IMPLEMENTACIÓN DE UN PROTOTIPO DE NUBE PARA BRINDAR INFRAESTRUCTURA COMO SERVICIO (IAAS) EN EL LABORATORIO DE TELEMÁTICA DE LA UCAB

Luis David Casique, Isaac Piña

Escuela de Ingeniería en Telecomunicaciones, Universidad Católica Andrés Bello
Caracas, Venezuela
ldcasique.17@est.ucab.edu.ve
igpina.18@est.ucab.edu.ve

***Resumen* — El presente artículo recopila el conjunto de actividades y procesos realizados desde el mes de octubre del año 2021 hasta la fecha, para la elaboración del proyecto de implementación de una nube prototipo, con la infraestructura computacional del Laboratorio de Telemática de la institución para brindar Infraestructura como Servicio (*Infrastructure as a Service, IaaS*).**

El proyecto fue implementado con el sistema operativo UBUNTU, la aplicación gestora de recursos de cómputo *MICROSTACK*, implementada en modalidad de nodos múltiples, cinco computadoras del Laboratorio de Telemática y su infraestructura de red utilizada para la impartición de las asignaturas de laboratorios.

Previo al proyecto, se realizó un despliegue de una nube pequeña de forma local, utilizando computadoras pertenecientes a los autores del proyecto, con el fin de ensayar, aprender y familiarizarse con el uso de las herramientas UBUNTU y *MICROSTACK*. Posteriormente, se implementó la nube en el laboratorio de telemática, utilizando cinco de sus computadoras conectadas como un macro bloque de cómputo, y se creó un despliegue completo que permitió virtualizar un ambiente de redes y realizar pruebas de distintas capas, virtualizando el *hardware* de las computadoras que conforman la nube.

Fundamentalmente, el proyecto busca introducir a la escuela de ingeniería en telecomunicaciones en el *cloud computing* y añadir el estudio del mismo a la carrera, y posteriormente, para sentar las bases de una futura mejora e implementación como servicio oficial para la comunidad universitaria, tanto para alumnos como profesores. Adicionalmente, se encontraron actividades impartidas en las asignaturas del laboratorio y otros usos prácticos de la nube que le saquen el mayor provecho posible, dentro de las limitaciones de *hardware* de los equipos que conforman a la misma.

Palabras clave: Infraestructura como Servicio, *MICROSTACK*, nube, despliegue, nodos.

I. INTRODUCCIÓN

La escuela de ingeniería en telecomunicaciones tiene el interés de abordar el tema de computación en la nube a la brevedad posible e incluirlo en sus estudios académicos. La implementación de un prototipo de nube, permite introducirse de lleno en este tópico y evaluar una factible aplicabilidad académica. De esto trata el presente Trabajo de Grado, resumido debidamente el presente artículo dividido en seis secciones, de las que se invita cordialmente a su lectura.

II. PLANTEAMIENTO DEL PROYECTO

A. Problema

La computación en la nube, a lo largo de los últimos años, se ha hecho presente en las actuales implementaciones de telecomunicaciones. El crecimiento de esta tecnología ha provocado que su implementación se haya establecido como una competencia de cada vez mayor importante dentro del ejercicio profesional de la ingeniería en telecomunicaciones. Por ello, resulta de interés incorporar el estudio teórico/práctico de dichos tópicos dentro de las asignaturas de la índole telemática, dado que actualmente no son abordados en las mismas [1] [2] [3], en aras de robustecer el perfil de los egresados de la carrera.

Para introducirse de lleno al tema descrito, se ha implementado un prototipo de nube que permite brindar Infraestructura como Servicio (*Infrastructure as a service*, IaaS) como una herramienta que servirá de forma esencial para estudiar sus características de implementación y utilización, evaluar la eficacia de ejecutar sistemas operativos sobre instancias de la y sentar las bases para aprovechar de manera eficiente los recursos del Laboratorio de Telemática, y en futuros proyectos, garantizar el acceso a los alumnos a los recursos, de cómputo y de red, que se ajusten a sus necesidades académicas. Se utilizó la aplicación *MICROSTACK*, un compendio de módulos y servicios de *OPENSTACK* [4], de código abierto, para el sistema operativo UBUNTU

B. *Objetivos*

Objetivo general:

Implementar prototipo de nube basada en *MICROSTACK*, para brindar Infraestructura Como Servicio en el Laboratorio de Telemática de la Universidad Católica Andrés Bello.

Objetivos Específicos:

- Investigar los recursos y servicios basados en la nube como la Infraestructura Como Servicio (IaaS) y la herramienta de cómputo en la nube MICROSTACK.
- Realizar la instalación local y configuración del sistema operativo Ubuntu y de MICROSTACK.
- Estudiar y experimentar con los comandos de ejecución de la consola de MICROSTACK.
- Realizar pruebas de habilitación y levantamiento de servicios basados en la nube en dispositivos locales. Creación de instancias para la solicitud de recursos como perfiles virtuales.
- Establecer conexión remota desde dispositivos externos los dispositivos locales en los que se implementó la nube.
- Instalar e implementar MICROSTACK en las computadoras del Laboratorio de Telemática.
- Realizar solicitudes de recursos de hardware a las computadoras del laboratorio desde un dispositivo externo a la nube.
- Determinar parámetros para evaluar rendimiento de las instancias de la nube.
- Evaluar el desempeño y rendimiento de las instancias.
- Estudiar la posibilidad de emplear la infraestructura de nube para ejecutar las plataformas de simulación empleadas en las cátedras de Laboratorio de Telemática I y Laboratorio de Telemática II.

III. MARCO TEÓRICO

Considerando la envergadura del presente proyecto, el mismo debe contar con un robusto y pertinente sustento teórico, partiendo de la idea fundamental que es la Infraestructura como servicio, tocando los temas externos e internos relacionados a las redes de computadoras y sistemas operativos.

A. *Redes definidas por software*

Es un enfoque de arquitecturas de red que busca separar de forma lógica la gestión y control de una red de su hardware físico. Esto permite que un amplio conjunto de aplicaciones y servicios que utilizan Interfaces de Programación de Aplicaciones (API) abiertas programe el comportamiento de la red, lo cual brinda a los proveedores de servicios el control completo sobre las redes, y los equipos que las conforman, de forma consistente, dinámica y escalable, independientemente del hardware o de la tecnología de red subyacente. Mediante la implementación de estándares sobre las interfaces programables, SDN permite la gestión y control de las redes de forma unificada [5].

B. *Computación en la nube (Cloud Computing)*

Se define como un conjunto de recursos de computadora (redes, servidores, almacenamiento, aplicaciones y servicios), que son brindados a usuarios, según sus requerimientos, como un servicio bajo demanda. Los recursos son accesibles de forma remota por medio de una red externa, como *Internet* [6] [7] [8].

La computación en la nube también está basada en capas, las cuales pueden observarse en el siguiente esquema:

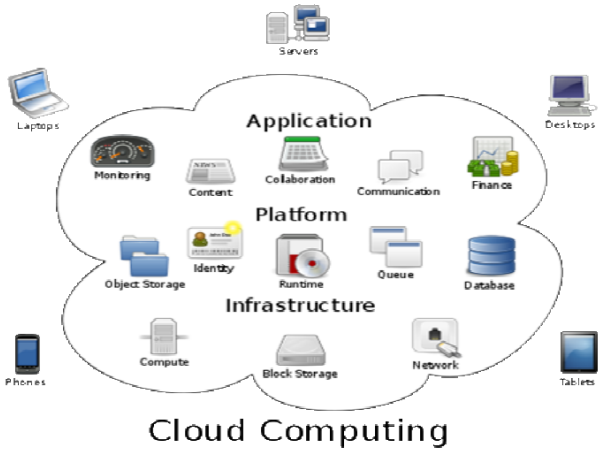

Fig. 1 Capas de la computación en la nube [7].

C. *Tipos de Nubes*

Las nubes se pueden clasificar según el tipo de servicio o contenido que ofrece, según el tipo de identidad, grupo o empresa que las implementan y según la dimensión o sectorización del público hacia el que están dirigidas. Principalmente, distinguen tres tipos de nubes:

1) Nube Pública:

Es el tipo de nube más conocido generalmente. En ella, se comparten recursos y ofrece servicios al público general (o un importante público industrial [6]) a través de Internet [9].

2) Nube Privada

Es una nube implementada, gestionada y operada por una organización privada [7], donde se encuentran físicamente los servidores y equipos que la conforman. En general, son implementadas para ofrecer hardware e infraestructura de red, plataformas, y aplicaciones.

3) Nube Híbrida

Este tipo de nube cuenta con características tanto de nube pública como privada (o comunitaria), combinando las aplicaciones locales con las que se alojan en *Internet*.

D. *Infraestructura como Servicio (Infrastructure as a Service, IaaS)*

Es de los servicios basados en nube en los que se le da mayor control del hardware al usuario. Tiene la capacidad de proveer procesamiento, almacenamiento, redes, y otros servicios informáticos de *hardware* fundamentales. En este nivel de servicio, el consumidor puede implementar, y ejecutar *software* de cualquier tipo, incluyendo sistemas operativos y aplicaciones [6].

Es el tipo de servicio más avanzado de los que se proporcionan bajo computación en la nube, en comparación con el *software* como servicio (*Software as a Service*, SaaS) y la plataforma como servicio (*Platform as a Service*, PaaS).

E. *Sistemas Operativos*

Son capas de *software* cuyo trabajo es administrar todos y cada uno de los recursos y módulos de *hardware* de una computadora, además de proporcionar al usuario un modelo de computadora mejor, más simple, amigable y pulcro [10].

El programa con el que los usuarios generalmente interactúan se denomina ***shell***, cuando está basado en texto, y **GUI** (*Graphical User Interface*; Interfaz gráfica de usuario) cuando utiliza elementos gráficos o íconos [10].

1) Linux UBUNTU

UBUNTU es un moderno sistema operativo, de código abierto basado en Linux que puede ser instalado y utilizado en servidores empresariales, computadoras personales de escritorio o portátiles y para aplicaciones de computación en la nube e IoT (*Internet of Things*; Internet de las cosas). Está patrocinado por Canonical, una compañía británica de programación de ordenadores, su misión principal es "*llevar el software libre a la audiencia más amplia*" [11].

F. *Software de Gestión de Infraestructura de nube OPENSTACK*

Es una plataforma de código abierto basada en la nube, que controla grandes grupos de cómputo, almacenamiento y recursos de redes a lo largo de un *datacenter*, gestionado y provisionado mediante un conjunto de Interfaces de Programación de Aplicaciones (API) [12].

Consta de un conjunto de módulos y paquetes que permiten provisionar Infraestructura como Servicio [13].

Fig. 2 Logotipo de *OPENSTACK* [14].

1) MICROSTACK

Es una versión simplificada de *OPENSTACK* para el sistema operativo UBUNTU, de sencilla instalación a través de un *snap*, que es la integración de una aplicación y sus dependencias principales en un mismo paquete. Esta versión permite la implementación de nubes sencillas, que van desde aquellas de un nodo, hasta las de múltiples nodos [15], y cuenta con una completa documentación para la utilización mediante la consola de UBUNTU [4].

MICROSTACK puede ser implementado en dos modalidades:

Configuración de Nodo Simple: Consiste en un nodo de control que proporciona todo lo que el usuario necesite, incluida la función de nodo de cómputo.

Configuración Multi-Nodo o *Clustering*: Permite al usuario instalar *MICROSTACK* en varias máquinas (nodos) y agruparlos para proporcionar un clúster, un macro grupo de cómputo.

2) Módulos y servicios de MICROSTACK

El paquete snap de *MICROSTACK*, consta de los siguientes módulos y servicios, cada uno con un rol específico:

KeyStone: Gestiona las identidades dentro del entorno de la nube [16].

Glance: Gestiona imágenes de sistemas operativos en distintos formatos [16].

Neutron: Es el servicio de redes TCP/IP y virtualización de los elementos y dispositivos que las conforman [16].

Nova: Es el servicio de cómputo de *OPENSTACK*. Responsable de la programación, aprovisionamiento y finalización de máquinas virtuales, denominadas instancias, que conforman la nube [16].

Cinder: Crea, gestiona y elimina volúmenes de memora de almacenamiento para las instancias que conforman la nube [16].

IV. Marco Metodológico

El presente proyecto, tomando sus características en cuenta y consideración, entra en la categoría de **proyecto especial**, debido a que lleva a cabo la creación de una herramienta que sentará las bases para la solución de un problema, la cual busca dar respuesta a necesidades e intereses de tipo cultural [17].

El proyecto requirió de una elaboración estructurada dividida en cinco fases de desarrollo para cumplir bloques de actividades específicas, las cuales son:

A. *Fase I: Investigación teórica documental.*

Fase en la que se investigó, recopiló y se documentó cada aspecto teórico necesario sustentar la implementación del proyecto. Esta fase fue continua y constante durante el desarrollo del mismo, dado que en la implementación de las fases posteriores se tuvo que seguir investigando y ampliando los fundamentos recopilados.

Las acciones llevadas a cabo fueron las siguientes:

1) Investigación sobre los conceptos asociados al levantamiento nubes para brindar Infraestructura como Servicio (IaaS).

2) Indagación en antecedentes de levantamientos de nubes con OPENSTACK y MICROSTACK, en general, y sus aplicaciones en el área académica.

3) Recopilación de guías de uso de MICROSTACK, y de sus comandos de consola.

4) Determinación de las características de hardware de las computadoras del laboratorio de telemática.

5) Definición de los parámetros a medir para evaluar rendimiento de plataformas de nube, y de computadora.

B. *Fase II: Configuración de las Herramientas.*

Se interactuó con los entornos y herramientas con las que se llevó a cabo la implementación del proyecto tal como son los programas y sistemas operativos, en las computadoras de los autores del proyecto. Las acciones realizadas fueron:

1) Instalación y configuración de la versión 20.04.3 LTS del sistema operativo UBUNTU en computadoras locales, creación de particiones de disco duro, y asignación de recursos de memoria y procesamiento.

2) Ajustes de configuración y rendimiento por medio de la consola de UBUNTU. Gestión de la conectividad a redes locales, y a redes externas como Internet.

3) Descarga e instalación de la herramienta MICROSTACK desde la consola de UBUNTU. Visualización de tutoriales de configuración.

C. *Fase III: Aprendizaje y experimentación local de MICROSTACK.*

Se experimentó con los distintos servicios de computación en la nube utilizando la aplicación *MICROSTACK*, en los dispositivos locales utilizados en la fase II (sección III.B). Se estudió la documentación existente, comandos y funciones para cada servicio ofrecido. Las actividades llevadas a cabo fueron:

1) Lectura y estudio de la documentación de OPENSTACK y de MICROSTACK para el conocimiento de sus características, módulos y servicios ofrecidos.

2) Estudio de los módulos de MICROSTACK necesarios para la implementación de la nube.

3) Descarga e instalación los paquetes de MICROSTACK que permitieron ofrecer los recursos de hardware por medio de la nube.

4) Habilitación las funciones de Infraestructura como Servicio, y realizar pruebas con otras computadoras de la misma red de área local. Medición el rendimiento de la conexión remota a la infraestructura con base en parámetros definidos.

D. *Fase IV: Montaje de la nube en el Laboratorio de Telemática.*

Se realizó la instalación y levantamiento de la nube en el laboratorio de telemática en la universidad, utilizando cinco computadoras de su infraestructura; cuatro de ellas, como nodos de cómputo, y la restante, como nodo de control. Las actividades desarrolladas fueron:

1) Instalación de MICROSTACK en las computadoras del laboratorio de telemática.

2) Configuración de los módulos de MICROSTACK con los que se habrán experimentado en la fase III.

3) Levantamiento de la infraestructura como servicio en las cinco computadoras que conformarán la nube y configuración del nodo de control.

4) Creación de instancias de prueba.

E. *Fase V: Realización de pruebas de rendimiento.*

Estando implementada la nube en el laboratorio de telemática, se definieron los parámetros para evaluar el rendimiento y la calidad del servicio y se aplicaron los mismos. Se desplegó un entorno virtualizado de computadoras y redes en los que se realizaron las pruebas descritas. Las acciones desarrolladas en esta fase, fueron:

1) Determinación de tres parámetros de medición de rendimiento de instancias y máquinas virtuales de la nube.

2) Comparación de rendimiento de instancias en la nube con respecto de las máquinas virtuales manejadas con VirtualBox, para determinar cuál de las dos opciones ofrece el mejor desempeño.

La duración de cada una de las fases realizadas, se presentan en el siguiente diagrama:

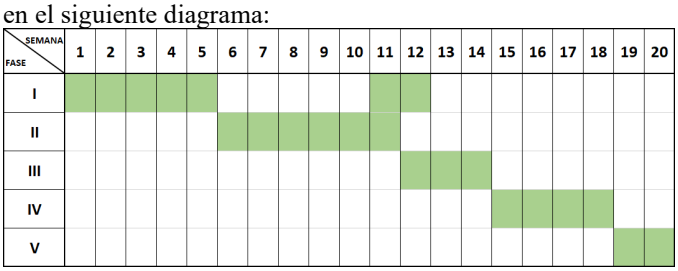

Fig. 3 - Planificación para la realización del proyecto.

V. Desarrollo

En la presente sección, se recopila la ejecución de las actividades realizadas en el desarrollo del proyecto, desde su análisis teórico, hasta su implementación. Se presentan relacionadas y agrupadas en las mismas fases descritas en la planificación metodológica plasmada en el capítulo anterior (ver sección III), no obstante, muchas de estas actividades no contribuyen únicamente a sus respectivas fases, sino también a otros aspectos presentes en toda la realización del proyecto.

A. *Fase de investigación teórica documental:*

Esta fase fue desarrollada durante la mayoría del período de elaboración del proyecto, realizándose en simultáneo con varias de las demás fases del mismo. Sirvió fundamentalmente para recolectar una amplia información organizada para elaborar el marco teórico del proyecto y para familiarizarse con cada uno de los tópicos relacionados al mismo.

Los temas fundamentales recopilados son:

- Redes de computadoras.
- Modelo de Capas TCP/IP.
- Redes definidas por *software*.
- Computación en la nube.
- Servicios brindados en la nube.
- Infraestructura como servicio.
- Recursos de *hardware.*
- Sistema Operativo UBUNTU.
- *OPENSTACK* y *MICROSTACK*.

B. *Fase de configuración de las herramientas:*

Se instalaron y configuraron las herramientas con las que se llevaron a cabo el proyecto, con el fin de aprender y familiarizarse con ellas, creando una "micro nube" en computadoras locales, pertenecientes a los autores del proyecto.

Las herramientas instaladas fueron:

1) Sistema Operativo UBUNTU, versión 20.04.3 LTS

Se escogió esta versión debido a que, al momento de su elección, era la última versión con soporte a largo plazo, marcado por las siglas LTS (*Long Term Support*).

Se realizó una partición de la memoria secundaria de la computadora para dar lugar a todos los archivos y programas que conforman al sistema operativo, y que el mismo cumpla con los requisitos mínimos para la instalación de la aplicación *MICROSTACK.*

2) Paquete snap MICROSTACK.

En su última versión disponible para el momento (244), por medio de la propia consola de UBUNTU, a través del comando de super usuario: `$ sudo snap install microstack --beta`.

C. *Fase de aprendizaje y experimentación local con MICROSTACK:*

1) *Inicialización de MICROSTACK*

Por medio de la ejecución del comando de consola: `~$ sudo microstack init --auto –control`, se habilitaron los módulos, se ajustaron las variables de entorno y configuraciones de *clustering* de la aplicación, como se muestra en la siguiente figura:

```
Waiting for 192.168.0.115:9292
2022-01-24 17:15:37,444 - microstack_init - INFO - Adding cirros image ...
2022-01-24 17:15:40,994 - microstack_init - INFO - Creating security group rules ...
2022-01-24 17:15:53,707 - microstack_init - INFO - Configuring the Cinder services...
2022-01-24 17:17:05,180 - microstack_init - INFO - Running Cinder DB migrations...
2022-01-24 17:18:19,282 - microstack_init - INFO - restarting libvirt and virtlogd ...
2022-01-24 17:18:51,601 - microstack_init - INFO - Complete. Marked microstack as initialized!
root@LuisDavid-Desktop:~#
```

Fig. 4 *MICROSTACK* marcado como inicializado.

2) *Despliegue de ambiente virtual*

Luego de la iniciación de *MICROSTACK*, se procedió a realizar un amplio despliegue en el que se ajustaron y configuraron sus siguientes herramientas y parámetros: **Servicios**, ***dashboard***, **plantillas**, **identidades**, **multi-provisión**, **redes**, **instancias**, **almacenamiento** y **cuotas**. Este despliegue fue realizado de forma satisfactoria siguiendo la documentación oficial de Canonical [18].

3) *Creación de instancias de prueba*

Con el despliegue anterior culminado, se procedió a crear algunas instancias funcionales, utilizando los parámetros creados en el despliegue, utilizando el *dashboard* de *OPENSTACK*, el cual, permite observar el proceso de creación, como se muestra en la siguiente figura:

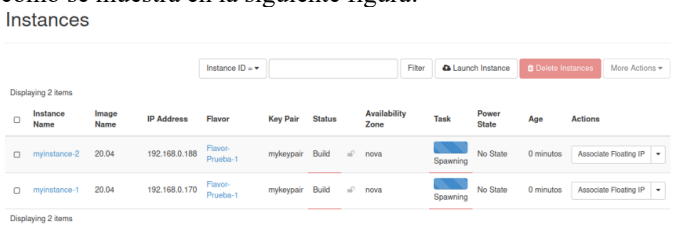

Fig. 5 Instancias de prueba creadas en *MICROSTACK*.

4) *Adición de nodo de cómputo*

El despliegue de ensayo en la "micro nube" concluyó con la adición de un nodo de cómputo, con el fin de familiarizarse con la creación de clústeres en la nube. Se consiguió un segundo equipo denominado "macxy-desktop", con características similares al nodo de control, se le descargó *MICROSTACK* y estableció conexión con el nodo de control para combinar sus características y crear un clúster con los dos nodos.

Para ello, en el nodo de control se generó una cadena de conexión (*connection string*), para ser ingresada en el nodo de cómputo para establecer el clúster. Se generó la cadena con el comando:

```
$ sudo microstack add-compute
```

Luego, en el nodo de cómputo, se ingresó la cadena de conexión en el proceso de inicialización de *MICROSTACK* siguiente comando: `$ sudo microstack init --auto --compute --join <cadena>`

En seguida, al acabar la inicialización, se mostró en pantalla el mensaje: `Complete. Marked microstack as initializated!`

5) *Instancias de prueba en el nodo de cómputo*

Una vez se añadió el nodo de cómputo, se procedió a probar la creación de una instancia de prueba denominada "test", equipada con la imagen de CirrOS de Linux, utilizando su *hardware*, para ello, se empleó el siguiente comando:

```
$ microstack launch cirros -name test --availability-zone nova:macxy-desktop
```

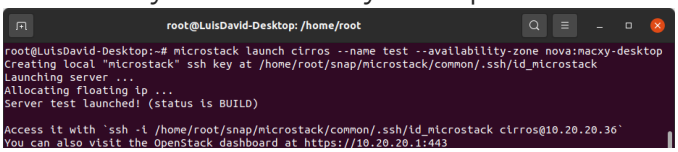

Fig. 6 Instancia "*test*" creada en el nodo de cómputo.

El clúster de la nube, se muestra en la siguiente figura:

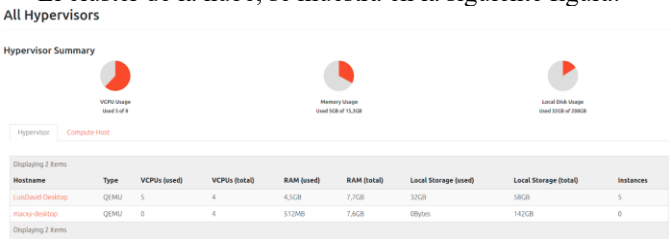

Fig. 7 Visualización de ambos nodos de la nube como hipervisores desde el *dashboard* de *OPENSTACK*.

D. *Fase de Montaje de la nube en el Laboratorio de Telemática*

Tras haber ensayado con la exitosa implementación de la "micro nube", se procedió a implementar la nube en el laboratorio de telemática. Para ello, se utilizaron cinco computadoras de su infraestructura, en las que cuatro de ellas, son nodos de cómputo y la restante, el nodo de control de la nube que también provee su *hardware* como uno de cómputo.

1) *Diseño de red independiente*

Implementar computadoras para aprovisionamiento de Infraestructura como Servicio, hace que las mismas funcionen como servidores dentro de sus redes, los cuales deben estar operativos y disponibles para funcionar correctamente y no generar fallas.

Para garantizar esta condición de operatividad continua, se conectaron las computadoras de la nube a las dos infraestructuras de red creadas en el laboratorio; una de ellas, gestionada por el Departamento de Tecnologías de Información (DTI), para contar con conexión a Internet en los nodos; y una segunda infraestructura de red independiente del DTI dentro del laboratorio, para el montaje de la nube sobre la misma. La conexión de las computadoras de la nube se realizó por medio de un *switch* CISCO *Catalyst* 3750 *Series*, como se muestra a continuación:

Fig. 8 Conexión de los nodos al *switch*.

A cada nodo, se le asignó una respectiva dirección IP, perteneciente al bloque 192.168.0.0/24.

2) Descarga de MICROSTACK

Después de configurar la red de la nube, se descargó *MICROSTACK* en cada uno de los nodos que la conforman, por medio del comando:

```
$ sudo snap install microstack --beta.
```

a) Nodo de control

El nodo de control fue inicializado, a diferencia del caso de la "micro nube", de forma manual, para especificar la dirección IP de la red independiente del DTI, por medio del comando `$ sudo microstack init`. Se ingresó la configuración del *clustering*, el rol de nodo de control y la dirección IP del mismo.

Posteriormente, se generaron cuatro cadenas de conexión para adjuntar los cuatro nodos de cómputo a la nube, de forma similar a la "micro nube", por medio del comando: `$ sudo microstack add-compute.`

b) Nodos de cómputo

De igual forma, los nodos de cómputo también fueron inicializados de forma manual, para especificar sus direcciones IP, a diferencia del nodo de cómputo de la "micro nube", por medio del comando: `$ sudo microstack init`. Se ingresó la configuración del *clustering*, el rol de nodo de cómputo, la dirección IP y la cadena de conexión para cada nodo de cómputo, generadas en el nodo de control.

3) Despliegue final

Luego de conformar el clúster de la nube con el nodo de control y los cuatro nodos de cómputo, se procedió a realizar un acondicionamiento del entorno similar al realizado en la "micro nube" (ver sección V.C.2). Se crearon nuevos dominios, proyectos y usuarios con distintos roles, y habilitación de soporte multidominios, descarga de imágenes de distintos sistemas operativos, entre otros parámetros.

a) Imágenes de sistemas operativos

Adicionalmente a la imagen de CirrOS, se descargaron imágenes de los siguientes sistemas operativos:

- UBUNTU 20.04.3 LTS [19].
- UBUNTU 18.04.3 LTS [19].
- Debian 12 [20].
- Centos Stream 9 [21].
- Fedora 36 [22].
- Windows 10 [23].

E. *Fase de realización de pruebas*

Posteriormente a la creación de la nube multi nodo, se procedió a probar y evaluar su rendimiento y funcionalidad con una serie de evaluaciones de desempeño.

1) Pruebas de rendimiento

Para probar el rendimiento de la infraestructura de la nube, se seleccionaron tres parámetros que permiten estudiar esta característica, los cuales son:

- Tiempo de creación de instancias.
- Tiempo de encendido de instancias.
- Latencias de instancias y los nodos físicos de la nube.

Se diseñó una topología para la realización de estas pruebas, conformada por cinco instancias, utilizando el dominio *default* esta vez, como se muestra a continuación:

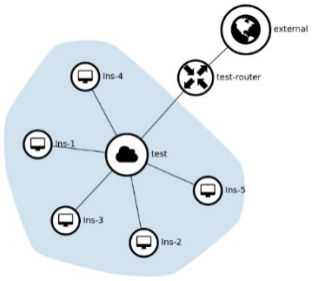

Fig. 9 Topología para pruebas de rendimiento.

TABLA I

ESPECIFICACIONES PARA LAS PRUEBAS DE RENDIMIENTO.

Instancia	***Flavors***		**Red**
1	*TG_Flavor*	*m1.small*	"test"; 192.168.222.0/24
2			
3			
4			
5			

2) Prueba de enrutamiento estático

Neutron, módulo gestor de las funciones de red, permite la adición y establecimiento de rutas estáticas de forma nativa por medio del *dashboard* de *OPENSTACK*, esto permite la comunicación y convergencia de cada segmento de las topologías desplegadas en cada proyecto dentro de la nube, como la que se muestra a continuación:

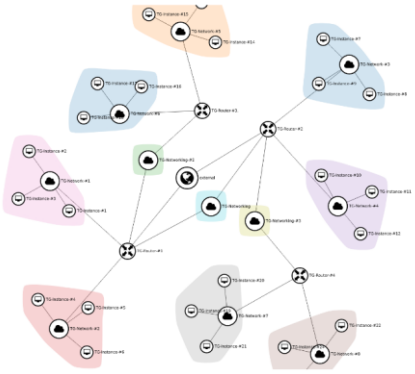

Fig. 10 Topología de prueba de enrutamiento estático.

Se ingresó al menú de cada uno de los *routers* de la red y se añadieron las rutas estáticas correspondientes para la

convergencia completa de la red, como se muestra a continuación, el caso del *router* "TG-Router-#1":

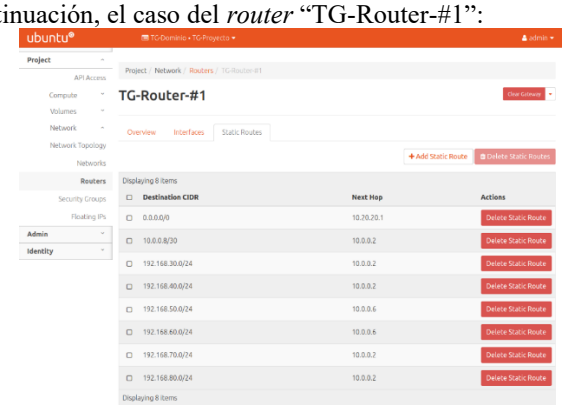

Fig. 11 Rutas estáticas para el router "TG-Router-#1".

La comunicación ICMP de los elementos que conforman la topología, luego de esta converger, se muestra en el próximo capítulo.

3) Prueba de enrutamiento dinámico

A diferencia del caso de enrutamiento estático, *Neutron* no cuente con una función que permita ejecutar protocolos de enrutamiento dinámico en los *routers*, por lo que se trabajó con instancias que funcionan como enrutadores, equipadas con la imagen del S.O. Debian 10 y el *software* de enrutamiento dinámico *FRRouting*, un *fork* del proyecto original GNU *Zebra*, desarrollado por Kunihiro Ishiguro entre 1996 y 2005, para implementar el protocolo *Open Shortest First Path* (OSPF) [24] [25].

a) Configuración en el nodo de control

Para contar con conexión a Internet en las instancias, y así descargar el software *FRRouting*, se procedió a añadir un par de funciones en el nodo de control para realizar *postrouting* y enmascaramiento NAT, con el objetivo de redirigir el tráfico dirigido a la nube por medio de la interfaz virtual "br-ex", y la habilitación del *IPforwarding*, que permite que el tráfico entrante por una interfaz del nodo, salga por otra diferente.

b) Descarga del software FRRouting

Se creó una instancia a la que le fue descargada FRR por medio de los siguientes comandos:

```
# apt-get update:
# apt-get upgrade:
# apt-get install FRR:
```

c) Configuración de FRRouting

Ya descargado, se procedió a realizar un conjunto de pasos de configuración que se mencionan a continuación:

- Habilitación de los *daemons* de los protocolos que ofrece *FRR*, en el fichero "/etc/frr/daemons".
- Habilitación de *IP Forwarding*.
- Adición de los puertos TCP/UDP que utilizan los protocolos de *FRR*, en el fichero "/etc/services".
- Copiado de los archivos de configuración principales de los *daemons* de interés, al directorio "/etc/frr/".
- Reinicio del *software FRRouting*.

d) Definición de topología de red

Diseñada con la idea de plasmar un caso que requiera establecer rutas en los *routers* para la convergencia de la red, la topología en la que se implementó *FRR* es la siguiente:

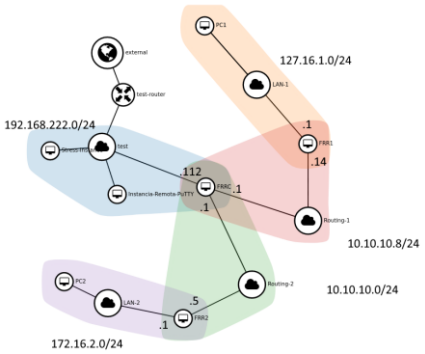

Fig. 12 Topología para enrutamiento dinámico.

e) Configuración de OSPF

Luego de crear la red descrita, se implementó OSPF en las instancias FRR1, FRR2 y FRRC, se publicaron las redes directamente conectadas a cada una y OSPF se encargó de la convergencia de la red y el establecimiento dinámico de las rutas, como se muestra a continuación, el caso de FRR1:

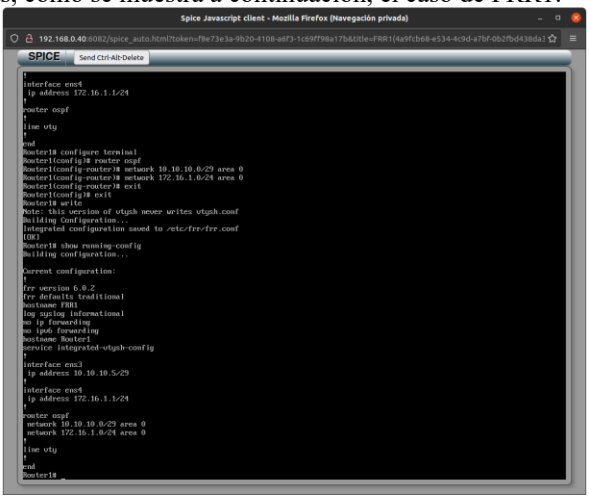

Fig. 13 Configuración de *FRR* en instancia "FRR1".

f) Establecimiento de rutas

Luego que estas configuraciones fueron guardadas en las memorias de las instancias, al cabo de unos segundos, se establecieron las rutas dinámicas de las redes desconocidas por cada una de ellas, como se muestra a continuación, el caso de FRR1:

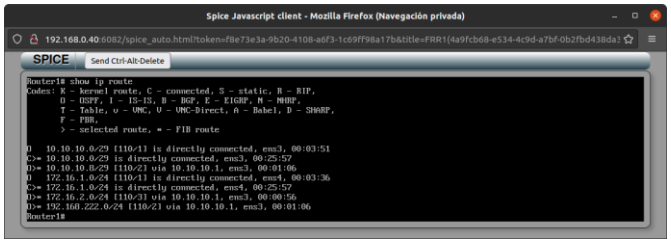

Fig. 14 Rutas dinámicas de FRR1.

Con el establecimiento de las rutas, el enrutamiento dinámico de la red está completo.

VI. Resultados

La presente sección recopila un conjunto de resultados obtenidos de las actividades ejecutadas durante el desarrollo del proyecto. Se presenta como la titulación del resultado específico de cada uno de los resultados, seguida de una descripción detallada de los mismos.

A. Resultados de pruebas locales.

a) Latencia de la micro nube

Latencia entre instancias y el nodo de control

Este parámetro fue medido con el envío de paquetes ICMP entre el nodo de control y las instancias creadas, los resultados se muestran a continuación:

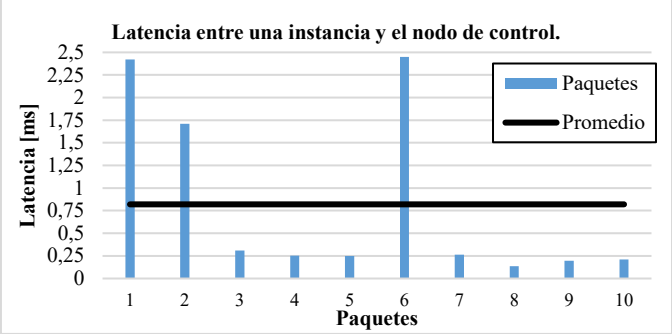

Gráfico 1. Latencia entre el nodo de control de la "micro nube" y "*myinstance1*", en milisegundos.

Nótese que, en general, la mayoría de los paquetes tardó menos de medio milisegundo en ir y volver entre ambos dispositivos, y esto se debe a que no existe un salto físico entre ellos, sino saltos lógicos relacionados a la virtualización de funciones de red de *MICROSTACK*; no obstante, paquetes 1, 2 y 6, de mayor latencia, incrementan el promedio de estos de forma considerable.

Latencia entre instancias distintas

Se crearon dos instancias pertenecientes a dos redes separadas por un *router* central, al medir dos veces la latencia entre ellas, se obtuvo el siguiente resultado:

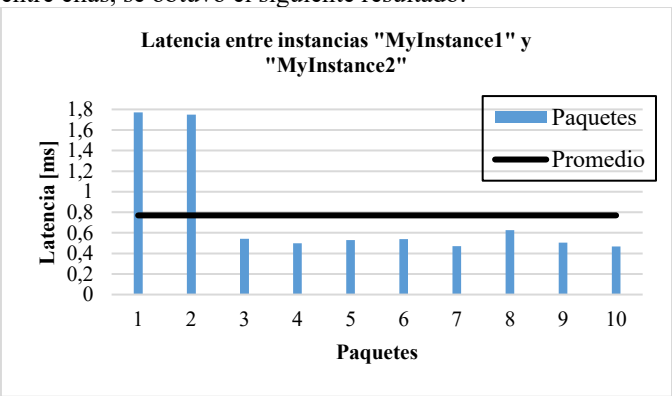

Gráfico 2. Latencia de envío de paquetes desde "*MyInstance1*" hasta "*MyInstance2*", en milisegundos.

La latencia es levemente mayor en este caso, debido a la virtualización de un *router* central que separa a las instancias, lo cual implica un salto adicional de los paquetes que viajan de una instancia a la otra.

b) Acceso remoto a la micro nube

Conexión desde el dispositivo externo hasta el nodo de control:

Utilizando la aplicación *PuTTY*, ingresando la dirección IP del nodo de control e indicando el protocolo SSH, se estableció conexión desde el dispositivo externo hasta el nodo de control.

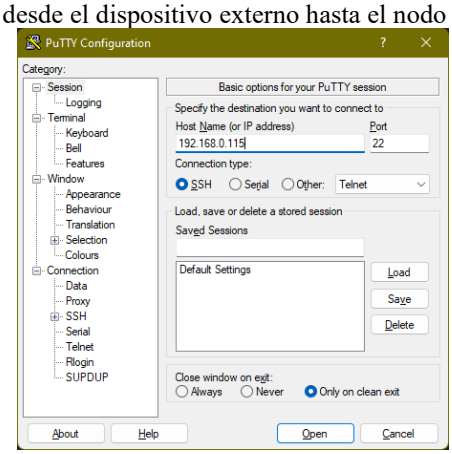

Fig. 15 Parámetros de conexión SSH hacia el nodo de control, por medio de *PuTTY*.

Luego de abrir la conexión, la ventana solicita los datos del usuario para el inicio de sesión en el nodo de control y su correspondiente contraseña. Se ingresaron las credenciales del usuario "luis-david-casique" y se logró el acceso de forma satisfactoria, permitiendo ver parámetros tanto del equipo como de la nube desde un dispositivo externo, como se muestra a continuación:

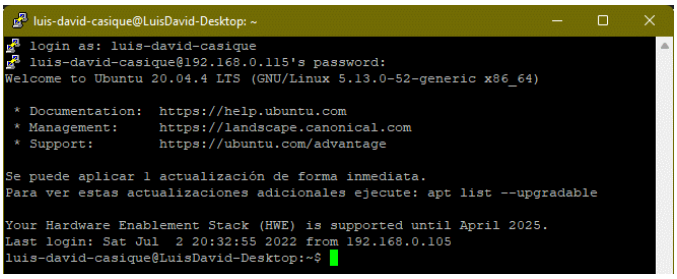

Fig. 15 Acceso remoto al nodo de control por medio de la aplicación *PuTTY*.

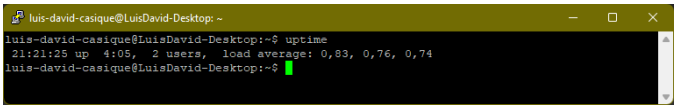

Fig. 16 Visualización de hora y usuarios del nodo de control por medio de *PuTTY*. Nótese que muestra que hay dos usuarios utilizando el equipo: el usuario local y el usuario remoto.

Conexión remota a las instancias usando *PuTTY*:

Luego de acceder al nodo de control desde *PuTTY*, es posible acceder a las instancias creadas en la nube ejecutando una segunda conexión SSH directamente con ellas, por medio de los archivos correspondientes a las llaves de acceso o *Key Pairs* y sus direcciones IP flotantes, como se muestra a continuación, el caso de "*MyInstance1*":

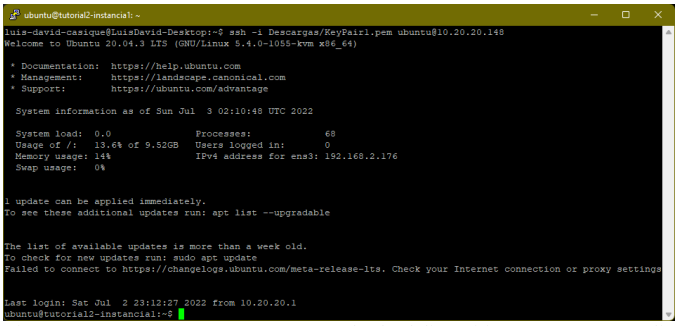

Fig. 17 Acceso remoto a "*MyInstance1*" desde el dispositivo externo, por medio de *PuTTY*.

B. *Resultados de implementación en el laboratorio*

A partir de esta sección, se plasman los resultados de la implementación de la nube en las computadoras del laboratorio de Telemática, seguidos de los resultados de las pruebas de rendimiento y usabilidad

1) Creación exitosa del clúster de la nube

De igual forma que en la "micro nube", visualizando los servicios de cómputo de la nube, puede comprobarse la creación del clúster multi nodo, como se muestra a continuación:

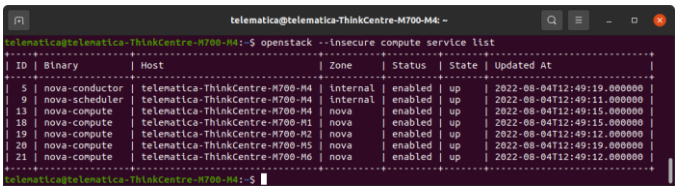

Fig. 18 Servicios de cómputo en el clúster del laboratorio.

2) Rendimiento

a) Tiempo de creación de instancias

La medición del tiempo de creación de las instancias descritas en la prueba arrojó los siguientes resultados, los cuales fueron clasificados de acuerdo al tipo de prueba en la que se recogieron los datos:

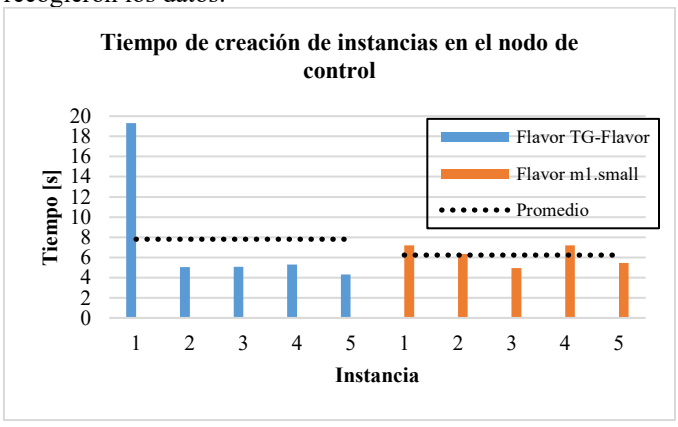

Gráfico 3. Tiempo de creación de instancias en el nodo de control, en segundos.

Como puede observarse, la primera instancia equipada con el *flavor* de menor capacidad, "TG-Flavor", fue la que más tiempo tardó en crearse, con una diferencia considerable con el resto de instancias de ambos *flavors*. Esto se debe a que fue la primera instancia creada en el día de la realización de esta prueba, con la creación de esta instancia se realizó la primera solicitud a *Nova* de utilizar el *hardware* del nodo de control de la nube, proceso que requirió un tiempo mayor en la activación y ejecución de los algoritmos y programas dentro de las memorias de los nodos que conforman la nube por primera vez en la jornada, tardando por esta causa, más tiempo que las instancias posteriores.

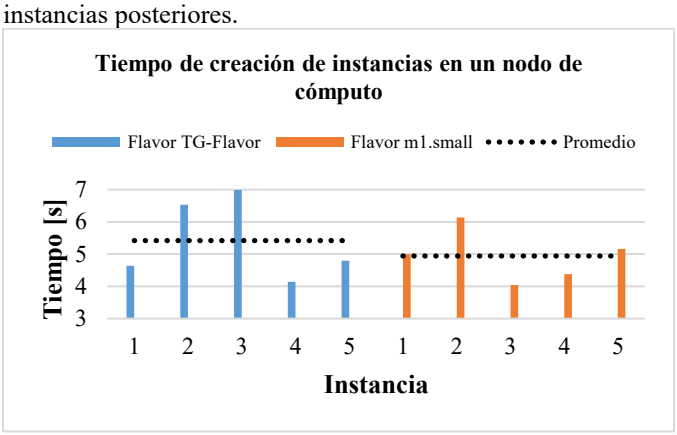

Gráfico 4. Tiempo de creación de instancias en un nodo de cómputo, en segundos.

Nótese en los tiempos de esta prueba que, la instancia que tardó más tiempo en crearse fue la tercera equipada con el *flavor* "TG-Flavor"; no obstante, sin una diferencia considerable como en la primera instancia del caso anterior.

Nótese también que, en promedio, los tiempos de creación de las instancias de los dos *flavors*, en esta prueba no tienen una diferencia perceptible.

b) Tiempo de encendido de las instancias

Encendiendo las instancias con las que desarrolló la prueba anterior, la medida de los tiempos respectivos devolvió los siguientes datos:

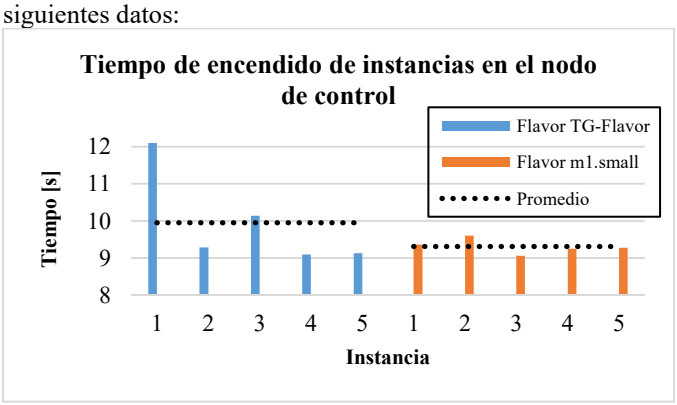

Gráfico 5. Tiempo de encendido de instancias en el nodo de control, en segundos.

El tiempo de encendido es considerablemente mayor que el tiempo de creación, esto se debe a que *MICROSTACK* emula todos los procesos que se involucran en el encendido de una computadora, independientemente de ser máquinas virtuales.

Al igual que en la medición del tiempo de creación, la instancia que demoró más en encender fue la primera equipada con el *flavor* menor, "TG-Flavor", no obstante, sin una diferencia tan importante como en el tiempo de creación. Esto se debe al mismo principio descrito en la prueba de tiempo de

creación, fue la primera instancia en solicitar a *Nova* el proceso de encendido.

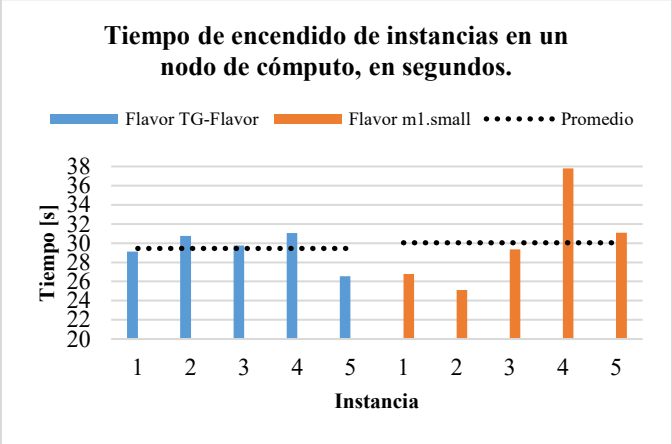

Gráfico 6. Tiempo de encendido de instancias en un nodo de cómputo, en segundos.

Nótese que, el tiempo de encendido de las instancias creadas en los nodos de cómputo es considerablemente mayor que cuando están creadas en el nodo de control. Esto se debe a pese a cada nodo de cómputo funciona como un hipervisor *Nova* gestiona algunos servicios de control de los nodos de cómputo, desde el nodo de control, y al estar este separado físicamente de los nodos de cómputo, hace uso de los cables *Ethernet* que conectan los nodos de la nube para comunicarse con los mismos; que tiene una capacidad máxima de velocidad de transmisión de datos masivamente menor a la velocidad de trabajo de las memorias físicas de una computadora.

Esta limitación, en el caso del encendido de las instancias, se traduce en un retardo adicional en el tiempo de encendido, que requiere de la ignición y corrido de múltiples procesos iniciales en la computadora que se ven ralentizados por la capacidad del medio físico que interconecta a los nodos de la nube.

c) Comparación con el entorno de VirtualBox

Luego de medir los tiempos descritos en *MICROSTACK*, se replicaron en la aplicación *VirtualBox*, donde se obtuvieron los siguientes resultados:

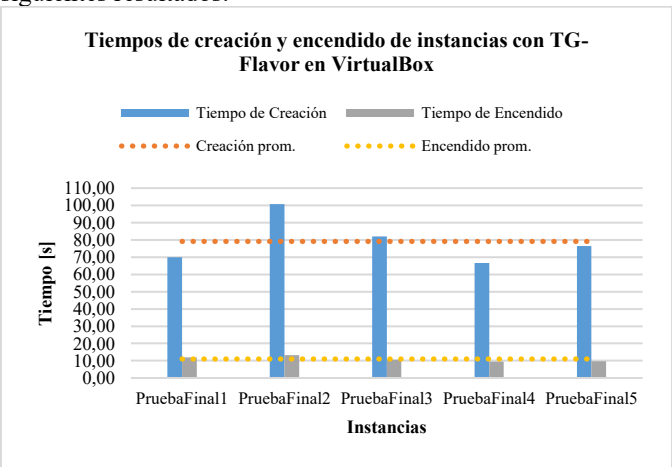

Gráfico 7. Tiempos de creación y encendido de instancias con TG-Flavor en *VirtualBox*, en segundos.

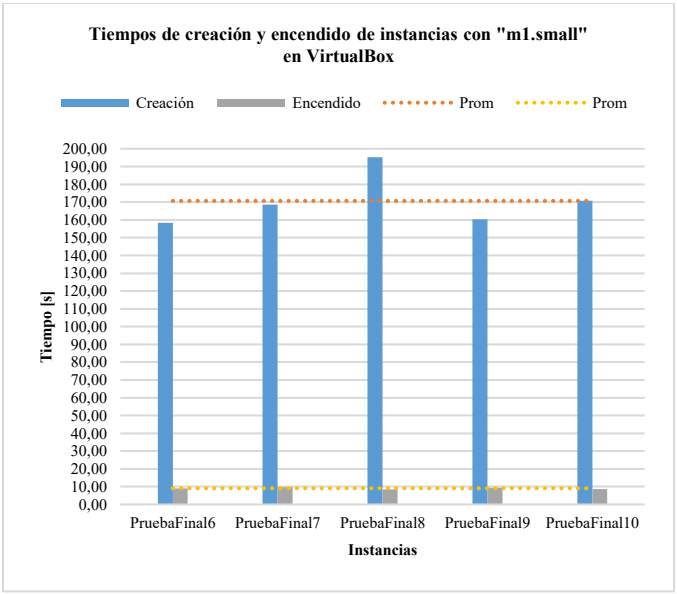

Gráfico 8. Tiempos de creación y encendido de instancias con *flavor* "m1.small" en *VirtualBox*, en segundos.

Como puede observarse, los tiempos de creación de las máquinas virtuales en este entorno son considerablemente mayores que los de las instancias de la nube. Esto ocurre principalmente por las diferencias en los procesos de creación de las mismas en ambos entornos; por un lado, en *MICROSTACK*, es posible crear instancias en tiempos reducidos porque permite el asignado de recursos de cómputo con los perfiles de configuración predefinidos, los *flavors*, utilizando un conjunto de características de *hardware* de los nodos físicos que ya ha sido preparado por *Nova* para su utilización en la nube; mientras que, en la plataforma de *Oracle*, es necesario crear de forma manual las particiones de disco duro y memoria principal para cada una de las instancias, lo cual ocupa una mayor cantidad de tiempo en la virtualización de los discos físicos para cada instancia, lo cual es una ventaja considerable a favor de *MICROSTACK*.

Por otra parte, los tiempos de encendido de las instancias son más acordes y similares a los registrados con *MICROSTACK* en el nodo de control, dado que las máquinas de *VirtualBox* fueron creadas físicamente usando un mismo nodo, incluso son inversamente proporcionales a las características de memorias asignadas a las mismas, lo cual es consecuente con que el uso de memorias de mayor capacidad, reduce los tiempos de carga de las computadoras, entre ellos, el encendido.

3) Latencias

a) Latencia inter nodo

Por medio del envío y recepción de cuatro paquetes ICMP desde el nodo de control hasta cada nodo de cómputo, la latencia general de la nube, medida con el objetivo de estimar el retardo de las instancias que se creen en cada uno de los mismos al momento de utilizarlas, arrojó los siguientes resultados:

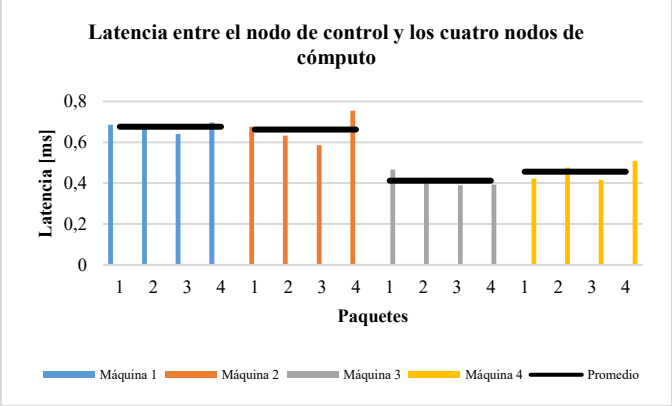

Gráfico 9. Latencia entre el nodo de control y los nodos de cómputo.

La importancia de medir esta latencia, a pesar de ser perceptiblemente baja, radica en que las instancias creadas en nodos distintos mostraron una latencia entre sí mayor que aquellas que fueron creadas en el mismo nodo, siendo la diferencia entre ambas latencias aproximadamente igual al promedio de latencias inter nodo medida. Los casos descritos serán mostrados a continuación:

b) Entre instancias creadas en el nodo de control

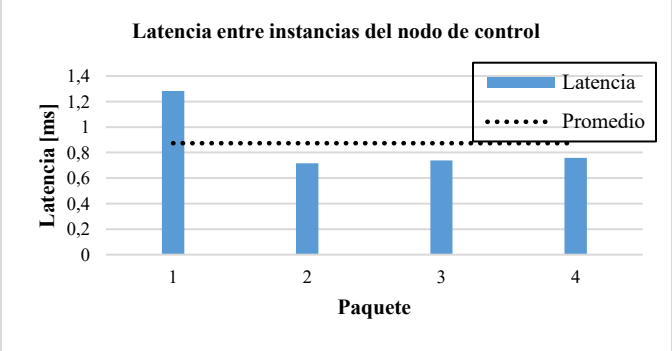

Gráfico 10. Latencia entre instancias creadas en el nodo de control, en milisegundos.

La latencia es baja porque, al estar embebidas en el mismo nodo de control, los paquetes sólo realizan un salto hacia la instancia de destino. Cabe destacar que en este caso se repite la mayor latencia del primer paquete.

c) Entre instancias creadas en el mismo nodo de cómputo

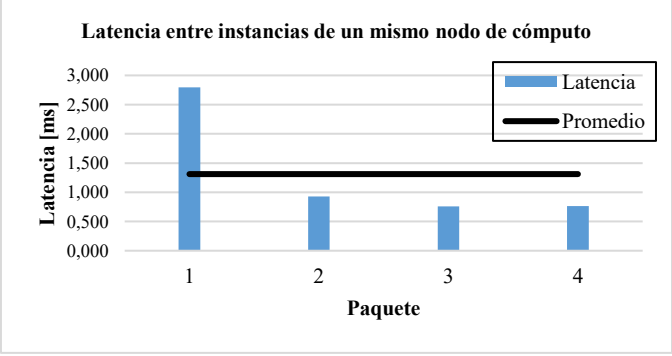

Gráfico 11. Latencia entre instancias creadas en el mismo nodo de cómputo.

Aquí se cumple el mismo criterio que en el nodo de control, sólo existe un salto entre los paquetes que recorren las instancias, y el primer paquete tarda más en ir y volver.

d) Entre instancia del nodo de control e instancia del nodo de cómputo

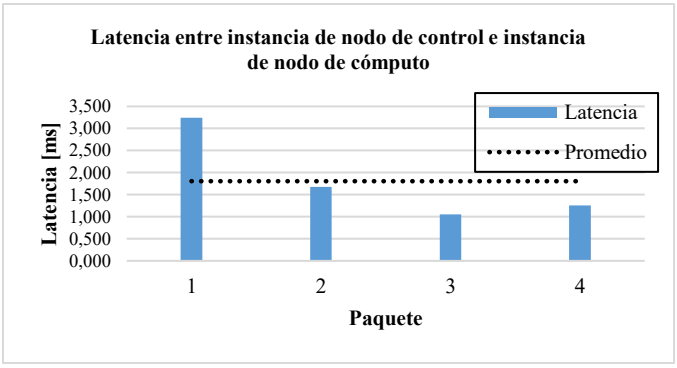

Gráfico 12. Latencia entre instancias del nodo de control y del nodo de cómputo.

Como puede observarse, las latencias crecen con respecto a los dos casos anteriores debido a que, al estar las dos instancias en cuestión embebidas en el nodo de control y en un nodo cómputo, los paquetes ejecutan dos saltos en su recorrido de una instancia a otra, añadiendo una latencia adicional similar a los promedios de las anteriores pruebas. La mayor duración del primer paquete también es reflejada en los datos.

e) Entre instancias de nodos de cómputo distintos

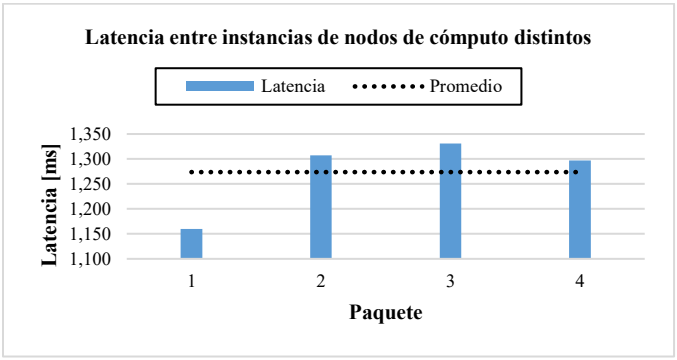

Gráfico 13. Latencia entre instancias de dos nodos de cómputo distintos.

En este caso, las latencias son similares al del caso anterior, debido a que los paquetes igualmente realizan dos saltos en su recorrido de una instancia hacia la otra. Nótese también que el primer paquete fue el de menor latencia, esto se debe a que, al momento de realizar esta última medición, ya se habían enviado y recibido algunos paquetes entre las instancias involucradas, por lo que el primer paquete reflejado en esta medición tardó en realizar su recorrido un tiempo similar al promedio, llamativamente en este caso, siendo el menor de todos.

4) Acceso remoto a la nube

De igual forma que en la "micro nube", se accedió a la infraestructura de la nube de forma remota por medio de la aplicación de SSH, *PuTTY*, desde una computadora adicional, conectada a la misma red de la nube. Igualmente, mediante el acceso SSH al nodo de control, como se muestra a continuación:

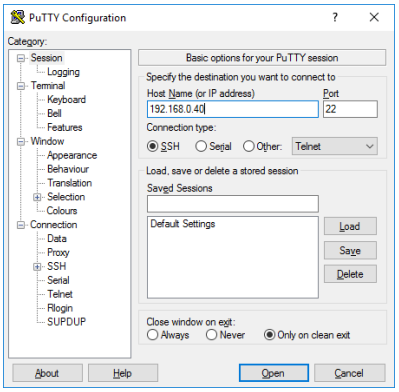

Fig. 19 Ingreso de dirección IP del nodo de control desde *PuTTY*.

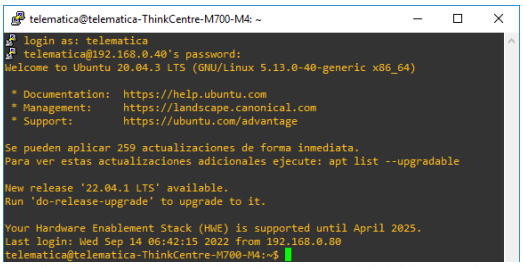

Fig. 20 Acceso SSH al nodo de control del laboratorio.

Pudiendo incluso, crear instancias desde *PuTTY*:

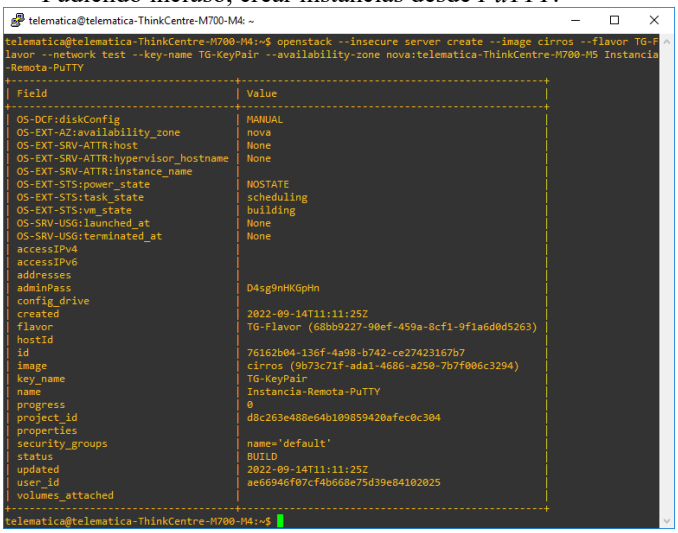

Fig. 21 Creación remota de instancia desde *PuTTY*.

De igual forma, accesibles vía SSH:

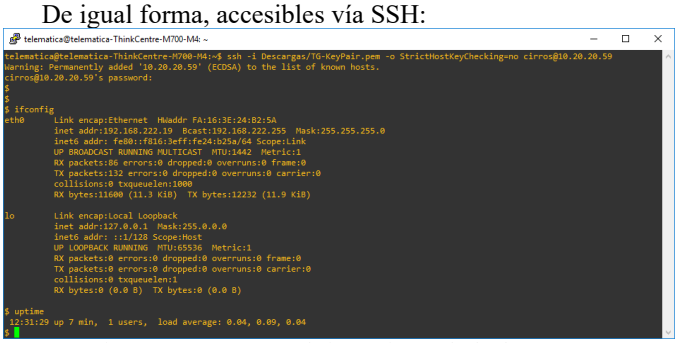

Fig. 22 Acceso remoto a la instancia creada desde *PuTTY*.

C. *Actividades de las cátedras de laboratorio que pueden sacar provecho a la nube*

1) *Diseño de redes y subredes basadas en IP*

La plataforma de *MICROSTACK/OPENSTACK* permite el diseño de redes de varias topologías y dimensiones de acuerdo con el tipo de despliegue que el usuario desee implementar. Se podría hacer uso de la herramienta para la creación de redes de distintas dimensiones con su respectivo direccionamiento IP, utilizando instancias y enrutadores como principales dispositivos virtualizados. El estudiante puede entenderse y familiarizarse con la virtualización de funciones de red y el diseño de las mismas de forma interactiva y didáctica.

2) *Enrutamiento estático*

La prueba de establecimiento de rutas de red estáticas realizada en el capítulo anterior, permitió que todas las instancias de la red se comuniquen entre sí, como se muestra a continuación, un ejemplo:

Fig. 23 Comunicación entre #TG-Instance-#1" y "TG-Instance-#16", pertenecientes a las redes "TG-Network-#1" y TG-Network-#6" respectivamente.

Las instancias de la figura anterior, están separadas física y lógicamente por los cuatro enrutadores de la topología, por lo que los paquetes debieron recorrer y realizar los saltos establecidos de forma estática en cada uno de ellos, que, al ser recibidos los paquetes ICMP, el enrutamiento estático se concluye como exitoso.

3) *Enrutamiento dinámico*

La prueba de enrutamiento dinámico, realizada con la aplicación *FRRouting*, arrojó los siguientes resultados:

a) *Establecimiento de rutas*

El protocolo OSPF, implementado en esta prueba, completó la convergencia de la red, al establecer las rutas dinámicas en las instancias enrutadoras en las que se equipó, como se muestra a continuación:

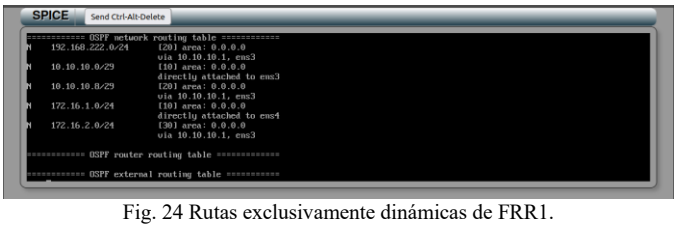

Fig. 24 Rutas exclusivamente dinámicas de FRR1.

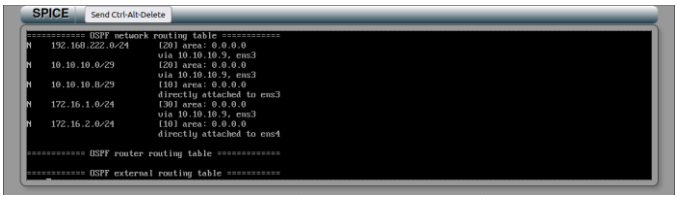

Fig. 25 Rutas exclusivamente dinámicas de FRR2.

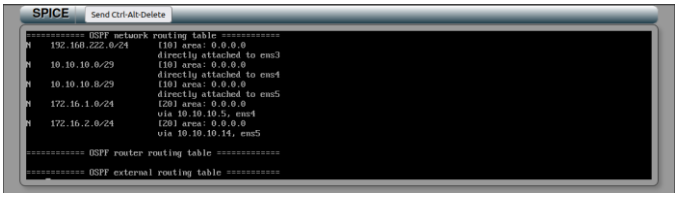

Fig. 26 Rutas exclusivamente dinámicas de FRRC.

Con el establecimiento de las rutas, cada instancia enrutadora conoce la ruta por la que deben redirigir los paquetes entrantes y salientes para llegar a cada una de las redes que conforman la topología, lo que quiere decir que el protocolo de enrutamiento dinámico sirvió de forma satisfactoria.

b) Comunicación ICMP

No obstante, la comunicación ICMP entre las instancias enrutadoras, arrojó los siguientes resultados:

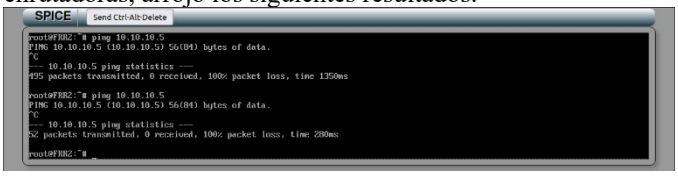

Fig. 27 Comunicación entre FRR2 y FRR1 reiterativa fallida.

Como puede observarse, pese a estar establecidas, las instancias no son capaces de intercambiar paquetes cuando deben saltar al menos una instancia enrutadora, a diferencia de las comunicaciones directas punto a punto, como se muestra a continuación:

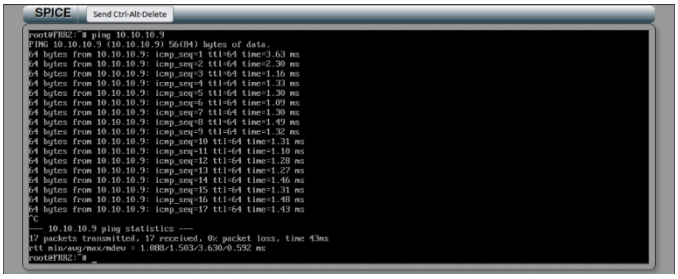

Fig. 28 Comunicación exitosa entre FRR2 y FRRC, adyacentes.

Para analizar a profundidad esta comunicación fallida, se procedió a ejecutar un protocolo de captura de paquetes dentro de la red, por medio del capturador "*tcpdump*". Se escogió analizar los paquetes en la instancia FRRC, debido a que es un elemento central de toda la topología, por el que deben pasar la mayoría de los paquetes para llegar desde un extremo de la red al otro, se escogió la interfaz "ens5", directamente conectada a FRR2. Se analizó en dos casos: cuando se realiza una solicitud ICMP que tiene como destino la interfaz descrita, y cuando tiene destino a FRR1:

Con solicitud ICMP con destino a dicha interfaz de FRRC

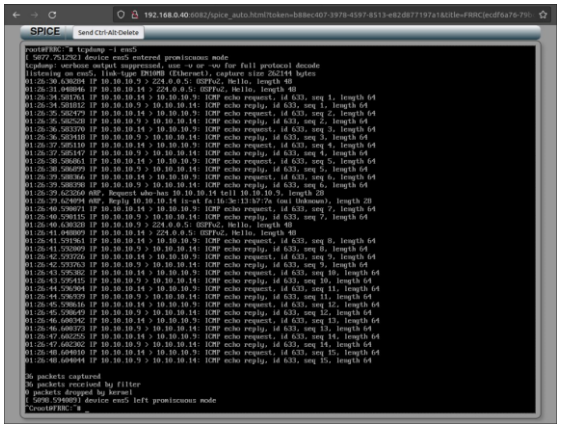

Fig. 29 Captura de paquetes ICMP con "*tcpdump*".

Como puede observarse, "*tcpdump*" captura los paquetes ICMP que se le llegaron directamente desde FRR2 a FRRC, al estar directamente conectados, los paquetes llegan sin ninguna dificultad, se distingue la distinción de los paquetes ICMP al observarse las solicitudes (*request*), respuestas (*reply*) y solicitudes de direcciones de capa de enlace (*Who has…*)

Con solicitud ICMP con destino a FRR1

Fig. 30 Captura general de paquetes en interfaz ens5 de FRRC.

Como puede observarse, la interfaz captura de forma continua los paquetes *hello* de OSPF con sus horas exactas de recepción. No obstante, nótese que la captura no registra los paquetes ICMP de ningún tipo, tanto de solicitud. como de respuesta y tampoco de solicitud de dirección de capa de enlace.

Cuando FRR2 intenta comunicarse con FRR1, FRRC no recibe sus paquetes de solicitud, por lo que no es capaz de reenviarlos hacia su destino en FRR1.

De esta situación, se concluye que el uso de instancias como elementos enrutadores, hace forzar la gestión de las funciones de red a *Nova*, gestor del cómputo de las instancias, en lugar de ser *Neutron* el gestor de las funciones de red, entre las que se incluye el direccionamiento de paquetes incluyendo la función de virtualización de elementos enrutadores embebida, de forma nativa; caso contrario de FRR, que es un *software* que fue forzado a usarse en la infraestructura de la nube.

4) Comparación entre los recursos utilizados originalmente para las prácticas de laboratorio con respecto a los recursos de la nube

Luego de haber desplegado la nube y analizar las distintas actividades de los laboratorios de telemática I y II, las prácticas pueden utilizar la infraestructura de la nube de la siguiente manera, mostrada en las siguientes tablas:

TABLA II

ACTIVIDADES DEL LABORATORIO DE TELEMÁTICA I QUE LE SACAN PROVECHO A LA NUBE [1].

Asig.	Práctica	Herramienta actual	¿Puede usarse la nube?	Observación / Uso de *MICROSTACK*
Lab. de telemática I	#1 - Análisis de las unidades de datos de protocolo (PDU) (modelo tcp/ip) usando *Wireshark*	*Wireshark*, analizador de paquetes TCP/IP	Sí	Es posible equipar las instancias con un analizador de paquetes, como "tcp-dump" nativo para Linux, o incluso el mismo *WireShark.*
	#2 - *Ethernet* clásica - Estudio de la capa de acceso al medio, el *hub*, dominios de colisión y protocolo de resolución de direcciones	No aplica	No	*MICROSTACK* no realiza simulación de dispositivos de capa física ni de acceso al medio.
	#3 - *Ethernet* conmutada - Estudio del switch, direcciones MAC y dominios de *Broadcast*.	CISCO *Packet Tracer*	No	El enfoque de diseño de las redes en *MICROSTACK* está en la capa de red, de forma nativa, este no provee virtualización de *switches* en los que se estudie la capa de enlace.
	#4 - Red de área local virtual (VLAN) - Reducción de dominios de broadcast en *switch.*	*Switches* físicos	No	*MICROSTACK* no provee simulación de dispositivos de capa enlace, como el *switch*.
	#5 - Estudio del protocolo IP - Direcciones IP, máscaras, subredes, VLSM.	Wireshark, analizador de paquetes TCP/IP	Sí	*MICROSTACK* ofrece una plataforma para la creación de redes de distintas dimensiones lógicas, con distintas máscaras de subred y delimitación de direcciones IP. Además de la utilización de analizadores de paquetes.
	#6 - Introducción al enrutamiento estático - Rutas por interfaz de salida, rutas por dirección de siguiente salto y rutas por defecto.	VirtualBox - Máquinas virtuales con S.O. basados en Linux	Sí	Por defecto, *Neutron*, módulo gestor de las funciones de red, ofrece la virtualización de routers en los que se puede configurar rutas estáticas. No obstante, sólo es posible por medio de la dirección IP de siguiente salto.
	#7 - Estudio de los protocolos TCP-UDP - Protocolos de capas superiores, conexiones cliente/servidor, servidores DHCP y SSH	VirtualBox - Máquinas virtuales con S.O. basados en Linux	Sí	*Neutron* tiene embebida una función de servidor DHCP para los despliegues realizados. Al poder crear instancias con imágenes de distintos sistemas operativos, pueden utilizarse para la ejecución de protocolos de este tipo, como el acceso remoto SSH.

TABLA III

ACTIVIDADES DEL LABORATORIO DE TELEMÁTICA II QUE LE SACAN PROVECHO A LA NUBE [2].

Asig.	Práctica	Herramienta actual	¿Puede usarse la nube?	Observación / Uso de *MICROSTACK*
Lab. de telemática II	#1 - Configuración básica del Router - Nombre de equipo, direcciones IP, contraseñas y guardados de configuración	*Graphical Network Simulator 3 (GNS3)* y *VirtualBox* con Máquinas Linux.	Sí	*Neutron* ofrece configuración básica de los routers vortuales que provee, no obstante, a nivel mucho más básico que el estudiado en la práctica. Si se desea estudiar a un nivel similar, es necesario utilizar una instancia con *software* de enrutamiento, como *Quagga* o *FRRouting*, que no funcionan del todo.
	#2 - Repaso de enrutamiento estático - Rutas con direcciones de siguiente salto, rutas con interfaz de salida, rutas con interfaces *Ethernet* y rutas predeterminadas.	CISCO *Packet Tracer*	Parcialmente	*Neutron* ofrece función de enrutamiento estático, no obstante, sólo se puede implementar con dirección de siguiente salto, y no se simula la especicación del tipo de medio físico para las interfaces de los *routers.*
	#3 - Protocolo de enrutamiento dinámico - RIPv2, convergencias, métricas y parámetros.	CISCO *Packet Tracer* y *WireShark*	Parcialmente.	Por defecto, *Neutron* no ofrece función de enrutamiento dinámico, por lo que se debe utilizar una instancia como router equipada con un software de enrutamiento, como *Quagga* o *FRRouting*, los mismos soportan RIPv2 y la convergencia de la red, sin embargo, sin comunicación ICMP.
	#4 - Protocolo *Open Shortest Path First* (OSPF) - Paquetes *hello*, adyacencias, vecindades, costos, configuraciones y parámetros.	CISCO *Packet Tracer* / VirtualBox con Máquinas Linux equipadas con *Zebra.*	Parcialmente	Al igual que con RIPv2, se puede usar una instancia Linux equipada con Quagga o FRRouting para implementar OSPF y observar los parámetros descritos y la convergencia de la red, no obstante, sin comunicación ICMP.
	#5 - Filtrado de Paquetes y NAT - Listas de control de acceso, bloqueo y habilitación de paquetes, reglas y cadenas de *firewall*	CISCO *Packet Tracer* / VirtualBox con Máquinas Linux equipadas con *Zebra.*	Parcialmente	*MICROSTACK* provee funciones de cadenas de *firewall* para las instancias creadas, no obstante, se pueden implementar de forma manual utilizando el firewall de Linux en las mismas. No obstante, si se usan instancias como routers, no habrá comunicación ICMP, como en los casos de los protocolos de enrutamiento dinámico.

VII. Conclusiones y Recomendaciones

Del presente proyecto realizado, se pueden destacar y concluir algunos aspectos importantes y relevantes, que parten desde detalles y observaciones sobre el desarrollo del producto y su utilización, hasta para implementaciones o despliegues similares al desarrollado o para futuras y factibles implementaciones o mejoras al producto actual.

A. Conclusiones

En la actualidad, el mundo de la telemática se encuentra expuesto al crecimiento continuo y de gran escala de las plataformas de nube, las cuales permiten suministrar servicios a clientes finales y proveedores de servicios de telecomunicaciones e Internet, a través de diferentes formas, dentro de las cuales resaltan la provisión de infraestructura como servicio (IaaS), de plataforma como servicio (PaaS), y de *software* como servicio (SaaS).

Teniendo presentes las ideas previas, se seleccionó el *software MICROSTACK* para el despliegue de la nube, y haciendo uso de este último se consiguió implementar en el Laboratorio de Telemática una infraestructura de nube con cinco (5) computadoras, las cuales cuentan con el sistema operativo UBUNTU; una actúa como nodo de control y de cómputo, y el resto como nodos de cómputo exclusivamente. A partir de dicha nube, fue posible reservar recursos de cómputo, almacenamiento y red.

Lo anteriormente descrito permitió determinar que *MICROSTACK* hace posible llevar a cabo la instalación simple de una versión preconfigurada y reducida de *OPENSTACK*, dando lugar al despliegue rápido y eficaz de una nube, que puede ser útil en escenarios donde se requiera una nube sin funcionalidades de mayor complejidad, y en caso de llegar a necesitarse, se puede ampliar la nube mediante el esquema convencional de instalación de módulos de *OPENSTACK*. La gestión de la nube no constituye mayor dificultad, pues *MICROSTACK* incorpora el *dashboard* o interfaz gráfica de *OPENSTACK*.

En el mismo orden de ideas previo, se debe mencionar que *MICROSTACK* permite el despliegue de nubes de diferentes dimensiones, desde nubes con un solo nodo, como la "micro nube", hasta nubes con múltiples nodos, como la implementada en el Laboratorio de Telemática.

La nube desplegada permitió hacer un uso más eficiente de los recursos de *hardware* de las computadoras del Laboratorio de Telemática, a la vez que incorporó en el banco de conocimiento de la Escuela los tópicos de computación en la nube, sin comprometer los usos cotidianos y prácticos que actualmente se les da a las máquinas, pues los procesos de gestión y operación de la nube tienen lugar en segundo plano.

Por otro lado, para evaluar el desempeño y rendimiento de la infraestructura de la nube, se realizaron un grupo de pruebas que consistieron en la medición de los tiempos de creación y encendido de instancias, y fueron comparados con los tiempos de creación y encendido de máquinas virtuales tradicionales, utilizando el software *VirtualBox*. En ambos entornos, las pruebas fueron realizadas utilizando dos configuraciones de memorias para las instancias, correspondientes a dos *flavors* distintos de *MICROSTACK*.

Tras medir los tiempos de creación de instancias y analizar sus resultados, se concluye que en la virtualización de los distintos recursos y funciones que provee la infraestructura de la nube para el aprovisionamiento de instancias con *MICROSTACK*, no se encontraron diferencias perceptibles en los tiempos de creación de las instancias con distintas configuraciones de *flavors,* creadas en los diferentes nodos que la conforman, y al ser comparados con los medidos en *VirtualBox*, se percibe una diferencia considerable en función de la configuración de memorias asignada a cada máquina virtual.

En cuanto a los tiempos de encendido de las instancias se concluye que la nube tiene un desempeño regular que depende directamente del tipo de nodo en el que se hayan creado las instancias. Si se crean en el nodo de control, los tiempos medidos son comparables, sin diferencias perceptibles, con los medidos en el entorno de *VirtualBox*; sin embargo, cuando las instancias se crean en un nodo de cómputo, el tiempo de encendido es considerablemente mayor, independientemente de la capacidad de memorias del *flavor* utilizado.

Adicionalmente, para complementar el desempeño de la implementación de la nube, se midieron las latencias existentes entre cada uno de los nodos que la conforman y las instancias creadas dentro de cada uno de los mismos. De las pruebas, se concluye que la implementación de nodos múltiples no supone una latencia importante que pueda lastrar mínimamente la comunicación de los elementos virtualizados.

Para evaluar la accesibilidad remota de la nube, empleando PuTTY desde una computadora externa a la nube, fue posible establecer conexión remota con el nodo de control, y desde esa sesión remota, como se refleja en los resultados, se logró interactuar con los recursos de la nube desde dicha computadora y ejecutar sus funciones, como la creación y eliminación de instancias, a las que incluso se logró acceder nuevamente con una nueva conexión SSH en cascada, luego de haber establecido la primera. Con estas acciones, se concluye que la nube es accesible de forma remota y puede gestionar sus recursos de esta forma con éxito.

Otra de las premisas principales por las que se implementó la nube fue evaluar su posible uso académico, para ello, se procedió a probar la efectividad de la nube al ejecutar un grupo de aplicaciones y funcionalidades impartidas en las prácticas de Laboratorio de Telemática I y II. De esta prueba se observó que, la infraestructura de nube implementada permite llevar a cabo, de manera limitada, algunos de los tópicos abordados en las prácticas de los laboratorios mencionados, más no todos.

Las pruebas realizadas sobre las prácticas de los Laboratorio de Telemática demostraron que es posible implementar esquemas de enrutamiento estático sin inconvenientes y con resultados positivos utilizando las funcionalidades embebidas del módulo *Neutron*; sin embargo, se encontraron limitantes al desplegar topologías que empleasen protocolos de enrutamiento dinámico como consecuencia de las restricciones del módulo *Neutron* en *MICROSTACK*, el cual no admite que las instancias actúen como enrutadores, al no permitir la

comunicación de paquetes entre instancias no adyacentes. Tras analizar estos resultados, se concluye que el proyecto tiene potencial para ser utilizado como la herramienta de estudio del tópico de implementación de esquemas de *cloud computing*, de manera independiente a las prácticas de laboratorio, que es la competencia que se quiere abordar y añadir al estudio académico en la carrera. Esto permitirá desarrollar las habilidades de implementación de nubes, mediante experiencias de ejemplo y modelo de forma independiente a las prácticas de laboratorio ya existentes.

B. Recomendaciones

Luego de finalizar el recorrido de la realización del presente proyecto, para futuras implementaciones similares o mejoras del mismo, se recomiendan los siguientes aspectos.

1) Conocimiento del S.O. UBUNTU

Fundamentalmente, se recomienda la documentación o estudio previo de los comandos de consola de UBUNTU u otros sistemas derivados de LINUX, específicamente aquellos relacionados a: Manipulación de archivos y directorios, jerarquías de permisos, funciones de red (direccionamiento IP, enrutamiento, captura de paquetes), *firewall*, e instalación de paquetes y programas.

2) Recomendaciones de entorno de despliegue

Para combatir y evitar las demás eventualidades mencionadas, externas al *software* de UBUNTU, se recomienda tomar las siguientes acciones y medidas para acondicionar el entorno:

- Disponer de una versión reciente del sistema UBUNTU, al menos la versión 20.04.3 LTS, y preferiblemente la última disponible con soporte a largo plazo. Se recomienda equipar la misma en todos los nodos.
- Utilizar la misma versión de la aplicación *MICROSTACK*, proveniente del mismo canal de instalación, se recomienda específicamente la versión que se indica en la página oficial de los desarrolladores: --beta [42].
- Utilizar una infraestructura de red libre de cualquier *software* de gestión masiva de seguridad en la red. Evitar los corta fuegos estrictos, servidores proxy, etc.
- Implementar direccionamiento IP estático, no variable.
- Verificar el óptimo funcionamiento de los medios físicos de transmisión de datos, tanto guiados, como no guiados:
  - De utilizar comunicación inalámbrica, disminuir lo más posible la cantidad de obstáculos que generen pérdidas o interferencia en las señales de radio, que puedan repercutir en la conexión de los nodos.
  - De utilizar comunicación cableada, asegurar que los cables *Ethernet* o *patch cords*, estén bien construidos, con las puntas y conectores bien colocados y sin roturas o dobleces bruscos.

3) Recomendaciones para presunta mejora del proyecto

Estas recomendaciones contemplan varios aspectos a tomar en cuenta si se considera implementar una mejora en un futuro a mediano plazo del proyecto, distribuidas en las siguientes categorías:

Para ampliar el abanico de funciones de cada módulo del *software* gestor de la nube, se recomienda lo siguiente:

- Descargar cada módulo de *OPENSTACK* de forma independiente, fuera del paquete *snap* de *MICROSTACK*.
- Reutilizar los módulos previamente instalados en *MICROSTACK*.

a) Para funcionalidad de enrutamiento dinámico

Se recomienda implementar las funciones que ofrece la instalación independiente del módulo *Neutron* de *OPENSTACK*. El mismo, cuenta con un conjunto de funciones embebidas, deshabilitadas por defecto, que permiten implementar protocolos de enrutamiento dinámico, específicamente el Protocolo de Puerta de Enlace de Frontera (*Border Gateway Protocol*, BGP). Existe documentación oficial de *Neutron* para implementar este protocolo [43] [44], por lo que se recomienda su revisión y aplicación en caso de necesitar su uso.

4) Recomendaciones para el uso académico de la nube

a) Estudio independiente de las prácticas

Si se reservan espacios en las asignaturas de laboratorio para el estudio de la nube utilizando *MICROSTACK*, se recomiendan las siguientes acciones:

- Elaborar prácticas específicas para llevar a cabo el estudio de la implementación y uso de la infraestructura de nube.
- Evitar migrar completamente las prácticas actuales desde las herramientas de su estudio para la infraestructura de la nube.
- No adaptar *software* que no sea compatible de forma nativa con *MICROSTACK*, o que no esté embebido en el mismo, debido a los conflictos de compatibilidad que pueden presentarse.

b) Forma sugerida de segmentación de asignaturas, prácticas y usuarios

Esta plataforma, además de los servicios que provisiona, podría soportar el segmentado de los múltiples proyectos, diseños e implementaciones que el/los estudiantes de las diferentes asignaturas desarrollen por medio de la gestión de identidades de uso de la nube pertenecientes al servicio *Keystone*, tales, como: Dominios, proyectos, grupos, usuarios y contraseñas.

La gestión de los proyectos y prácticas de laboratorio que usen la nube podrían gestionar sus respectivas identidades de la siguiente forma ejemplificada:

- Creación de múltiples **dominios**, uno para cada **asignatura**.
- Creación e inclusión de **usuarios** en su **dominio respectivo**, **de acuerdo a la asignatura** que estén cursando.
- Creación de **grupos** de usuarios por cada **mesa del laboratorio** y su respectiva inclusión de los usuarios correspondientes a los estudiantes que las conformen.
- Creación y distinción de **proyectos** por **cada práctica de laboratorio**, con sus respectivos elementos privados o compartidos, de acuerdo si se necesitan utilizar en prácticas posteriores.

REFERENCIAS

[1] Universidad Católica Andrés Bello, "Programa de Asignatura - Telemática I," [Online]. Available: http://w2.ucab.edu.ve/tl_files/Ingenieriatelecom/Pensum/sexto_semestre/TELE-00047.pdf. [Accessed 2 Octubre 2021].

[2] Universidad Católica Andrés Bello, "Programa de Asignatura - Telemática II," [Online]. Available: http://w2.ucab.edu.ve/tl_files/Ingenieriatelecom/Pensum/septimo_semestre/TELE-00052.pdf. [Accessed 2 Octubre 2021].

[3] Universidad Católica Andrés Bello, "Programa de Asignatura - Telemática IV," [Online]. Available: http://w2.ucab.edu.ve/tl_files/Ingenieriatelecom/Pensum/noveno_semestre/TELE-00089.pdf. [Accessed 2 Octubre 2021].

[4] Canonical Limited, "*MICROSTACK* - *OPENSTACK* in a snap," Canonical Limited, [Online]. Available: https://*MICROSTACK*.run/docs. [Accessed 2 Octubre 2021].

[5] D. A. Tong and K. Wade, "Guía de NFV y SDN para operadores y proveedores de servicio," 2017. [Online]. Available: https://media.ciena.com/documents/Blue+Planet+Essentials_NFV+and+SDN+Guide_033017_A5_es_LA.pdf. [Accessed 2 Octubre 2021].

[6] D. Marinescu, Cloud ComputingTheory and Practice, Cambridge, MA.: Elsevier Inc., 2018.

[7] B. Kezherashvili, "Computación en la Nube," *trabajo de fin de máster.*

[8] ManagementSolutions, "La nube: oportunidades y retos para los integrantes de la cadena de valor," 2012. [Online]. Available: https://www.managementsolutions.com/sites/default/files/publicaciones/esp/La-nube.pdf. [Accessed 2 Octubre 2021].

[9] Microsoft Azure, "¿Qué es la nube?," [Online]. Available: https://azure.microsoft.com/es-es/overview/what-is-the-cloud/. [Accessed 2 Octubre 2021].

[10] A. S. Tanembaun, Sistemas operativos modernos, Tercera ed., Naucalpan de Juárez: PEARSON EDUCACIÓN, 2009.

[11] Ubuntu, "Our Mission | Ubuntu," 2022. [Online]. Available: https://ubuntu.com/community/mission. [Accessed 30 Enero 2022].

[12] *OPENSTACK*, "Open Source Cloud Computing Platform Software - *OPENSTACK*," [Online]. Available: https://www.*OPENSTACK*.org/software/. [Accessed 2 Octubre 2021].

[13] Red Hat, "El concepto de *OPENSTACK*," Red Hat, Inc., [Online]. Available: https://www.redhat.com/es/topics/*OPENSTACK*. [Accessed 2 Octubre 2021].

[14] *OPENSTACK*, "Download The *OPENSTACK* Logo - *OPENSTACK* is open source software for creating private and public clouds," 2022. [Online]. Available: https://www.*OPENSTACK*.org/brand/*OPENSTACK*-logo/logo-download/. [Accessed 24 Septiembre 2022].

[15] Canonical Ltd., "*MICROSTACK* - *OPENSTACK* in a snap," Canonical Ltd., [Online]. Available: https://*MICROSTACK*.run/. [Accessed 2 Octubre 2021].

[16] Canonical Limited, "Learn about *OPENSTACK* services and their functions | Ubuntu," 2022. [Online]. Available: https://ubuntu.com/tutorials/learn-about-*OPENSTACK*-services-and-their-functions#3-explore-the-keystone-service. [Accessed 11 Marzo 2022].

[17] Universidad Pedagógica Experimental Libertador, Manualde Trabajos de Gradode Especialización y Maestría y Tesis Doctorales, Caracas: Fondo Editorial de la Universidad Pedagógica Experimental Libertador, 2016.

[18] Canonical Limited, "*OPENSTACK* tutorials on Ubuntu | *OPENSTACK* | Ubuntu," 2022. [Online]. Available: https://ubuntu.com/*OPENSTACK*/tutorials. [Accessed 24 Julio 2022].

[19] Canonical Ltd, "Manage instances templates, including images and flavors | Ubuntu," 2022. [Online]. Available: https://ubuntu.com/tutorials/manage-instance-templates-including-images-and-flavors#2-manage-images. [Accessed 13 Marzo 2022].

[20] Debian, "Index of /cdimage/*OPENSTACK*," 2022. [Online]. Available: http://cdimage.debian.org/cdimage/*OPENSTACK*/. [Accessed 30 Agosto 2022].

[21] CentOS, "CentOS Cloud images," 2022. [Online]. Available: https://cloud.centos.org/centos/9-stream/. [Accessed 30 Agosto 2022].

[22] Fedora, "Fedora Cloud," 2022. [Online]. Available: https://alt.fedoraproject.org/cloud/. [Accessed 30 Agosto 2022].

[23] Microsoft Corporation, "Download Windows 10," 2022. [Online]. Available: https://www.microsoft.com/en-us/software-download/windows10. [Accessed 30 Agosto 2022].

[24] e. a. Kunihiro Ishiguro, "Quagga - A routing software package for TCP/IP networks," [Online]. Available: https://cs.uns.edu.ar/~ldm/data/rc/info/quagga.pdf. [Accessed 9 Septiembre 2022].

[25] FRRouting Project, "FRRouting," 2022. [Online]. Available: https://frrouting.org/. [Accessed 16 Septiembre 2022].

[26] Canonical Limited, "Install *OPENSTACK* on your workstation and launch your first instance | Ubuntu," 2022. [Online]. Available: https://ubuntu.com/tutorials/install-*OPENSTACK*-on-your-workstation-and-launch-your-first-instance#1-overview. [Accessed 10 Julio 2022].

[27] OpenNebula Systems, "OpenNebula – Open Source Cloud & Edge Computing Platform," [Online]. Available: https://opennebula.io/. [Accessed 08 Julio 2022].

[28] *OPENSTACK*, "Welcome to neutron-dynamic-routing's documentation! --- neutron-dynamic-routing 21.1.0.dev2 documentation," 2022. [Online]. Available: https://docs.*OPENSTACK*.org/neutron-dynamic-routing/latest/index.html. [Accessed 18 Septiembre 2022].

[29] *OPENSTACK*, "BGP dynamic routing --- Neutron 21.1.0.dev2 documentation," 2022. [Online]. Available: https://docs.*OPENSTACK*.org/neutron/latest/admin/config-bgp-dynamic-routing.html. [Accessed 18 Septiembre 2022].

[30] Universidad Católica Andrés Bello, "Malla Curricular Escuela de Ingeniería de Telecomunicaciones Octubre 2021," 20 Septiembre 2021. [Online]. Available: http://w2.ucab.edu.ve/tl_files/Ingenieriatelecom/202215/Grafo%202022215.pdf. [Accessed 30 Septiembre 2021].

[31] BBC News Mundo, "Coronavirus | Maduro ordena la cuarentena de Caracas y de otros 6 estados de Venezuela por el covid-19," BBC, 15 Marzo 2020. [Online]. Available: https://www.bbc.com/mundo/noticias-america-latina-51902733. [Accessed 2 Octubre 2021].

[32] D. J. W. Andrew S. Tanenbaum, Redes de Computadoras, Quinta ed., Naucalpan de Juárez: PEARSON EDUCACIÓN, 2012.

[33] Oracle Corporation España, "Introducción al conjunto de protocolos TCP/IP - Guía de administración del sistema: Servicios IP," 2010. [Online]. Available: https://docs.oracle.com/cd/E24842_01/html/820-2981/ipov-6.html#ipov-13. [Accessed 13 Marzo 2022].

[34] Á. López Villegas, *Modelos OSI y TCP/IP,* Caracas: Material de clase, Escuela de Ingeniería en Telecomunicaciones, Universidad Católica Andres Bello, 2019.

[35] *MICROSTACK*, "*OPENSTACK* for the edge, micro clouds and developers | Single-Node," 2022. [Online]. Available: https://*MICROSTACK*.run/docs/single-node. [Accessed 14 Junio 2022].

[36] *MICROSTACK*, "*OPENSTACK* for the edge, micro clouds and developers | Multi-node," 2022. [Online]. Available: https://*MICROSTACK*.run/docs/multi-node. [Accessed 14 Junio 2022].