

Centro Universitario de Occidente
Redes de Computadoras 1
Ing. Francisco Rojas
Primer Semestre 2024



Proyecto Redes

- 201830498 - Oscar Antonio de León Urizar.
201832069 - Marcos Andrés Aguare Bravo.
201930643 - Jeffrey Kenneth Menendez Castillo
201930693 - Leví Isaac Hernández Sapón.
201930699 - Erick Daniel Morales Xicará.
201931707 - Luis Emilio Maldonado Rodriguez.

Abstracción de información

EJEMPLO SENCILLO

Ejemplo simplificado del funcionamiento del CRC: parte 1 (transmisión)

- El nivel de red le entrega al nivel de enlace un paquete con los datos, concretamente el número 302
- El nivel de enlace ha acordado con su homólogo en el otro extremo utilizar un CRC de un dígito, y como generador el número 3. Por tanto sabe que tiene que enviar 302x
- Elige x tal que el número resultante sea divisible por 3. Tomará la primera posibilidad:
prueba con $0 \rightarrow 3020$, no es divisible
Prueba con $1 \rightarrow 3021$, este sí
- El nivel de enlace transmite una trama que contiene 3021

Proyecto

Redes de computadoras 1
Proyecto Final

Sistema operativo host: Linux
Lenguaje de programación: Script de configuración
Formato de respuesta: Texto
Grupos: Ambiente de trabajo

Especificaciones:

Linux debemos todo en modo texto, sin interfaz gráfica
Script de configuración
Texto
5 personas por grupo (cada persona tiene un Puente)
Máquinas virtualizadas: KVM, Qemu o Proxmox

HACER UNA RED CORPORATIVA

• CONFIGURAR

ES UN PUENTE, COMO UN SWITCH.

DE UN LADO CONEXIÓN FÍSICA Y DEL OTRO LADO WiFi. SWITCH INALÁMBRICO.

MÁQUINA VIRTUAL

SWITCH VIRTUAL

LX = Linux/Debian

PARA ENTRAR de 2 INTERFACES FÍSICAS

DISTRIBUIR PAQUETES

MIN 1 EQUIPO /

TODAS LAS REDES SE COMUNICAN, POR MEDIO DE UN RUTADOR

2 FÍSICAS Y 1 VIRTUAL

HOST PARA CADA VM

ESTADO DE LAS CONEXIONES PARA SABER DONDE MANDAR EL PAQUETE

• RJ45/UTP, ALUMBRICO SIEMPRE INALAMBRICO.

Segmento A (6 equipos)
Segmento B (40 equipos)
Segmento C (30 equipos)
Segmento D (8 equipos)

Distribución de puntuación

Envío	20
Fallover	30
Enlace WiFi	30
Dispositivo	10
Seguridad	10

Fecha de entrega y evaluación: 02/mayo/2021

EMPEZAR CON UN

UNA MÁQUINA SWITCCH (EN RUTADOR)

Y UNA CLIENTE.

Z MÁQUINAS VIRTUALES POR EQUIPO

¿Las conexiones físicas son líneas verdes?

Objetivo del Proyecto

El propósito principal de este proyecto es explorar la virtualización de redes con QEMU, destacando el uso de iptables para la gestión del tráfico, y la implementación de una variedad de sistemas operativos, incluyendo Kali Linux, Ubuntu y Debian. Desde la configuración de máquinas virtuales hasta la creación de puntos de acceso inalámbricos, este proyecto nos llevará a través de un viaje de aprendizaje emocionante en el mundo de las redes virtualizadas. Y permitiendo así la construcción de una pequeña red tomando en cuenta los temas de enrutamiento, failover, enlace wifi, diseño y seguridad.

Marco Teórico

Aquí describiremos algunos temas necesarios para poder entender los temas utilizados para la elaboración del proyecto:

Sistema Operativo Debian 11 para el Uso de Redes:

Debian 11, conocido como "Bullseye", es una distribución de Linux estable y ampliamente utilizada que proporciona un entorno robusto para el desarrollo y despliegue de aplicaciones, incluyendo aplicaciones de red. Su enfoque en la estabilidad y la libertad del software lo convierte en una opción popular para servidores y estaciones de trabajo. Para el contexto de este proyecto, utilizaremos Debian 11 como sistema operativo base para configurar y gestionar diversos aspectos de la red, como enrutamiento, filtrado de paquetes y servicios de red.

Sistema Operativo Kali:

Kali Linux es una distribución de Linux diseñada específicamente para pruebas de penetración y auditoría de seguridad. Basada en Debian, Kali Linux proporciona una amplia gama de herramientas de seguridad preinstaladas que son útiles para identificar vulnerabilidades en sistemas y redes. En este proyecto, utilizaremos Kali Linux para configurar y administrar la seguridad de la red, así como para implementar herramientas de análisis y monitoreo de tráfico.

Enrutamiento:

El enrutamiento es el proceso de dirigir el tráfico de red entre diferentes redes o subredes. Los routers son dispositivos clave en una red que facilitan este proceso, tomando decisiones sobre la mejor ruta para enviar paquetes de datos. En el contexto de este proyecto, exploramos el enrutamiento estático y dinámico para dirigir el tráfico entre máquinas virtuales y segmentos de red virtualizados.

iptables y la Librería IP de Linux:

iptables es una herramienta de firewall para el kernel de Linux que permite configurar reglas de filtrado, traducción de direcciones de red (NAT) y otras funcionalidades relacionadas con el tráfico de red. Utilizaremos iptables para implementar políticas de seguridad en nuestras máquinas virtuales Debian 11 y Kali Linux, asegurando la integridad y confidencialidad de los datos que atraviesan la red virtual. La librería IP de Linux proporciona una interfaz para interactuar con la pila de protocolos de red del kernel Linux, permitiendo un control granular sobre el tráfico de red a nivel de software.

OpenSSH:

OpenSSH es una suite de herramientas de conectividad de red que permite la transferencia segura de datos a través de una red no segura utilizando el protocolo SSH (Secure Shell). Utilizaremos OpenSSH para acceder de forma remota a nuestras máquinas virtuales Debian 11 y Kali Linux, facilitando la configuración y gestión de la red desde un entorno de línea de comandos remoto.

Hotspot Configurado en Kali:

Un hotspot es un punto de acceso inalámbrico que proporciona conectividad a Internet a dispositivos cercanos a través de una red inalámbrica. En este proyecto, configuraremos un hotspot en Kali Linux utilizando herramientas como Hostapd y dnsmasq. Esto nos permitirá ofrecer una conexión Wi-Fi segura y gestionar el tráfico inalámbrico dentro de nuestra red virtualizada, proporcionando a los clientes virtuales acceso a Internet de manera controlada y segura.

Enlace Wi-Fi y Diseño:

El enlace Wi-Fi permite la conexión inalámbrica entre dispositivos dentro de la red. En este proyecto, diseñaremos y configuraremos enlaces Wi-Fi seguros utilizando estándares de cifrado y autenticación robustos. Además, planificamos el diseño de la red virtualizada, considerando aspectos como la topología de la red, la segmentación de subredes y la asignación de direcciones IP, para garantizar un rendimiento óptimo y una gestión eficiente del tráfico.

Seguridad:

La seguridad de la red es una consideración crítica en cualquier entorno, virtualizado o no. Implementaremos medidas de seguridad en varios niveles, incluyendo la autenticación de usuarios, el cifrado de datos, la segmentación de red y la detección y prevención de intrusiones. Además, realizaremos pruebas de penetración y evaluaciones de seguridad para identificar y mitigar posibles vulnerabilidades en la red virtualizada.

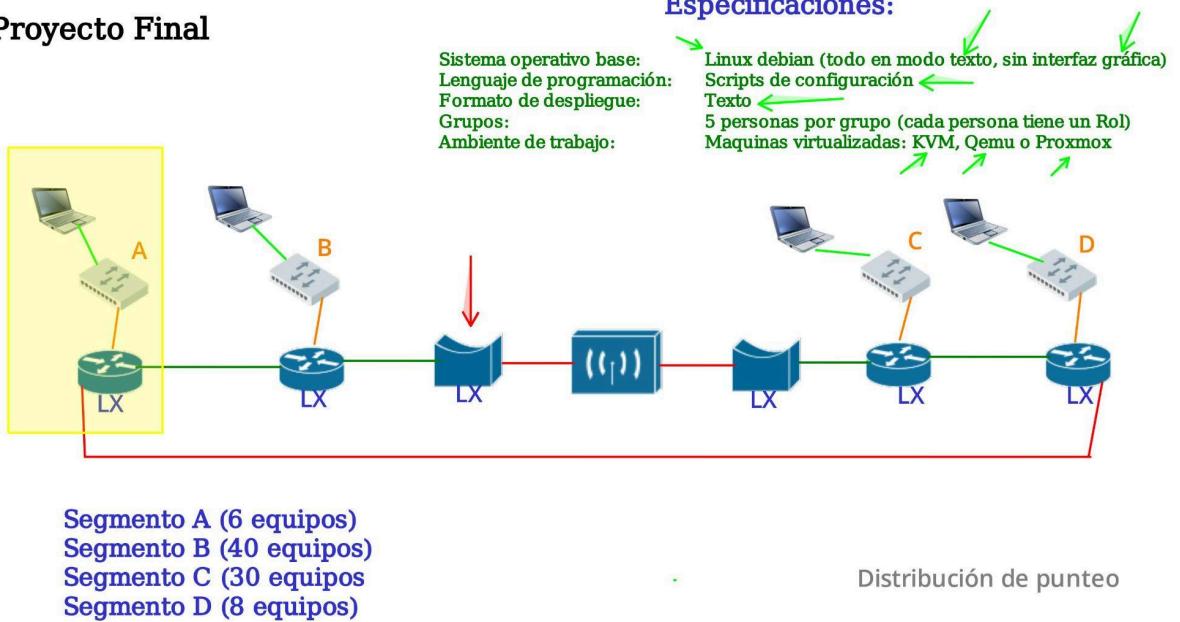
QEMU (Quick Emulator):

QEMU es una herramienta de virtualización de código abierto que permite la emulación de hardware y la virtualización de sistemas operativos. Ofrece la capacidad de ejecutar sistemas operativos invitados en diferentes arquitecturas de CPU, lo que lo convierte en una opción flexible y versátil para crear entornos de virtualización. En el contexto de este proyecto, utilizaremos QEMU para crear y gestionar máquinas virtuales que actuarán como clientes, servidores y dispositivos de red en nuestra red virtualizada.

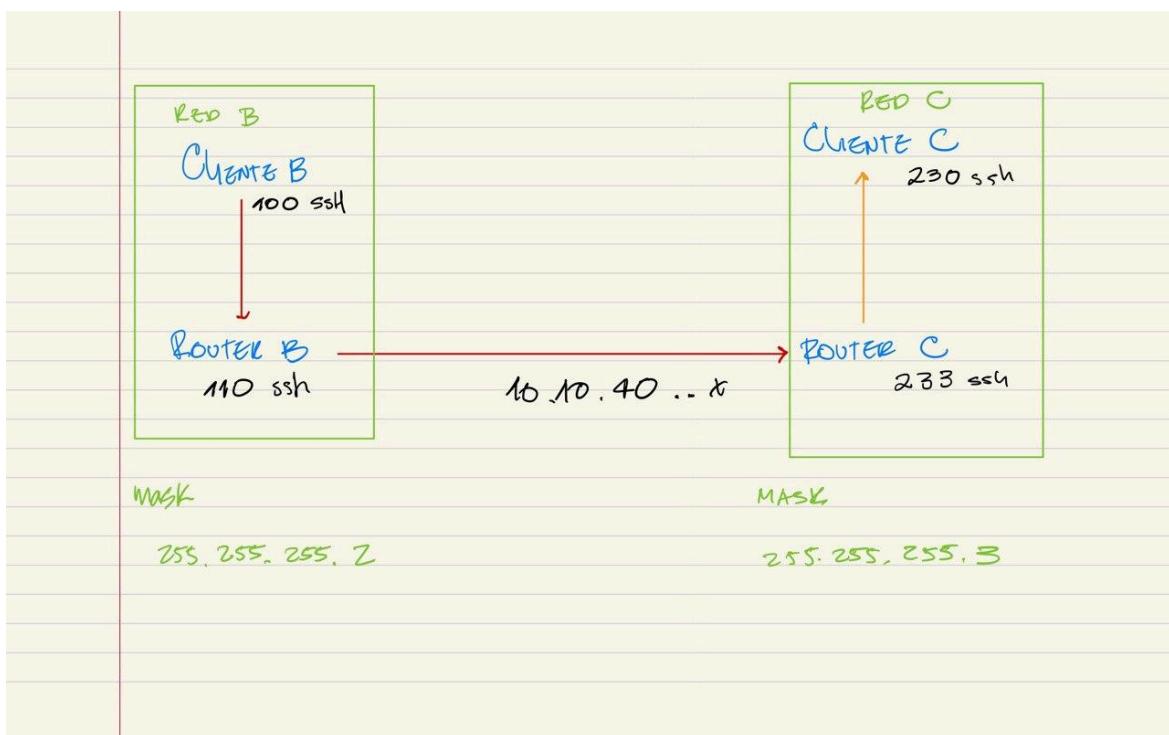
Problema

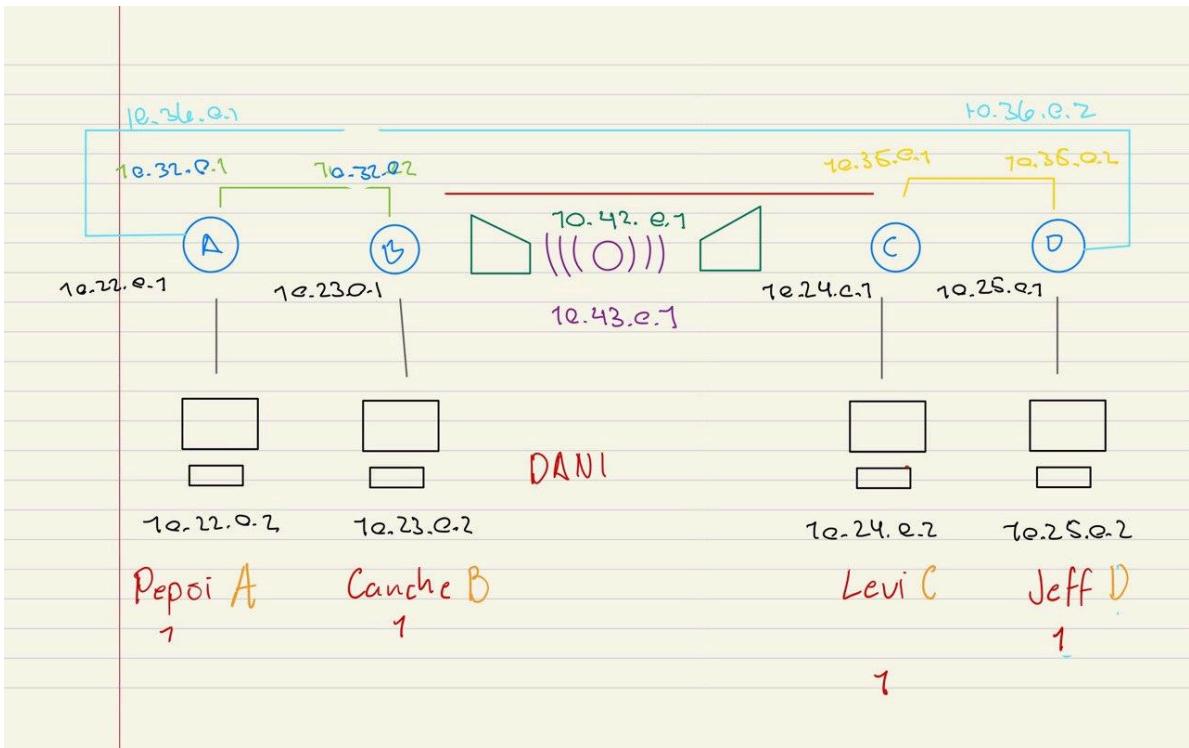
Redes de computadoras 1

Proyecto Final



Solución



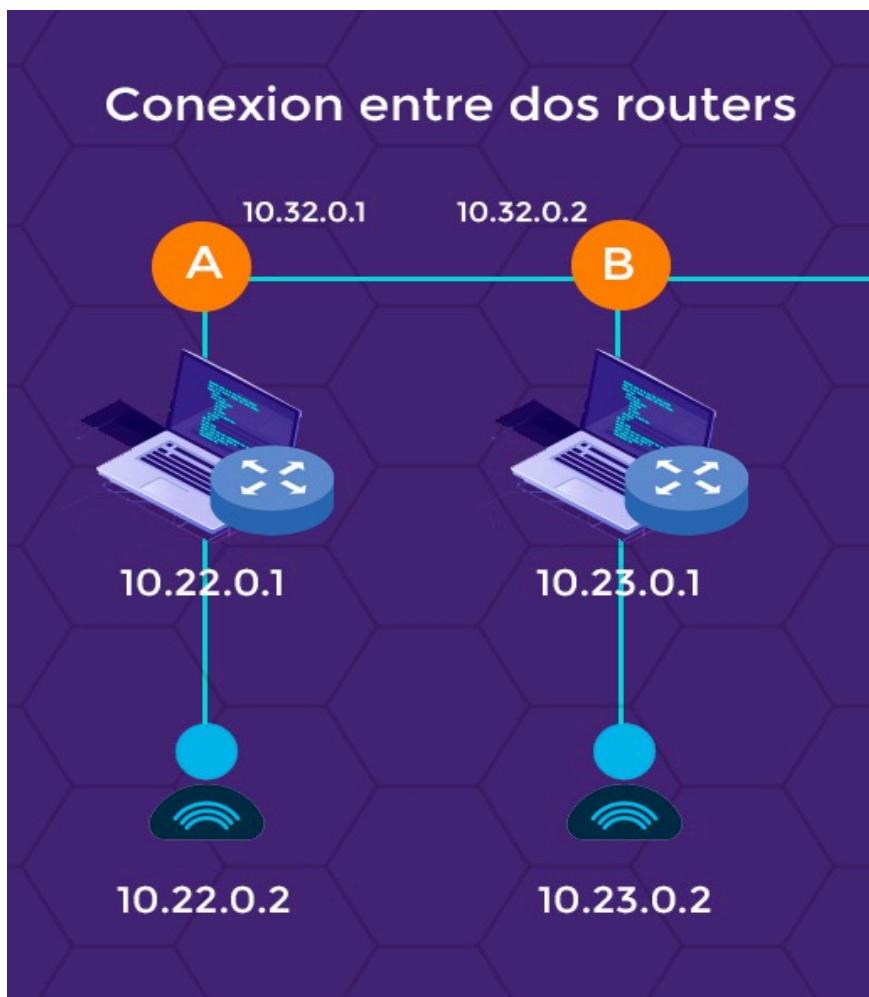


Para la resolución de este problema decidimos dividir por pedazos las redes y así poder trabajar paso a paso.

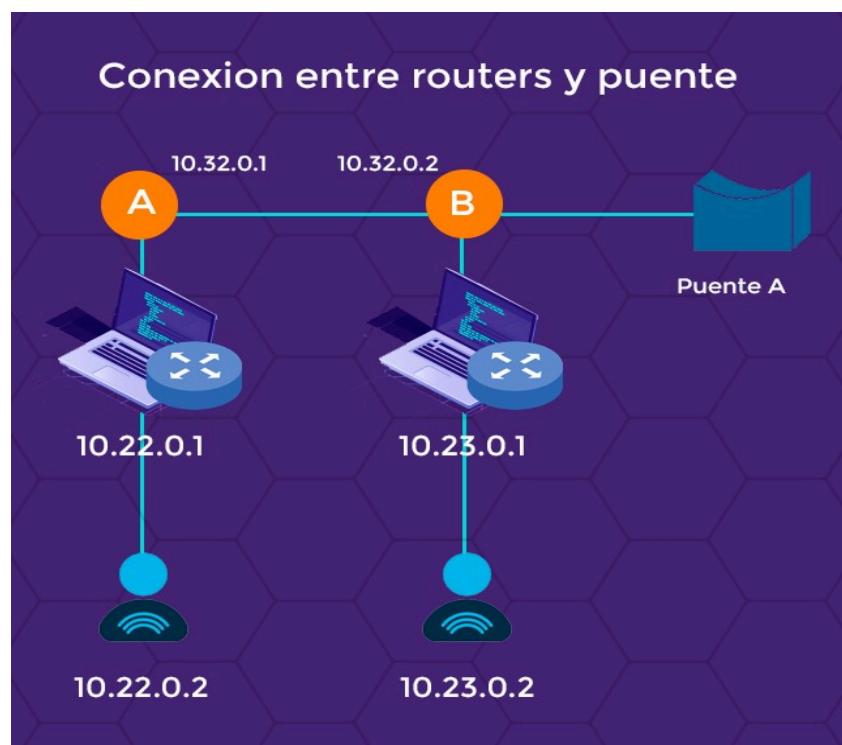
1. Ver la conexión del cliente con su router mediante una red privada



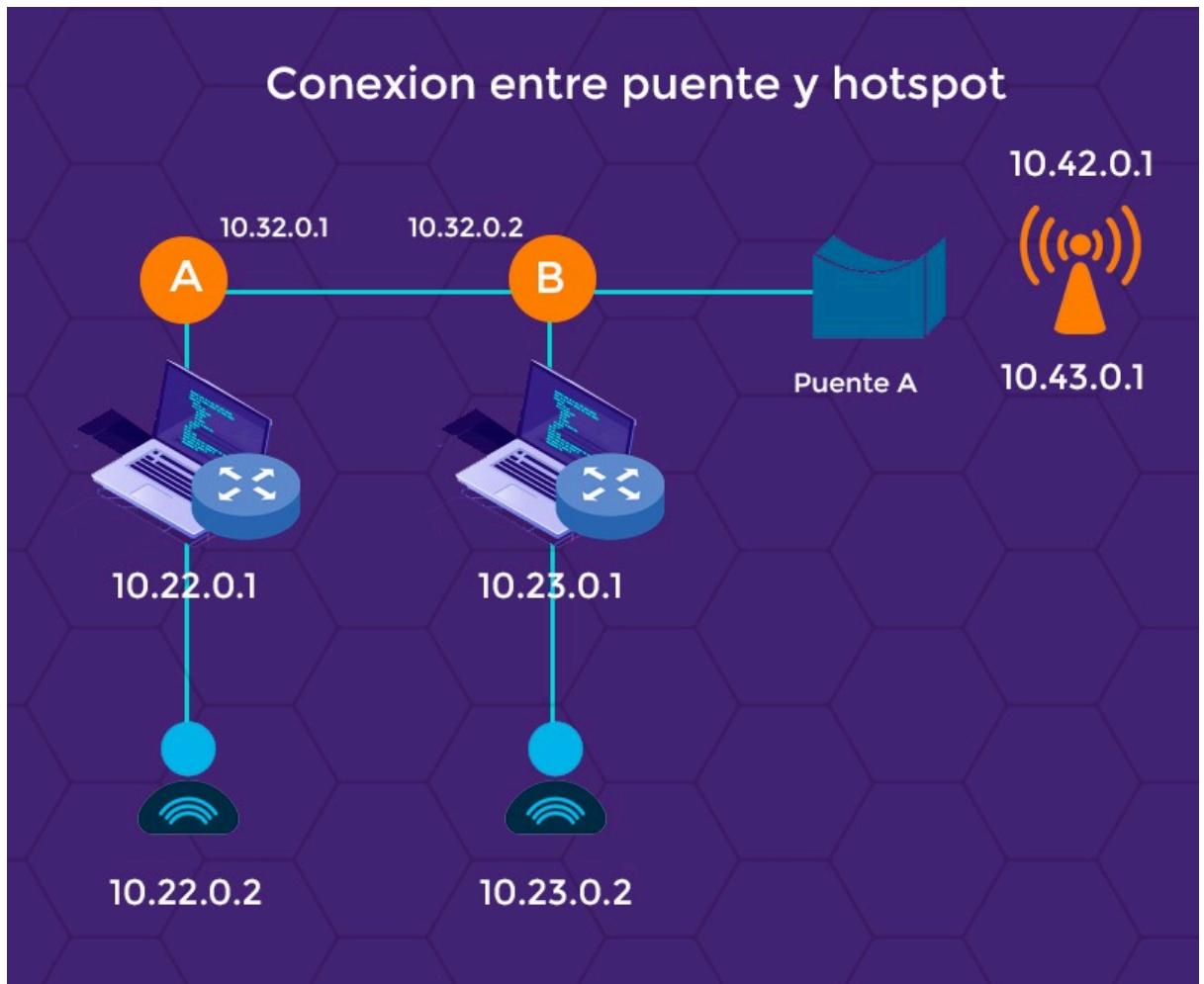
2. “Replicamos” esta configuración para crear la red B, seguidamente procedemos a conectarlas por la interfaz ethernet



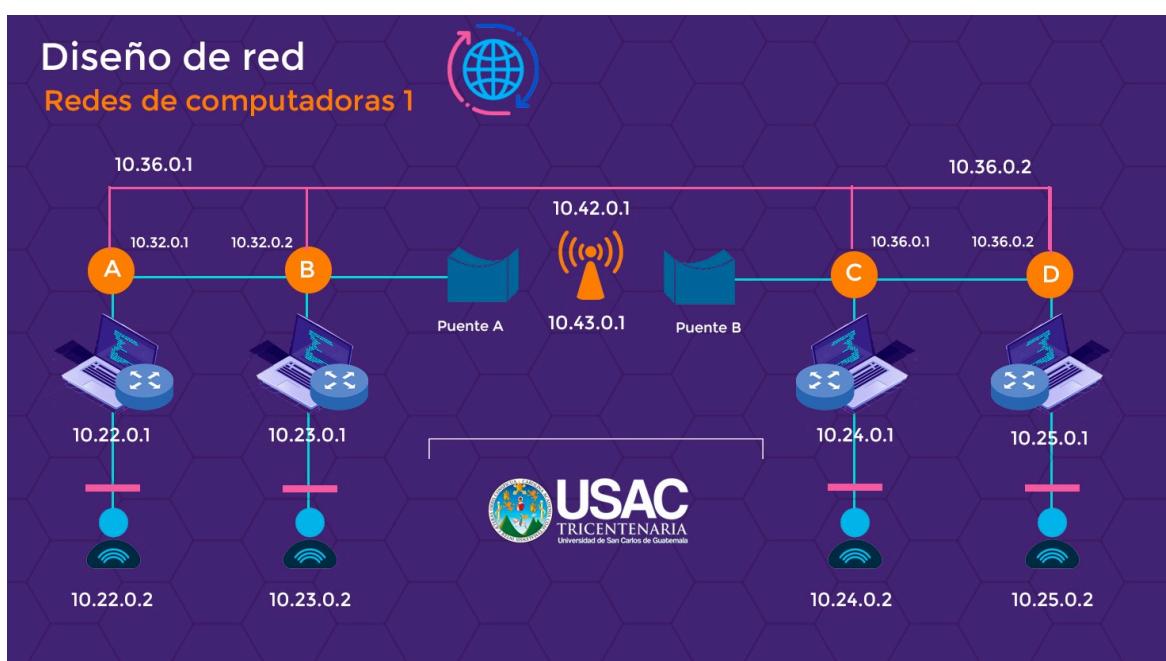
3. La configuración anterior, se debe conectar al puente transparente



4. Y el puente anterior, debe estar conectado a nuestro hotspot configurado en kali linux



5. Finalmente debemos replicar esta parte, para la otra parte y quedaría conectada la red.

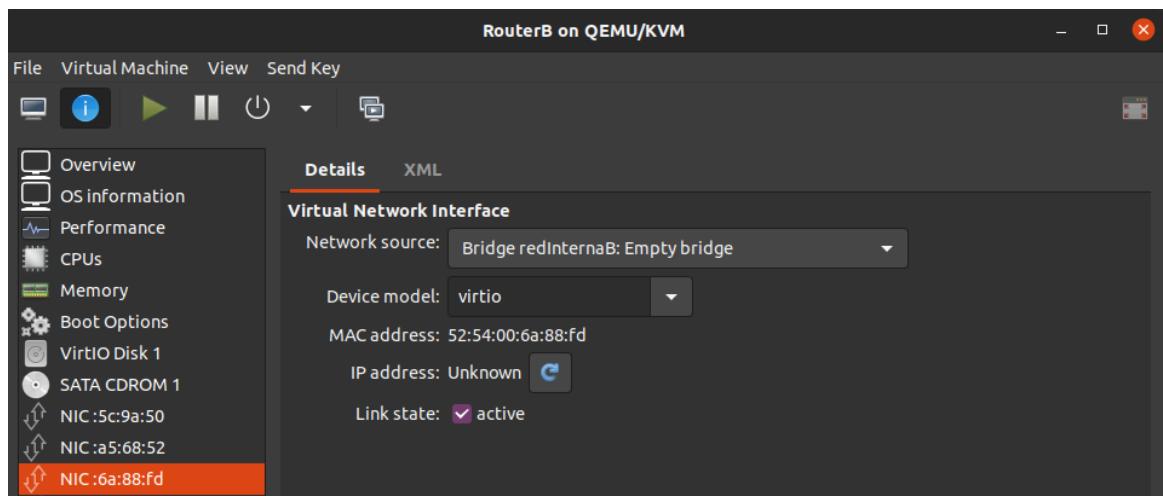


Empezaremos a explicar la solución del proyecto:

1. Tenemos que crear una interfaz llamada RedInterna (A,B,C,D), para poder tratar cada red privada, por lo cual el router tendrá acceso mediante una red privada a su cliente, pero el router se comunicará con los demás routers mediante una red pública.

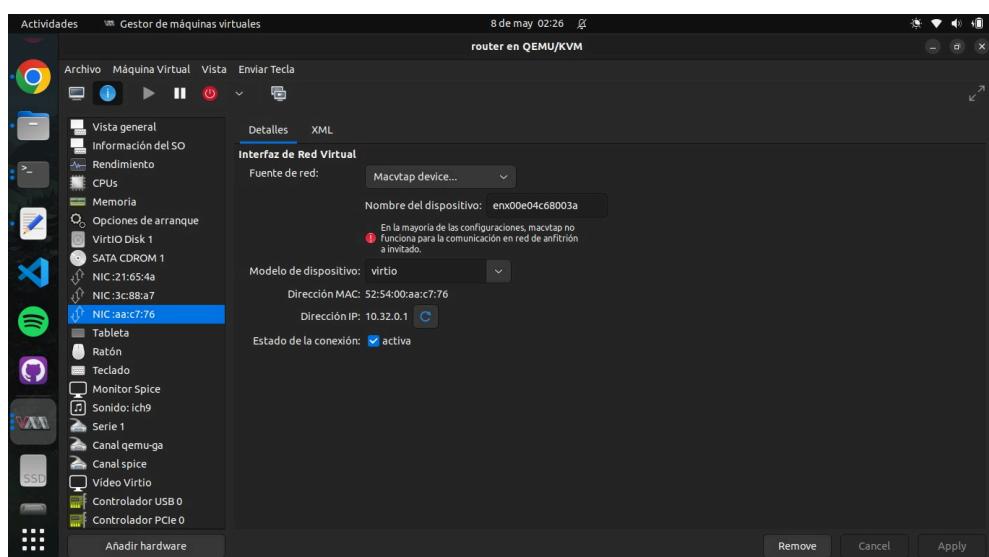
```
sudo brctl addbr redInternaB  
sudo ip link set dev redInternaB up  
  
sudo systemctl restart NetworkManager.service
```

2. Debemos añadir esta red interna a qemu para poder así manejar las diferentes redes privadas de A,B,C,D



A su vez, los computadores que actuarán como A Y D se les debe añadir la interfaz de wlan, para que así estas máquinas virtuales utilicen estas interfaces como si fueran de ellas.

Mientras que las máquinas virtuales que actuarán como router deberán agregar la interfaz de ethernet para usar esta interfaz a su vez como si fuera de él.



Script cliente, este script recibe la flag del equipo que se desea configurar y el nombre de la interfaz que fue generada en la máquina virtual

```
INTERFAZ=$2

if [ $1 == "A" ]
then

    ip addr add 10.22.0.2/29 dev $INTERFAZ
    ip link set $INTERFAZ up

    ip route add default via 10.22.0.1 dev $INTERFAZ

elif [ $1 == "B" ]
then

    ip addr add 10.23.0.2/26 dev $INTERFAZ
    ip link set $INTERFAZ up

    ip route add default via 10.23.0.1 dev $INTERFAZ

elif [ $1 == "C" ]
then
    ip addr add 10.24.0.2/27 dev $INTERFAZ
    ip link set $INTERFAZ up
    ip route add default via 10.24.0.1 dev $INTERFAZ
elif [ $1 == "D" ]
then
    ip addr add 10.25.0.2/29 dev $INTERFAZ
    ip link set $INTERFAZ up
    ip route add default via 10.25.0.1 dev $INTERFAZ
fi
```

Y de ahí tenemos el script para los router's A Y D que serían los “extremos”

```
#!/bin/bash
INTERFAZ_RED_INTERNA=$2
INTERFAZ_OTRO_ROUTER=$3
sysctl -w net.ipv4.ip_forward=1
if [ $1 == "A" ]
then
```

```

# AGREGAMOS LA IP DE LA RED INTERNA A LA INTERFAZ Y LEVANTAMOS
ip addr add 10.22.0.1/29 dev $INTERFAZ_RED_INTERNA
ip link set $INTERFAZ_RED_INTERNA up
# AGREGAMOS LA IP PÚBLICA CON LA SE COMUNICARÁ A CON B Y
LEVANTAMOS
ip addr add 10.32.0.1/30 dev $INTERFAZ_OTRO_ROUTER
ip link set $INTERFAZ_OTRO_ROUTER up
# DIRIGIMOS TODO EL TRÁFICO DEL ROUTER AL SIGUIENTE ROUTER (B)
ip route add default via 10.32.0.2 dev $INTERFAZ_OTRO_ROUTER
# pendiente ver si necesita msaquerada
elif [ $1 == "D" ]
then
# AGREGAMOS LA IP DE LA RED INTERNA A LA INTERFAZ Y LEVANTAMOS
ip addr add 10.25.0.1/29 dev $INTERFAZ_RED_INTERNA
ip link set $INTERFAZ_RED_INTERNA up
# AGREGAMOS LA IP PÚBLICA CON LA SE COMUNICARÁ D CON C Y
LEVANTAMOS
ip addr add 10.35.0.2/30 dev $INTERFAZ_OTRO_ROUTER
ip link set $INTERFAZ_OTRO_ROUTER up
# DIRIGIMOS TODO EL TRÁFICO DEL ROUTER AL SIGUIENTE ROUTER (C)
ip route add default via 10.35.0.1 dev $INTERFAZ_OTRO_ROUTER
# pendiente ver si necesita msaquerade
fi

```

```

# CONFIGURAMOS LA TABLA NAT PARA ENMASCARAR EL PAQUETE LUEGO DE
ENRUTARLO
# AL ROUTER PERO ANTES DE ENVIARLO
# iptables -t nat -A POSTROUTING -o $INTERFAZ_OTRO_ROUTER -j
MASQUERADE

```

Este script recibe como flags, la red que se está configurando, la interfaz de la red interna, y la interfaz de la red pública

De ahí tenemos el script para las router's B Y C los cuales son los que se conectarán a los puentes transparentes.

```

#!/bin/bash
INTERFAZ_INTERNA=$2
INTERFAZ_OTRO_ROUTER=$3
INTERFAZ_PUENTE=$4
sysctl -w net.ipv4.ip_forward=1

```

```

if [ $1 == "B" ]
then
    # ASIGNAMOS LA IP DE LA RED INTERNA A LA INTERFAZ Y LEVANTAMOS
    ip addr add 10.23.0.1/26 dev $INTERFAZ_INTERNA
    ip link set $INTERFAZ_INTERNA up

    # ASIGNAMOS LA IP CON LA QUE SE CONECTARÁ B CON A
    ip addr add 10.32.0.2/30 dev $INTERFAZ_OTRO_ROUTER
    ip link set $INTERFAZ_OTRO_ROUTER up

    # REDIRECCIONAMOS EL TRÁFICO RELACIONADO CON A A SU IP PÚBLICA
    ip route add 10.22.0.0/29 via 10.32.0.1 dev
$INTERFAZ_OTRO_ROUTER

    # REDIRECCIONAMOS TODO EL DEMÁS TRÁFICO HACIA EL PUENTE
    ip route add default via 10.42.0.1 dev $INTERFAZ_PUENTE

elif [ $1 == "C" ]
then
    # ASIGNAMOS LA IP DE LA RED INTERNA A LA INTERFAZ Y LEVANTAMOS
    ip addr add 10.24.0.1/27 dev $INTERFAZ_INTERNA
    ip link set $INTERFAZ_INTERNA up

    # ASIGNAMOS LA IP CON LA QUE SE CONECTARÁ C A D
    ip addr add 10.35.0.1/30 dev $INTERFAZ_OTRO_ROUTER
    ip link set $INTERFAZ_OTRO_ROUTER up

    # REDIRECCIONAMOS EL TRÁFICO RELACIONADO CON D A SU IP PÚBLICA
    ip route add 10.25.0.0/29 via 10.35.0.2 dev
$INTERFAZ_OTRO_ROUTER

    # REDIRECCIONAMOS TODO EL DEMÁS TRÁFICO HACIA EL PUENTE
    ip route add default via 10.42.0.1 dev $INTERFAZ_PUENTE

fi

# CONFIGURAMOS LA TABLA NAT PARA ENMASCARAR EL PAQUETE LUEGO DE
# ENRUTARLO
# AL PUENTE PERO ANTES DE ENVIARLO
iptables -t nat -A POSTROUTING -o $INTERFAZ_PUENTE -j MASQUERADE

```

Y por último tenemos el script para la configuración del puente transparente:

```
#!/bin/bash

IP_ROUTER=$1
INTERFAZ=$2

if [ $1 == "D" ]
then
    ip route add 10.23.0.0/26 via $IP_ROUTER dev $INTERFAZ
    ip route add 10.22.0.0/29 via $IP_ROUTER dev $INTERFAZ
    iptables -t nat -A PREROUTING -d 10.23.0.2 -j DNAT --to-destination
$IP_ROUTER
    iptables -t nat -A PREROUTING -d 10.22.0.2 -j DNAT --to-destination
$IP_ROUTER

else
    ip route add 10.24.0.0/27 via $IP_ROUTER dev $INTERFAZ
    ip route add 10.25.0.0/29 via $IP_ROUTER dev $INTERFAZ
    iptables -t nat -A PREROUTING -d 10.24.0.2 -j DNAT --to-destination
$IP_ROUTER
    iptables -t nat -A PREROUTING -d 10.25.0.2 -j DNAT --to-destination
$IP_ROUTER
fi
```

Este script recibe como flags, la ip del router y la interfaz que utiliza que en este caso sería la interfaz wlan.

Conclusión

Este proyecto ha sido una inmersión completa en el vasto campo de la virtualización de redes, utilizando herramientas como QEMU, sistemas operativos como Debian 11 y Kali Linux, y tecnologías como iptables, enrutamiento y puntos de acceso Wi-Fi. Al reflexionar sobre esta experiencia y considerar el marco teórico que hemos explorado, se pueden extraer varias conclusiones importantes:

- **Amplio Conocimiento de la Virtualización de Redes:** A través de la práctica directa con herramientas como QEMU y la implementación de diferentes sistemas operativos como Debian 11 y Kali Linux, he adquirido un conocimiento sólido de los fundamentos de la virtualización de redes. Comprender los conceptos detrás de la creación y gestión de redes virtuales es fundamental para abordar desafíos en entornos de red cada vez más complejos.
- **Aplicación Efectiva del Marco Teórico:** El marco teórico proporcionado ha sido fundamental para orientar mis acciones y decisiones en este proyecto. Desde la comprensión de los principios del enrutamiento hasta la aplicación de reglas de iptables para la seguridad de la red, cada aspecto teórico ha encontrado una aplicación práctica en la configuración y gestión de nuestra red virtualizada.
- **Desarrollo de Habilidades Prácticas y Transversales:** Este proyecto me ha brindado la oportunidad de desarrollar habilidades prácticas esenciales en el campo de las redes, así como habilidades transversales como la resolución de problemas, la planificación y la gestión de proyectos. La capacidad de configurar y administrar redes virtuales complejas es una habilidad valiosa en el mundo tecnológico actual y en constante evolución.
- **Exploración de Diversos Escenarios de Uso:** A lo largo de este proyecto, hemos explorado una amplia gama de escenarios de uso, desde la configuración de enrutadores hasta la implementación de puntos de acceso Wi-Fi. Esta exploración nos ha permitido comprender las aplicaciones prácticas de la virtualización de redes en diferentes contextos, desde entornos domésticos hasta entornos empresariales.

Anexos

