

Memoria de prácticas

Medición y Estimación de la Matriz de Tráfico para la Planificación Óptima

DISEÑO Y DIMENSIONAMIENTO DE REDES

Enrique Alcalá-Zamora Castro — enriqueaz@correo.ugr.es

Juan Pablo Cano López — jupacanolopez@correo.ugr.es

ÍNDICE

1-OBJETIVOS/ INTRODUCCIÓN

2-ACTIVIDAD 1: CONFIGURACIÓN DE LA ESTRUCTURA DE LA RED

3-ACTIVIDAD 2: TRANSMISIÓN DE TRÁFICO CON LA HERRAMIENTA IPERF

4-ACTIVIDAD 3: MEDIDAS DE TRÁFICO, NETFLOW, NFCAPD, NFDUMP

5-ACTIVIDAD 4: OBTENCIÓN DE LA MATRIZ DE TRÁFICO CON PYTHON

6-ACTIVIDAD 5: OPTIMIZACIÓN DE LA RED DESCRITA CON LA MATRIZ DE TRÁFICO OBTENIDA ANTERIORMENTE

OBJETIVOS/ INTRODUCCIÓN

Las prácticas de la asignatura *Diseño y Dimensionado de Redes* consisten en obtener una matriz de tráfico de una red a partir de medidas reales en esta, habiendo realizado la configuración de la estructura de VLANs y conexiones indicadas en la topología de red de la *actividad 1* y posteriormente realizar una optimización de red con esa matriz como parámetro de entrada en la *actividad 6*.

La *actividad 2* engloba desde el momento en que se configura la estructura de red hasta que los PCs se encuentran transmitiendo datos, la *actividad 3* desde el punto anterior hasta que se obtienen los archivos csv con las medidas y la *actividad 4* termina con la obtención de la matriz de tráfico en python a partir de los archivos csv previos.

ACTIVIDAD 1: CONFIGURACIÓN DE LA ESTRUCTURA DE LA RED

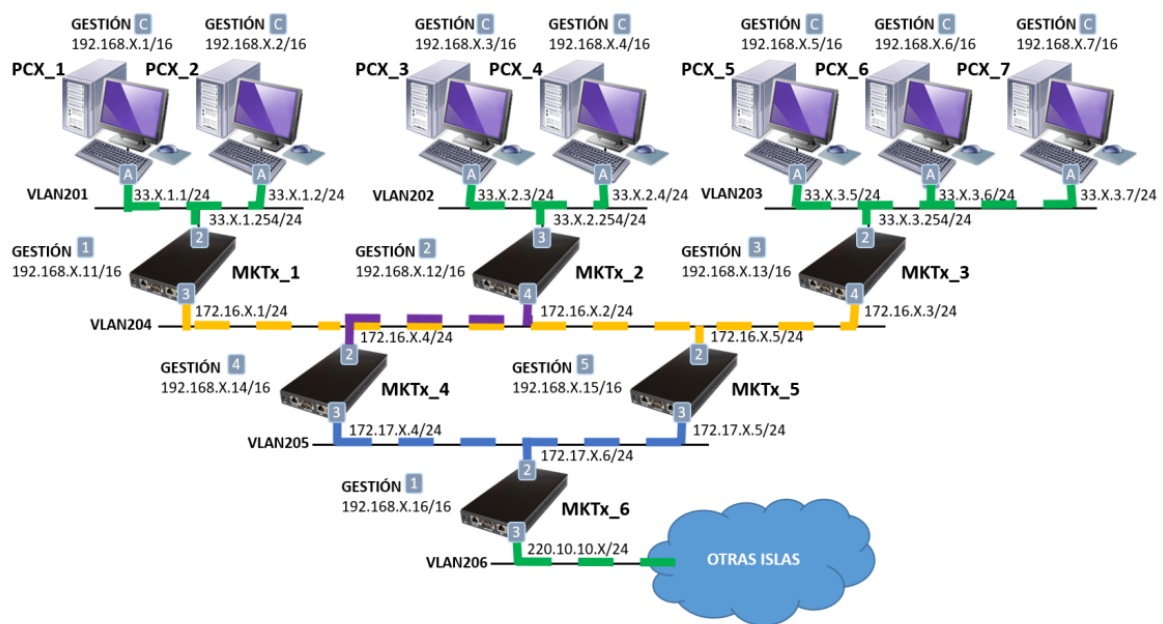


Fig. 1. Topología de la red a configurar, nótese que la conexión entre nodos se realizará según los caminos marcados en esta.

```
#!/bin/sh

PC=$1
ISLA=$2

if [ $PC -lt 3 ]
then
    LAN=1
elif [ $PC -lt 5 ]
then
    LAN=2
else
    LAN=3
fi

# ejecutar con sudo
echo "Acquire::http::proxy \"http://\";" > /etc/apt/apt.conf
if [ ! -f /etc/apt/sources.list.bak0 ]
then
    sed -i.bak0 "s/archive\.ubuntu\.com/192\.168\.33\.21/g" /etc/apt/sources.list
    sed -i "s/ubuntu/ubuntu\/mirror\/archive\.ubuntu\.com\/ubuntu/g" /etc/apt/sources.list
    sed -i "s/deb http/deb [trusted=yes, arch=amd64] http/g" /etc/apt/sources.list
    apt update
    echo "192.168.33.21 eihal.ugr.es eihal eihal.labredes.pri" >> /etc/hosts
fi

#Instala traceroute

apt-get -y --allow-unauthenticated install traceroute

sudo route del default
sudo route add default gw 33.$ISLA.$LAN.254
echo $LAN
```

Fig. 2. Script que configurará la ruta por defecto de cada uno de los PC, adicionalmente configura el uso de los repositorios del laboratorio para la instalación de herramientas como *traceroute*, *iperf* o *nfdump/nfcapd*.

```
#!/bin/sh

PC=$1
ISLA=$2

if [ $PC -lt 3 ]
then
    LAN=1
elif [ $PC -lt 5 ]
then
    LAN=2
else
    LAN=3
fi

#####Configuracion de los enrutadores Mikrotik

#R1
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.11 "/ip route add dst-address=33.$ISLA.2.0/24 gateway=172.16.$ISLA.5"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.11 "/ip route add dst-address=33.$ISLA.3.0/24 gateway=172.16.$ISLA.3"

#R2
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.12 "/ip route add dst-address=33.$ISLA.1.0/24 gateway=172.16.$ISLA.4"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.12 "/ip route add dst-address=33.$ISLA.3.0/24 gateway=172.16.$ISLA.4"

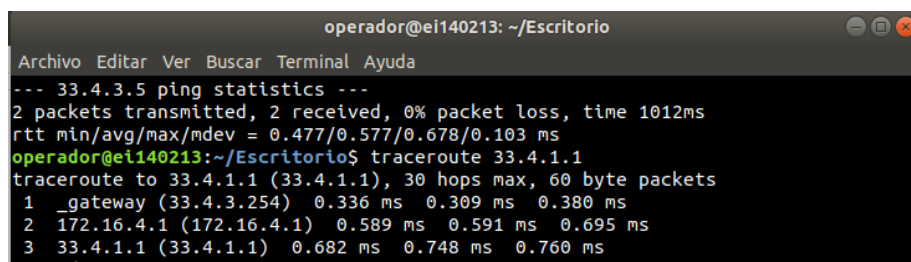
#R3
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.13 "/ip route add dst-address=33.$ISLA.1.0/24 gateway=172.16.$ISLA.1"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.13 "/ip route add dst-address=33.$ISLA.2.0/24 gateway=172.16.$ISLA.5"

#R4
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.14 "/ip route add dst-address=33.$ISLA.1.0/24 gateway=172.17.$ISLA.5"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.14 "/ip route add dst-address=33.$ISLA.2.0/24 gateway=172.16.$ISLA.2"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.14 "/ip route add dst-address=33.$ISLA.3.0/24 gateway=172.17.$ISLA.5"

#R5
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.15 "/ip route add dst-address=33.$ISLA.1.0/24 gateway=172.16.$ISLA.1"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.15 "/ip route add dst-address=33.$ISLA.2.0/24 gateway=172.17.$ISLA.4"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.15 "/ip route add dst-address=33.$ISLA.3.0/24 gateway=172.16.$ISLA.3"

#R6
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.16 "/ip route add dst-address=33.$ISLA.1.0/24 gateway=172.17.$ISLA.5"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.16 "/ip route add dst-address=33.$ISLA.2.0/24 gateway=172.17.$ISLA.4"
ssh -o "StrictHostKeyChecking=no" admin@192.168.$ISLA.16 "/ip route add dst-address=33.$ISLA.3.0/24 gateway=172.17.$ISLA.5"
```

Fig. 3. Script de configuración de los routers en el que se seguirán los caminos especificados en (1). Este script realiza para cada uno de los routers la conexión ssh (parámetro -o para deshabilitar el chequeo y confirmación de claves cada vez que cambia en el host), a los cuales se les ejecuta de forma personalizada el comando de adición de rutas según cada LAN de destino.



```
operador@ei140213: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
--- 33.4.3.5 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1012ms
rtt min/avg/max/mdev = 0.477/0.577/0.678/0.103 ms
operador@ei140213:~/Escritorio$ traceroute 33.4.1.1
traceroute to 33.4.1.1 (33.4.1.1), 30 hops max, 60 byte packets
 1 _gateway (33.4.3.254) 0.336 ms 0.309 ms 0.380 ms
 2 172.16.4.1 (172.16.4.1) 0.589 ms 0.591 ms 0.695 ms
 3 33.4.1.1 (33.4.1.1) 0.682 ms 0.748 ms 0.760 ms
operador@ei140213:~/Escritorio$
```

Fig. 4. Traceroute ejecutado desde PC6 a PC1 para comprobar que efectivamente, hay conexión, además de que se siguen los caminos definidos en (1).

```
operador@el140213: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
64 bytes from 33.4.3.5: icmp_seq=5 ttl=64 time=0.470 ms
64 bytes from 33.4.3.5: icmp_seq=6 ttl=64 time=0.474 ms
^C
--- 33.4.3.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5100ms
rtt min/avg/max/mdev = 0.341/0.424/0.474/0.053 ms
operador@el140213:~/Escritorio$ ping -R 33.4.1.1
PING 33.4.1.1 (33.4.1.1) 56(124) bytes of data.
64 bytes from 33.4.1.1: icmp_seq=1 ttl=62 time=1.25 ms
RR: 33.4.3.6
    172.16.4.3
    33.4.1.254
    33.4.1.1
    33.4.1.1
    172.16.4.1
    33.4.3.254
    33.4.3.6

64 bytes from 33.4.1.1: icmp_seq=2 ttl=62 time=1.10 ms (same route)
^C
--- 33.4.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 1.108/1.181/1.255/0.081 ms
```

Fig. 5. Ping con parámetro -R nos muestra los nodos de salida de los paquetes ICMP, lo cual es una alternativa más rápida que *traceroute* para comprobar los caminos configurados. Nuevamente este ejemplo procede de una conexión desde PC6 a PC1.

```
operador@el140213: ~/Escritorio
Archivo Editar Ver Buscar Terminal Ayuda
operador@el140213:~/Escritorio$ ping -R 33.4.2.3
PING 33.4.2.3 (33.4.2.3) 56(124) bytes of data.
64 bytes from 33.4.2.3: icmp_seq=1 ttl=60 time=1.67 ms
RR: 33.4.3.6
    172.16.4.3
    172.17.4.5
    172.16.4.4
    33.4.2.254
    33.4.2.3
    33.4.2.3
    172.16.4.2
    172.17.4.4
```

Fig. 6. Ping con parámetro -R desde PC6 a PC3, en el que vemos que transcurre por R3, R4, R5 y R2, tal y como se requería en (1).

ACTIVIDAD 2: TRANSMISIÓN DE TRÁFICO CON LA HERRAMIENTA IPERF

Una vez que las rutas han sido configuradas y que cada una de las estaciones tienen visibilidad entre sí, el siguiente paso es generar tráfico mediante *iperf* en cada uno de los ordenadores. Para ello, cada PC ejecutará *iperf* tanto en calidad de cliente (envío de tráfico) como de servidor (recepción de este por parte de otros PC).

A modo de observación, dado a que será un paso necesario para los apartados siguientes, se ha observado (mediante *Wireshark*) que el tamaño de los paquetes originados por esta herramienta tienen un **tamaño por defecto de 1500 bytes**, lo cual nos será de utilidad para el tratamiento de los datos de la matriz de tráfico.

```
#Instalacion de iperf
apt-get -y --allow-unauthenticated install iperf

#Iperf modo demonio

sudo iperf -s -D -p5001
sudo iperf -s -D -p5002
sudo iperf -s -D -p5004
sudo iperf -s -D -p5005
sudo iperf -s -D -p5006
```

Fig. 7. Script de configuración de *iperf* a modo de servidor (-s) y demonio (-d) tras su instalación en cada uno de los PC. Este se pondrá a escuchar en cada puerto dedicado a los clientes remotos (uno por cada PC excepto este mismo, visualizando el caso de PC3).

```
#!/bin/sh

PC=$1
ISLA=$2

if [ $PC -lt 3 ]
then
LAN=1
elif [ $PC -lt 5 ]
then
LAN=2
else
LAN=3
fi

iperf -t 3600 -c 33.$ISLA.1.1 -p 500$1 &
iperf -t 3600 -c 33.$ISLA.1.2 -p 500$1 &
iperf -t 3600 -c 33.$ISLA.2.4 -p 500$1 &
iperf -t 3600 -c 33.$ISLA.3.5 -p 500$1 &
iperf -t 3600 -c 33.$ISLA.3.6 -p 500$1 &
```

Fig. 8. Script de configuración de *iperf* a modo de cliente (-c) para cada uno de los PC salvo el emisor, el cual enviará tráfico durante una hora (-t en segundos) desde el puerto destinado a cada emisor (-p), en segundo plano (&).

Los del resto de clientes son análogos al de PC3, siendo la única variación que no se incluirá la ip del PC que ejecutará el script, del mismo modo que PC3 no incluye una sentencia para conectarse a 33.4.2.3, su propia IP.

Se ejecutan primero los script de servidor en todos los PCs desde el terminal y seguidamente los script de los clientes en cada PC, teniendo especial atención la clarificación anterior, en el que emplearemos una versión de este script modificado para el caso de cada PC.

Finalmente se incluyen varias capturas de pantalla de una captura tráfico de wireshark que muestran tráfico llegando correctamente a diferentes PCs.

TraficoTodoslosPC.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.port == 5001

No.	Time	Source	Destination	Protocol	Length	Info
2167	0.0916317...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
201...	0.8352013...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
202...	0.8378210...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
202...	0.8391334...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
203...	0.8417588...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
203...	0.8423546...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
81	0.0032861...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
203...	0.8432254...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
204...	0.8468029...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353
204...	0.8481331...	33.4.3.6	33.4.1.1	TCP	68	5001 → 41384 [ACK] Seq=1 Ack=143353

> Frame 2167: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0

> Linux cooked capture v1

> Internet Protocol Version 4, Src: 33.4.3.6, Dst: 33.4.1.1

> Transmission Control Protocol, Src Port: 5001, Dst Port: 41384, Seq: 1, Ack: 143353, Len: 0

Fig. 9. Tráfico Iperf con origen PC6 y destino PC1 desde el puerto 5001.

TraficoTodoslosPC.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.port == 5002

No.	Time	Source	Destination	Protocol	Length	Info
212...	0.8811360...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
213...	0.8855877...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
214...	0.8889922...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
215...	0.8933298...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
216...	0.8967725...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
217...	0.9008034...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
218...	0.9047519...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
218...	0.9085767...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
2159	0.0913686...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809
219...	0.9125196...	33.4.3.6	33.4.1.2	TCP	68	5002 → 34302 [ACK] Seq=1 Ack=645809

> Frame 21238: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0

> Linux cooked capture v1

> Internet Protocol Version 4, Src: 33.4.3.6, Dst: 33.4.1.2

> Transmission Control Protocol, Src Port: 5002, Dst Port: 34302, Seq: 1, Ack: 645809, Len: 0

Fig. 10. Tráfico Iperf con origen PC6 y destino PC2 desde el puerto 5002.

TraficoTodoslosPC.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.port == 5003

No.	Time	Source	Destination	Protocol	Length	Info
272...	1.1350851...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
272...	1.1370144...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
273...	1.1402805...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
3072	0.1292612...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
261	0.0108180...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
273...	1.1410293...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
274...	1.1448587...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
275...	1.1470774...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
275...	1.1495097...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209
276...	1.1505422...	33.4.3.6	33.4.2.3	TCP	68	5003 → 41412 [ACK] Seq=1 Ack=1442209

> Frame 27248: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0

> Linux cooked capture v1

> Internet Protocol Version 4, Src: 33.4.3.6, Dst: 33.4.2.3

> Transmission Control Protocol, Src Port: 5003, Dst Port: 41412, Seq: 1, Ack: 1442209, Len: 0

Fig. 11. Tráfico Iperf con origen PC6 y destino PC3 desde el puerto 5003.

TraficoTodoslosPC.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.port == 5004

No.	Time	Source	Destination	Protocol	Length	Info
258...	1.0744525...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=829
259...	1.0782815...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=832
259...	1.0795193...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=835
259...	1.0816827...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=838
2548	0.1081825...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=839
263...	1.0978727...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=841
264...	1.0997644...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=844
264...	1.1006958...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=847
265...	1.1070191...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=849
266...	1.1095145...	33.4.3.6	33.4.2.4	TCP	68	5004 → 48930 [ACK] Seq=1 Ack=852

> Frame 25816: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0

> Linux cooked capture v1

> Internet Protocol Version 4, Src: 33.4.3.6, Dst: 33.4.2.4

> Transmission Control Protocol, Src Port: 5004, Dst Port: 48930, Seq: 1, Ack: 829705, Len: 0

Fig. 12. Tráfico Iperf con origen PC6 y destino PC4 desde el puerto 5004.

TraficoTodoslosPC.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

tcp.port == 5005

No.	Time	Source	Destination	Protocol	Length	Info
189...	7.9168512...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=35030017 W:
189...	7.9174760...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=35032913 W:
189...	7.9180993...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=35035809 W:
189...	7.9188480...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=35038705 W:
189...	7.9193665...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=35041601 W:
201...	0.8371207...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=3504161 Win:
1875	0.0788517...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=350417 Win:
189...	7.9198645...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=35044497 W:
189...	7.9202595...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=35047393 W:
189...	7.9209038...	33.4.3.6	33.4.3.5	TCP	68	5005 → 59578 [ACK] Seq=1 Ack=35050289 W:

> Frame 25757: 1516 bytes on wire (12128 bits), 1516 bytes captured (12128 bits) on interface any, id 0

> Linux cooked capture v1

> Internet Protocol Version 4, Src: 33.4.3.5, Dst: 33.4.3.6

> Transmission Control Protocol, Src Port: 59578, Dst Port: 5005, Seq: 4504729, Ack: 1, Len: 1448

> Data (1448 bytes)

Fig. 13. Tráfico Iperf con origen PC6 y destino PC5 desde el puerto 5005.

ACTIVIDAD 3: MEDIDAS DE TRÁFICO, NETFLOW, NFCAPD, NFDUMP

En esta parte se realiza la medición del tráfico. Para ello se comienza entrando a los routers de acceso R1, R2 y R3, los directamente conectados con las LANs y habilitando netflow con el comando *ip traffic-flow set enabled=yes.*, como muestra la siguiente captura.

Netflow es una utilidad que captura datos estadísticos sobre el tráfico entrante y saliente de estos nodos con un mínimo *overhead* sobre el dispositivo, en el que cada cierto período de muestreo envía paquetes con estos datos a un colector, el cual configuraremos en esta sección.


```

operador@el140213: ~/Escritorio/PC6
Archivo Editar Ver Buscar Terminal Ayuda
MMM   MMM   III KKK KKK RRRRRR 000 000 TTT   III KKK KKK
MMM   MMM   III KKK KKK RRR  RRR 000000 TTT   III KKK KKK

MikroTik RouterOS 6.42.10 (c) 1999-2018      http://www.mikrotik.com/

[?]          Gives the list of available commands
command [?]  Gives help on the command and list of arguments

[Tab]        Completes the command/word. If the input is ambiguous,
              a second [Tab] gives possible options

/            Move up to base level
..           Move up one level
/command     Use command at the base level

[admin@MKT_4_3] > ip traffic-flow
[admin@MKT_4_3] /ip traffic-flow> set enabled=yes
[admin@MKT_4_3] /ip traffic-flow> print
enabled: yes
interfaces: all
cache-entries: 64k
active-flow-timeout: 30m
inactive-flow-timeout: 15s
[admin@MKT_4_3] /ip traffic-flow>

```

Fig. 14. Habilitación de Netflow en el router de acceso mediante *ip traffic-flow set enabled=yes*.

Posteriormente se añade la IP en la red de gestión del PC donde se quieren recibir los datos de ese flujo de tráfico y un puerto del PC al que conectarse como se muestra en la siguiente captura de pantalla

```

operador@el140213: ~/Escritorio/PC6
Archivo Editar Ver Buscar Terminal Ayuda

[admin@MKT_4_2] /ip traffic-flow> target add
copy-from port v9-template-refresh version
disabled src-address v9-template-timeout dst-address
[admin@MKT_4_2] /ip traffic-flow> target add 1
disabled dst-address
[admin@MKT_4_2] /ip traffic-flow> target add dst-address=192.168.4.6
copy-from port v9-template-refresh version
disabled src-address v9-template-timeout
[admin@MKT_4_2] /ip traffic-flow> target add dst-address=192.168.4.6 port=2055 v
ersion=9
[admin@MKT_4_2] /ip traffic-flow> print
as-value file interval without-paging
[admin@MKT_4_2] /ip traffic-flow> print detailed
expected end of command (line 1 column 7)
[admin@MKT_4_2] /ip traffic-flow> print detail
expected end of command (line 1 column 7)
[admin@MKT_4_2] /ip traffic-flow>
ipfix target edit export get monitor print set
[admin@MKT_4_2] /ip traffic-flow> target
[admin@MKT_4_2] /ip traffic-flow target> print detail
Flags: X - disabled
0 src-address=0.0.0.0 dst-address=192.168.4.6 port=2055 version=9
v9-template-refresh=20 v9-template-timeout=30m
[admin@MKT_4_2] /ip traffic-flow target>

```

Fig. 15. Habilitación de un colector de tráfico Netflow sobre la interfaz de gestión del mismo (192.168.4.6:2055).

En nuestro caso fue PC6 y el puerto 2055, que luego se cambió al 7678 debido a que el puerto por defecto se estaría utilizando por todos los PC colectores, lo cual produciría problemas

Seguidamente se utiliza la herramienta *nfcapd* para que se comience a transmitir tráfico.

La llamada a *nfcapd* se realiza con la siguiente estructura:

nfcapd -p [puerto configurado anteriormente] -l [carpeta de guardado] -t [tiempo captura nueva]

En nuestro caso quedó como:

nfcapd -p 7678 -l capturatrafico -t 60s

Posteriormente se debe realizar un tratamiento a estas capturas obtenidas para que sean legibles, una transformación de los archivos a formato csv. Esto se realiza con la herramienta *nfdump*, que debe ser previamente instalada con *sudo apt-get install nfdump*.

Una vez instalada realizamos el script de clasificación del tráfico según la LAN origen y la LAN destino que se muestra a continuación.

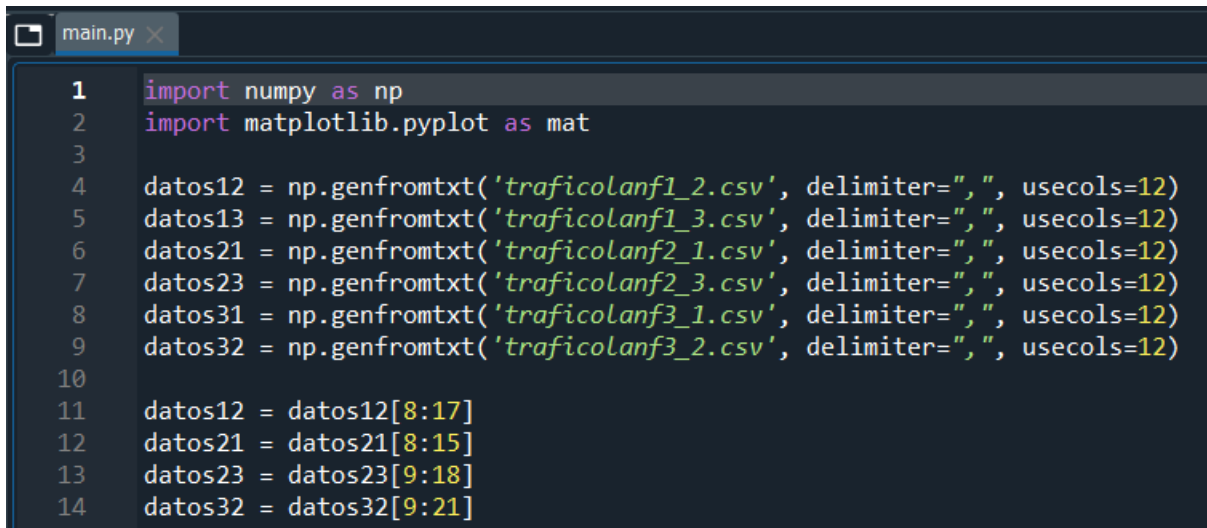
```
for variable in nfcapd.20*
do
#trafico lan1 lan2
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.1.1 and dst ip 33.4.2.3' -o csv >> traficolanf1_2
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.1.1 and dst ip 33.4.2.4' -o csv >> traficolanf1_2
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.1.2 and dst ip 33.4.2.3' -o csv >> traficolanf1_2
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.1.2 and dst ip 33.4.2.4' -o csv >> traficolanf1_2
#trafico lan1 lan3
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.1.1 and dst ip 33.4.3.5' -o csv >> traficolanf1_3
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.1.1 and dst ip 33.4.3.6' -o csv >> traficolanf1_3
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.1.2 and dst ip 33.4.3.5' -o csv >> traficolanf1_3
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.1.2 and dst ip 33.4.3.6' -o csv >> traficolanf1_3
#trafico lan2 lan1
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.2.3 and dst ip 33.4.1.1' -o csv >> traficolanf2_1
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.2.3 and dst ip 33.4.1.2' -o csv >> traficolanf2_1
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.2.4 and dst ip 33.4.1.1' -o csv >> traficolanf2_1
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.2.4 and dst ip 33.4.1.2' -o csv >> traficolanf2_1
#trafico lan2 lan3
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.2.3 and dst ip 33.4.3.5' -o csv >> traficolanf2_3
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.2.3 and dst ip 33.4.3.6' -o csv >> traficolanf2_3
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.2.4 and dst ip 33.4.3.5' -o csv >> traficolanf2_3
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.2.4 and dst ip 33.4.3.6' -o csv >> traficolanf2_3
#trafico lan3 lan1
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.3.5 and dst ip 33.4.1.1' -o csv >> traficolanf3_1
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.3.5 and dst ip 33.4.1.2' -o csv >> traficolanf3_1
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.3.6 and dst ip 33.4.1.1' -o csv >> traficolanf3_1
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.3.6 and dst ip 33.4.1.2' -o csv >> traficolanf3_1
#trafico lan3 lan2
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.3.5 and dst ip 33.4.2.3' -o csv >> traficolanf3_2
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.3.5 and dst ip 33.4.2.4' -o csv >> traficolanf3_2
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.3.6 and dst ip 33.4.2.3' -o csv >> traficolanf3_2
nfdump -r $variable -q -A srcip,dstip 'src ip 33.4.3.6 and dst ip 33.4.2.4' -o csv >> traficolanf3_2
done
```

Fig. 16. Script para el uso de *nfdump* sobre las muestras obtenidas previamente. Para ello diferenciaremos entre duplas de LAN origen y destino, además de que eliminaremos las cabeceras de los datos para agilizar el proceso de análisis (-q).

Una vez ejecutado este script se obtienen los archivos .csv que se utilizarán en la actividad 5.

ACTIVIDAD 4: OBTENCIÓN DE LA MATRIZ DE TRÁFICO CON PYTHON

Esta actividad consiste en la realización del script de python para determinar la matriz de tráfico. Se va a proceder a incluir capturas del script y explicar las sentencias escritas en este.



```

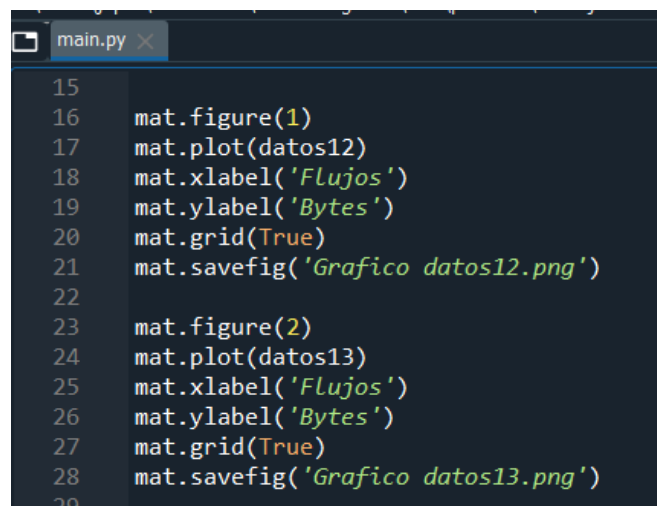
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  datos12 = np.genfromtxt('traficolanf1_2.csv', delimiter=",", usecols=12)
5  datos13 = np.genfromtxt('traficolanf1_3.csv', delimiter=",", usecols=12)
6  datos21 = np.genfromtxt('traficolanf2_1.csv', delimiter=",", usecols=12)
7  datos23 = np.genfromtxt('traficolanf2_3.csv', delimiter=",", usecols=12)
8  datos31 = np.genfromtxt('traficolanf3_1.csv', delimiter=",", usecols=12)
9  datos32 = np.genfromtxt('traficolanf3_2.csv', delimiter=",", usecols=12)
10
11 datos12 = datos12[8:17]
12 datos21 = datos21[8:15]
13 datos23 = datos23[9:18]
14 datos32 = datos32[9:21]

```

Fig. 17. Script Python en el que se usan las librerías *numpy* y *pyplot* (posteriormente), para delimitar el uso de datos pertenecientes a la columna correspondiente al tráfico recibido.

En primer lugar se abren todos los archivos csv obtenidos, solo quedándonos con la columna 12, ya que analizando los archivos y evaluando los datos en cada columna, se determina que es en esta columna donde se incluye el flujo de tráfico en bytes medido.

Seguidamente se eliminan las filas en las que el tamaño de paquete no corresponde con 1500 bytes, porque no se trata del tráfico iperf transmitido sino del generado por otro tipo de protocolos como icmp (al realizar pings entre PCs).



```

15
16  mat.figure(1)
17  mat.plot(datos12)
18  mat.xlabel('Flujos')
19  mat.ylabel('Bytes')
20  mat.grid(True)
21  mat.savefig('Grafico datos12.png')
22
23  mat.figure(2)
24  mat.plot(datos13)
25  mat.xlabel('Flujos')
26  mat.ylabel('Bytes')
27  mat.grid(True)
28  mat.savefig('Grafico datos13.png')
29

```

Fig. 18. Script Python (continuación) en el que se representa mediante *pyplot* el flujo de datos en bytes para cada dupla de LANs.

Los gráficos se muestran a continuación.

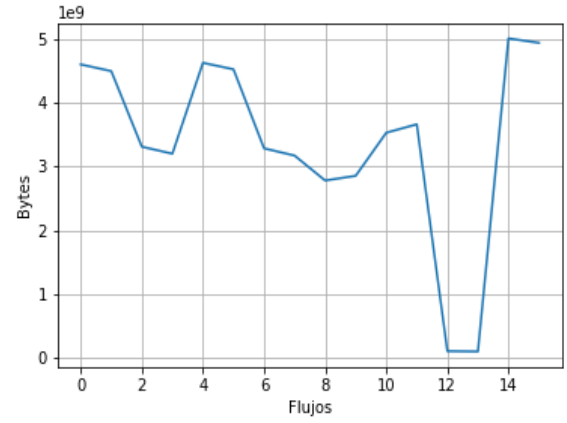
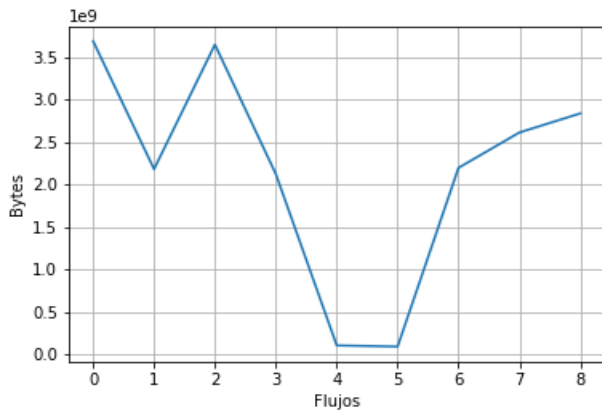


Fig. 19. Flujo de bytes generado para la dupla LAN1,LAN2 (izquierda) y LAN1,LAN3 (derecha).

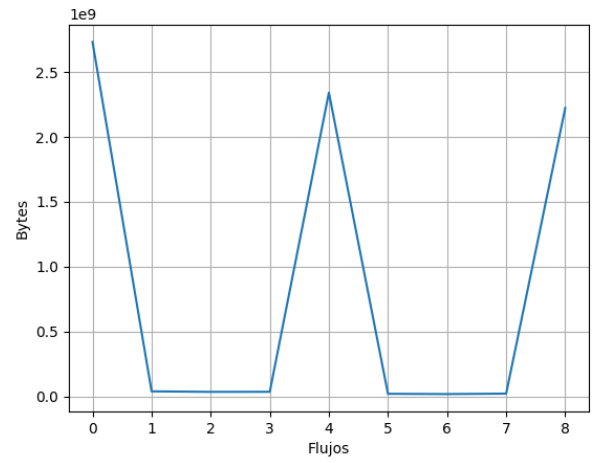
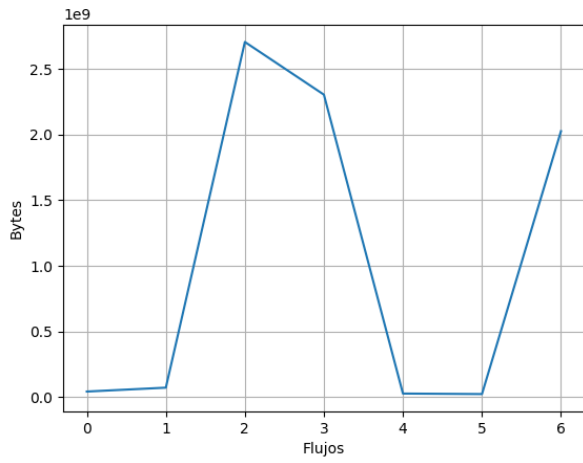


Fig. 20. Flujo de bytes generado para la dupla LAN2,LAN1 (izquierda) y LAN2,LAN3 (derecha).

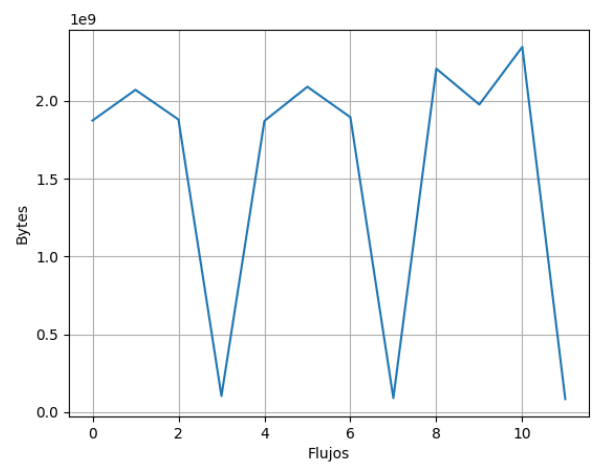
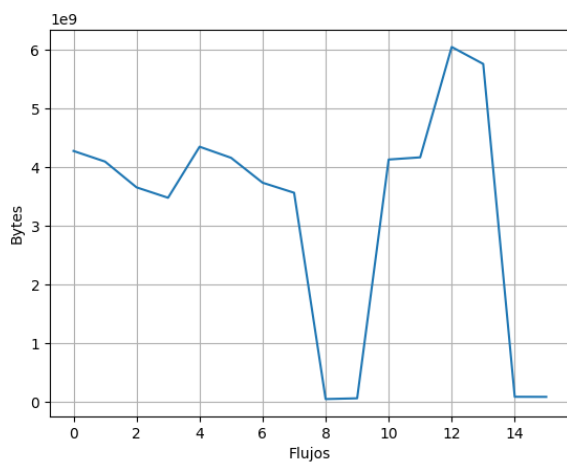


Fig. 21. Flujo de bytes generado para la dupla LAN3,LAN1 (izquierda) y LAN3,LAN2 (derecha).

```

Matriz_media = [[0,0,0],
                [0,0,0],
                [0,0,0]]

Matriz_99 =     [[0,0,0],
                [0,0,0],
                [0,0,0]]

Media12 = (sum(datos12)/len(datos12))*2
Media13 = (sum(datos13)/len(datos13))*2
Media21 = (sum(datos21)/len(datos21))*2
Media23 = (sum(datos23)/len(datos23))*2
Media31 = (sum(datos31)/len(datos31))*2
Media32 = (sum(datos32)/len(datos32))*2

P99_12 = np.percentile(datos12,99)
P99_13 = np.percentile(datos13,99)
P99_21 = np.percentile(datos21,99)
P99_23 = np.percentile(datos23,99)
P99_31 = np.percentile(datos31,99)
P99_32 = np.percentile(datos32,99)

Matriz_media[0][1]=Media12
Matriz_media[0][2]=Media13
Matriz_media[1][0]=Media21
Matriz_media[1][2]=Media23
Matriz_media[2][0]=Media31
Matriz_media[2][1]=Media32

Matriz_99[0][1]=P99_12
Matriz_99[0][2]=P99_13
Matriz_99[1][0]=P99_21
Matriz_99[1][2]=P99_23
Matriz_99[2][0]=P99_31
Matriz_99[2][1]=P99_32

```

MATRICES DE TRÁFICO ORIGINAL(BYTES)

MATRIZ MEDIA

```

[0,          4324919445, 6779747473]
[2056597135, 0,          1660316423]
[6464277400, 3082383861, 0          ]

```

MATRIZ PERCENTIL 99

```

[0,          3682339080, 5002332854]
[2682399210, 0,          2700369360]
[6006501805, 2331661579, 0          ]

```

Fig. 22. Obtención de la matriz media y percentil 99 de tráfico en bytes en Python. Para ello se utilizan las funciones de media y cálculo de percentiles de la librería *numpy*.

ACTIVIDAD 5: OPTIMIZACIÓN DE LA RED DESCRITA CON LA MATRIZ DE TRÁFICO OBTENIDA ANTERIORMENTE

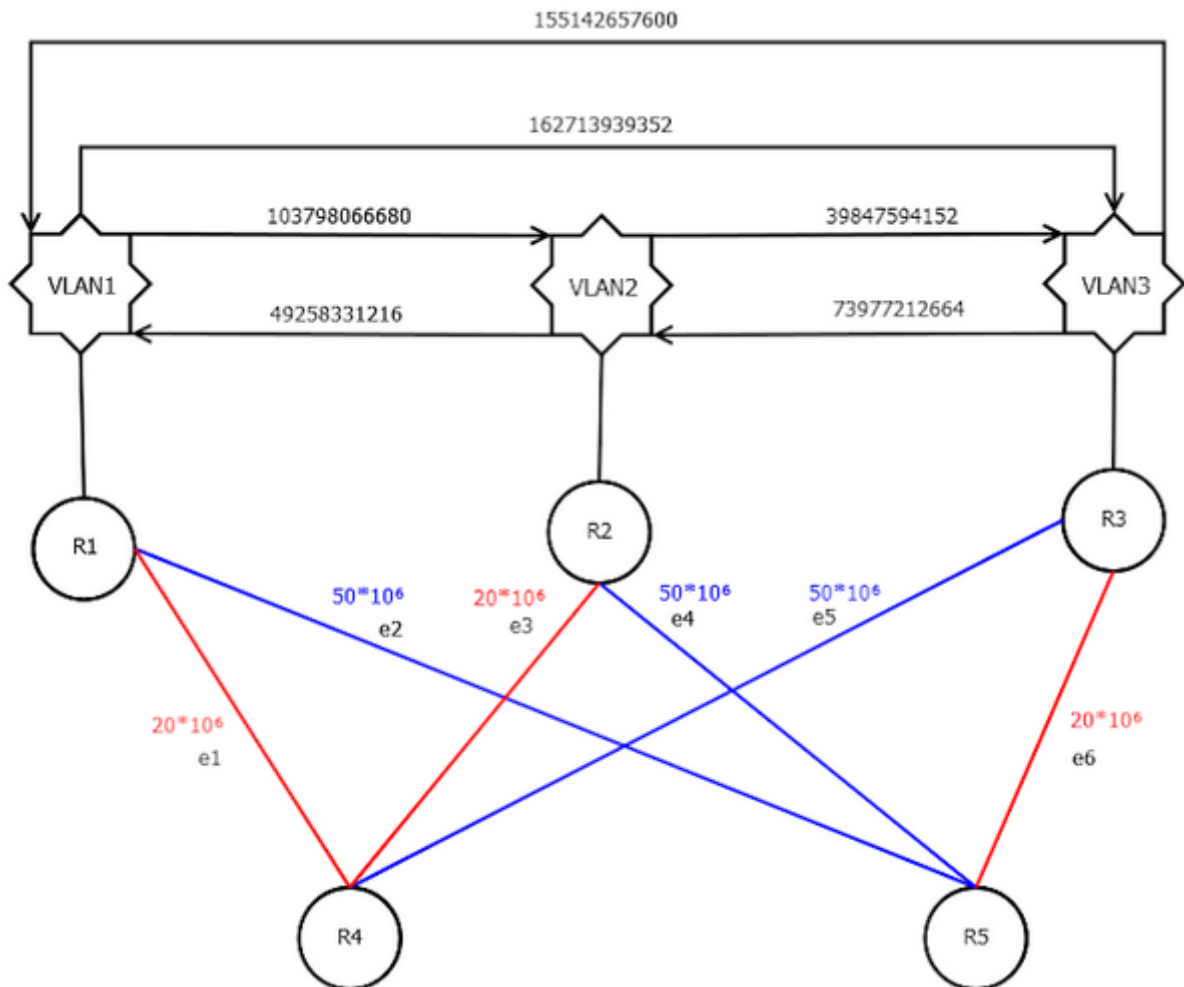


Fig. 23. Topología de red de optimización, donde R1, R2 y R4 están situados en Europa y R3 y R5 situados en EEUU. Es por ello que los enlaces e1, e3 y e6 son intracontinentales y e2,e4 y e5 son intercontinentales.

Con los siguientes datos

Característica de los enlaces	Intracontinental	Intercontinental
Capacidad	20 Mbps	50 Mbps
CAPEX	20.000 €	100.000 €
OPEX	1.000 € * 5 años	10.000 € * 5 años

H =		0	103798066680	162713939352	
		49358331216	0	39847594152	
		155142657600	55959877896	0	

Fig. 24. La matriz de tráfico requerida para este problema proviene de pasar a bits los datos y de su multiplicación por un factor de 3 (se esperaba que el tráfico se triplicará en 5 años). **Donde las filas determinan la LAN origen y las columnas la LAN destino.**

Para lo cual se consideran, según (23) los siguientes caminos posibles

x11 = R1, R4, R2
 x12 = R1, R5, R2
 x13 = R1, R4, R3, R5, R2
 x14 = R1, R5, R3, R4, R2
 x15 = R1, R4, R5, R2
 x16 = R1, R5, R4, R2

x21 = R1, R5, R3
 x22 = R1, R4, R3
 x23 = R1, R5, R2, R4, R3
 x24 = R1, R4, R2, R5, R3
 x25 = R1, R4, R5, R3
 x26 = R1, R5, R4, R3

x31 = R2, R4, R1
 x32 = R2, R5, R1
 x33 = R2, R4, R3, R5, R1
 x34 = R2, R5, R3, R4, R1
 x35 = R2, R4, R5, R1
 x36 = R2, R5, R4, R1

Fig. 25. (De izquierda a derecha) Flujo de camino para una H: **H1**: LAN1 → LAN2 / **H2**: LAN1 → LAN3 / **H3**: LAN2 → LAN1.

x41 = R2, R5, R3
 x42 = R2, R4, R3
 x43 = R2, R5, R1, R4, R3
 x44 = R2, R4, R1, R5, R3
 x45 = R2, R4, R5, R3
 x46 = R2, R5, R4, R3

x51 = R3, R4, R1
 x52 = R3, R5, R1
 x53 = R3, R4, R2, R5, R1
 x54 = R3, R5, R2, R4, R1
 x55 = R3, R5, R4, R1
 x56 = R3, R4, R5, R1

x61 = R3, R4, R2
 x62 = R3, R5, R2
 x63 = R3, R4, R1, R5, R2
 x64 = R3, R5, R1, R4, R2
 x65 = R3, R5, R4, R2
 x66 = R3, R4, R5, R2

Fig. 26. (De izquierda a derecha) Flujo de camino para una H: **H4**: LAN2 → LAN3 / **H5**: LAN3 → LAN1 / **H6**: LAN3 → LAN2.

El problema se plantea como sigue

Sea una función de coste

$$F = \sum_{e \in \{intra\}} (\xi_{intra} + K_{intra}) \cdot y_e + \sum_{e \in \{inter\}} (\xi_{inter} + K_{inter}) \cdot y_e = \sum_e (\xi_e + K_e) \cdot y_e$$

Sujeto a (FCD)

$$\sum_p x_{dp} = h_d$$

$$x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} = H1$$

$$x_{21} + x_{22} + x_{23} + x_{24} + x_{25} + x_{26} = H2$$

$$x_{31} + x_{32} + x_{33} + x_{34} + x_{35} + x_{36} = H3$$

$$x_{41} + x_{42} + x_{43} + x_{44} + x_{45} + x_{46} = H4$$

$$x_{51} + x_{52} + x_{53} + x_{54} + x_{55} + x_{56} = H5$$

$$x_{61} + x_{62} + x_{63} + x_{64} + x_{65} + x_{66} = H6$$

$$\sum_d x_{de} \leq C_e y_e$$

***Ver (23)**

e1

$$x_{11} + x_{13} + x_{15} + x_{22} + x_{24} + x_{25} + x_{31} + x_{34} + x_{36} + x_{43} + x_{44} + x_{51} + x_{54} + x_{55} + x_{63} + x_{64} \leq 20 \cdot 10^6 y_1, \text{intra}$$

e2

$$x_{12} + x_{14} + x_{16} + x_{21} + x_{23} + x_{26} + x_{32} + x_{33} + x_{35} + x_{43} + x_{44} + x_{52} + x_{53} + x_{56} + x_{63} + x_{64} \leq 50 \cdot 10^6 y_2, \text{inter}$$

e3

$$x_{11} + x_{14} + x_{16} + x_{23} + x_{24} + x_{31} + x_{33} + x_{35} + x_{42} + x_{44} + x_{45} + x_{53} + x_{54} + x_{61} + x_{64} + x_{65} \leq 20 \cdot 10^6 y_3, \text{intra}$$

e4

$$x_{12} + x_{13} + x_{15} + x_{23} + x_{24} + x_{32} + x_{34} + x_{36} + x_{41} + x_{43} + x_{46} + x_{53} + x_{54} + x_{62} + x_{63} + x_{66} \leq 50 \cdot 10^6 y_4, \text{inter}$$

e5

$$x_{13} + x_{14} + x_{22} + x_{23} + x_{26} + x_{33} + x_{34} + x_{42} + x_{43} + x_{46} + x_{51} + x_{53} + x_{56} + x_{61} + x_{63} + x_{66} \leq 50 \cdot 10^6 y_5, \text{inter}$$

e6

$$x_{13} + x_{14} + x_{21} + x_{24} + x_{25} + x_{33} + x_{34} + x_{41} + x_{44} + x_{45} + x_{52} + x_{54} + x_{55} + x_{62} + x_{64} + x_{65} \leq 20 \cdot 10^6 y_6, \text{intra}$$

Constantes (Entradas):

Característica	Valor
hd	Demanda d
ξ	OPEX e
K	CAPEX e

Variables (Salidas)

Característica	Valor
x_{dp}	Flujo de tráfico por camino
y_e	Módulos en e

El problema se resuelve mediante *glpk* en Octave

[illegible]

Fig. 27. Script de resolución en Octave.

Obteniendo los siguientes resultados

A	double	12x42	[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]	FMIN	
B	double	1x12	[1.0380e+11, 1.6271e+11, 4.9358e+10, 3.9848e+10, 1.5514e+11, 7.3977e+10, 0, 0, 0, 0]		1
C	double	42x1	[0; 0; 0; 0; 0; 0; 0; 0; 0; 0]	1	1.6592e+09
CTYPE	char	1x12	SSSSSSUUUUUU		
FMIN	double	1x1	1.6592e+09		
LB	double	1x42	0:0:0	XOPT	
UB	double	1x42	[Inf, Inf, Inf, Inf, Inf, Inf, Inf, Inf, Inf, Inf]		1
VARTYPE	char	1x42	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCIIIIII	37	13827
XOPT	double	42x1	[0; 0; 0; 0; 1.0380e+11; 0; 0; 4.9405e+10; 0; 9.5107e+09]	38	1479.5
ans	double	1x42	[0, 0, 0, 0, 1.0380e+11, 0, 0, 4.9405e+10, 0, 9.5107e+09]	39	10225
				40	2266.2
				41	2611.4
				42	4174.4

Fig. 28. (Izquierda) Valores tenidos en cuenta en tiempo de ejecución del problema de optimización y (derecha) sección de resultados que muestra el número mínimo de cables por enlace para satisfacer la demanda. **Tener en cuenta que el uso de variables enteras para la resolución del problema ha causado problemas a la hora de ejecutar el script, ocasionando bloqueos indefinidos, por lo que se ha tenido que realizar con variables continuas, tras lo cual se aproximaron a su valor unitario más próximo.**

Enlace (<i>e</i>)	Nº de cables (<i>y</i>)
<i>e1</i> (INTRA)	13.827
<i>e2</i> (INTER)	1.480
<i>e3</i> (INTRA)	10.225
<i>e4</i> (INTER)	2.267
<i>e5</i> (INTER)	2.612
<i>e6</i> (INTRA)	4.715