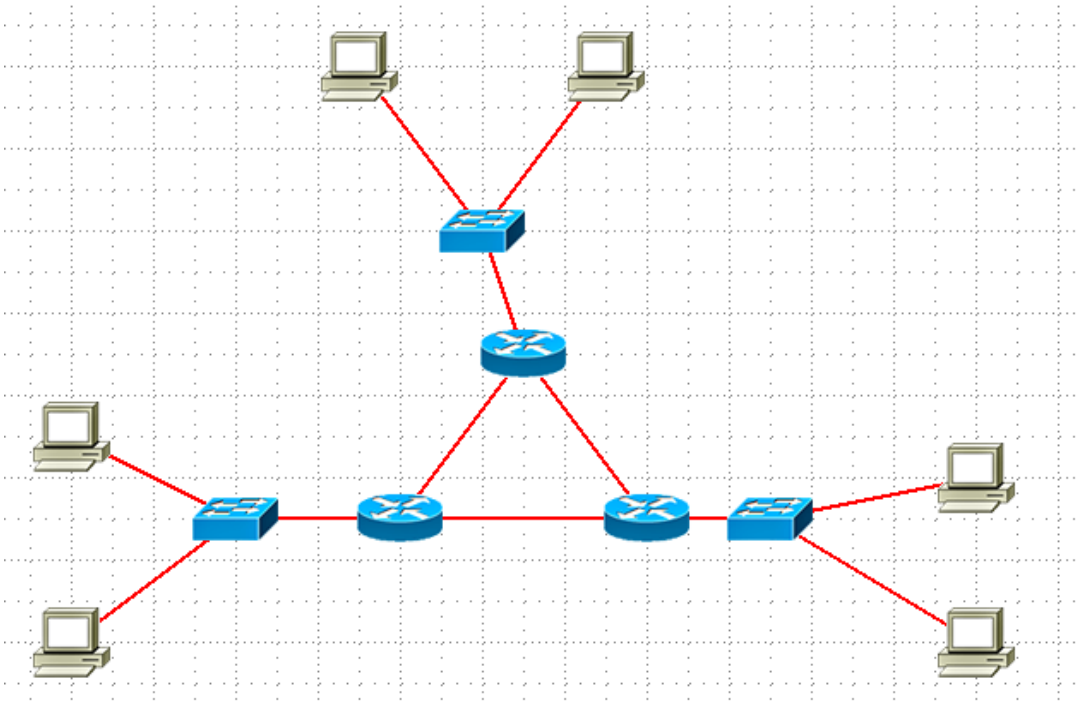

Ilustración 35 OSPF Imunes

6.6.2. Educational Network Simulator

No tiene las capacidades técnicas para este servicio.

6.6.3. Labtainer

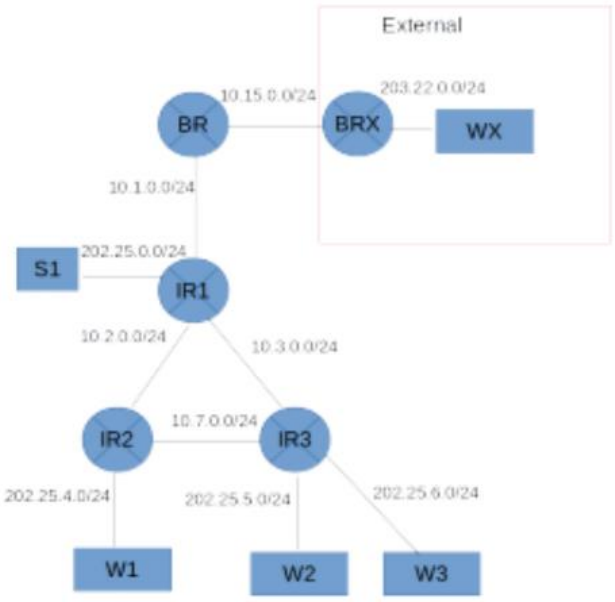

Ilustración 36 OSPF Labtainer

6.7. PING

6.7.1. Imunes

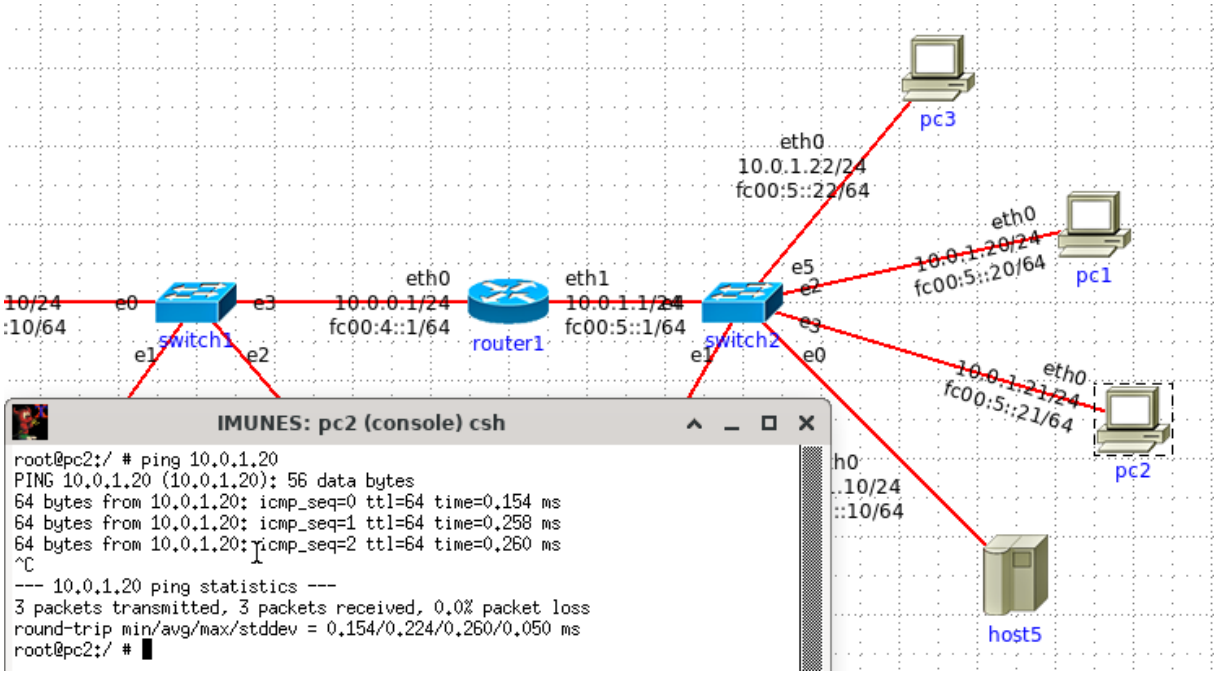

Ilustración 37 PING Imunes

6.7.2. Educational Network Simulator

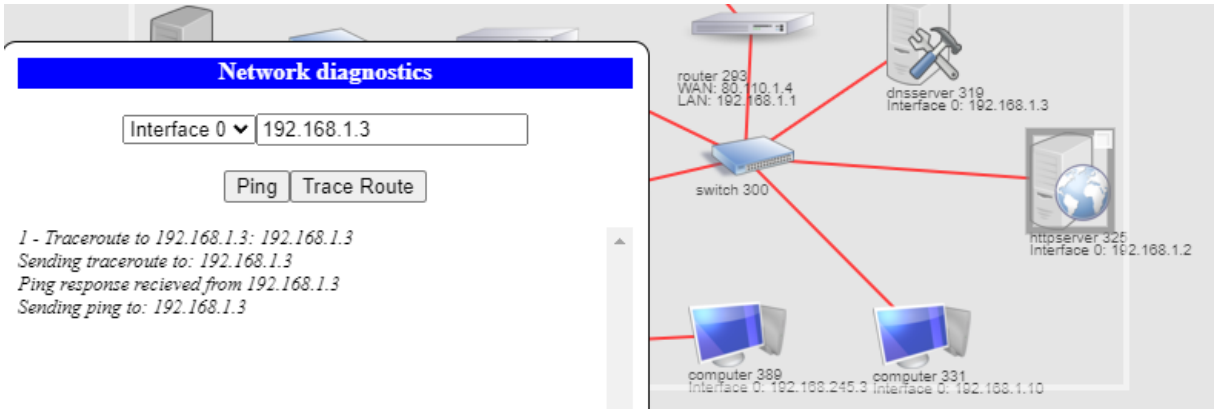

Ilustración 38 PING ENS

6.7.3. Labtainer

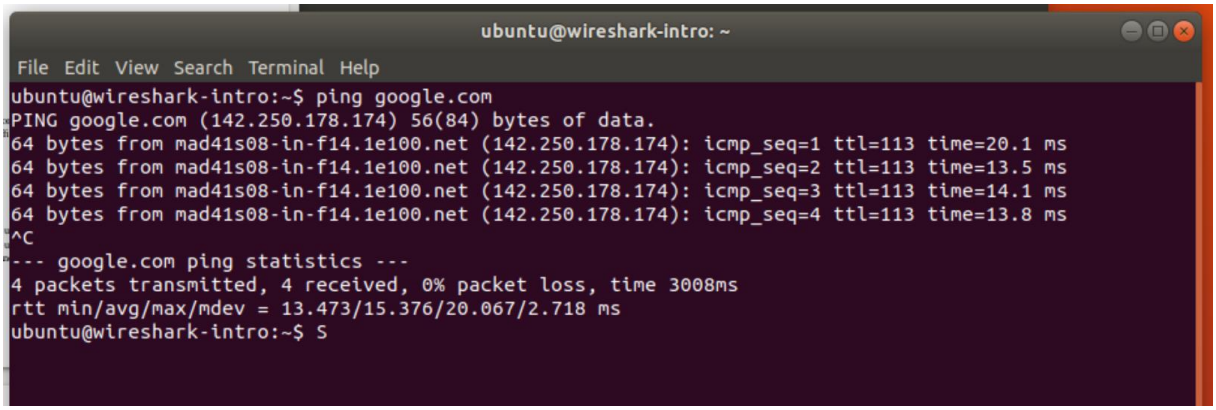

Ilustración 39 PING Labtainer

6.8. WIRESHARK

6.8.1. Imunes

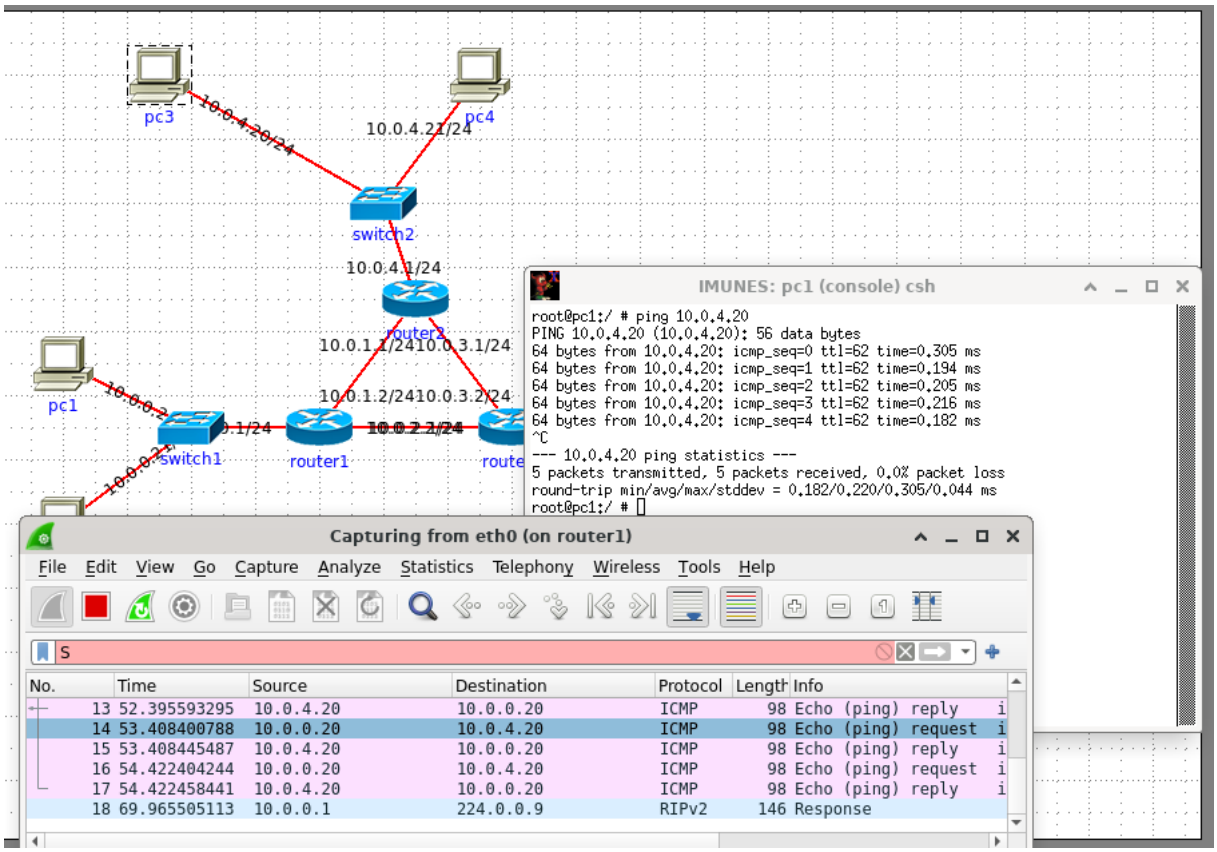

Ilustración 40 Wireshark Imunes

6.8.2. Educational Network Simulator

No tiene las capacidades técnicas para este servicio.

6.8.3. Labtainer

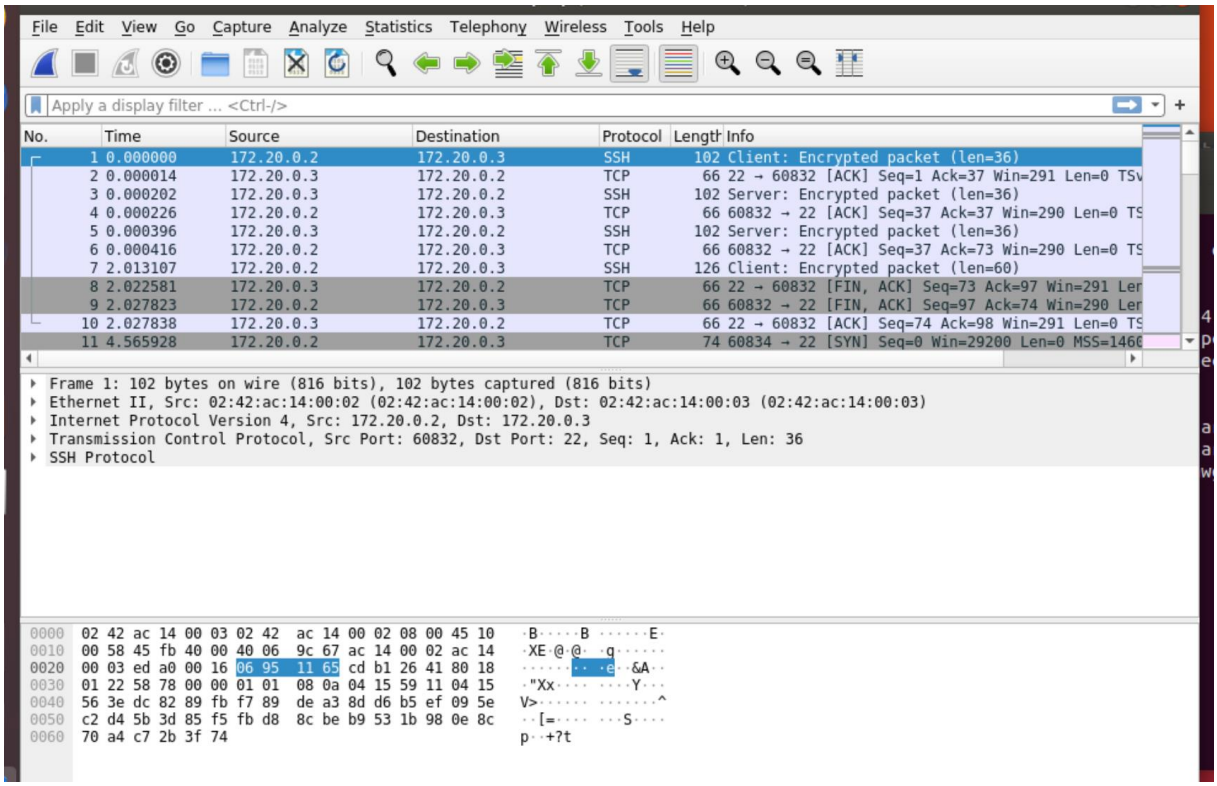

Ilustración 41 Wireshark Labtainer

7. Conclusión

Existen muchas soluciones que aportan ventajas e inconvenientes al aprendizaje de redes, cada tiene características y funcionalidades únicas.

Para poder comparar soluciones tan diferentes, durante el proyecto se ha armonizado al máximo los criterios de evaluación extrayendo de la forma mas objetiva posible cual es la mejor solución para los objetivos expuestos.

Como herramientas seleccionadas tenemos Imunes, Labtainers y Educational Network Simulator. Ha sido imposible seleccionar una ya que el conjunto de estas tres hace, desde la perspectiva del autor, el proceso de aprendizaje de redes un proceso liviano e incluso divertido tanto para estudiante como para profesor.

Cada una de estas tres herramientas ayudan en una faceta diferente del proceso de aprendizaje.

En la fase inicial cuando el alumno entra en contacto con el mundo de las redes disponer de una herramienta tan sencilla como la de ENS permite al alumno conocer todos los términos básicos sin verse abrumado por demasiada información.

Una vez el alumno avanza teóricamente se debe introducir Imunes. En esta herramienta ya encontrará protocolos y dispositivos más avanzados y le permitirá, de forma sencilla, comenzar a gestionar y montar topologías simulando entornos bastante fidedignos.

La última fase de enseñanza se debe realizar con Labtainers. Permitirá al alumno profundizar en los conocimientos teóricos ya adquiridos de una forma diferente y única.

Con estas soluciones conseguimos acercar el mundo práctico de redes informáticas al mundo académico, con 0 costes y teniendo en cuenta las capacidades técnicas disponibles.

8. Bibliografía

[1] T. Scheepers, Virtualization and Containerization of Application, Twente Enschede.

[2] «Aniruddh M; Anubhav Dinkar; Shashank C Mouli; Sahana B; Abhay A Deshpande,» de *IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, 2021.

[3] Cloonix, «Cloonix Official Documentation,» 09 January 2022. [En línea]. Available: http://clownix.net/.

[4] srl-labs, «ContainerLab Documentation,» 2022. [En línea]. Available: https://containerlab.dev/. [Último acceso: 24 04 2022].

[5] Coreemu, «Core Official Documentation,» 2022. [En línea]. Available: http://coreemu.github.io/core/.

[6] Coreemu, «Core Official Github Page,» 22 Marzo 2022. [En línea]. Available: https://github.com/coreemu/core. [Último acceso: Junio 2022].

[7] ENS, «ENS Official Website,» 2022. [En línea]. Available: http://malkiah.github.io/NetworkSimulator/. [Último acceso: Junio 2022].

[8] Next Generation, «Eve Official Documentation,» 2022. [En línea]. Available: https://www.eve-ng.net/.

[9] Eve-ng, «Eve-ng Official Documentation,» 2022. [En línea]. Available: https://www.eve-ng.net/. [Último acceso: Junio 2022].

[10] srl-labs, «Github,» 2022. [En línea]. Available: https://github.com/srl-labs/containerlab. [Último acceso: 12 Feb 2022].

[11] GNS3, «GNS3 Official Website,» 2022. [En línea]. Available: https://www.gns3.com/. [Último acceso: Junio 2022].

[12] Imunes, «Imunes Official Website,» 2022. [En línea]. Available: http://imunes.net/. [Último acceso: Junio 2022].

[13] U. Shah, Performance Evaluation of MPLS in a Virtualized Service Provider Core (with/without Class of Service), 2017.

[14] G. R. Robert Koch, «ECCWS European Conference on Cyber Warfare,» de *An overview of Linux Container Based Network Emulation,* 2016.

[15] A. R. D. K. Rajdeep Dua, «Virtualization vs Containerization to Support PaaS,» de *IEEE International Conference on Cloud Engineering*, 2014.

[16] P. S. S. B. S. S. N. G. Bachiega, «Container-Based Performance Evaluation: A Survey and Challenges,» de *2018 IEEE International Conference on Cloud Engineering (IC2E)*, 2018.

[17] B. Linkletter, «brianlinkletter,» 05 03 2021. [En línea]. Available: https://www.brianlinkletter.com/. [Último acceso: Enero 2022].

[18] O. K. Eason, Information Systems Development Methodologies Transitions: An Analysis of Waterfall to Agile Methodology, University of New Hampshire, 2016.

[19] Juniper, «Wistar Official Github,» 2022. [En línea]. Available: https://github.com/Juniper/wistar. [Último acceso: Mayo 2022].

[20] NS-3, «NS-3 Official Website,» 2022. [En línea]. Available: https://www.nsnam.org/. [Último acceso: Mayo 2022].

[21] Nre Labs, «Nre Labs Official Documentation,» 2022. [En línea]. Available: https://nrelabs.io/.

[22] Mininet, «Mininet Official Website,» 2022. [En línea]. Available: http://mininet.org/. [Último acceso: Mayo 2022].

[23] Naval Postgraduate School, «Labtainers Official Website,» 2022. [En línea]. Available: https://nps.edu/web/c3o/labtainers. [Último acceso: Junio 2022].

[24] Kathara, «Kathara Official Website,» 2022. [En línea]. Available: https://www.kathara.org/. [Último acceso: Mayo 2022].

[25] Imunes, «Imunes Official Website,» 2022. [En línea]. Available: https://github.com/imunes/. [Último acceso: Junio 2022].