UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Laboratorio 2: Capa de red y transporte

Integrantes:	Alex Muñoz
	Alberto Rodríguez
Curso:	Sistemas de Comunicación
Sección	0-L-1
Profesor:	Carlos González
Ayudante:	Catalina Morales

28 de Agosto de 2020

Tabla de contenidos

1. Introducción

Dentro de la informática y la telecomunicación, existen protocolos de comunicaciones, estas consisten como un conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre sistemas, en palabras más simples, se puede definir como el "idioma" en el se comunican computadores dentro de una misma red. Algunos de ellos son el modelo OSI y el modelo TCP/IP, los cuales están compuestos por capas, cada una con funcionalidades específicas. este laboratorio se centra en dos de estas, la capa de red y la capa de transporte

La capa de red es la encargada de definir, resolver los problemas de enrutamiento y todo lo relacionado con el envío de paquetes entre redes. Los paquetes de la red viajan a través de Internet a diferentes host (Stallings, 2004). La forma mantener a todos unidos a internet, es mediante el protocolo de la capa de red, IP (Protocolo de Internet).

La capa que esta un nivel mas arriba que la capa de red en el modelo, es la capa de transporte, la cual proporciona servicios a los usuarios del servicio de transporte. Es la encargada de efectuar el transporte de datos del emisor al receptor, independiente de las rutas que tenga que usar el paquete. (Stallings, 2004)

Los objetivos principales es poder aprender como funciona el envió de datos y el funcionamiento de distintos protocolos a través de la capa de red y la de transporte. Explicado esto mas detalladamente, uno es poder comprender cual es la ruta que realizan paquetes enviados desde nuestra red local hasta cualquier otra dirección del mundo y cuanto demora en realizarlo. El otro objetivo es analizar como funciona el protocolo TCP (protocolo de la capa de transporte), cuando se ingresa a distintas paginas web y que es lo que estas devuelve a nuestra red.

Para poder desarrollar todo lo anterior a cabo, se utilizan diversas herramientas de análisis de redes. Estas son:

- nmap: Se utiliza para averiguar la topología de la red (es decir, que computadores están activos en la red local, que puertos están abierto y que servicios hay disponibles)
- Wireshark: Un analizador de protocolos que permite realizar capturas del tráfico de red. En este caso, se usará para ver protocolos de red y el protocolo de transporte TCP.

2. Marco teórico

2.1. Protocolo de Internet versión 4 (IPv4)

El protocolo IPv4 vive dentro de la capa de red, encargada del ruteo, control de flujo y solución de errores al transmitir datagramas. Para solventar el problema de ruteo se necesita una dirección única y global, esto para evitar problemas si existen dos direcciones iguales. IPv4 aporta una cabecera en el paquete, la cual aporta datos de control y utilidad como lo pueden ser: Versión del protocolo IP, protocolo de la capa posterior, el largo total, un identificador, entre otros. Además de las direcciones IP de la fuente y de destino, las cuales se representan utilizando 32 bits (o sea, 4.294.967.296 posibles combinaciones) como lo muestra la figura 1.

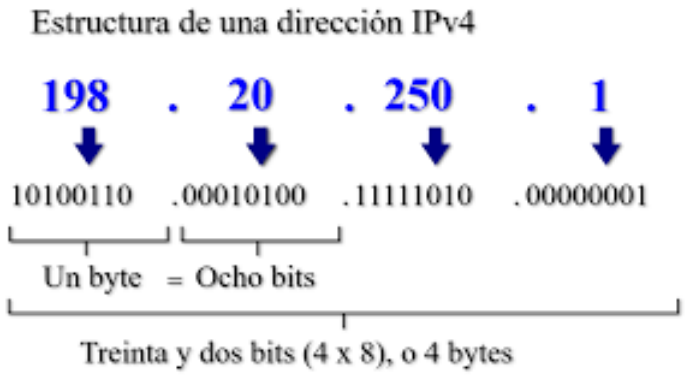


Figura 1: Estructura de dirección IP en IPv4.

Sin embargo, 32 bits no ha sido suficiente y se ha optado por utilizar subredes con un sistema de IP pública e IP privada. Debido a esto es que cuando se averigua la IP del computador y se compara con la de otro por lo general estas son muy similares (generalmente comenzando por 192.168.1).

2.2. Protocolo TCP

TCP o Protocolo de Control de Transmisión, es un protocolo de internet encargado de informar del destino de los datos permitiendo la creación de conexiones seguras. Es unos de los protocolos fundamentales de internet, es una forma segura de intercambio de datos al

requerir la autorización entre cliente y servidor (o emisor y receptor), antes de producirse la transferencia.

TCP tiene un funcionamiento sencillo que consta de tres fases, La primera establece la conexión con la autorización de ambas partes, para después, iniciarse la transferencia de información.

En el protocolo TCP los datos se entregan en el mismo orden en el que se enviaron. Para ello, divide la información en diferentes paquetes que se envían por la ruta más rápida hacia su destino. Así, con una separación en capas, se identifica la procedencia del tráfico es más fácil y evitar la saturación de la red. Además, sirve de capa intermedia entre una aplicación y el protocolo IP, supliendo las carencias de seguridad del protocolo de red (consulta aquí cuál es tu dirección IP).

TCP sirve también como mecanismo que permite diferenciar las aplicaciones, ya sean emisoras o receptoras, dentro de una misma máquina. Para ello, recurre al concepto de puerto. A pesar de tener ya más de cuatro décadas, sigue empleándose en todas las comunicaciones que se producen en la red de redes: internet. Y, sin su desarrollo, el internet que conocemos actualmente sería muy diferente.

3. Desarrollo y resultados

3.1. Capa de red

3.1.1. Averiguar y analizar rango de direcciones de la red local

Comenzando por conocer la dirección IP del computador utilizado junto a su rango de direcciones de la red local utilizando el comando *ifconfig* para ver la configuración de las interfaces de red, la cual devolvió como salida la terminal 3.1

Terminal 3.1: Salida de ifconfig

```
[root@localhost]# ifconfig
enp3s0    Link encap:Ethernet  HWaddr d8:cb:8a:9b:2a:a1
          inet addr:192.168.1.217  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a9dc:e905:6a0b:7941/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500
          RX packets:128362 errors:0 dropped:757 overruns:0 frame:0
          TX packets:57708 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:152691790 (145.6 MiB)  TX bytes:11173362 (10.6 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536
          RX packets:416 errors:0 dropped:0 overruns:0 frame:0
          TX packets:416 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:141523 (138.2 KiB)  TX bytes:141523 (138.2 KiB)
```

Al realizar una operación binaria AND entre la dirección IP del computador (192.168.1.217) con la máscara dada (255.255.255.0) se obtiene que el rango de direcciones la red local corresponde a 192.168.1.1-254 (debido a que la dirección 192.168.1.255 corresponde a la dirección de broadcast), por lo tanto corresponde utilizar la herramienta nmap para analizar dicho rango y obtener los dispositivos conectados a esa red. Dicha salida se encuentra en la terminal 3.2

Terminal 3.2: Análisis del rango de direcciones de la red local

```
[root@localhost]# nmap 192.168.1.1-254
Starting Nmap 7.80 ( [https://nmap.org](https://nmap.org/) ) at 2020-08-01 20:21 -04
Nmap scan report for 192.168.1.1
Host is up (0.00053s latency).
Not shown: 991 filtered ports
PORT STATE SERVICE
80/tcp open http
113/tcp closed ident
143/tcp closed imap
199/tcp closed smux
995/tcp closed pop3s
1723/tcp closed pptp
3389/tcp closed ms-wbt-server
8080/tcp closed http-proxy
8888/tcp closed sun-answerbook

Nmap scan report for 192.168.1.29
Host is up (0.012s latency).
All 1000 scanned ports on 192.168.1.29 are closed

Nmap scan report for 192.168.1.72
Host is up (0.0045s latency).
Not shown: 995 closed ports
PORT STATE SERVICE
1080/tcp open socks
1163/tcp open sddp
3000/tcp open ppp
3001/tcp open nessus
9998/tcp open distinct32

Nmap scan report for 192.168.1.188
Host is up (0.038s latency).
Not shown: 998 closed ports
PORT STATE SERVICE
22/tcp open ssh
80/tcp open http

Nmap scan report for 192.168.1.217
Host is up (0.000053s latency).
All 1000 scanned ports on 192.168.1.217 are closed

Nmap done: 255 IP addresses (5 hosts up) scanned in 19.55 seconds
```

A partir del análisis de la red local, se obtiene que están conectados 3 hosts aparte del computador donde se realizarán las pruebas (192.168.1.217, con conexión por cable) y el router de la red (192.168.1.1). Los otros 3 host corresponden a dispositivos móviles conectados a través de Wifi.

Al comenzar a analizar los paquetes utilizando la herramienta Wireshark es posible

36	7.266413565	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.6? Tell 192.168.1.217
37	7.266415170	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.5? Tell 192.168.1.217
38	7.266416710	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.4? Tell 192.168.1.217
39	7.266418346	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.3? Tell 192.168.1.217
40	7.266420315	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.2? Tell 192.168.1.217
41	8.279738496	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.2? Tell 192.168.1.217
42	8.279745318	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.3? Tell 192.168.1.217
43	8.279747548	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.4? Tell 192.168.1.217
44	8.279755974	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.5? Tell 192.168.1.217
45	8.279767832	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.6? Tell 192.168.1.217
46	8.279770252	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.7? Tell 192.168.1.217
47	8.279772523	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.8? Tell 192.168.1.217
48	8.279775069	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.9? Tell 192.168.1.217
49	8.279777398	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.10? Tell 192.168.1.217
50	8.279779692	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.11? Tell 192.168.1.217
51	8.279782101	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.23? Tell 192.168.1.217
52	8.279784604	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.22? Tell 192.168.1.217
53	8.279787021	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.21? Tell 192.168.1.217
54	8.279790252	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.20? Tell 192.168.1.217
55	8.279793684	Micro-St_9b:2a:a1	Broadcast	ARP	42 Who has 192.168.1.19? Tell 192.168.1.217

Figura 2: Paquetes ARP detectados por Wireshark al utilizar nmap

observar que esta herramienta funciona, en general, de la siguiente manera: Primero enviando paquetes ARP a la dirección de broadcast para comprobar los hosts disponibles (figura 2) para posteriormente analizar los puertos de cada uno de los hosts encontrados utilizando paquetes TCP (figura 3).

3.1.2. Ping a distintas direcciones

Posteriormente, se utiliza la herramienta ping para extraer datos de un dirección de la red local (se escoge 192.168.1.29) y tres sitios web (uno nacional y dos internacionales), de los cuales se escogen: biobiochile.cl, gov.za (página del gobierno sudafricano) y gosuslugi.ru (página web rusa), posteriormente se utilizó el parámetro -t en la herramienta ping para modificar el valor del TTL (Time To Live) de cada paquete y averiguar cuál es el mínimo para cada una de las direcciones escogidas. Para cada una de las direcciones se dejó que ping transmitiera 10 veces, lo cual entrega el resultado del cuadro 1.

La opción -s en ping permite modificar el tamaño del paquete que se envía a la dirección especificada, como se puede ver en la figura 4 al realizar el comando *ping gov.za -s 1000* provoca que ping envie paquetes ICMP de largo 1042, mientras que *ping gov.za* provoca que se envíen paquetes de largo 98.

8425	24.002257063	192.168.1.217	192.168.1.1	TCP	74 40972 → 33 [SYN] Seq=0 Win=64
8426	24.002345030	192.168.1.217	192.168.1.1	TCP	74 37760 → 1719 [SYN] Seq=0 Win=
8427	24.004065314	192.168.1.217	192.168.1.1	TCP	74 36028 → 259 [SYN] Seq=0 Win=6
8428	24.004188873	192.168.1.217	192.168.1.1	TCP	74 37720 → 31337 [SYN] Seq=0 Win
8429	24.004257977	192.168.1.217	192.168.1.1	TCP	74 55646 → 6547 [SYN] Seq=0 Win=
8430	24.013804296	192.168.1.217	192.168.1.1	TCP	74 44658 → 82 [SYN] Seq=0 Win=64
8431	24.013994542	192.168.1.217	192.168.1.1	TCP	74 44896 → 1147 [SYN] Seq=0 Win=
8432	24.018861450	192.168.1.217	192.168.1.1	TCP	74 50478 → 3690 [SYN] Seq=0 Win=
8433	24.018959164	192.168.1.217	192.168.1.1	TCP	74 33000 → 41511 [SYN] Seq=0 Win
8434	24.019045673	192.168.1.217	192.168.1.1	TCP	74 34806 → 6969 [SYN] Seq=0 Win=
8435	24.019117136	192.168.1.217	192.168.1.1	TCP	74 32984 → 5102 [SYN] Seq=0 Win=
8436	24.021937885	192.168.1.217	192.168.1.1	TCP	74 49432 → 4129 [SYN] Seq=0 Win=
8437	24.022036164	192.168.1.217	192.168.1.1	TCP	74 55578 → 1175 [SYN] Seq=0 Win=
8438	24.024283136	192.168.1.217	192.168.1.1	TCP	74 36122 → 2021 [SYN] Seq=0 Win=
8439	24.029048533	192.168.1.217	192.168.1.1	TCP	74 55456 → 2009 [SYN] Seq=0 Win=
8440	24.032048951	192.168.1.217	192.168.1.1	TCP	74 35120 → 636 [SYN] Seq=0 Win=6
8441	24.032146315	192.168.1.217	192.168.1.1	TCP	74 36794 → 10617 [SYN] Seq=0 Win

Figura 3: Paquetes TCP detectados por Wireshark al utilizar nmap

583	37.466126648	192.168.1.217	163.195.1.225	ICMP	1042 Echo (ping) request
592	37.815764800	163.195.1.225	192.168.1.217	ICMP	1042 Echo (ping) reply
599	38.467566897	192.168.1.217	163.195.1.225	ICMP	1042 Echo (ping) request
606	38.816625438	163.195.1.225	192.168.1.217	ICMP	1042 Echo (ping) reply
615	39.468466426	192.168.1.217	163.195.1.225	ICMP	1042 Echo (ping) request
625	39.818050024	163.195.1.225	192.168.1.217	ICMP	1042 Echo (ping) reply
641	40.469443043	192.168.1.217	163.195.1.225	ICMP	1042 Echo (ping) request
650	40.818808057	163.195.1.225	192.168.1.217	ICMP	1042 Echo (ping) reply
699	44.036257215	192.168.1.217	163.195.1.225	ICMP	98 Echo (ping) request
707	44.385790309	163.195.1.225	192.168.1.217	ICMP	98 Echo (ping) reply
716	45.036090070	192.168.1.217	163.195.1.225	ICMP	98 Echo (ping) request
723	45.384603811	163.195.1.225	192.168.1.217	ICMP	98 Echo (ping) reply
730	46.036410623	192.168.1.217	163.195.1.225	ICMP	98 Echo (ping) request
738	46.385458994	163.195.1.225	192.168.1.217	ICMP	98 Echo (ping) reply
745	47.036281702	192.168.1.217	163.195.1.225	ICMP	98 Echo (ping) request
752	47.385542275	163.195.1.225	192.168.1.217	ICMP	98 Echo (ping) reply
763	48.036091125	192.168.1.217	163.195.1.225	ICMP	98 Echo (ping) request

Figura 4: Wireshark al realizar ping a gov.za con la opción -s y sin ella

Cuadro 1: Resumen ping

	192.168.1.29	biobiochile.cl	gov.za	gosuslugi.ru
IP	192.168.1.29	190.153.242.131	163.195.1.225	109.207.1.97
TTL	64	56	46	232
TTL mínimo	1	9	22	17
Tiempo promedio	89.879ms	2.918ms	349.029ms	253.088ms
Tamaño respuesta	64 bytes	64 bytes	64 bytes	64 bytes

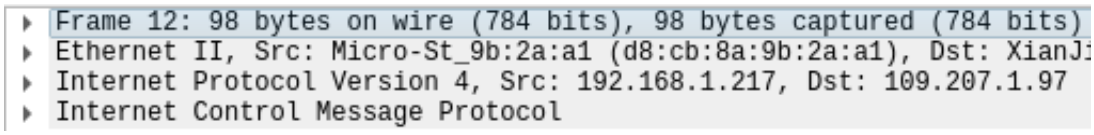


Figura 5: Contenido de paquete ICMP visto en Wireshark

En general, el comando ping se basa puramente en el protocolo ICMP, el cual vive dentro de la capa de red. Sin embargo, en caso de que se utilice ping en conjunto con un dominio (por ejemplo: *ping google.com*), primero es necesario utilizar el protocolo DNS (Domain Name System) de la capa de aplicación para traducir dicho dominio a una dirección IP pública, luego se envian paquetes ICMP encapsulados luego de la cabecera IP con las direcciones fuente y de destino, tal como lo muestra Wireshark en la figura 5. Posteriormente enviados los paquetes, el servidor debe de responder dichos paquetes ICMP (a menos que dicho servidor se encuentre configurado para ignorar paquetes ICMP) cómo lo muestra la figura 4.

Con respecto a los paquetes DNS que se realizan al inicio de ping, cabe destacar que dicho comando envía dos paquetes DNS al servidor DNS 131.221.32.2, donde un paquete posee una consulta estándar A (encargada de obtener direcciones IPv4) y el otro es una consulta estándar AAAA (este se encarga de obtener direcciones IPv6). El servidor responde cada consulta por separado (figura 14), en este caso el paquete respuesta a la consulta estándar A posee en su interior la dirección IPv4 del sitio solicitado (figura 7), mientras que para la consulta estándar AAAA no posee respuesta al interior de su paquete, pues dicho dominio no soporta IPv6 (figura 8).

A partir de esto se puede decir que el comando ping llega hasta la capa de red si se utiliza una dirección IP de manera directa (Cómo puede ser *ping 192.168.1.1*), pero

```
192.168.1.217    131.221.32.2     DNS      72 Standard query 0xad97 A gosuslugi.ru
192.168.1.217    131.221.32.2     DNS      72 Standard query 0x0b95 AAAA gosuslugi.ru
131.221.32.2     192.168.1.217    DNS     132 Standard query response 0x0b95 AAAA gosuslugi.ru SOA ns1.gosusl
131.221.32.2     192.168.1.217    DNS     234 Standard query response 0xad97 A gosuslugi.ru A 109.207.1.97 NS
```

Figura 6: Paquetes con consultas DNS

```
▾ Domain Name System (response)
    Transaction ID: 0xad97
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 4
    Additional RRs: 4
  ▾ Queries
    ▸ gosuslugi.ru: type A, class IN
  ▾ Answers
    ▸ gosuslugi.ru: type A, class IN, addr 109.207.1.97
  ▸ Authoritative nameservers
  ▸ Additional records
    [Request In: 8]
    [Time: 0.001699386 seconds]
```

Figura 7: Respuesta de servidor DNS a consulta A

```
▾ Domain Name System (response)
    Transaction ID: 0x0b95
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 1
    Additional RRs: 0
  ▾ Queries
    ▸ gosuslugi.ru: type AAAA, class IN
  ▾ Authoritative nameservers
    ▸ gosuslugi.ru: type SOA, class IN, mname ns1.gosuslugi.ru
    [Request In: 9]
    [Time: 0.001625112 seconds]
```

Figura 8: Respuesta de servidor DNS a consulta AAAA

si se utiliza un dominio (*ping google.com*), esta llega hasta la capa de aplicación porque es necesario utilizar DNS para resolver dicho dominio a una dirección IP. Y por lo tanto, la función de la herramienta ping es comprobar la calidad y estabilidad de una conexión con otra dirección IP.

Cabe destacar que el TTL (Time To Live) corresponde un valor utilizado para que el paquete no se encuentre deambulando por internet de manera indefinida, por lo que se le asigna un número que designa la cantidad de routers por los que pasará (por cada router se le descuenta 1 al TTL) y si dicho valor llega a 0, es descartado. Por ese motivo es que en el cuadro 1 la dirección IP de la red local solamente necesita de TTL igual a uno para alcanzar a su destino, mientras que en los otros casos, se necesitan valores más altos.

El tiempo promedio es afectado principalmente por el tamaño del paquete al utilizar ping con la opción -s, por ejemplo, al intentar utilizar ping al dominio biobiochile.cl con la opción -s igual a 60 se obtiene que el tiempo de respuesta medio es de 2.784ms con desviación estándar de 0.548ms, mientras que con la opción -s igual a 65000 el tiempo de respuesta medio aumentó a 16.693ms con desviación estándar de 0.558ms (en ambos casos se dejó que el comando ping enviara 100 paquetes ICMP). Además del tamaño del paquete, también es necesario tener en cuenta la topología de la red, pues para un servidor chileno es más probable que el proveedor de internet posea conexiones más directas que para un servidor sudafricano (tal como se puede apreciar en el cuadro 1, donde biobiochile.cl posee tiempo promedio de respuesta de 2.918ms, mientras que la página del gobierno sudafricano posee 349.029ms de tiempo de respuesta medio)

3.1.3. Ruta y localización de distintas direcciones

Para una visión mas exacta de cuál es la ruta (direcciones IP) por donde viajan los paquetes se utiliza el comando *traceroute -I* hacia una cierta dirección, esta si bien puede ser escrita con su IP o con su nombre de dominio (ej: google.com), para esta prueba se usarán tres direcciones, www.baidu.com (buscador chino), gosuslugi.ru (página web rusa), gov.za (página del gobierno sudafricano). Ahora nuestra IP del computador es, 192.168.0.5.

La lista de IP junto con el tiempo que demoran los paquetes se ilustran en las terminales 3.3, 3.4 y 3.5

Terminal 3.3: Salida del comando traceroute -I baidu.com

```
[root@localhost]# traceroute -I baidu.com
traceroute to baidu.com (220.181.38.148), 64 hops max
  1   192.168.0.1  2,138ms  1,841ms  3,805ms
  2   10.139.128.1  11,481ms  15,084ms  12,211ms
  3   190.208.229.61  19,580ms  14,768ms  15,670ms
  4   200.14.204.98  17,468ms  14,522ms  14,561ms
  5   200.27.128.205  19,920ms  108,613ms  13,586ms
  6   66.110.72.133  190,653ms  204,927ms  204,488ms
  7   63.243.152.61  204,424ms  185,299ms  224,660ms
  8   66.198.154.177  306,252ms  205,008ms  306,858ms
  9   216.6.87.1  205,309ms  207,125ms  201,009ms
 10   216.6.87.29  204,955ms  182,510ms  227,868ms
 11   202.97.92.169  305,881ms  306,955ms  306,676ms
 12   202.97.59.109  409,619ms  409,782ms  409,040ms
 13   202.97.12.57  440,227ms  481,057ms  511,740ms
 14   *  202.97.34.157  478,866ms  516,784ms
 15   180.149.128.122  367,547ms  *  *
 16   36.110.246.197  411,251ms  *  *
 17   *  *  *
 18   220.181.17.90  373,158ms  461,547ms  408,481ms
 19   220.181.38.148  421,613ms  409,423ms 400,333ms
```

Terminal 3.4: Salida del comando traceroute -I gosuslugi.ru

```
[root@localhost]# traceroute -I gosuslugi.ru
traceroute to gosuslugi.ru (109.207.1.97), 64 hops max
  1   192.168.0.1  1,941ms  2,902ms  2,658ms
  2   10.139.128.1  12,569ms  80,721ms  14,599ms
  3   190.208.229.57  14,687ms  19,488ms  15,180ms
  4   190.208.9.102  15,401ms  15,960ms  13,015ms
  5   200.27.128.201  14,802ms  14,563ms  65,537ms
  6   168.143.191.220  205,017ms  203,844ms  204,703ms
  7   129.250.2.62  205,215ms  203,630ms  208,475ms
  8   *  4.68.37.221  268,300ms  114,730ms
  9   *  *  *
 10   195.122.183.218  228,966ms  275,164ms  223,142ms
 11   188.254.25.77  289,075ms  409,161ms  307,406ms
 12   109.207.0.194  368,155ms  348,166ms  307,133ms
 13   109.207.3.6  306,690ms  306,891ms  306,844ms
 14   *  *  *
 15   *  *  *
 16   109.207.1.97  318,730ms  304,702ms  309,560ms
```

Terminal 3.5: Salida del comando traceroute -I gov.za

```
[root@localhost]# traceroute -I gov.za
traceroute to gov.za (163.195.1.225), 64 hops max
  1   192.168.0.1  2,575ms  1,725ms  1,852ms
  2   10.139.128.1  15,617ms  15,032ms  15,077ms
  3   190.208.229.57  14,729ms  14,917ms  121,502ms
  4   190.208.9.102  18,698ms  13,859ms  16,158ms
  5   200.27.128.201  15,374ms  13,421ms  *
  6   168.143.191.220  196,293ms  206,029ms  203,276ms
  7   129.250.3.149  207,472ms  201,455ms  204,177ms
  8   *  *  *
  9   129.250.2.110  309,456ms  309,907ms  305,080ms
 10   129.250.2.185  415,012ms  302,273ms  307,341ms
 11   *  83.231.146.210  273,696ms  409,162ms
 12   *  5.11.10.102  394,748ms  598,084ms
 13   5.11.10.171  409,105ms  409,323ms  409,226ms
 14   *  46.17.232.5  376,602ms  409,649ms
 15   *  *  *
 16   41.164.161.106  412,383ms  408,117ms  408,743ms
 17   163.195.3.2  410,179ms  413,132ms  391,528ms
 18   163.195.1.225  423,131ms  408,009ms  411,486ms
```

Verlo solo con direcciones IP de cada router puede ser confuso acerca de cual es la localización geográfica de cada una de estas direcciones. Las siguientes figuras muestran un mapa geográfico con la ruta para llegar a la IP de destino de cada una de estas direcciones, el buscador chino baidu (figura 9), página web rusa gosuslugi (figura 10)y la página del gobierno sudafricano (figura 11.

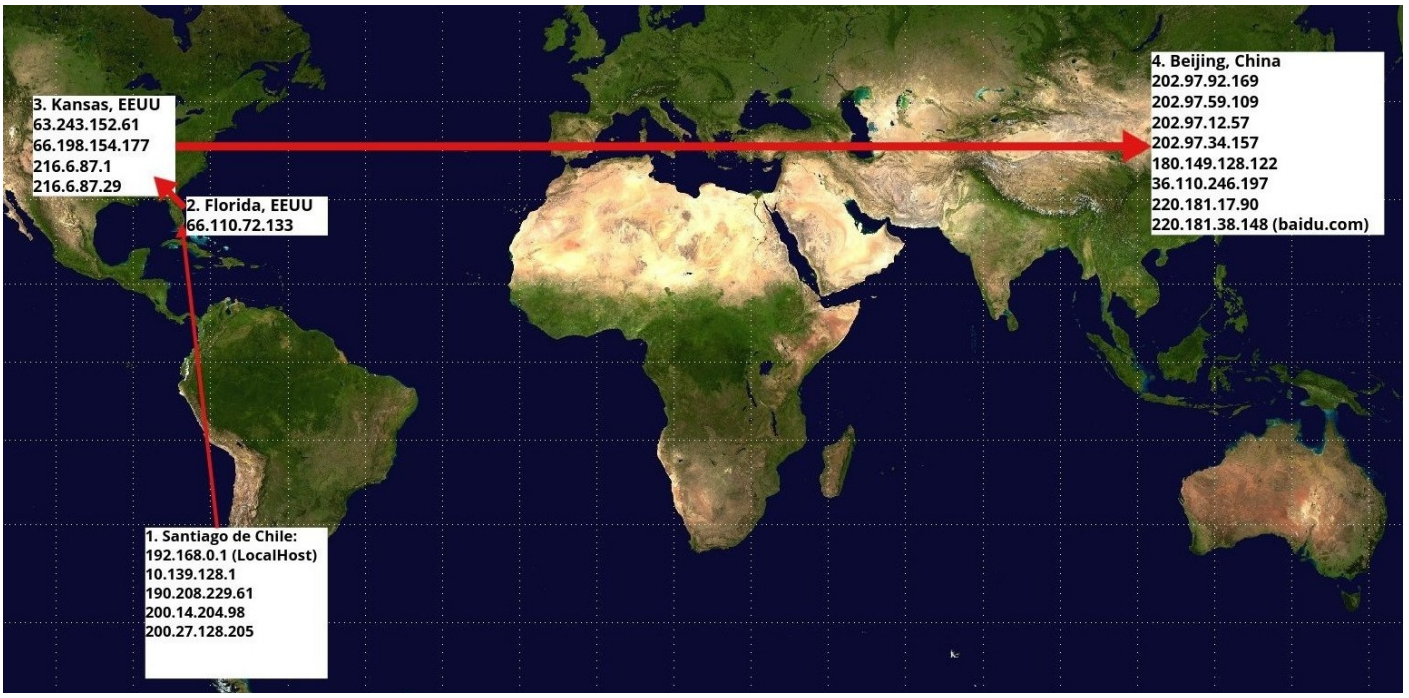


Figura 9: Mapa de la ruta para llegar a baidu.com

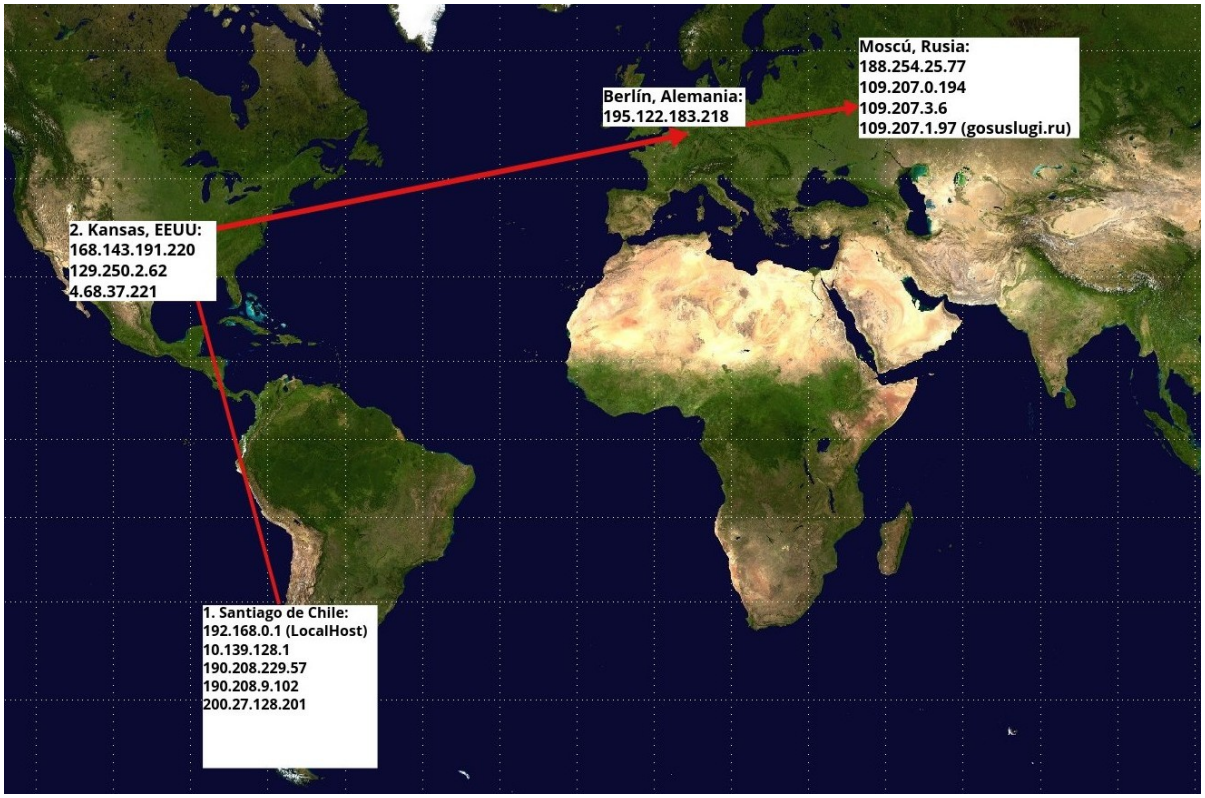


Figura 10: Mapa de la ruta para llegar a gosuslugi.ru

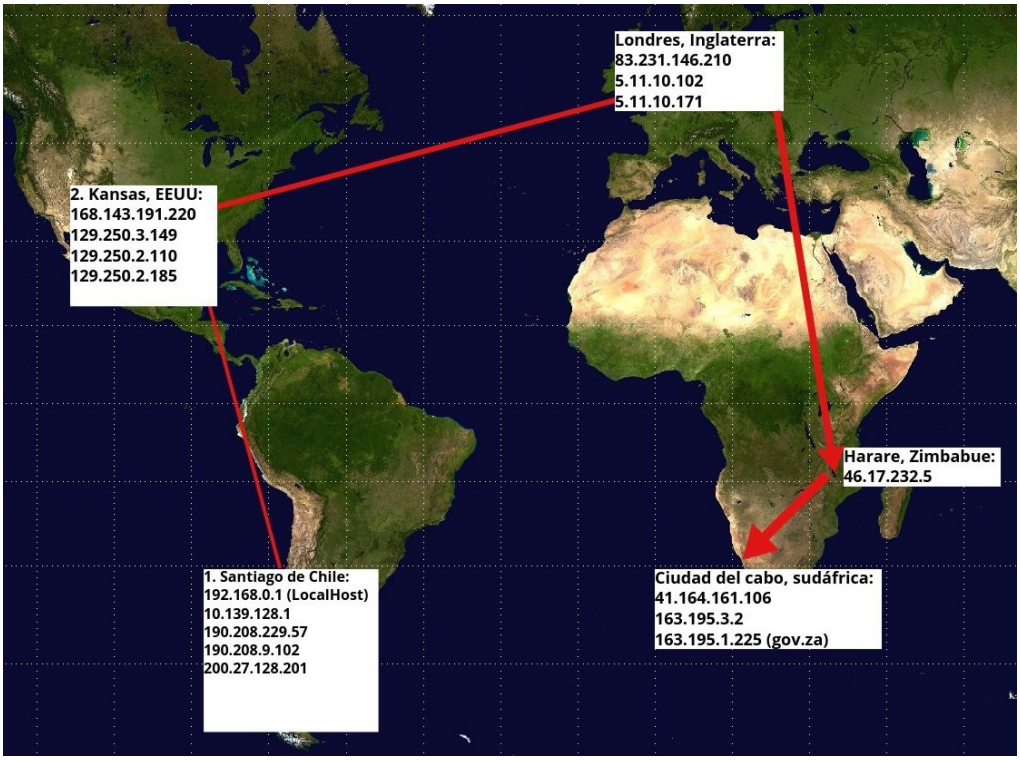


Figura 11: Mapa de la ruta para llegar a gov.za

Después de ver los mapas podemos ver el comportamiento de la red utilizando Wireshark durante el transcurso de la ejecución de los comando. (figura 12 muestra las últimas direcciones de red para llegar la página gov.za).

237.848092…	163.195.3.2	192.168.0.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
237.848326…	192.168.0.5	163.195.1.225	ICMP	66	Echo (ping) request id=0x2660, seq=49/12544, ttl=17 (no response found!)
238.253755…	163.195.3.2	192.168.0.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
238.253884…	192.168.0.5	163.195.1.225	ICMP	66	Echo (ping) request id=0x2660, seq=50/12800, ttl=17 (no response found!)
238.663076…	163.195.3.2	192.168.0.5	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
238.663422…	192.168.0.5	163.195.1.225	ICMP	66	Echo (ping) request id=0x2660, seq=51/13056, ttl=18 (reply in 20967)
239.072881…	163.195.1.225	192.168.0.5	ICMP	66	Echo (ping) reply id=0x2660, seq=51/13056, ttl=45 (request in 20966)
239.073164…	192.168.0.5	163.195.1.225	ICMP	66	Echo (ping) request id=0x2660, seq=52/13312, ttl=18 (reply in 20969)
239.482727…	163.195.1.225	192.168.0.5	ICMP	66	Echo (ping) reply id=0x2660, seq=52/13312, ttl=45 (request in 20968)
239.482984…	192.168.0.5	163.195.1.225	ICMP	66	Echo (ping) request id=0x2660, seq=53/13568, ttl=18 (reply in 20971)
239.892593…	163.195.1.225	192.168.0.5	ICMP	66	Echo (ping) reply id=0x2660, seq=53/13568, ttl=45 (request in 20970)

Figura 12: Wireshark al realizar traceroute a gov.za

Al igual que el comando ping, *traceroute* se basa puramente en el protocolo ICMP, tal como muestra el Wireshark de la figura 12, en envían paquetes ICMP encapsulados luego de la cabecera IP con la direcciones fuente y de destino. Se puede observar también que hay veces que el TTL (Time To Live) es excedido mientras esta en transito, entonces es descartado, por lo tanto, se envía de nuevo el paquete el cual ahora no tiene problemas con el TTL.

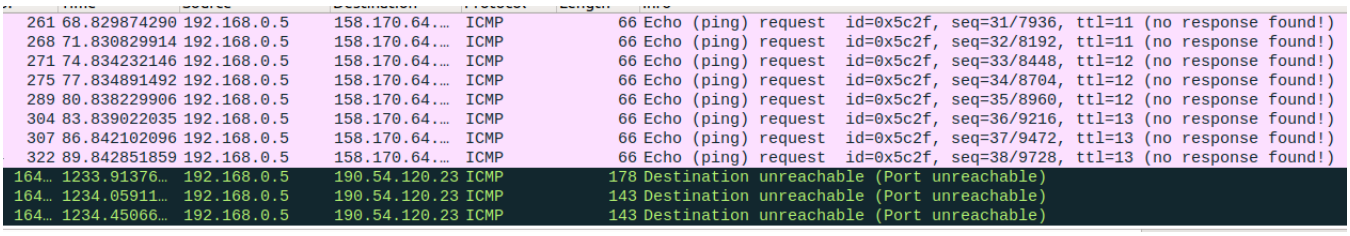
Para explicar mas específicamente lo que hace el comando *traceroute*, su objetivo es enviar paquetes a un destino, pero mientras este llega a su destino final, irá solicitando a cada uno de los enrutadores que se encuentre en el camino una respuesta al paso de estos paquetes. De esta forma obtendremos información sobre cada nodo por el que pase el paquete como su dirección IP, Nombre de dominio, si lo tiene, y latencia o tiempo de conexión entre nuestro equipo y cada uno de los nodos del camino. Este comando tiene varios flags, el utilizado en este laboratorio fue el *traceroute -I* el cual indica que se realizará en modo ICMP , por eso Wireshark muestra que se usa el protocolo ICMP.

Puede existir el caso de servidores que no respondan a este comando, porque están configurados para no responder a paquetes ICMP. Se realizará un ejemplo con la página de la Universidad de Santiago (usach.cl) la cual está configurada como se mencionó anteriormente. EL terminal 3.6 muestra como nunca llega a la dirección destino, se podría seguir esperando respuesta pero no tiene mucho sentido ya que nunca llegará.

Terminal 3.6: Salida del comando traceroute -I usach.cl

```
[root@localhost]# traceroute -I usach.cl
traceroute to usach.cl (158.170.64.150), 64 hops max
  1   192.168.0.1  2,889ms  2,523ms  1,843ms
  2   10.139.128.1  15,534ms  14,298ms  12,383ms
  3   190.208.229.65  17,451ms  19,650ms  14,789ms
  4   190.208.9.102  15,752ms  13,864ms  14,805ms
  5   190.208.9.101  14,923ms  16,006ms  18,759ms
  6   *  *  *
  7   *  *  *
  8   *  *  *
  9   *  *  *
 10   *  *  *
 11   *  *  *
 12   *  *  *
 13   *  *  *
 14   *  *  *
```

Al analizar Wireshark (figura 13), se puede observar como nunca recibe respuesta de la IP destino, hasta que dice que el destino es inalcanzable.



261	68.829874290	192.168.0.5	158.170.64.…	ICMP	66	Echo (ping) request id=0x5c2f, seq=31/7936, ttl=11 (no response found!)
268	71.830829914	192.168.0.5	158.170.64.…	ICMP	66	Echo (ping) request id=0x5c2f, seq=32/8192, ttl=11 (no response found!)
271	74.834232146	192.168.0.5	158.170.64.…	ICMP	66	Echo (ping) request id=0x5c2f, seq=33/8448, ttl=12 (no response found!)
275	77.834891492	192.168.0.5	158.170.64.…	ICMP	66	Echo (ping) request id=0x5c2f, seq=34/8704, ttl=12 (no response found!)
289	80.838229906	192.168.0.5	158.170.64.…	ICMP	66	Echo (ping) request id=0x5c2f, seq=35/8960, ttl=12 (no response found!)
304	83.839022035	192.168.0.5	158.170.64.…	ICMP	66	Echo (ping) request id=0x5c2f, seq=36/9216, ttl=13 (no response found!)
307	86.842102096	192.168.0.5	158.170.64.…	ICMP	66	Echo (ping) request id=0x5c2f, seq=37/9472, ttl=13 (no response found!)
322	89.842851859	192.168.0.5	158.170.64.…	ICMP	66	Echo (ping) request id=0x5c2f, seq=38/9728, ttl=13 (no response found!)
164…	1233.91376…	192.168.0.5	190.54.120.23	ICMP	178	Destination unreachable (Port unreachable)
164…	1234.05911…	192.168.0.5	190.54.120.23	ICMP	143	Destination unreachable (Port unreachable)
164…	1234.45066…	192.168.0.5	190.54.120.23	ICMP	143	Destination unreachable (Port unreachable)

Figura 13: Wireshark al realizar traceroute a usach.cl

3.2. Capa de Transporte

3.2.1. Funcionamiento TCP

A continuación se analizará una conexión con la página web http://www.example.com/ con la ayuda de Wireshark para estudiar los paquetes que llegan y salen debido a dicha conexión.

Source	Destination	Protocol	Length	Info
192.168.1.217	131.221.32.2	DNS	75	Standard query 0x8f2b A www.example.com
131.221.32.2	192.168.1.217	DNS	139	Standard query response 0x8f2b A www.example
192.168.1.217	93.184.216.34	TCP	74	48234 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1
93.184.216.34	192.168.1.217	TCP	74	80 → 48234 [SYN, ACK] Seq=0 Ack=1 Win=65535
192.168.1.217	93.184.216.34	TCP	66	48234 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
192.168.1.217	93.184.216.34	HTTP	488	GET / HTTP/1.1
93.184.216.34	192.168.1.217	TCP	66	80 → 48234 [ACK] Seq=1 Ack=423 Win=67072 Len
93.184.216.34	192.168.1.217	HTTP	1071	HTTP/1.1 200 OK (text/html)
192.168.1.217	93.184.216.34	TCP	66	48234 → 80 [ACK] Seq=423 Ack=1006 Win=64128
192.168.1.217	93.184.216.34	HTTP	418	GET /favicon.ico HTTP/1.1
93.184.216.34	192.168.1.217	HTTP	1079	HTTP/1.1 404 Not Found (text/html)
192.168.1.217	93.184.216.34	TCP	66	48234 → 80 [ACK] Seq=775 Ack=2019 Win=64128

Figura 14: Paquetes captados por Wireshark al ingresar a http://www.example.com/

Es posible observar en la figura 14 , al igual que en la sección 3.1.2, se inicia enviando paquetes DNS con una consulta estándar A (la que se encarga de obtener direcciones IPv4) al servidor DNS por defecto (131.221.32.2) preguntando por la dirección del dominio http://www.example.com/. El servidor responde, entregando la IP 93.184.216.34 (figura 15)

Una vez la dirección IP es conocida, se procede a realizar el three-way handshake de TCP para inicializar la conexión (esto corresponde a las tres primeras flechas en la figura 16), iniciando los números de secuencia y tamaño de ventana para tanto el cliente como el servidor (Seq y Len en la figura 16).

Posterior a el establecimiento de la conexión, el cliente procede a utilizar un método GET para obtener el contenido de la página web al servidor (las líneas marcadas en rojo de la figura 17), posteriormente el servidor responde reconociendo que le llegaron 422

```
▾ Domain Name System (response)
    Transaction ID: 0x8f2b
  ▸ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 2
    Additional RRs: 0
  ▸ Queries
  ▾ Answers
    ▸ www.example.com: type A, class IN, addr 93.184.216.34
  ▸ Authoritative nameservers
    [Request In: 1]
    [Time: 0.001569463 seconds]
```

Figura 15: Respuesta de servidor DNS a http://www.example.com/

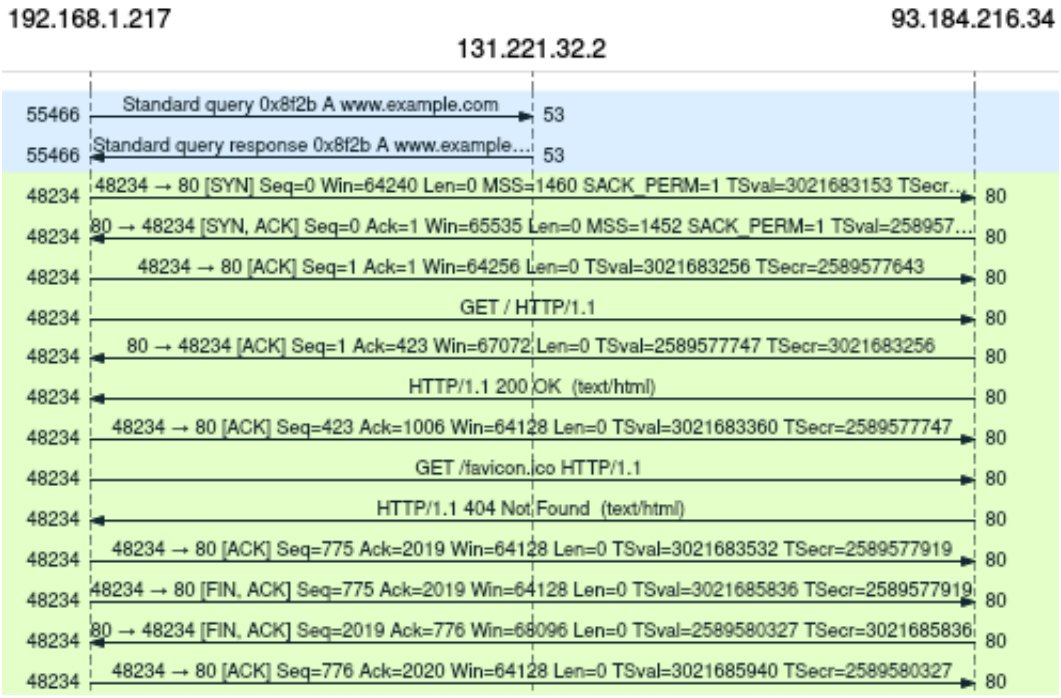


Figura 16: Flujo de la conexión a http://www.example.com/

```
GET / HTTP/1.1
Host: www.example.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=
Accept-Encoding: gzip, deflate
Accept-Language: es,es-ES;q=0.9,en;q=0.8

HTTP/1.1 200 OK
Content-Encoding: gzip
Age: 322399
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Mon, 24 Aug 2020 21:21:28 GMT
Etag: "3147526947+gzip"
Expires: Mon, 31 Aug 2020 21:21:28 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (mic/9B22)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 648

<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Ari

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
```

Figura 17: Primer GET en la conexión a http://www.example.com/

bytes de dicho GET, luego de esto, procede a enviar la respuesta con el protocolo HTTP (la cuál contiene el HTML y CSS de la página web en texto plano) a la solicitud realizada por el cliente (las líneas marcadas con azul en la figura 17).

Una vez enviado el contenido de la página web, el cliente reconoce los bytes enviados por el servidor y realiza otro método GET para obtener la imagen favicon.co (que corresponde al icono que aparece en las pestañas del navegador al ingresar a una página web), pero el servidor le responde que no existe dicha imagen (404 not found) y le envía de nuevo los contenidos de http://www.example.com/ (figura 18), luego el cliente reconoce que le llegaron los bytes y comienza el fin de la conexión por parte del cliente (las últimas tres flechas en la figura 16) haciendo uso de la bandera FIN de TCP, el servidor responde también con una bandera FIN (y reconociendo que le llegó el FIN del cliente) y finalmente, el cliente responde reconociendo el FIN del servidor, cerrando finalmente la conexión.

Cómo se pudo apreciar, los datos viajan a través de internet como texto plano,

```
GET /favicon.ico HTTP/1.1
Host: www.example.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://www.example.com/
Accept-Encoding: gzip, deflate
Accept-Language: es,es-ES;q=0.9,en;q=0.8

HTTP/1.1 404 Not Found
Content-Encoding: gzip
Accept-Ranges: bytes
Age: 404456
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Mon, 24 Aug 2020 21:21:28 GMT
Expires: Mon, 31 Aug 2020 21:21:28 GMT
Last-Modified: Thu, 20 Aug 2020 05:00:32 GMT
Server: ECS (mic/9A9B)
Vary: Accept-Encoding
X-Cache: 404-HIT
Content-Length: 648

<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, s

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
```

Figura 18: Segundo GET en la conexión a http://www.example.com/

por ello es necesario utilizar encriptación para proteger los datos que son enviados tanto por el cliente como el servidor. Por parte de los protocolos utilizados en esta comunicación, principalmente se usa DNS para obtener la dirección IP del dominio (capa de aplicación), TCP para una comunicación confiable entre el cliente y servidor (capa de transporte) y HTTP para enviar los datos de la página web (En este caso el HTML y CSS) en formato texto plano (capa de aplicación).

3.2.2. Funcionamiento FTP

Se ingresará al sitio ftp://test.rebex.net/pub/example/ (con usuario: demo y contraseña: password) para analizar la transferencia de archivos a través del protocolo FTP. Cabe destacar que los protocolos principalmente utilizados a lo largo de esta comunicación son: DNS, TCP (para transferencia de datos confiable, capa de transporte) y FTP (File Transfer Protocol, encargado de controlar y transferir archivos entre dos computadores), dichos protocolos se pueden ver en la figura 19.

Source	Destination	Protocol	Length	Info
192.168.1.217	131.221.32.2	DNS	74	Standard query 0xb8d5 A test.rebex.net
XianJizh_d0:bd:d8	Broadcast	0xfffa	60	Ethernet II
131.221.32.2	192.168.1.217	DNS	165	Standard query response 0xb8d5 A test.rebex.net A 195.
192.168.1.217	195.144.107.198	TCP	74	50912 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_P
195.144.107.198	192.168.1.217	TCP	74	21 → 50912 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1
192.168.1.217	195.144.107.198	TCP	66	50912 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=354
195.144.107.198	192.168.1.217	FTP	93	Response: 220 Microsoft FTP Service
192.168.1.217	195.144.107.198	TCP	66	50912 → 21 [ACK] Seq=1 Ack=28 Win=64256 Len=0 TSval=35
192.168.1.217	195.144.107.198	FTP	82	Request: USER anonymous
192.168.1.217	104.42.39.77	TCP	66	42878 → 443 [ACK] Seq=1 Ack=1 Win=501 Len=0 TSval=3033
192.168.1.217	34.73.232.153	TCP	66	38486 → 443 [ACK] Seq=1 Ack=1 Win=1667 Len=0 TSval=423
34.73.232.153	192.168.1.217	TCP	66	[TCP ACKed unseen segment] 443 → 38486 [ACK] Seq=1 Ack
104.42.39.77	192.168.1.217	TCP	66	[TCP ACKed unseen segment] 443 → 42878 [ACK] Seq=1 Ack
195.144.107.198	192.168.1.217	FTP	104	Response: 331 Password required for anonymous.
192.168.1.217	195.144.107.198	TCP	66	50912 → 21 [ACK] Seq=17 Ack=66 Win=64256 Len=0 TSval=3
192.168.1.217	195.144.107.198	FTP	91	Request: PASS chrome@example.com
195.144.107.198	192.168.1.217	FTP	91	Response: 530 User cannot log in.
192.168.1.217	195.144.107.198	TCP	66	50912 → 21 [ACK] Seq=42 Ack=91 Win=64256 Len=0 TSval=3
192.168.1.217	195.144.107.198	FTP	72	Request: QUIT
195.144.107.198	192.168.1.217	FTP	80	Response: 221 Goodbye.
195.144.107.198	192.168.1.217	TCP	66	21 → 50912 [FIN, ACK] Seq=105 Ack=48 Win=66048 Len=0 T
192.168.1.217	195.144.107.198	TCP	66	50912 → 21 [FIN, ACK] Seq=48 Ack=106 Win=64256 Len=0 T
195.144.107.198	192.168.1.217	TCP	66	21 → 50912 [ACK] Seq=106 Ack=49 Win=66048 Len=0 TSval=
192.168.1.217	195.144.107.198	TCP	74	50914 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_P
XianJizh_d0:bd:d8	Broadcast	0xfffa	60	Ethernet II
195.144.107.198	192.168.1.217	TCP	74	21 → 50914 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1
192.168.1.217	195.144.107.198	TCP	66	50914 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=354
195.144.107.198	192.168.1.217	FTP	93	Response: 220 Microsoft FTP Service
192.168.1.217	195.144.107.198	TCP	66	50914 → 21 [ACK] Seq=1 Ack=28 Win=64256 Len=0 TSval=35
192.168.1.217	195.144.107.198	FTP	82	Request: USER anonymous

Figura 19: Paquetes captados por Wireshark al conectarse con el servidor FTP

Para establecer la conexión con el servidor, se comienza con el three-way handshake de TCP para establecer la conexión entre el cliente y el servidor (el flujo de la figura 20), posteriormente el servidor le avisa al cliente que está utilizando el servicio FTP de Mi-

crosoft, el cliente reconoce eso y le envía al servidor una solicitud para ingresar con el usuario anonymous, el servidor responde que anonymous necesita una contraseña, el cliente reconoce y envía la contraseña por defecto chrome@example.com, el servidor responde que no se puede ingresar, por lo tanto el cliente reconoce y solicita cerrar la conexión (QUIT), el servidor le responde con un adiós y posteriormente se vuelve a establecer exactamente la misma conexión. Posterior a estos dos intentos, el cliente vuelve a intentar establecer la conexión, pero esta vez con el usuario demo y contraseña password (figura 21), logrando ingresar correctamente al servidor FTP, posteriormente el cliente solicita el directorio actual y los archivos dentro de él y termina la conexión.

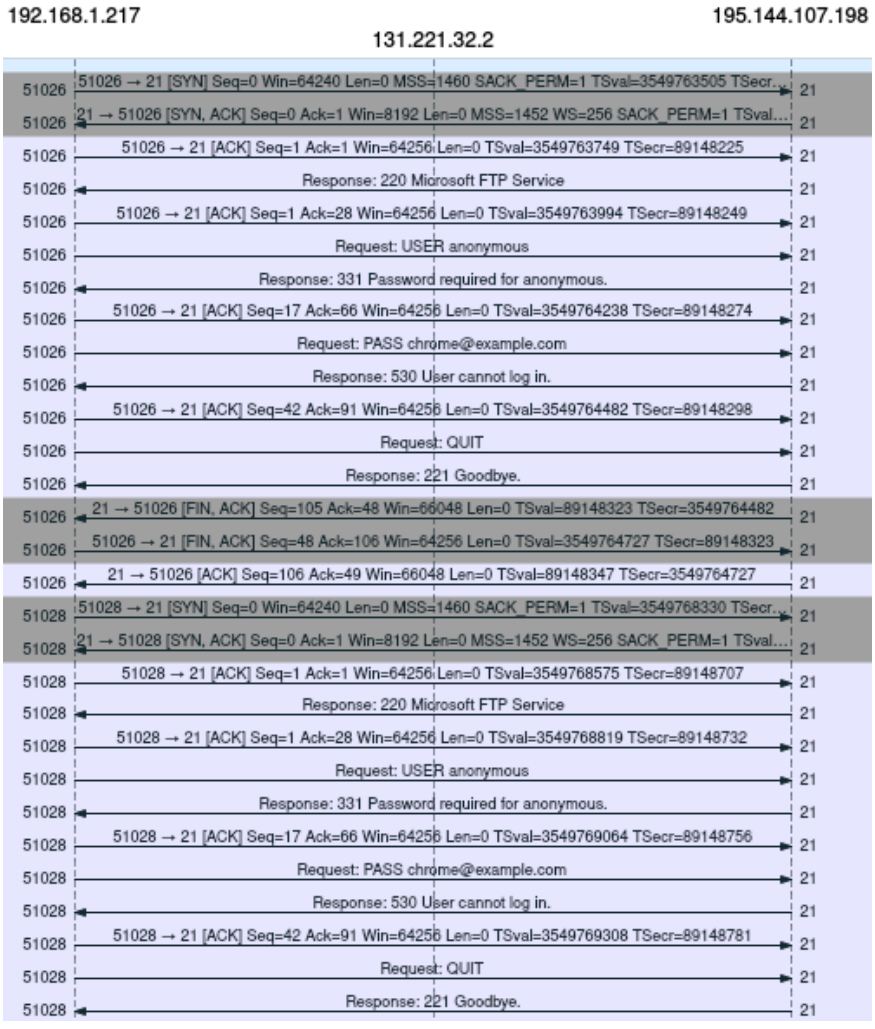


Figura 20: Inicio de flujo de la conexión FTP

Una vez el cliente escoge cuál archivo descargar, se vuelve a establecer la conexión

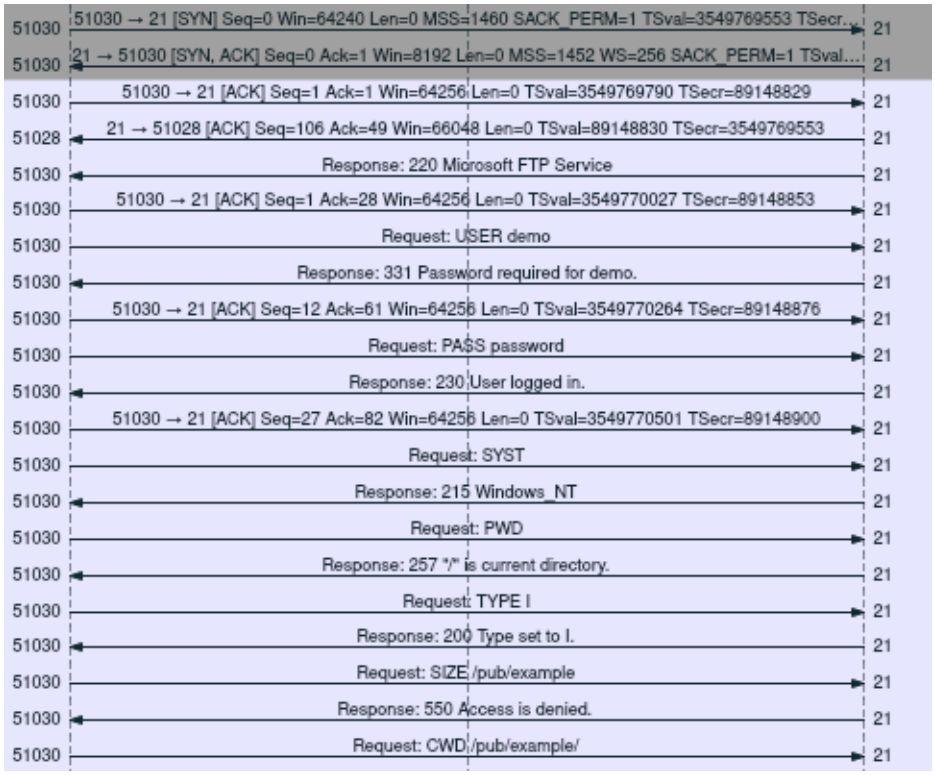


Figura 21: Flujo de la conexión FTP: Conexión de usuario

de la manera explicada anteriormente, y el cliente le solicita a el servidor el archivo WinFormClient.png (el más pesado, 78.1 KiB), el servidor responde diciendo que la conexión ya está establecida y se inicializará la transferencia (figura 22, el cliente lo reconoce, provocando que el servidor comience a transferir el archivo (1440 bytes en cada transferencia), el cliente reconoce los datos, para luego repetirse el ciclo del servidor enviando 1440 bytes y el cliente reconociendo dichos bytes, hasta que se complete totalmente la transferencia.

Al finalizar la transferencia el servidor (figura 23) envía un paquete TCP con la bandera de FIN para finalizar la conexión y una respuesta anunciando que la transferencia de datos se ha completado, el cliente reconoce, envía un paquete con la bandera FIN y reconociendo la solicitud del servidor, finalmente el servidor reconoce el FIN del cliente y finaliza la conexión.

Dentro de la conexión entre cliente y servidor, el parámetro Seq hace referencia al número de bytes que enviará en el paquete, mientras que en la respuesta el parámetro Ack indica el número de byte siguiente a recibir, esto es útil para comprobar si se perdieron paquetes durante la conexión.

Figura 22: Flujo de la conexión FTP: Transferencia de WinFormClient.png

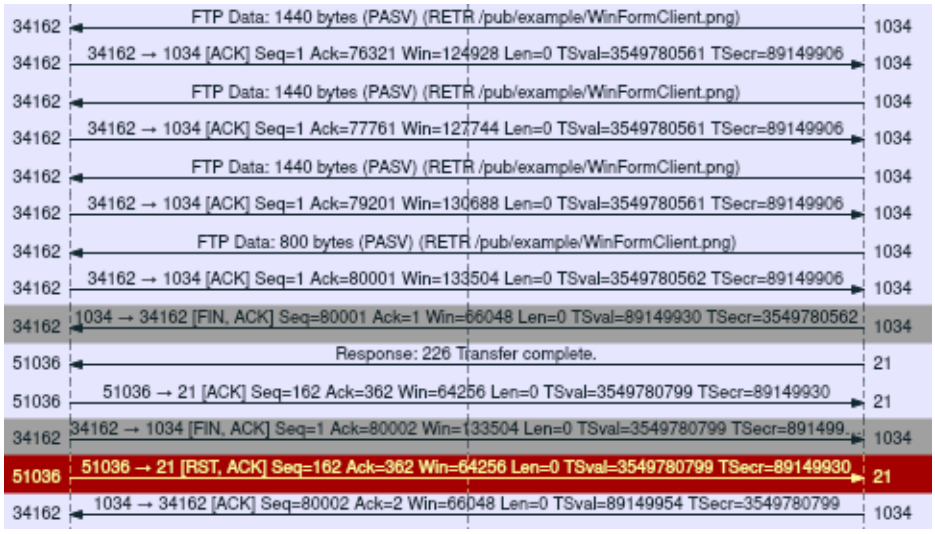


Figura 23: Flujo de la conexión FTP: Fin de transferencia de WinFormClient.png

El archivo transferido pesa 80.000 bytes, es posible ver en la figura 23 que el valore

de ACK al finalizar la transferencia de datos es 80.001, significando que hubo una cantidad de 80.000 bytes transferidos por parte del servidor, significando que se transfirió completamente el archivo. Cabe destacar que no ocurrió ningún error durante la etapa de transferencia del archivo WinFormClient.png.

4. Análisis de resultados

4.1. Parte 1

Podemos observar en el terminal 3.1 la máscara de red (255.255.255.0) podemos obtener el rango de direcciones de la red local, y después al ejecutar el comando *map* ver cuales son las direcciones dentro de nuestra red local. Al realizar los comandos podemos observar en el wireshark (figura 2) que se utiliza el protocolo ARP de la misma forma de la que se vio en clases, es decir, enviando paquetes a la dirección broadcast para comprobar que host están disponibles, y le llega de respuesta la dirección MAC de cada uno de estos.

Posteriormente utilizando el comando *ping* se pudo observar como al ir modificando el tamaño del paquete de las distintas direcciones escogidas, cambia el tiempo promedio de cada una de estas. Igualmente se puede ver como el TTL mínimo de una dirección es mayor al encontrarse en un continente distinto al nuestro.

Lo anterior dicho se puede complementar cuando se realizar el comando *traceroute*, donde se puede observar por cuantas direcciones (routers) tiene que pasar para llegar a la IP de destino, y ver en los mapas (figuras 9, 10 y 11) las distancias que tiene que recorrer para llegar a esta. Por eso tiene un TTL mínimo mayor que otras direcciones, por ejemplo las páginas nacionales.

Al igual como vimos en clases todas las direcciones obtenidas siguen la estructura del protocolos de internet versión 4 (IPv4), no se pudo observar ninguna que utilizara la estructura del protocolo versión 6 (IPv6).

4.2. Parte 2

Con respecto al funcionamiento de TCP, el resultado tiene sentido, debido a que al inicializar la conexión entre cliente y servidor, primero es necesario obtener la dirección IP de dicha conexión utilizando el protocolo DNS, luego realizar el three-way handshake de TCP para comenzar a transmitir información entre los dos puntos, una vez finalizada la transmisión, se utiliza la bandera FIN para terminar la conexión entre ambos dispositivos. Durante todo este proceso no ocurrió ningún tipo de error, esto se puede saber porque en la figura 16 no hay ningún paquete TCP con bandera RST activada. Debido a todo esto, se

obtiene que los resultados obtenidos durante esta sección de funcionamiento TCP es correcto según lo aprendido durante las clases.

Por otra parte, en la sección del funcionamiento FTP, los resultados obtenidos tienen sentido pues pues los datos son transportados utilizando el protocolo TCP del flujo presente de las figuras 20, 21, 22 y 23, son consistentes con lo estudiado en clases y lo discutido en el anterior párrafo. Sin embargo, La conexión debe de volver a ser establecida varias veces durante el proceso, esto puede ser debido al modo incógnito del navegador utilizado o a la configuración establecida en el servidor FTP investigado. Por otra parte, durante la transmisión de archivo, no ocurrieron errores y los valores de Ack y Seq tienen sentido respecto a lo enseñado en clases.

5. Conclusiones

Luego de la experiencia realizada pudimos analizar y entender de mejor manera los conceptos y funcionamientos de protocolos a nivel de la capa de red y de transporte, utilizando las herramientas mencionadas a lo largo del laboratorio.

Al obtener la información de los distintos dispositivos conectados a la red local, pudimos vercom funciona el protocolo ARP, se entiende como se forma una red de área local (LAN), que dispositivos de nosotros la componen, y como se comunican a través direcciones MAC e IP. También vimos el uso del protocolo ICMP para enviar paquetes a las distintas direcciones ubicada en distintos continentes, y como obtener el respectivo IP de cada una de estas junto con el tiempo que demora en obtenerlas. Comprender como es la ruta (conjunto de routers) en la que tiene que ir un paquete para llegar a la dirección destino.

También pudimos comprender de forma "real" como funciona el protocolo TCP de la capa de transporte, como esta interacciona con la capa de aplicación y con la de red. Al igual que en clases se pudo ver el flujo de como un cliente (nosotros) quiere acceder a un servidor en específico, detallando como es la conexión entre ellos.

Todo lo anterior se pudo analizar gracias a la herramienta Wireshark, que gracias a ella se ve específicamente el tráfico de red y cuales son los protocolos utilizando para cada prueba realizada.

Finalmente, podemos decir que gracias al este segundo laboratorio hemos reforzado nuestro conocimientos acerca de la capa de red y transporte y como estas funcionan para poder comunicar los dispositivos que usamos día a día. Se espera en un próximo trabajo de laboratorio poder complementar lo aprendido hoy sobre estas capas, y ver como se comunican los dispositivos en la última capa del modelo, la capa de aplicación.

Bibliografía

Stallings, W. (2004). *Comunicaciones y Redes de Computadores*, volume 7. Pearson.