

Práctica 6

Capa de Transporte

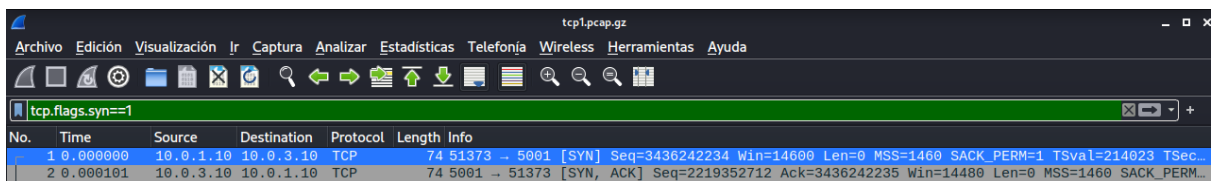
Grupo z

Integrantes: Herrera Francisco, Domé Luis y Gagliardi Pablo.

9. (Ejercicio de promoción) Para la captura dada, responder las siguientes preguntas.

NOTA: para quienes hagan la promoción, este será un ejercicio entregable. En la entrega deberán estar todas las preguntas respondidas y debidamente justificadas. En los puntos donde es necesario ejecutar comandos, los mismos deberán adjuntarse a la entrega.

a. ¿Cuántos intentos de conexiones TCP hay?



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.10	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=3436242234 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=214023 TSecr=...
2	0.000101	10.0.3.10	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SACK_PERM=...

En la imagen podemos ver 1 intento de conexión TCP (pedido con el flag SYN).

El que además tienen el flag ACK es la respuesta del receptor.

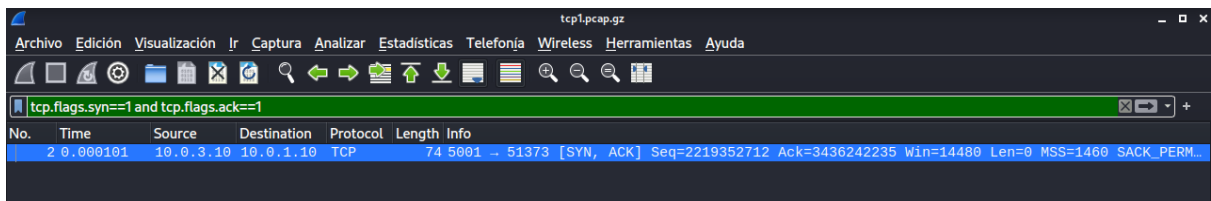
b. ¿Cuáles son la fuente y el destino (IP:port) para c/u?

La fuente es 10.0.1.10:51373 y el destino es 10.0.3.10:5001

c. ¿Cuántas conexiones TCP exitosas hay en la captura? Cómo diferencia las exitosas de las que no lo son? ¿Cuáles flags encuentra en cada una?

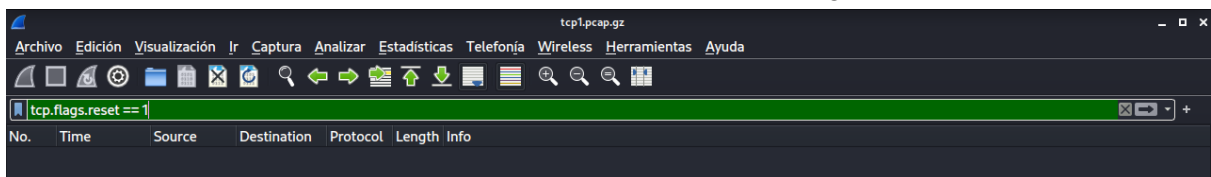
Tenemos 1 conexión TCP exitosas en la captura. Las diferenciamos ya que tienen encendidos los flag SYN y ACK, lo cual significa que la conexión se realizó con éxito. Si hubiese un problema con la conexión, se responde con el flag RST, el cual nos indica que se debe volver a intentar la conexión.

Podemos filtrar las conexiones exitosas por el flag SYN y ACK:



No.	Time	Source	Destination	Protocol	Length	Info
2	0.000101	10.0.3.10	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SACK_PERM=...

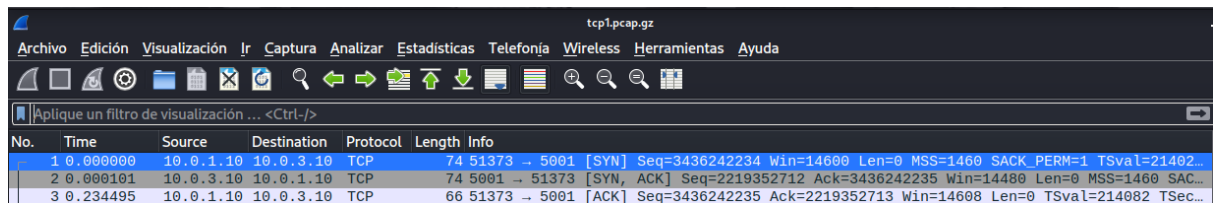
Podemos filtrar las conexiones que no fueron exitosas por el flag RST (si las hubiera).



No.	Time	Source	Destination	Protocol	Length	Info
-----	------	--------	-------------	----------	--------	------

d. Dada la primera conexión exitosa responder:

i. ¿Quién inicia la conexión?



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.10	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=3436242234 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=21402...
2	0.000101	10.0.3.10	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SAC...
3	0.234495	10.0.1.10	10.0.3.10	TCP	66	51373 → 5001 [ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=0 TSval=214082 TSec...

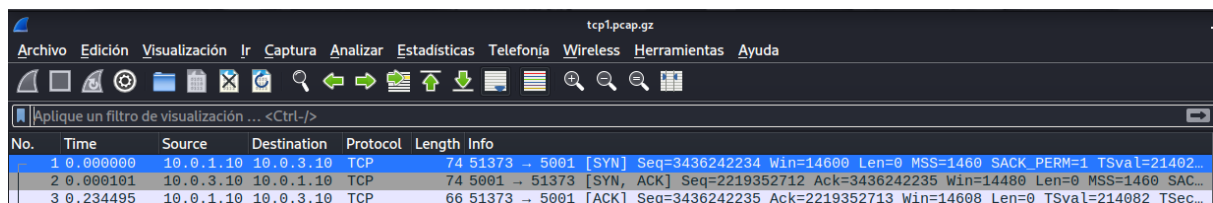
La conexión es iniciada por IP:Port = 10.0.1.10:51373 en el segmento N° 1 de la captura.

ii. ¿Quién es el servidor y quién el cliente?

El cliente, 10.0.1.10:51373, es quien quiere iniciar la conexión y hace el primer paso del 3 way handshake.

El servidor 10.0.3.10:5001, quien lo *recibe*, es el encargado de informar que se ha establecido dicha conexión.

iii. ¿En qué segmentos se ve el 3-way handshake?

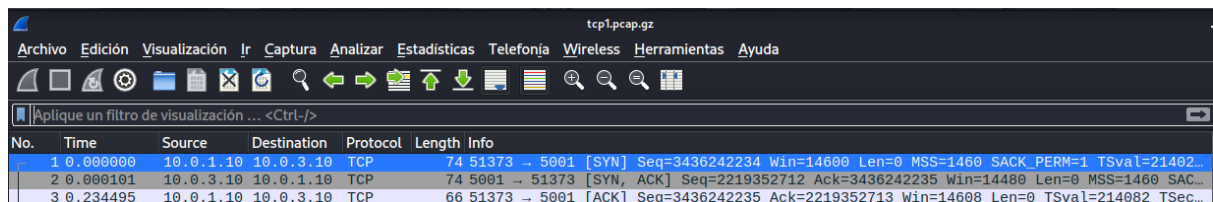


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.10	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=3436242234 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=21402...
2	0.000101	10.0.3.10	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SAC...
3	0.234495	10.0.1.10	10.0.3.10	TCP	66	51373 → 5001 [ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=0 TSval=214082 TSec...

Como se ve en la imagen, tenemos los 3 segmentos 1, 2 y 3, correspondientes al 3-way handshake: el paquete que inicia la conexión con el flag SYN por parte del cliente, el segmento con los flags SYN y ACK por parte del servidor confirmando la conexión, y por último el segmento con el flag ACK por parte del cliente avisando que recibió correctamente la confirmación de la conexión.

iv. ¿Cuáles ISNs (Número de Secuencia Inicial) se intercambian?

El ISN es el primer número de secuencia que se envía (el cual es determinado por el SO) al establecer una conexión.

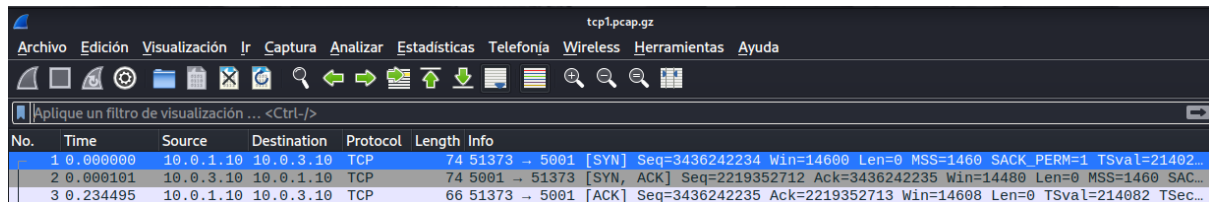


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.10	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=3436242234 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=21402...
2	0.000101	10.0.3.10	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SAC...
3	0.234495	10.0.1.10	10.0.3.10	TCP	66	51373 → 5001 [ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=0 TSval=214082 TSec...

Como se ve en la captura el ISN del cliente es 3436242234 y el del servidor es 2219352712.

v. ¿Cuál MSS se negoció?

La cantidad máxima de datos que pueden tomarse y colocarse en un segmento está limitada por el tamaño máximo de segmento o MSS (es la cantidad máxima de datos de la capa de aplicación en el segmento, no el tamaño máximo del segmento TCP incluyendo las cabeceras).

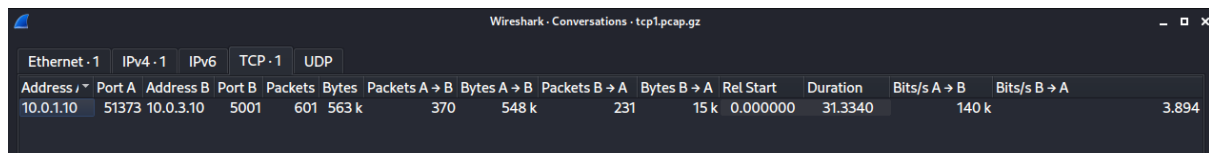


tcp1.pcap.gz

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.10	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=3436242234 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=21402
2	0.000101	10.0.3.10	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SACK
3	0.234495	10.0.1.10	10.0.3.10	TCP	66	51373 → 5001 [ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=0 TSval=214082 TSecr=

El MSS del cliente es de 1460 bytes. y el del servidor también es 1460 bytes.

vi. ¿Cuál de los dos hosts envía la mayor cantidad de datos (IP:port)?

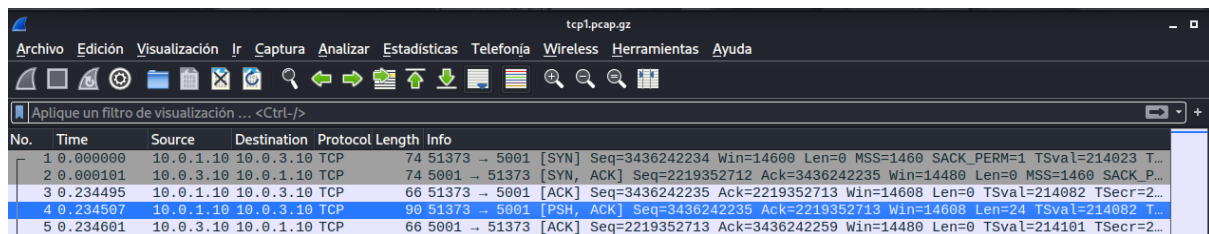


Wireshark · Conversations · tcp1.pcap.gz

Address / Port A	Port B	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel. Start	Duration	Bits/s A → B	Bits/s B → A
10.0.1.10	51373	10.0.3.10	5001	601	563 k	370	548 k	231	15 k	0.000000	31.3340	140 k	3.894

Para poder ver esto, fuimos a *estadísticas* → *conversaciones* → *TCP* y como se puede ver donde está resaltado en la imagen: se envían un total de 563kb. El cliente envía 548kb al servidor; y el servidor envía 15kb al cliente. Llegamos a la conclusión de que el cliente (10.0.1.10:51373) envía la mayor cantidad de datos.

e. Identificar primer segmento de datos (origen, destino, tiempo, número de fila y número de secuencia TCP).

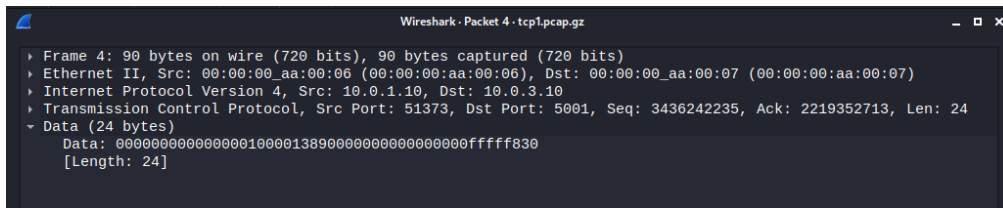


tcp1.pcap.gz

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.10	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=3436242234 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=214023 T...
2	0.000101	10.0.3.10	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SACK_P...
3	0.234495	10.0.1.10	10.0.3.10	TCP	66	51373 → 5001 [ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=0 TSval=214082 TSecr=2...
4	0.234507	10.0.1.10	10.0.3.10	TCP	90	51373 → 5001 [PSH, ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=24 TSval=214082 T...
5	0.234601	10.0.3.10	10.0.1.10	TCP	66	5001 → 51373 [ACK] Seq=2219352713 Ack=3436242259 Win=14480 Len=0 TSval=214101 TSecr=2...

El primer segmento de datos es el Número=4, Src=10.0.1.10:51373, Dst=10.0.3.10:5001, tiempo=0.234507, Seq= 3436242235. Los segmentos 1, 2 y 3 representan al 3-way handshake y no envían datos.

i. ¿Cuántos datos lleva?



El segmento lleva 24 bytes.

ii. ¿Cuándo es confirmado (tiempo, número de fila y número de secuencia TCP)?

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.1.10	10.0.3.10	TCP	74	51373 → 5001 [SYN] Seq=3436242234 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=214023 T...
2	0.000101	10.0.3.10	10.0.1.10	TCP	74	5001 → 51373 [SYN, ACK] Seq=2219352712 Ack=3436242235 Win=14480 Len=0 MSS=1460 SACK_P...
3	0.234495	10.0.1.10	10.0.3.10	TCP	66	51373 → 5001 [ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=0 TSval=214082 TSecr=2...
4	0.234507	10.0.1.10	10.0.3.10	TCP	90	51373 → 5001 [PSH, ACK] Seq=3436242235 Ack=2219352713 Win=14608 Len=24 TSval=214082 T...
5	0.234601	10.0.3.10	10.0.1.10	TCP	66	5001 → 51373 [ACK] Seq=2219352713 Ack=3436242259 Win=14480 Len=0 TSval=214101 TSecr=2...
6	0.234666	10.0.1.10	10.0.3.10	TCP	1514	51373 → 5001 [ACK] Seq=3436242259 Ack=2219352713 Win=14608 Len=1448 TSval=214082 TSec...

Es confirmado por el segmento Número=5, Tiempo=0.234601 y Seq=2219352713.

iii. La confirmación, ¿qué cantidad de bytes confirma?

La confirmación confirma 24 bytes, esto se puede ver en el número del ACK, que es el número de secuencia del segmento enviado por el cliente + los 24 bytes que este contenía (3436242235 + 24 = 3436242259). Notar en el segmento 5, ACK tiene ese valor.

f. ¿Quién inicia el cierre de la conexión? ¿Qué flags se utilizan? ¿En cuáles segmentos se ve (tiempo, número de fila y número de secuencia TCP)?

No.	Time	Source	Destination	Protocol	Length	Info
603	31.177540	10.0.1.10	10.0.3.10	TCP	178	51373 → 5001 [FIN, PSH, ACK] Seq=3436766435 Ack=2219352713 Win=14608 Len=112 TSval=221046 ...
604	31.181675	10.0.3.10	10.0.1.10	TCP	66	5001 → 51373 [FIN, ACK] Seq=2219352713 Ack=3436766548 Win=240368 Len=0 TSval=221838 TSecr=...

El cierre de conexión lo inicia el cliente, utilizando el flag FIN. Esto se ve en el segmento con número de fila 603, con tiempo 31.177540 y número de secuencia 3436766435. En el segmento con número de fila 604, tiempo 31.181675 y número de secuencia 2219352713 se finaliza el cierre de la conexión.

10. (Ejercicio de promoción) Responda las siguientes preguntas respecto del mecanismo de control de flujo.

NOTA: para quienes hagan la promoción, este será un ejercicio entregable. En la entrega deberán estar todas las preguntas respondidas y debidamente justificadas. En los puntos donde es necesario ejecutar comandos, los mismos deberán adjuntarse a la entrega.

a. ¿Quién lo activa? ¿De qué forma lo hace?

Lo activa el receptor, porque es quien tiene el conocimiento del tamaño de su buffer y es quién le informa al emisor que se debe reducir la cantidad de datos que se envían.

Se lleva a cabo modificando el valor del campo windows size. Cuando se empieza a reducir el tamaño de la ventana el mecanismo se considera activo.

b. ¿Qué problema resuelve?

Si no se hace un control de flujo, el emisor puede que envíe muchos paquetes que se pierdan y deban ser retransmitidos, debido a que el receptor no tenía espacio suficiente en su buffer (Rx Buffer). Esto causa una gran congestión innecesaria en la red que debe ser evitada.

c. ¿Cuánto tiempo dura activo y qué situación lo desactiva?

El control está activo toda la conexión, pero se considera desactivado cuando se deja de achicar la ventana a causa del control.

11. (Ejercicio de promoción) Responda las siguientes preguntas respecto del mecanismo de control de congestión.

NOTA: para quienes hagan la promoción, este será un ejercicio entregable. En la entrega deberán estar todas las preguntas respondidas y debidamente justificadas. En los puntos donde es necesario ejecutar comandos, los mismos deberán adjuntarse a la entrega.

a. ¿Quién lo activa el mecanismo de control de congestión? ¿Cuáles son los posibles disparadores?

Lo activa el emisor al detectar ciertos eventos que se dan por un nivel de congestión que hay que arreglar. Los posibles disparadores del control son 3 ACKs duplicados (o 4 para el mismo segmento) o un timeout (expira el RTO).

b. ¿Qué problema resuelve?

Al medir la congestión de la red y adaptar los tamaños de las ventanas y otros parámetros, en función de la congestión medida, se logra bajar la cantidad de paquetes que se envían y por problemas de red se van a perder, consiguiendo un mejor uso de la red en general.

c. Diferencie slow start de congestion-avoidance.

En slow start se incrementa de forma exponencial el tamaño de la ventana hasta el umbral. Congestion-avoidance realiza un incremento de 1 en 1 de la ventana de transmisión hasta que ocurre un error en la red.

12. (Ejercicio de promoción) Para la captura dada, responder las siguientes preguntas.

NOTA: para quienes hagan la promoción, este será un ejercicio entregable. En la entrega deberán estar todas las preguntas respondidas y debidamente justificadas. En los puntos donde es necesario ejecutar comandos, los mismos deberán adjuntarse a la entrega.

a. ¿Cuántas comunicaciones (srcIP,srcPort,dstIP,dstPort) UDP hay en la captura?

Wireshark · Conversations · udp1.pcap.gz

Ethernet · 2

IPv4 · 1

IPv6

TCP

UDP · 4

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	B
10.0.2.10	39513	10.0.4.10	8001	1	46	1	46	0	0	0.150472	0.0000		—
10.0.2.10	49772	10.0.4.10	8002	1	47	1	47	0	0	7.158236	0.0000		—
10.0.2.10	34901	10.0.4.10	8000	1	49	1	49	0	0	14.716749	0.0000		—
10.0.2.10	57402	10.0.4.10	8003	5	238	3	144	2	94	28.543325	35.8844		32

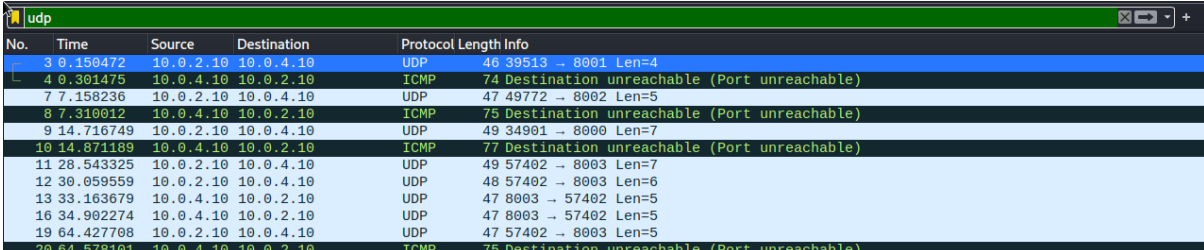
En la imagen podemos ver que hay 4 conversaciones UDP registradas en la captura.

1. La fuente es 10.0.2.10:39513 y el destino es 10.0.4.10:8001
2. La fuente es 10.0.2.10:49772 y el destino es 10.0.4.10:8002
3. La fuente es 10.0.2.10:34901 y el destino es 10.0.4.10:8000
4. La fuente es 10.0.2.10:57402 y el destino es 10.0.4.10:8003

b. ¿Cómo se podrían identificar las exitosas de las que no lo son?

Podemos identificar como exitosas a las comunicaciones que tuvieron una respuesta. Pero de las que no obtuvieron respuesta, no se puede afirmar que no hayan sido exitosas, ya que, por ejemplo, puede haberse perdido la confirmación.

Como se ve en la imagen a través del protocolo ICMP se indica que no se encontró el puerto del emisor (al que se le intenta enviar la respuesta), pero esto no asegura que la conversación no fue exitosa, ya que tal vez el servidor no tenía nada que responder.



No.	Time	Source	Destination	Protocol	Length	Info
3	0.150472	10.0.2.10	10.0.4.10	UDP	46	39513 → 8001 Len=4
4	0.301475	10.0.4.10	10.0.2.10	ICMP	74	Destination unreachable (Port unreachable)
7	7.158236	10.0.2.10	10.0.4.10	UDP	47	49772 → 8002 Len=5
8	7.310012	10.0.4.10	10.0.2.10	ICMP	75	Destination unreachable (Port unreachable)
9	14.716749	10.0.2.10	10.0.4.10	UDP	49	34901 → 8000 Len=7
10	14.871189	10.0.4.10	10.0.2.10	ICMP	77	Destination unreachable (Port unreachable)
11	28.543325	10.0.2.10	10.0.4.10	UDP	49	57402 → 8003 Len=7
12	30.059559	10.0.2.10	10.0.4.10	UDP	48	57402 → 8003 Len=6
13	33.163679	10.0.4.10	10.0.2.10	UDP	47	8003 → 57402 Len=5
16	34.902274	10.0.4.10	10.0.2.10	UDP	47	8003 → 57402 Len=5
19	64.427708	10.0.2.10	10.0.4.10	UDP	47	57402 → 8003 Len=5
20	64.578101	10.0.4.10	10.0.2.10	ICMP	75	Destination unreachable (Port unreachable)

c. ¿UDP sigue el modelo cliente/servidor?

Si, UDP sigue el modelo cliente/servidor, ya que tenemos un cliente que envía un pedido y un servidor que lo recibe y puede o no contestarle.

d. ¿Qué servicios o aplicaciones suelen utilizar este protocolo?

Las aplicaciones que utilizan este protocolo son: aplicaciones de video/voz, aplicaciones de streaming, aplicaciones de telefonía por internet, aplicaciones que utilicen el protocolo TFTP (Trivial File Transfer Protocol) o DNS (Domain Name System) y aplicaciones que utilicen comunicaciones broadcast o multicast.

e. ¿Qué hace el protocolo UDP en relación al control de errores?

El protocolo UDP no implementa un sistema de control de errores propiamente dicho (como si lo hace TCP), pero brinda un control a nivel de bits con el campo checksum.

f. Con respecto a los puertos vistos en las capturas, ¿observa algo particular que lo diferencie de TCP?

No hay nada particular que lo diferencia de TCP. Igualmente hay que tener en cuenta que UDP (al solo utilizar ip:port de destino para direccionar los paquetes que recibe) podría enviar el puerto origen en 0 en el caso de no esperar una respuesta, ya que el host de destino no lo utilizaría.

g. Dada la primera comunicación en la cual se ven datos en ambos sentidos (identificar el primer datagrama):

Como se ve en la imagen, a partir del segmento número 11 se puede ver la primera comunicación con datos viajando en ambos sentidos (respuesta en segmento 13).

No.	Time	Source	Destination	Protocol	Length	Info
3	0.159472	10.0.2.10	10.0.4.10	UDP	46	39513 → 8001 Len=4
4	0.301475	10.0.4.10	10.0.2.10	ICMP	74	Destination unreachable (Port unreachable)
7	7.158236	10.0.2.10	10.0.4.10	UDP	47	49772 → 8002 Len=5
8	7.319912	10.0.4.10	10.0.2.10	ICMP	75	Destination unreachable (Port unreachable)
9	14.716749	10.0.2.10	10.0.4.10	UDP	49	34901 → 8000 Len=7
10	14.871189	10.0.4.10	10.0.2.10	ICMP	77	Destination unreachable (Port unreachable)
11	28.543325	10.0.2.10	10.0.4.10	UDP	49	57402 → 8003 Len=7
12	30.059559	10.0.2.10	10.0.4.10	UDP	48	57402 → 8003 Len=6
13	33.163679	10.0.4.10	10.0.2.10	UDP	47	8003 → 57402 Len=5
16	34.902274	10.0.4.10	10.0.2.10	UDP	47	8003 → 57402 Len=5
19	64.427708	10.0.2.10	10.0.4.10	UDP	47	57402 → 8003 Len=5
20	64.578101	10.0.4.10	10.0.2.10	ICMP	75	Destination unreachable (Port unreachable)

i. ¿Quién envía el primer datagrama (srcIP,srcPort)?

No.	Time	Source	Destination	Protocol	Length	Info
11	28.543325	10.0.2.10	10.0.4.10	UDP	49	57402 → 8003 Len=7
12	30.059559	10.0.2.10	10.0.4.10	UDP	48	57402 → 8003 Len=6
13	33.163679	10.0.4.10	10.0.2.10	UDP	47	8003 → 57402 Len=5
16	34.902274	10.0.4.10	10.0.2.10	UDP	47	8003 → 57402 Len=5
19	64.427708	10.0.2.10	10.0.4.10	UDP	47	57402 → 8003 Len=5
20	64.578101	10.0.4.10	10.0.2.10	ICMP	75	Destination unreachable (Port unreachable)

El primer datagrama es enviado por el IP fuente srcIP= 10.0.2.10 con puerto srcPort=57402, es decir segmento 11 de la captura, que se ve resaltado, que a su vez es el primer segmento de la conexión mencionada anteriormente. Ya que UDP no es un protocolo orientado a la conexión, no es necesario hacer el 3 way handshake.

ii. ¿Cuántos datos se envían en un sentido y en el otro?

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.2.10	57402	10.0.4.10	8003	5	238	3	144	2	94	28.543325	35.8844	32	20

Siendo:

- A. aquel con srcIP= 10.0.2.10 y srcPort=57402.
- B. aquel con srcIP= 10.0.4.10 y srcPort=8003.

Vemos en la captura de la conversación que:

- En total, en la conversación se envían 238 Bytes, en 5 paquetes.
- Bytes enviados A→B = 144Bytes.
- Bytes enviados B→A = 94Bytes.

h. ¿Se puede calcular un RTT?

No es posible calcular el tiempo de ida y vuelta (RTT), ya que este es la cantidad de tiempo que le toma a una señal ser enviada sumado al tiempo en que tarda en recibir el acuso de recibido, "Acknowledgement" o "ACK", el protocolo UDP no cuenta con el flag ACK.