

# Proyecto 1: Diseño e implementación de una red de datos

Santiago Andres Mesa Niño

Giovanny Andres Duran Renteria

Facultad de Ingeniería

Pontificia Universidad Javeriana

Bogotá D.C., Colombia

[giovanny.durandr@javeriana.edu.co](mailto:giovanny.durandr@javeriana.edu.co)

o

Facultad de Ingeniería

Pontificia Universidad Javeriana

Bogotá D.C., Colombia

[santiagoa.mesan@javeriana.edu.co](mailto:santiagoa.mesan@javeriana.edu.co)

o

Samuel Nemes Moreno

Facultad de Ingeniería

Pontificia Universidad Javeriana

Bogotá D.C., Colombia

[s\\_nemes@javeriana.edu.co](mailto:s_nemes@javeriana.edu.co)

Luna Rengifo Moreno

Facultad de Ingeniería

Pontificia Universidad Javeriana

Bogotá D.C., Colombia

[lrengifom@javeriana.co](mailto:lrengifom@javeriana.co)

**Abstract—** This project designs and implements the data network for Company XYZ, ensuring efficient communication between two buildings and all their departments. Technologies such as VLANs, NAT, VLSM, and IPv4 and IPv6 protocols are used, in addition to redundant links to ensure availability. Functional WEB, DHCP, and DNS servers are also configured. The entire design is developed and tested in GNS3, with complete technical documentation in IEEE format.

**Keywords—** Diseño de redes, GNS3, VLAN, NAT, VLSM, IPv4, IPv6, redundancia, Spanning Tree Protocol, DHCP, DNS, servidor web, infraestructura de red, simulación de redes, documentación técnica, conectividad IP

## INTRODUCCIÓN

En el contexto actual de transformación digital, las redes de datos juegan un papel fundamental en el soporte operativo de las organizaciones. El presente proyecto plantea el diseño y la implementación de una red de datos para la empresa XYZ, con el fin de mejorar la comunicación interna, la disponibilidad de servicios y el acceso seguro a internet. Adicionalmente, el diseño contempla la infraestructura de dos edificios, la creación de subredes específicas para cada departamento, la implementación de VLAN's para la movilidad de los usuarios, y la configuración de servidores críticos bajo un esquema de direccionamiento IP propio. Finalmente, la solución propuesta

busca garantizar la redundancia, la escalabilidad y la alta disponibilidad, utilizando herramientas de simulación como GNS3 y aplicando los conceptos aprendidos en la asignatura de Comunicaciones y Redes.

## DISEÑO DE LA RED

### A. Topología inicial

El diseño inicial de la topología de red puede apreciar en la figura uno en donde se simuló ambos edificios, segmentados tanto por ubicación (sur y norte), Y por los pisos (1 y 2), igualmente se segmentó con cuadrados de diferentes colores a qué tipo de edificio, habitación y VLAN Correspondían, por ejemplo el edificio uno es el cuadrado grande amarillo y la VLAN de gerencia a pesar de que está en los 2 edificios está demarcada por el rectángulo morado dando referencia a que pertenecen al mismo departamento, este diseño de red fue mejorado eventualmente para poder visualizar e implementar la redundancia en la red para evitar complicaciones futuras en caso de que algún enlace se ve afectado.

Adicionalmente en cada piso de cada edificio está un switch que es el que se encarga de distribuir las peticiones de los diferentes departamentos según su VLAN y se escogió el switch que está en el piso 2 del edificio uno (norte) sea el que se enlazan con el router para comunicarse con el nat y así salir a internet.

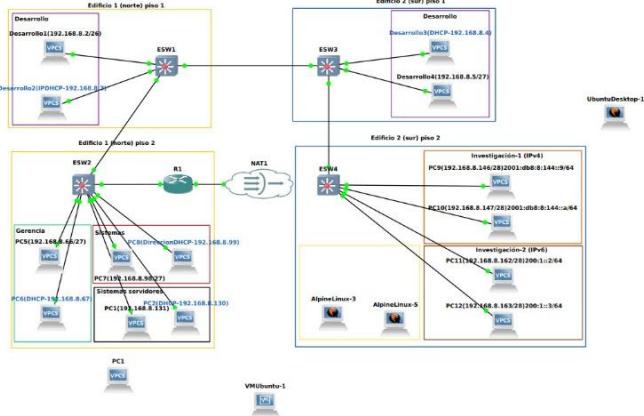


Figura 1 Diseño inicial de la topología de red

También, se planteó la siguiente tabla de enrutamiento, con el uso de VLSM y teniendo en cuenta todos los hosts necesarios para cada VLAN, dándolo como resultado:

VLAN	Nombre	Ubicación	Subred	Nº Host	Puerta de enlace	Rango de IPs (Sin subred, puerta de enlace ni broadcast)
2	Desarrollo	Norte (piso 1) y Sur (piso 1)	192.168.8.0 /26	40	192.168.8.1	192.168.8.2 - 192.168.8.62
3	Gerencia	Norte (piso 2)	192.168.8.64 /27	30	192.168.8.65	192.168.8.66 - 192.168.8.94
4	Sistemas	Norte (piso 2)	192.168.8.96 /27	30	192.168.8.97	192.168.8.98 - 192.168.8.126
5	Servidores (NO DNS, HTTP, DHC P)	Norte (piso 2)	192.168.8.128 /28	10	192.168.8.129	192.168.8.130 - 192.168.8.142
6	Investigación A	Sur (piso 2)	192.168.8.144 /28	10	192.168.8.145	192.168.8.146 - 192.168.8.158
7	Investigación B	Sur (piso 2)	192.168.8.160 /28	10	192.168.8.161	192.168.8.162 - 192.168.8.174

Tabla 1 Direcciones y distribuciones de la red

## B. Implementación de IPv4 e IPv6 en el departamento de Investigación

Lo primero que se debe tomar en consideración, es que el requerimiento implica el uso de una subred de investigación, la cual, esté dividida en 2, una de ellas con la finalidad de usar IPv6 y la otra con la finalidad de usar IPv4.

Para cumplir con este requerimiento, creamos 2 VLANs para cada división como se evidencia en la tabla de distribución de direcciones del proyecto, pero el requerimiento de IPv6 no se cumple hasta este punto. Ya después de tener configuradas todas las direcciones estáticas de IPv4, es necesario hacer uso del “Dual stack”, el cual, nos permite tener 2 tipos de direcciones en un mismo host, es decir, una dirección IPv4 y otra IPv6.

Primero, debemos configurar la subred que acogerá el protocolo IPv6, que, en este caso, será la de “Investigación 2”, añadiéndole una dirección IPv6 (puerta de enlace) para que se tenga el dual stack funcional.

Se debe ejecutar en el router el siguiente comando para que permita la entrada y enrutamiento de direcciones IPv6:

```
R1#enable
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#ipv6 unicast-routing
```

Figura 2 Enrutamiento de direcciones IPv6

Ahora, nos paramos en la interfaz de nuestra VLAN a configurar, en el puerto que tiene el enlace troncal:

```
R1(config)#interface fastEthernet 0/0.7
```

Figura 3 Enlace troncal

Se hará uso de la dirección 200:1::1/64, usada en clase, y a su vez, comprobamos que la dirección se esté guardando de manera correcta para nuestra subred deseada.

```

R1(config-subif)#ipv6 address 200:1::1/64
R1(config-subif)#do show ipv6 interface brief
FastEthernet0/0 [up/up]
FastEthernet0/0.2 [up/up]
FastEthernet0/0.3 [up/up]
FastEthernet0/0.4 [up/up]
FastEthernet0/0.5 [up/up]
FastEthernet0/0.6 [up/up]
FastEthernet0/0.7 [up/up]
  FE80::CE05:F9FF:FE50:0
  200:1::1
FastEthernet0/1 [up/up]
FastEthernet1/0 [administratively down/down]

```

Figura 4 Configuración Ipv6

Lo siguiente, será configurar las direcciones IP en cada uno de los VPCs de investigación, colocándoles una dirección IPv4 para que los demás puedan acceder a ella, y una dirección IPv6 como se pidió en el enunciado para la comunicación entre las máquinas de este departamento.

#### C. Enlaces redundantes y Spanning Tree

En nuestro contexto, los enlaces redundantes hacen referencia a la implementación de conexiones adicionales entre dispositivos de red, tales como switches y routers, con el fin de garantizar la disponibilidad y continuidad del servicio ante posibles fallos en los enlaces principales [1]. Esta práctica forma parte de los mecanismos de tolerancia a fallos, y busca asegurar que la red mantenga su funcionamiento incluso si uno de los enlaces experimenta una interrupción.

Estos enlaces redundantes se ven reflejados en la topología de nuestro proyecto, con los enlaces entre los switches como se evidencia en la figura 4.

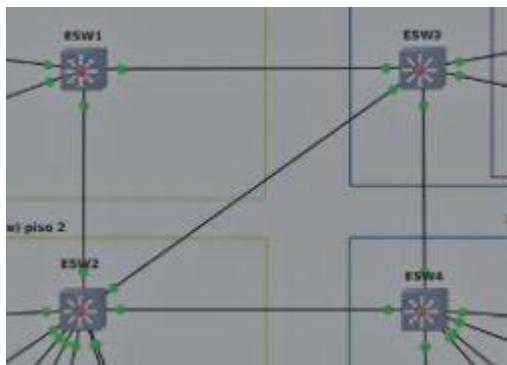


Figura 5 Diagrama de Conexión

La existencia de enlaces redundantes, sin embargo, puede derivar en un problema crítico: la formación de bucles de red. Estos bucles generan una circulación indefinida de tramas entre

los dispositivos conmutadores, lo cual satura la red, degrada el rendimiento y puede producir una caída total de los servicios [2]. Para evitar este comportamiento, se emplea el Protocolo Spanning Tree (STP).

El Spanning Tree Protocol (STP) es un protocolo de capa de enlace (Capa 2 del modelo OSI), definido por el estándar IEEE 802.1D, cuyo objetivo es prevenir los bucles de red en topologías conmutadas mediante la creación de una topología lógica sin bucles [3]. STP funciona identificando una estructura jerárquica denominada árbol de expansión (spanning tree), en la cual se designa un switch raíz (root bridge) y se determina un único camino activo hacia cada segmento de la red, bloqueando temporalmente los enlaces redundantes que no son necesarios en condiciones normales de operación [1].

En caso de que uno de los enlaces activos falle, el protocolo es capaz de recalcular la topología y reactivar uno de los enlaces previamente bloqueados, restableciendo la conectividad sin intervención manual. De este modo, STP equilibra la necesidad de redundancia con la estabilidad de la red [2].

## INSTALACIÓN Y CONFIGURACIÓN DE LOS SERVIDORES

### A. Servidor DNS

El Sistema de Nombres de Dominio (DNS por sus siglas en inglés) es el componente de las redes de datos, encargado de traducir nombres de dominio legibles, como lo es google.com, en direcciones IP comprensible para las maquinas. Su funcionamiento se establece mediante una estructura jerárquica y distribuida que permite consultar información de nombres de dominio en diversos niveles, empezando por servidores raíz hasta autoritativos. Un servidor DNS puede operar en diferentes roles, en los cuales encontramos, cache, reenviado, autoritativo y además puede gestionar zonas directas e inversas. Adicional su implementación en redes privadas y/o educativas facilita que la organización lógica de los dispositivos, el monitoreo de tráfico y la resolución de nombres locales no dependan

directamente de internet [4]. El servidor DNS también puede configurarse para responder a consultas desde IPv4 o IPv6, garantizando una alta disponibilidad gracias a su arquitectura [5].

Para llevar a cabo la gestión de consultas DNS necesarias dentro de la red definida para el proyecto, fue necesario instalar el paquete BIND9 en el sistema operativo Ubuntu de la Máquina Virtual con dirección 10.43.100.184. El Dominio de Nombres de Internet de Berkeley (BIND por sus siglas en Inglés) es el paquete que permite la resolución de nombres tanto directos como inversos, el manejo de zonas autoritativas y la configuración de reenviadores, lo cual es ideal para la elaboración del servidor DNS y la implementación que se requiere en el proyecto.

Tal como se muestra en la Figura 6, se utilizó el comando sudo apt install bind9-utils nano para realizar la instalación del paquete junto con el editor de texto *nano*, el cual se utilizará para más adelante para modificar archivos de configuración importantes.

```
vboxuser@ubuntu:~$ sudo apt install bind9-utils nano
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
nano ya está en su versión más reciente (7.2-2ubuntu0.1).
Eljado nano como instalado manualmente.
Se instalarán los siguientes paquetes NUEVOS:
bind9-utils
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 56 no actualizados.
Se necesita descargar 159 kB de archivos.
Se utilizarán 651 kB de espacio de disco adicional después de esta operación.
Desea continuar? [S/n] S
Get: http://archive.ubuntu.com/ubuntu nubie-updates/main amd64 bind9-utils amd64 1:9.18.30-0ubuntu0.24.04.2 [159 kB]
Descargados 159 kB en 1s (209 kB/s).
Seleccionando el paquete bind9-utils previamente no seleccionado.
(Leyendo la base de datos ... 63828 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../bind9-utils_1:9.18.30-0ubuntu0.24.04.2_amd64.deb ...

```

Figura 6 Instalación del paquete *bind9-utils* y *nano* mediante el gestor de paquetes APT en Ubuntu.

Una vez terminada la instalación, se verificó que el servicio asociado a BIND9 estuviera activo y en ejecución. Esto con el fin de garantizar que el servidor DNS se encuentra funcionando correctamente y este preparado para resolver consultas dentro de la red.

La Figura 7 muestra la ejecución del comando sudo systemctl status bind9, el cual confirma que el servicio

*named.service* se encuentra activo (active) y en ejecución (running), lo cual es muy importante antes de proceder con la configuración en materia del servidor.

```
vboxuser@ubuntu:~$ systemctl status bind9
● named.service - BIND Domain Name Server
   Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-04-17 16:49:59 UTC; 2min 18s ago
     Docs: man:named(8)
     Main PID: 2302 (named)
        Status: "running"
           Tasks: 17 (limit: 10455)
          Memory: 9.1M (peak: 10.1M)
            CPU: 176ms
           CGroup: /system.slice/named.service
                   └─2302 /usr/sbin/named -f -u bind

abr 17 16:49:59 ubuntu named[2302]: managed-keys-zone: loaded serial 0
abr 17 16:49:59 ubuntu named[2302]: zone 0.in-addr.arpa/IN: loaded serial 1
abr 17 16:49:59 ubuntu named[2302]: zone 127.in-addr.arpa/IN: loaded serial 1
abr 17 16:49:59 ubuntu named[2302]: zone 255.in-addr.arpa/IN: loaded serial 1
abr 17 16:49:59 ubuntu named[2302]: zone localhost/IN: loaded serial 2
abr 17 16:49:59 ubuntu named[2302]: all zones loaded
abr 17 16:49:59 ubuntu named[2302]: running
abr 17 16:49:59 ubuntu systemd[1]: Started named.service - BIND Domain Name Server.
abr 17 16:50:09 ubuntu named[2302]: managed-keys-zone: Unable to fetch DNSKEY set '.'.: timed out
abr 17 16:50:09 ubuntu named[2302]: resolver priming query complete: timed out
```

Figura 7 Verificación del estado del servicio BIND9 mediante *systemctl*.

Antes de permitir que el servidor DNS pueda recibir consultas, es necesario asegurarse de que el firewall permita dichas conexiones. Para ello, se ejecutó el comando sudo ufw allow bind9, lo cual habilita tanto el tráfico IPv4 como IPv6 hacia el servicio BIND9. Lo dicho anteriormente se muestra en la Figura 8.

```
vboxuser@ubuntu:~$ sudo ufw allow bind9
Rules updated
Rules updated (v6)
```

Figura 8 Comando para permitir tráfico DNS a través del firewall con UFW.

Posteriormente, se editó el archivo principal de configuración de opciones de BIND9, ubicado en /etc/bind/named.conf.options. Para ello, se empleó el editor de texto *nano* instalado anteriormente y como se evidencia en la Figura 9.

```
vboxuser@ubuntu:~$ sudo nano /etc/bind/named.conf.options
```

Figura 9 Apertura del archivo de configuración *named.conf.options* con privilegios de superusuario.

Dentro de este archivo se agregaron o modificaron varias líneas importantes, tal como se muestra en la Figura 10. En primer lugar, se definió la línea allow-query { any; }; la cual permite que cualquier dispositivo (sin importar su IP o protocolo, ya sea IPv4 o IPv6) pueda realizar consultas al servidor DNS. Este empleamiento es fundamental para garantizar que tanto las estaciones de trabajo como los dispositivos de prueba puedan comunicarse correctamente con el servidor DNS implementado.

```
allow-query { any; };
forwarders {
```

Figura 10 Fragmento del archivo named.conf.options con las linea clave incorporada para permitir consultas.

Después, se añadieron los reenviadores (forwarders) dentro del mismo archivo mencionado anteriormente, con el fin de especificar servidores DNS externos a los que se reenviarán las consultas que el servidor no pueda resolver de manera local. Esto se muestra en la Figura 11.

```
forwarders {
    8.8.8.8;
    8.8.4.4;
    10.2.1.10;
    10.2.1.30;
};
```

Figura 11 Configuración de los servidores reenviadores dentro de named.conf.options.

Para el caso de este servidor DNS, se configuraron cuatro direcciones IP: las dos primeras corresponden a los servidores públicos de Google (8.8.8.8 y 8.8.4.4), y las dos últimas a los servidores DNS primario y secundario de la universidad (10.2.1.10 y 10.2.1.30).

Luego, se comentó la línea dnssec-validation auto; y se dejó activa la opción dnssec-validation no;, como se observa en la Figura 12. Modificación necesaria para responder al hecho de que, al estar en un entorno académico, el servidor DNS

únicamente reenviará consultas sin validar firmas digitales (DNSSEC), ya que si se usara limitaría el ámbito del proyecto.

```
#dnssec-validation auto;
dnssec-validation no;
```

Figura 12 Desactivación de validación DNSSEC por tratarse de un entorno académico controlado.

Finalmente, se dejaron activas las líneas listen-on { any; } y listen-on-v6 { any; }, lo que permite al servidor DNS escuchar peticiones en todas las interfaces de red, tanto para IPv4 como para IPv6. Esto se muestra en la Figura 13.

```
listen-on { any; };
listen-on-v6 { any; };
```

Figura 13 Configuración para que el servidor DNS escuche en IPv4 y IPv6.

Una vez configurado el archivo principal del servidor DNS, se realiza una verificación utilizando el comando named-checkconf. Esto con el fin de detectar si quedaron errores de escritura antes de reiniciar el servidor. Después, se reinicia el servicio bind9 y se comprueba su estado para confirmar que está corriendo correctamente, tal como se observa en la Figura 14.

```
uboxuser@ubuntu:~$ sudo named-checkconf
uboxuser@ubuntu:~$ sudo systemctl restart bind9
uboxuser@ubuntu:~$ systemctl status bind9
● named.service - BIND Domain Name Server
   Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-04-17 17:35:28 UTC; 25s ago
     Docs: man:named(8)
 Main PID: 2558 (named)
   Status: "running"
      Tasks: 12 (limit: 10455)
     Memory: 6.8M (peak: 7.8M)
        CPU: 139ms
       CGroup: /system.slice/named.service
           └─2558 /usr/sbin/named -f -u bind

abr 17 17:35:28 ubantu named[2558]: configuring command channel from '/etc/bind/rndc.key'
abr 17 17:35:28 ubantu named[2558]: command channel listening on ::1:953
abr 17 17:35:28 ubantu named[2558]: named-keys-zone: loaded serial 2
abr 17 17:35:28 ubantu named[2558]: zone 0.in-addr.arpa/IN: loaded serial 1
abr 17 17:35:28 ubantu named[2558]: zone 127.in-addr.arpa/IN: loaded serial 1
abr 17 17:35:28 ubantu named[2558]: zone localhost/IN: loaded serial 2
abr 17 17:35:28 ubantu named[2558]: zone 255.in-addr.arpa/IN: loaded serial 1
abr 17 17:35:28 ubantu named[2558]: all zones loaded
abr 17 17:35:28 ubantu named[2558]: running
abr 17 17:35:28 ubantu systemd[1]: Started named.service - BIND Domain Name Server.
```

Figura 14 Verificación de sintaxis y estado del servicio BIND9 tras reinicio exitoso.

A continuación, se procede a editar el archivo named.conf.local, donde se declaran las zonas que serán administradas por el servidor DNS. En este caso, se crea una zona directa (servidorweb.local) que permitirá resolver nombres dentro del dominio que corresponde directamente al servidor web, así como su respectiva zona inversa (100.43.10.in-addr.arpa) para habilitar la resolución de direcciones IP hacia nombres, si así se requiere. Esta configuración se muestra en la Figura 15.

```
zone "servidorweb.local" IN {
    type master;
    file "/etc/bind/zonas/db.servidorweb.local";
};

zone "100.43.10.in-addr.arpa" IN {
    type master;
    file "/etc/bind/zonas/db.10.43.100";
};
```

Figura 15 Declaración de zonas directa e inversa en el archivo named.conf.local.

Una vez declaradas las zonas en el archivo named.conf.local, es necesario crear los archivos físicos que contendrán la información de resolución de nombres. Dado que el directorio /etc/bind/zonas no existe, primero se debe crear utilizando el comando mostrado en la Figura 16.

```
vboxuser@ubuuntu:~$ sudo mkdir /etc/bind/zonas
[sudo] password for vboxuser:
```

Figura 16 Creación del directorio de zonas en /etc/bind.

Posteriormente, se procede a crear el archivo correspondiente a la zona directa. Este archivo contiene los registros SOA (Start of Authority) y NS (Name Server), así como los registros A que definen las direcciones IP asociadas a los nombres de dominio definidos [5]. En este caso, se registran tanto el servidor principal como el cliente, tal como se aprecia en la Figura 17.

```
GNU nano 6.2
/etc/bind/zonas/db.servidorweb.local
$TTL 1d
@ IN SOA ns1.servidorweb.local. admin.servidorweb.local. (
        1
        12h
        15m
        3w
        2h      )
; Registro NS
        IN   NS   ns1.servidorweb.local.
ns1  IN   A    10.43.100.184
cliente IN  A   10.43.100.241
```

Figura 17 Archivo de zona directa db.servidorweb.local con registros SOA, NS y A.

Ahora, se crea también el archivo correspondiente a la zona inversa, el cual permitirá resolver direcciones IP hacia nombres. Este archivo también contiene su propio registro SOA, así como registros PTR que asocian direcciones IP con nombres canónicos definidos en la zona directa [5]. La estructura de este archivo se muestra en la Figura 18.

```
GNU nano 6.2
/etc/bind/zonas/db.10.43.100
$TTL 1d
@ IN SOA ns1.servidorweb.local. admin.servidorweb.local. (
        2025041701
        12h
        15m
        3w
        2h      )
; Registro NS
        IN   NS   ns1.servidorweb.local.
241  IN   PTR  cliente.servidorweb.local.
184  IN   PTR  ns1.servidorweb.local.
```

Figura 18 Archivo de zona inversa db.10.43.100 configurado con registros PTR

Una vez creados los archivos de zona, es necesario comprobar que su sintaxis y estructura no tengan ningún error. Para ello se utiliza el comando named-checkzone, que permite verificar cada archivo. En la Figura 19 se muestra el uso del comando para las zonas directa e inversa, donde ambas funcionan correctamente.

```
vboxuser@ubuuntu:~$ sudo named-checkzone servidorweb.local /etc/bind/zonas/db.servidorweb.local
zone servidorweb.local/IN: loaded serial 1
OK
vboxuser@ubuuntu:~$ sudo named-checkzone 100.43.10.in-addr.arpa /etc/bind/zonas/db.10.43.100
zone 100.43.10.in-addr.arpa/IN: loaded serial 2025041701
OK
vboxuser@ubuuntu:~$ sudo named-checkconf
```

Figura 19 Validación de los archivos de zona directa e inversa mediante named-checkzone.

Después de validar los archivos, se reinicia el servicio BIND9 para aplicar los cambios. Esto se muestra en la Figura 20.

```
vboxuser@ubuuntu:~$ sudo systemctl restart bind9
```

Figura 20 Reinicio del servicio BIND9 con systemctl.

Ya con todas las configuraciones necesarias del servidor DNS, se realizan pruebas para confirmar que el servidor DNS funciona correctamente por sí mismo. Para ello se utiliza el comando dig, tanto para resolver un nombre (cliente.servidorweb.local) como para hacer una consulta inversa con la dirección IP correspondiente, además de pruebas adicionales para verificar que por ejemplo se resuelvan consultas en IPv6. Como se observa en la Figura 21, las respuestas son correctas y el servidor DNS opera como se espera.

```
estudiante@NGENVA50: $ dig @127.0.0.1 cliente.servidorweb.local +short
10.43.100.241
estudiante@NGENVA50: $ dig -x 10.43.100.241 @127.0.0.1 +short
cliente.servidorweb.local.
estudiante@NGENVA50: $ dig -x 10.43.100.184 @127.0.0.1 +short
ns1.servidorweb.local.
estudiante@NGENVA50: $ dig AAAA google.com @127.0.0.1 +short
2800:3f0:4005:410::200e
estudiante@NGENVA50: $ dig google.com @127.0.0.1 +short
142.251.132.142
```

Figura 21 Pruebas de resolución directa e inversa con dig.

Ahora directamente centrado a la topología de red del proyecto asignamos y verificamos en un VPC que la dirección del DNS quede asignada correctamente como se muestra en la Figura 22. Después se hace ping a google.com y la dirección asignada para el servidor web (cliente.servidorweb.local) con el fin de asegurar que se puedan resolver nombres tanto asignados localmente en el servidor DNS como nombres de internet. Lo

anterior se evidencia en la Figura 23.

```
VPCS> show ip
```

NAME	:	VPCS[1]
IP/MASK	:	192.168.8.98/27
GATEWAY	:	192.168.8.97
DNS	:	10.43.100.184
MAC	:	00:50:79:66:68:06
LPORT	:	10096
RHOST:PORT	:	127.0.0.1:10097
MTU	:	1500

Figura 22 Asignación de IP del servidor DNS a un VPC de la topología del proyecto

```
VPCS> ping google.com
google.com resolved to 142.251.132.142
```

```
84 bytes from 142.251.132.142 icmp_seq=1 ttl=106 time=69.908 ms
84 bytes from 142.251.132.142 icmp_seq=2 ttl=106 time=72.056 ms
84 bytes from 142.251.132.142 icmp_seq=3 ttl=106 time=70.243 ms
84 bytes from 142.251.132.142 icmp_seq=4 ttl=106 time=72.406 ms
84 bytes from 142.251.132.142 icmp_seq=5 ttl=106 time=68.509 ms
```

```
VPCS> ping cliente.servidorweb.local
cliente.servidorweb.local resolved to 10.43.100.241
```

```
84 bytes from 10.43.100.241 icmp_seq=1 ttl=62 time=19.204 ms
84 bytes from 10.43.100.241 icmp_seq=2 ttl=62 time=20.265 ms
84 bytes from 10.43.100.241 icmp_seq=3 ttl=62 time=19.311 ms
84 bytes from 10.43.100.241 icmp_seq=4 ttl=62 time=12.781 ms
84 bytes from 10.43.100.241 icmp_seq=5 ttl=62 time=22.078 ms
```

Figura 23 Comprobación de conectividad mediante ping a google.com y resolución de nombre a cliente.servidorweb.local.

Para finalizar con las pruebas del servidor DNS, se consideran resolver otros nombres, entre los cuales están youtube.com y campusvirtual.javeriana.edu.co. Y con el objetivo de visualizar el intercambio de paquetes entre el cliente (en este caso la maquina con la dirección 192.168.8.3) y el servidor DNS se accedió a Wireshark para mostrar tanto las solicitudes a través del puerto 53 (el cliente) y la respuesta con la dirección correspondiente al nombre solicitado. LO anterior se evidencia en la Figura 24 y 25.

udp.port == 53				
No.	Time	Source	Destination	Protocol
492	868.063120	192.168.8.3	10.43.100.184	DNS
493	868.076318	10.43.100.184	192.168.8.3	DNS
849	1531.715254	192.168.8.3	10.43.100.184	DNS
850	1531.735451	10.43.100.184	192.168.8.3	DNS
1008	1808.400776	192.168.8.3	10.43.100.184	DNS
1009	1808.418793	10.43.100.184	192.168.8.3	DNS
1048	1863.351223	192.168.8.3	10.43.100.184	DNS
1049	1863.376989	10.43.100.184	192.168.8.3	DNS

Figura 24 Captura de Wireshark filtrando consultas y respuestas DNS desde un VPC al servidor DNS local.

```

Length Info
85 Standard query 0xdd99 A cliente.servidorweb.local
161 Standard query response 0xdd99 A cliente.servidorweb.local A 10.43.100.241
70 Standard query 0x7e7c A google.com
86 Standard query response 0x7e7c A google.com A 142.251.135.174
71 Standard query 0x77e0 A youtube.com
87 Standard query response 0x77e0 A youtube.com A 142.251.132.78
90 Standard query 0xfc18 A campusvirtual.javeriana.edu.co
106 Standard query response 0xfc18 A campusvirtual.javeriana.edu.co A 10.26.1.178

```

Figura 25 Detalle de las consultas DNS resueltas, incluyendo dominios externos como Google y sitios de la universidad.

### B. Servidor DHCP

El Protocolo de Configuración Dinámica de Host (DHCP, por sus siglas en inglés) es un protocolo de red que permite la asignación automática de parámetros de configuración IP a dispositivos en una red. Esto incluye la dirección IP, máscara de subred, puerta de enlace predeterminada y servidores DNS. Gracias a este protocolo, se evita la configuración manual de cada host, reduciendo errores y facilitando la administración de redes, especialmente en entornos grandes y dinámicos [6].

El funcionamiento del DHCP está basado en un proceso de cuatro fases conocido como DORA: Discover, Offer, Request y Acknowledge. En este intercambio, el cliente DHCP solicita una dirección IP al servidor, que responde ofreciendo una dirección disponible. Una vez aceptada por el cliente, el servidor confirma la asignación y finaliza el proceso. Esta asignación es temporal y se realiza bajo un tiempo de concesión, lo cual permite la reutilización eficiente de direcciones IP [6].

Para la configuración de nuestro servidor se siguieron algunos pasos que se describirán a continuación:

Como se observa en la Figura 26, se accedió al router mediante la consola, ingresando al modo privilegiado con el comando *enable*, seguido de *configure terminal* para entrar al modo de configuración global. Una vez allí, se aseguró que el servicio DHCP estuviera habilitado con el comando *service dhcp*:

```

R1#enable
R1#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)#service DHCP

```

Figura 26 Activación del servicio DHCP

Luego, se reservaron ciertas direcciones IP dentro de cada subred para evitar que el DHCP las asigne (las IP de las interfaces de gateway), como se observa en la Figura 27:

```

R1(config)#ip dhcp excluded-address 192.168.8.1
R1(config)#ip dhcp excluded-address 192.168.8.65
R1(config)#ip dhcp excluded-address 192.168.8.97
R1(config)#ip dhcp excluded-address 192.168.8.129
R1(config)#ip dhcp excluded-address 192.168.8.145
R1(config)#ip dhcp excluded-address 192.168.8.161

```

Figura 27 Exclusión direcciones IP

En tercera instancia, se configuró un pool para cada red VLAN, especificando la red, máscara de subred, puerta de enlace por defecto y servidor DNS:

```

R1(config)#ip dhcp pool VLAN2
R1(dhcp-config)# network 192.168.8.0 255.255.255.192
R1(dhcp-config)# default-router 192.168.8.1
R1(dhcp-config)# dns-server 10.43.100.184
R1(dhcp-config)#
R1(dhcp-config)#ip dhcp pool VLAN3
R1(dhcp-config)# network 192.168.8.64 255.255.255.224
R1(dhcp-config)# default-router 192.168.8.65
R1(dhcp-config)# dns-server 10.43.100.184
R1(dhcp-config)#
R1(dhcp-config)#ip dhcp pool VLAN4
R1(dhcp-config)# network 192.168.8.96 255.255.255.224
R1(dhcp-config)# default-router 192.168.8.97
R1(dhcp-config)# dns-server 10.43.100.184

```

Figura 28 Pool de direcciones VLAN 2, 3 y 4

```

R1(dhcp-config)#ip dhcp pool VLAN5
R1(dhcp-config)# network 192.168.8.128 255.255.255.240
R1(dhcp-config)# default-router 192.168.8.129
R1(dhcp-config)# dns-server 10.43.100.184
R1(dhcp-config)#
R1(dhcp-config)#ip dhcp pool VLAN6
R1(dhcp-config)# network 192.168.8.144 255.255.255.240
R1(dhcp-config)# default-router 192.168.8.145
R1(dhcp-config)# dns-server 10.43.100.184
R1(dhcp-config)#
R1(dhcp-config)#ip dhcp pool VLAN7
R1(dhcp-config)# network 192.168.8.160 255.255.255.240
R1(dhcp-config)# default-router 192.168.8.161
R1(dhcp-config)# dns-server 10.43.100.184
R1(dhcp-config)#

```

Figura 29 Pool de direcciones VLAN 5, 6 y 7

```

R1(config)#end
R1#write memory
Building configuration...
[OK]

```

Figura 30 Almacenamiento de la configuración

Finalmente, se utilizó el comando *show ip dhcp pool* para verificar que cada pool de direcciones estuviera bien configurado, como se observa en las figuras 31, 32 y 33:

```
R1#show ip dhcp pool
Pool VLAN2 :
Utilization mark (high/low) : 100 / 0
Subnet size (first/next) : 0 / 0
Total addresses : 62
Leased addresses : 0
Pending event : none
1 subnet is currently in the pool :
Current index IP address range Leased addresses
192.168.8.1 192.168.8.1 - 192.168.8.62 0

Pool VLAN3 :
Utilization mark (high/low) : 100 / 0
Subnet size (first/next) : 0 / 0
Total addresses : 30
Leased addresses : 0
Pending event : none
1 subnet is currently in the pool :
Current index IP address range Leased addresses
192.168.8.65 192.168.8.65 - 192.168.8.94 0
```

Figura 31 Configuración Pool VLAN 2 y 3

```
Pool VLAN4 :
Utilization mark (high/low) : 100 / 0
Subnet size (first/next) : 0 / 0
Total addresses : 30
Leased addresses : 1
Pending event : none
1 subnet is currently in the pool :
Current index IP address range Leased addresses
192.168.8.101 192.168.8.97 - 192.168.8.126 1

Pool VLAN5 :
Utilization mark (high/low) : 100 / 0
Subnet size (first/next) : 0 / 0
Total addresses : 14
Leased addresses : 0
Pending event : none
1 subnet is currently in the pool :
Current index IP address range Leased addresses
192.168.8.129 192.168.8.129 - 192.168.8.142 0
```

Figura 32 Configuración Pool VLAN 4 y 5

```
Pool VLAN6 :
Utilization mark (high/low) : 100 / 0
Subnet size (first/next) : 0 / 0
Total addresses : 14
Leased addresses : 0
Pending event : none
1 subnet is currently in the pool :
Current index IP address range Leased addresses
192.168.8.145 192.168.8.145 - 192.168.8.158 0

Pool VLAN7 :
Utilization mark (high/low) : 100 / 0
Subnet size (first/next) : 0 / 0
Total addresses : 14
Leased addresses : 0
Pending event : none
1 subnet is currently in the pool :
Current index IP address range Leased addresses
192.168.8.161 192.168.8.161 - 192.168.8.174 0
```

Figura 33 Configuración Pool VLAN 6 y 7

Ahora bien, para verificar el funcionamiento del servidor, entramos a la terminal de un Virtual PC ubicado en el departamento de sistemas (VLAN 4) y ejecutamos el comando *ip dhcp*. Como resultado, el dispositivo solicitó una dirección IP utilizando el protocolo DHCP, y el servidor respondió

exitosamente asignando la IP 192.168.8.99 con una máscara /27, correspondiente a la subred 192.168.8.96/27.

Seguidamente, con el comando *show ip*, confirmamos que la configuración fue recibida correctamente: IP, gateway, DNS y servidor DHCP coinciden con los valores definidos en el *ip dhcp pool VLAN4*. Todo este procedimiento se observa en la Figura 34:

```
VPCS> ip dhcp
DORA IP 192.168.8.99/27 GW 192.168.8.97

VPCS> show ip

NAME      : VPCS[1]
IP/MASK   : 192.168.8.99/27
GATEWAY   : 192.168.8.97
DNS       : 10.43.100.184
DHCP SERVER : 192.168.8.97
DHCP LEASE  : 86393, 86400/43200/75600
MAC        : 00:50:79:66:68:07
LPORT      : 10098
RHOST:PORT : 127.0.0.1:10099
MTU        : 1500
```

Figura 34 Uso del servidor DHCP en un Host

Este procedimiento se repite con otros Virtual PC'S en la topología.

### C. Servidor Web

Podemos considerar un servidor web como un sistema que almacena páginas web y entrega contenido a los usuarios cuando lo solicitan a través de un navegador, utilizando el protocolo HTTP [7], este servidor web es de suma importancia para nuestro proyecto, debido a que se nos indica explícitamente que la empresa necesita uno de estos para su red.

Para montar el servidor web, lo primero que se debe hacer es instalar software de servidor web, para el caso particular, escogeremos Apache, debido a su escalabilidad y fácil configuración.

En primer lugar, en la consola de la máquina virtual 10.43.100.241, se ejecutan los siguientes comandos para

instalar

dicho

software:

```
estudiante@ngenva107:~$ sudo apt install apache2
```

Figura 35 Instalación de Apache en el host..

Y acto seguido se comprueba la instalación y el correcto funcionamiento de este mismo con el comando:

```
estudiante@ngenva107:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor pres
   Active: active (running) since Fri 2025-04-04 15:27:07 -05; 3 days ago
     Docs: https://httpd.apache.org/docs/2.4/
Main PID: 3011433 (apache2)
   Tasks: 55 (limit: 19093)
  Memory: 6.4M
    CPU: 9.305s
   CGroup: /system.slice/apache2.service
           ├─1981878 /usr/sbin/apache2 -k start
           ├─1981879 /usr/sbin/apache2 -k start
           └─3011433 /usr/sbin/apache2 -k start
```

Figura 36 Comprobación del estado de la instalación

Ahora, cada vez que cualquier dispositivo quiera acceder a este mismo, solo deberá poner la IP en el buscador que desee (estando dentro de la red de la universidad), siendo que nos mostrará la siguiente interfaz:



Figura 37 Interfaz de comprobación de Apache

Ahora, para tener una prueba visual de que si se está ejecutando correctamente, se instalará un código con HTML y CSS que genere una interfaz visual, simulando una página web. El código para el HTML es el siguiente:

```
<!DOCTYPE html>

<html lang="es">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Proyecto de Redes</title>
```

```
<style>
```

```
 @import
```

```
url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap');
```

```
 body {
```

```
 margin: 0;
```

```
 padding: 0;
```

```
 background: linear-gradient(135deg, #e0f7fa,
```

```
#80deeaa);
```

```
 font-family: 'Poppins', sans-serif;
```

```
 display: flex;
```

```
 flex-direction: column;
```

```
 align-items: center;
```

```
 justify-content: center;
```

```
 min-height: 100vh;
```

```
 color: #004d40;
```

```
}
```

```
.container {
```

```
 background: #ffffff;
```

```
 padding: 3rem;
```

```
 border-radius: 20px;
```

```
 box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
```

```
 text-align: center;
```

```
 max-width: 600px;
```

```
 width: 90%;
```

```
}
```

```

h1 {
    margin-bottom: 1.5rem;
    font-size: 2.5rem;
    font-weight: 600;
    color: #00695c;
}

ul {
    list-style: none;
    padding: 0;
}

li {
    font-size: 1.2rem;
    margin: 0.5rem 0;
}

li::before {
    content: "🌐 ";
}

</style>

</head>

<body>

<div class="container">
    <h1>Proyecto de Redes</h1>
    <ul>
        <li>Santiago Mesa Niño</li>
        <li>Luna Rengifo Moreno</li>
        <li>Giovanny Andres Durán Renteria</li>
    </ul>
</div>

```

Y los comandos para la instalación de este archivo dentro del servidor web son los siguientes:

```

estudiante@ngenva107: $ sudo cp /home/estudiante/Downloads/proyecto_de_redes.html /var/www/html/

```

*Figura 38 Instalación de la página web en el servidor.*

Y ahora otorgamos permisos a Apache sobre el archivo con el siguiente comandos:

```

estudiante@ngenva107: $ sudo chown www-data:www-data /var/www/html/proyecto_de_redes.html

```

*Figura 37 Permisos a Apache*

Por último, así se vería la interfaz de la página.

*Figura 38 Visualización de la página en el servidor web*

#### ESCENARIO DE PRUEBAS

A continuación, se detallan las pruebas de conectividad y configuraciones realizadas en una red de prueba, incluyendo la comunicación entre dispositivos en diferentes VLANs y el acceso a Internet. También se describe la configuración de NAT en el router R1, asegurando la correcta traducción de direcciones y la conectividad entre las distintas redes.

*A. Pruebas de conectividad entre PC'S- Red de prueba*

```

#####
# DESARROLLO1 - 192.168.8.2
ping 192.168.8.3    # Desarrollo2 (misma VLAN)
ping 192.168.8.4    # Desarrollo3 (misma VLAN)
ping 192.168.8.5    # Desarrollo4 (misma VLAN)
ping 192.168.8.1    # Gateway VLAN 2
ping 192.168.8.66   # PC5 (Gerencia)
ping 192.168.8.98   # PC7 (Sistemas)
ping 192.168.8.130  # PC1 (Servidores)
ping 192.168.8.146  # PC9 (Investigación A)
ping 192.168.8.162  # PC11 (Investigación B)

#####
# DESARROLLO2 - 192.168.8.3
ping 192.168.8.2
ping 192.168.8.4
ping 192.168.8.5
ping 192.168.8.1
ping 192.168.8.67
ping 192.168.8.99
ping 192.168.8.131
ping 192.168.8.147
ping 192.168.8.163

#####
# PC5 - 192.168.8.66
ping 192.168.8.67  # PC6
ping 192.168.8.65  # Gateway VLAN 3
ping 192.168.8.2
ping 192.168.8.98
ping 192.168.8.130
ping 192.168.8.146

#####
ping 192.168.8.162
#####
# PC6 - 192.168.8.67
#####
# PC6 - 192.168.8.67
ping 192.168.8.66
ping 192.168.8.65
ping 192.168.8.64
ping 192.168.8.63
ping 192.168.8.62
ping 192.168.8.61
ping 192.168.8.60
ping 192.168.8.59
ping 192.168.8.58
ping 192.168.8.57
ping 192.168.8.56
ping 192.168.8.55
ping 192.168.8.54
ping 192.168.8.53
ping 192.168.8.52
ping 192.168.8.51
ping 192.168.8.50
ping 192.168.8.49
ping 192.168.8.48
ping 192.168.8.47
ping 192.168.8.46
ping 192.168.8.45
ping 192.168.8.44
ping 192.168.8.43
ping 192.168.8.42
ping 192.168.8.41
ping 192.168.8.40
ping 192.168.8.39
ping 192.168.8.38
ping 192.168.8.37
ping 192.168.8.36
ping 192.168.8.35
ping 192.168.8.34
ping 192.168.8.33
ping 192.168.8.32
ping 192.168.8.31
ping 192.168.8.30
ping 192.168.8.29
ping 192.168.8.28
ping 192.168.8.27
ping 192.168.8.26
ping 192.168.8.25
ping 192.168.8.24
ping 192.168.8.23
ping 192.168.8.22
ping 192.168.8.21
ping 192.168.8.20
ping 192.168.8.19
ping 192.168.8.18
ping 192.168.8.17
ping 192.168.8.16
ping 192.168.8.15
ping 192.168.8.14
ping 192.168.8.13
ping 192.168.8.12
ping 192.168.8.11
ping 192.168.8.10
ping 192.168.8.9
ping 192.168.8.8
ping 192.168.8.7
ping 192.168.8.6
ping 192.168.8.5
ping 192.168.8.4
ping 192.168.8.3
ping 192.168.8.2
ping 192.168.8.1
ping 192.168.8.0

#####
# Pruebas de conectividad hacia internet (8.8.8.8)
#####
# VLAN 2 - Desarrollo
# PC: Desarrollo1 - 192.168.8.2
ping 8.8.8.8
# PC: Desarrollo2 - 192.168.8.3
ping 8.8.8.8
# PC: Desarrollo3 - 192.168.8.4
ping 8.8.8.8
# PC: Desarrollo4 - 192.168.8.5
ping 8.8.8.8
#####
# VLAN 3 - Gerencia
# PC5 - 192.168.8.66
ping 8.8.8.8
# PC6 - 192.168.8.67

```

```

ping 8.8.8.8
#####
# VLAN 4 - Sistemas
# PC7 - 192.168.8.98
ping 8.8.8.8

# PC8 - 192.168.8.99
ping 8.8.8.8

#####
# VLAN 5 - Sistemas (Servidores)
# PC1 - 192.168.8.130
ping 8.8.8.8

```

encapsulation dot1Q 3  
ip address 192.168.8.65 255.255.255.224  
ip nat inside  
no shutdown

interface FastEthernet0/0.4  
encapsulation dot1Q 4  
ip address 192.168.8.97 255.255.255.224  
ip nat inside  
no shutdown

interface FastEthernet0/0.5  
encapsulation dot1Q 5  
ip address 192.168.8.129 255.255.255.240  
ip nat inside  
no shutdown

interface FastEthernet0/0.6  
encapsulation dot1Q 6  
ip address 192.168.8.145 255.255.255.240  
ip nat inside  
no shutdown

interface FastEthernet0/0.7  
encapsulation dot1Q 7  
ip address 192.168.8.161 255.255.255.240  
ip nat inside  
no shutdown

! ACL para definir qué redes se traducen  
access-list 1 permit 192.168.8.0 0.0.3.255

! Regla NAT PAT (sobrecarga)  
ip nat inside source list 1 interface FastEthernet0/1 overload

```

C. Configuración de Router 1, subinterfaces y NAT
CONFIGURACIÓN DE NAT EN R1

! Interfaz de salida hacia Internet (NAT1)
interface FastEthernet0/1
ip address dhcp
ip nat outside
no shutdown

SUBINTERFACES
! Subinterfaces para VLANs (router-on-a-stick)
interface FastEthernet0/0.2
encapsulation dot1Q 2
ip address 192.168.8.1 255.255.255.192
ip nat inside
no shutdown

interface FastEthernet0/0.3

```

```
end
write memory
```

## PROTOCOLO DE PRUEBAS Y RESULTADOS

Con el fin de validar la correcta operación de la red diseñada e implementada, se llevó a cabo un protocolo de pruebas estructurado. Estas pruebas permiten verificar la conectividad interna entre dispositivos, la comunicación entre VLANs, la correcta asignación de direcciones IP mediante DHCP, el acceso a servicios internos como el servidor web, así como la salida a Internet y la resolución de nombres de dominio. Además, se comprobó la compatibilidad tanto con IPv4 como con IPv6. Cada prueba se documenta a continuación, indicando su propósito, el procedimiento seguido, las direcciones IP utilizadas y los resultados obtenidos.

Se inició verificando la conectividad dentro de una misma VLAN. Para ello, se realizó un ping desde el dispositivo con dirección IP 192.168.8.2 perteneciente al departamento de Desarrollo (VLAN 2) hacia otro dispositivo de la misma VLAN con la dirección 192.168.8.3. El propósito fue comprobar que la comunicación interna entre dispositivos de una misma red segmentada funcionara correctamente. El resultado fue exitoso, ya que las respuestas de ping fueron recibidas sin pérdida de paquetes.

```
VPCS> ping 192.168.8.3
84 bytes from 192.168.8.3 icmp_seq=1 ttl=64 time=2.265 ms
84 bytes from 192.168.8.3 icmp_seq=2 ttl=64 time=2.165 ms
84 bytes from 192.168.8.3 icmp_seq=3 ttl=64 time=1.872 ms
84 bytes from 192.168.8.3 icmp_seq=4 ttl=64 time=2.888 ms
84 bytes from 192.168.8.3 icmp_seq=5 ttl=64 time=2.312 ms
```

Figura 38 Ping entre misma VLAN

10.16.982516	192.168.8.2	192.168.8.3	ICMP	98 Echo (ping) request id=0xa0b0, seq=1/256, ttl=64 (reply in 11)
11.15.986643	192.168.8.3	192.168.8.2	ICMP	98 Echo (ping) reply id=0xa0b0, seq=1/256, ttl=64 (request in 10)
12.17.988094	192.168.8.2	192.168.8.3	ICMP	98 Echo (ping) request id=0xa0b0, seq=2/512, ttl=64 (reply in 13)
13.17.990666	192.168.8.3	192.168.8.2	ICMP	98 Echo (ping) reply id=0xa0b0, seq=2/512, ttl=64 (request in 12)
14.17.997775	cc:01:36:dd:f1:01	Spanning-tree-(for - STP	ARP	60 Conf. Root = 32768/0;cc:01:36:dd:f1:01 Cost = 0 Port = 0x002a
15.891759	192.168.8.2	192.168.8.3	ICMP	98 Echo (ping) request id=0xa0b0, seq=3/768, ttl=64 (reply in 10)
16.18.993436	192.168.8.3	192.168.8.2	ICMP	98 Echo (ping) reply id=0xa0b0, seq=3/768, ttl=64 (request in 15)

Figura 39 Captura de tráfico ping

Posteriormente, se probó la comunicación entre diferentes VLAN'S. Se ejecutó un ping desde el dispositivo con

IP 192.168.8.3 de la VLAN 2 (Departamento de Desarrollo) hacia el dispositivo con IP 192.168.8.66 del departamento de Gerencia. Esta prueba tenía como objetivo validar el enrutamiento inter-VLAN configurado en el switch y el router. El resultado fue exitoso, confirmando que los dispositivos en diferentes VLANs pueden comunicarse adecuadamente.

```
VPCS> ping 192.168.8.66
```

```
84 bytes from 192.168.8.66 icmp_seq=1 ttl=63 time=29.127 ms
84 bytes from 192.168.8.66 icmp_seq=2 ttl=63 time=14.465 ms
84 bytes from 192.168.8.66 icmp_seq=3 ttl=63 time=14.338 ms
84 bytes from 192.168.8.66 icmp_seq=4 ttl=63 time=14.302 ms
84 bytes from 192.168.8.66 icmp_seq=5 ttl=63 time=16.813 ms
```

Figura 40 Ping entre diferentes VLANs

8 13.271181	Private_06:08:01	Private_06:08:01	ARP	64 Who has 192.168.8.1? Tell 192.168.8.3
9 13.282769	192.168.8.3	192.168.8.66	ARP	60 192.168.8.3 is at cc:01:36:dd:f1:01
10 13.282769	192.168.8.66	192.168.8.3	ICMP	98 Echo (ping) request id=0xb7fe, seq=1/256, ttl=64 (reply in 11)
11 13.282343			ICMP	98 Echo (ping) reply id=0xb7fe, seq=1/256, ttl=64 (request in 10)
12 14.313078	192.168.8.3	192.168.8.66	ICMP	98 Echo (ping) request id=0xb800, seq=2/512, ttl=64 (reply in 14)
14 14.327416	192.168.8.66	192.168.8.3	ICMP	98 Echo (ping) reply id=0xb800, seq=2/512, ttl=64 (request in 13)
15 14.327416	192.168.8.3	192.168.8.66	ICMP	98 Echo (ping) request id=0xb800, seq=3/768, ttl=64 (reply in 15)
16 15.341215	192.168.8.66	192.168.8.3	ICMP	98 Echo (ping) reply id=0xb800, seq=3/768, ttl=64 (request in 16)
17 15.341215	192.168.8.3	192.168.8.66	ICMP	98 Echo (ping) request id=0xb800, seq=4/1024, ttl=64 (reply in 17)
18 16.355249	192.168.8.66	192.168.8.3	ICMP	98 Echo (ping) reply id=0xb800, seq=4/1024, ttl=64 (request in 18)
19 16.355469	192.168.8.66	192.168.8.3	ICMP	98 Echo (ping) reply id=0xb800, seq=5/1280, ttl=64 (request in 19)
20 17.359524	192.168.8.3	192.168.8.66	ICMP	98 Echo (ping) request id=0xb800, seq=5/1280, ttl=64 (reply in 21)
21 17.375334	192.168.8.66	192.168.8.3	ICMP	98 Echo (ping) reply id=0xb800, seq=5/1280, ttl=64 (request in 20)

Figura 41 Captura de tráfico ping

También, se realizó esta prueba desde el host con IP 192.168.8.5 (edificio 2) hacia la máquina con IP 192.168.8.2 (edificio 1) pertenecientes a la misma VLAN (Desarrollo), lo cual, resultó de manera exitosa la comunicación entre VLANs, pero diferentes edificios.

```
VPCS> ping 192.168.8.2
```

```
84 bytes from 192.168.8.2 icmp_seq=1 ttl=64 time=3.024 ms
84 bytes from 192.168.8.2 icmp_seq=2 ttl=64 time=3.334 ms
84 bytes from 192.168.8.2 icmp_seq=3 ttl=64 time=2.837 ms
84 bytes from 192.168.8.2 icmp_seq=4 ttl=64 time=1.742 ms
84 bytes from 192.168.8.2 icmp_seq=5 ttl=64 time=2.541 ms
```

Figura 42 Ping entre edificios

17 25.0983230	192.168.8.5	192.168.8.2	ICMP	98 Echo (ping) request id=0x35ef, seq=1/256, ttl=64 (reply in 18)
18 25.098527	192.168.8.2	192.168.8.5	ICMP	98 Echo (ping) reply id=0x35ef, seq=1/256, ttl=64 (request in 17)
19 25.098527	192.168.8.5	192.168.8.2	ICMP	98 Echo (ping) request id=0x35ef, seq=2/512, ttl=64 (reply in 19)
20 26.0986994	192.168.8.5	192.168.8.2	ICMP	98 Echo (ping) request id=0x35ef, seq=2/512, ttl=64 (request in 21)
21 26.0986912	192.168.8.2	192.168.8.5	ICMP	98 Echo (ping) reply id=0x35ef, seq=2/512, ttl=64 (request in 20)
22 27.0987002	192.168.8.5	192.168.8.2	ICMP	98 Echo (ping) request id=0x35ef, seq=3/768, ttl=64 (reply in 22)
23 27.0983240	192.168.8.2	192.168.8.5	ICMP	98 Echo (ping) reply id=0x35ef, seq=3/768, ttl=64 (request in 23)
24 28.0984527	192.168.8.5	192.168.8.2	ICMP	98 Echo (ping) request id=0x35ef, seq=4/1024, ttl=64 (reply in 24)
25 28.0984527	192.168.8.2	192.168.8.5	ICMP	98 Echo (ping) reply id=0x35ef, seq=4/1024, ttl=64 (request in 25)
26 28.0996155	192.168.8.5	192.168.8.2	ICMP	98 Echo (ping) request id=0x35ef, seq=5/1280, ttl=64 (reply in 26)
27 29.099222	192.168.8.2	192.168.8.5	ICMP	98 Echo (ping) reply id=0x35ef, seq=5/1280, ttl=64 (request in 27)
28 29.099370	192.168.8.2	192.168.8.5	ICMP	98 Echo (ping) request id=0x35ef, seq=6/1920, ttl=64 (reply in 28)

Figura 43 Captura de tráfico ping

A continuación, se verificó el correcto funcionamiento de la resolución de nombres DNS. Desde el dispositivo 192.168.8.67, se ejecutó un ping hacia el dominio www.google.com. El objetivo era validar que los dispositivos pueden resolver nombres de dominio en lugar de trabajar únicamente con direcciones IP. El resultado fue exitoso, ya que

el dominio fue resuelto correctamente y se recibieron respuestas de ping.

```
VPCS> show ip
NAME      : VPCS[1]
IP/MASK   : 192.168.8.67/27
GATEWAY   : 192.168.8.65
DNS       : 10.43.100.184
DHCP SERVER : 192.168.8.65
DHCP LEASE  : 83363, 86400/43200/75600
MAC       : 00:50:79:66:68:05
LPORT     : 10110
RHOST:PORT : 127.0.0.1:10111
MTU       : 1500

VPCS> ping google.com
google.com resolved to 142.251.135.174

84 bytes from 142.251.135.174 icmp_seq=1 ttl=106 time=63.259 ms
84 bytes from 142.251.135.174 icmp_seq=2 ttl=106 time=69.122 ms
84 bytes from 142.251.135.174 icmp_seq=3 ttl=106 time=55.763 ms
84 bytes from 142.251.135.174 icmp_seq=4 ttl=106 time=65.019 ms
84 bytes from 142.251.135.174 icmp_seq=5 ttl=106 time=53.958 ms
```

Figura 44 Ping a nombre de dominio

Para garantizar el acceso a Internet, se realizó un ping desde un dispositivo interno, hacia una dirección IP pública conocida, 8.8.8.8 (servidor DNS público de Google). Esta prueba permitió comprobar que la red cuenta con traducción de direcciones (NAT) y salida correcta hacia la red externa. El ping fue exitoso, con respuestas satisfactorias.

```
VPCS> ping 8.8.8.8

84 bytes from 8.8.8.8 icmp_seq=1 ttl=106 time=70.409 ms
84 bytes from 8.8.8.8 icmp_seq=2 ttl=106 time=69.983 ms
84 bytes from 8.8.8.8 icmp_seq=3 ttl=106 time=64.072 ms
84 bytes from 8.8.8.8 icmp_seq=4 ttl=106 time=70.355 ms
84 bytes from 8.8.8.8 icmp_seq=5 ttl=106 time=69.847 ms
```

Figura 45 Ping a internet

También se realizó la prueba del servicio DHCP. Para ello, en un computador cliente se utilizó el comando ip dhcp para solicitar automáticamente una dirección IP. Al ejecutar este comando, el computador inició el proceso de negociación conocido como DORA (Discover, Offer, Request, Acknowledge), fundamental en el protocolo DHCP.

Primero, el cliente envió un mensaje Discover (Descubrir) para localizar servidores DHCP disponibles en la red. Posteriormente, el servidor respondió con un mensaje Offer (Oferta), ofreciendo una dirección IP y otros parámetros de red como la puerta de enlace y el servidor DNS. Luego, el cliente

envió un mensaje Request (Solicitud) para aceptar la oferta recibida. Finalmente, el servidor respondió con un mensaje Acknowledge (Confirmación), asignando formalmente la configuración de red al dispositivo. Esta prueba se ejecutó desde la máquina 192.168.8.4 (Desarrollo).

```
VPCS> ip dhcp
DORA IP 192.168.8.4/26 GW 192.168.8.1

VPCS> show ip
NAME      : VPCS[1]
IP/MASK   : 192.168.8.4/26
GATEWAY   : 192.168.8.1
DNS       : 10.43.100.184
DHCP SERVER : 192.168.8.1
DHCP LEASE  : 86393, 86400/43200/75600
MAC       : 00:50:79:66:68:02
LPORT     : 10104
RHOST:PORT : 127.0.0.1:10105
MTU       : 1500
```

Figura 46 Asignación de IP por DHCP

La correcta ejecución del comando “ip dhcp” y la recepción de los cuatro mensajes confirmaron que el servidor DHCP funciona adecuadamente y que la asignación dinámica de dirección IP se realiza correctamente. Esta prueba fue completada exitosamente.

También, se comprobó la conectividad con IPv4 e IPv6 por parte del departamento de Investigación, el primer comando se ejecuta para comprobar que el host tiene ambas direcciones guardadas para la comunicación.

```
VPCS> show ip

NAME      : VPCS[1]
IP/MASK   : 192.168.8.162/28
GATEWAY   : 192.168.8.161
DNS       :
MAC       : 00:50:79:66:68:0a
LPORT     : 10120
RHOST:PORT: 127.0.0.1:10121
MTU       : 1500

VPCS> show ipv6

NAME      : VPCS[1]
LINK-LOCAL SCOPE : fe80::250:79ff:fe66:680a/64
GLOBAL SCOPE    : 2001:1::2/64
DNS       :
ROUTER LINK-LAYER: cc:05:f9:50:00:00
MAC       : 00:50:79:66:68:0a
LPORT     : 10120
RHOST:PORT: 127.0.0.1:10121
MTU       : 1500
```

Figura 47 Dual-stack en VPC

Luego, desde esta máquina, se realiza un ping a un host con IPv4 y a uno con IPv6, esto para comprobar el correcto funcionamiento del dual stack y del enrutamiento.

```
VPCS> ping 192.168.8.163

84 bytes from 192.168.8.163 icmp_seq=1 ttl=64 time=1.129 ms
84 bytes from 192.168.8.163 icmp_seq=2 ttl=64 time=1.191 ms
84 bytes from 192.168.8.163 icmp_seq=3 ttl=64 time=1.178 ms
84 bytes from 192.168.8.163 icmp_seq=4 ttl=64 time=1.237 ms
84 bytes from 192.168.8.163 icmp_seq=5 ttl=64 time=1.248 ms

VPCS> ping 200:1::3

200:1::3 icmp6_seq=1 ttl=64 time=1.899 ms
200:1::3 icmp6_seq=2 ttl=64 time=1.856 ms
200:1::3 icmp6_seq=3 ttl=64 time=1.628 ms
200:1::3 icmp6_seq=4 ttl=64 time=1.347 ms
200:1::3 icmp6_seq=5 ttl=64 time=1.941 ms
```

Figura 48 Ping entre IPv4 e IPv6

Por último, se probó el servidor web, el cual, estaba montado sobre la máquina virtual de un integrante del equipo, que tiene la IP 10.43.100.241, y en el servidor DNS se le asignó un nombre de host a esta dirección (cliente.servidorweb.local), para que, al ingresarla, resolviera este nombre a la dirección que deseamos, como se ve en la siguiente imagen:

```
VPCS> ping 10.43.100.241

84 bytes from 10.43.100.241 icmp_seq=1 ttl=62 time=19.990 ms
84 bytes from 10.43.100.241 icmp_seq=2 ttl=62 time=15.863 ms
84 bytes from 10.43.100.241 icmp_seq=3 ttl=62 time=17.120 ms
84 bytes from 10.43.100.241 icmp_seq=4 ttl=62 time=22.061 ms
84 bytes from 10.43.100.241 icmp_seq=5 ttl=62 time=16.931 ms

VPCS> ping cliente.servidorweb.local
cliente.servidorweb.local resolved to 10.43.100.241

84 bytes from 10.43.100.241 icmp_seq=1 ttl=62 time=19.931 ms
84 bytes from 10.43.100.241 icmp_seq=2 ttl=62 time=20.584 ms
84 bytes from 10.43.100.241 icmp_seq=3 ttl=62 time=12.533 ms
84 bytes from 10.43.100.241 icmp_seq=4 ttl=62 time=13.774 ms
84 bytes from 10.43.100.241 icmp_seq=5 ttl=62 time=24.493 ms
```

Figura 49 Ping al servidor web y su dominio asignado en DNS

La conectividad se probó también en una interfaz gráfica montada en la topología, esta interfaz es de Ubuntu, con la versión 22.04, se accedió a la página de YouTube, desde el navegador, y también a la página que se ingresó en el servidor web, también, se realizaron pings a máquinas de la topología.

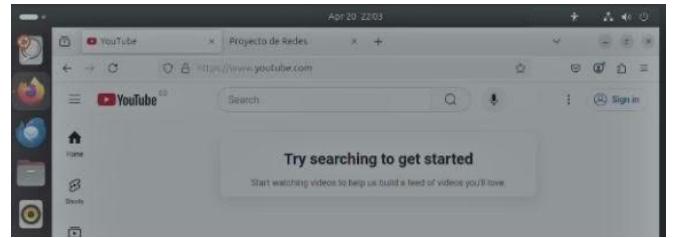


Figura 50 Acceso a internet desde Ubuntu

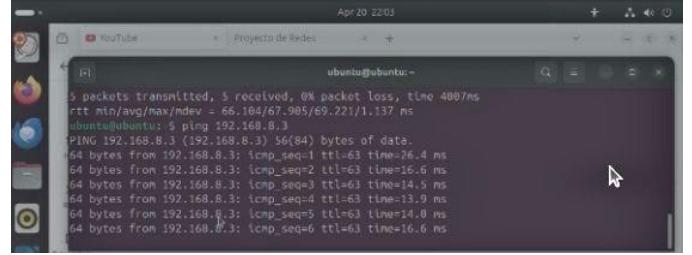


Figura 51 Ping a otra VLAN desde Ubuntu

## BIBLIOGRAFÍA

- [1] W. Stallings, Data and Computer Communications, 10th ed., Boston: Pearson, 2013.

- [2] A. S. Tanenbaum y D. J. Wetherall, Computer Networks, 5th ed., Upper Saddle River, NJ: Pearson, 2011.
- [3] IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Bridges, IEEE Std 802.1D-2004, 2004.
- [4] A. Mockapetris, “Domain names - implementation and specification,” *RFC 1035*, Internet Engineering Task Force (IETF), Nov. 1987. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc1035>
- [5] P. Albitz and C. Liu, *DNS and BIND*, 5th ed., Sebastopol, CA: O'Reilly Media, 2006.
- [6] R. Droms, Dynamic Host Configuration Protocol, RFC 2131, Mar. 1997. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc2131>
- [7] K. C. Laudon y J. P. Laudon, *Sistemas de información gerencial*, 15.<sup>a</sup> ed., México: Pearson Educación, 2020.

