

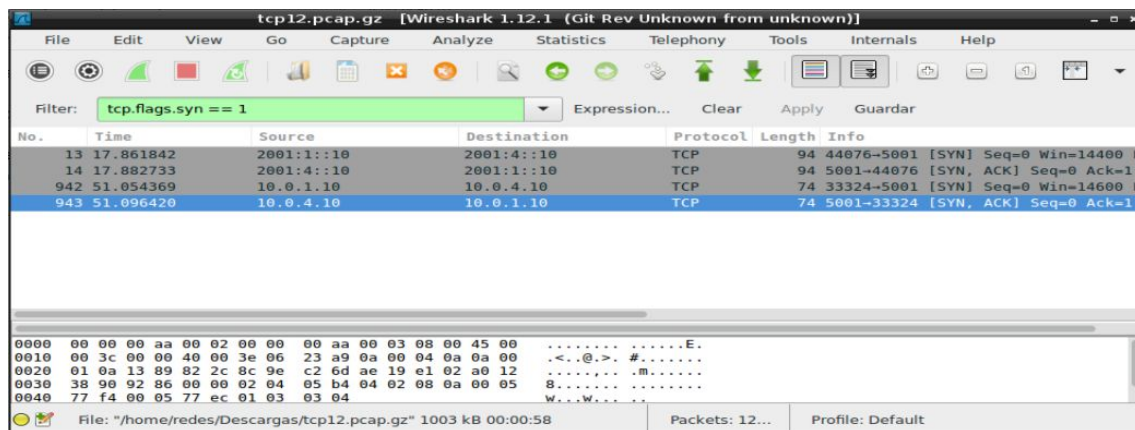
Practica 6

Ejercicio 9

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

a. ¿Cuántos intentos de conexiones TCP hay?

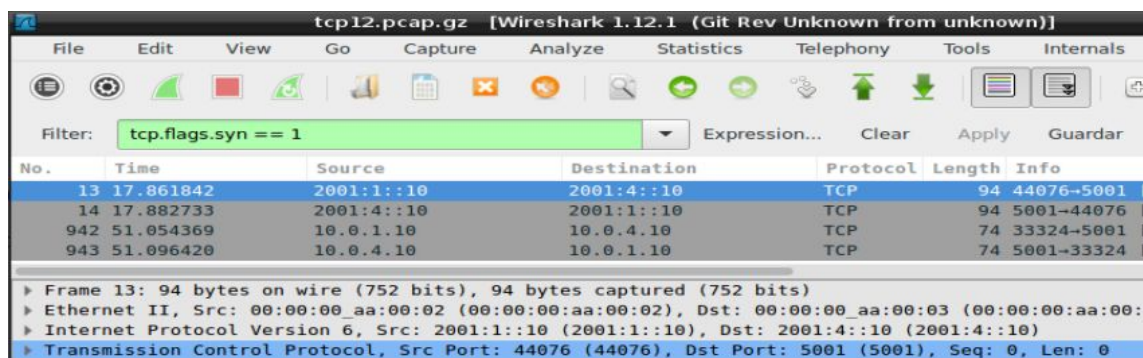
Para saber cuando hay un intento de conexión, debemos identificar la cantidad de mensajes que tienen el flag de SYN prendido, que piden establecer una conexión. En la imagen hay 2 intentos de conexión.



No.	Time	Source	Destination	Protocol	Length	Info
13	17.861842	2001:1::10	2001:4::10	TCP	94	44076->5001 [SYN] Seq=0 Win=14400 L
14	17.882733	2001:4::10	2001:1::10	TCP	94	5001->44076 [SYN, ACK] Seq=0 Ack=1
942	51.054369	10.0.1.10	10.0.4.10	TCP	74	33324->5001 [SYN] Seq=0 Win=14600 L
943	51.096420	10.0.4.10	10.0.1.10	TCP	74	5001->33324 [SYN, ACK] Seq=0 Ack=1

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

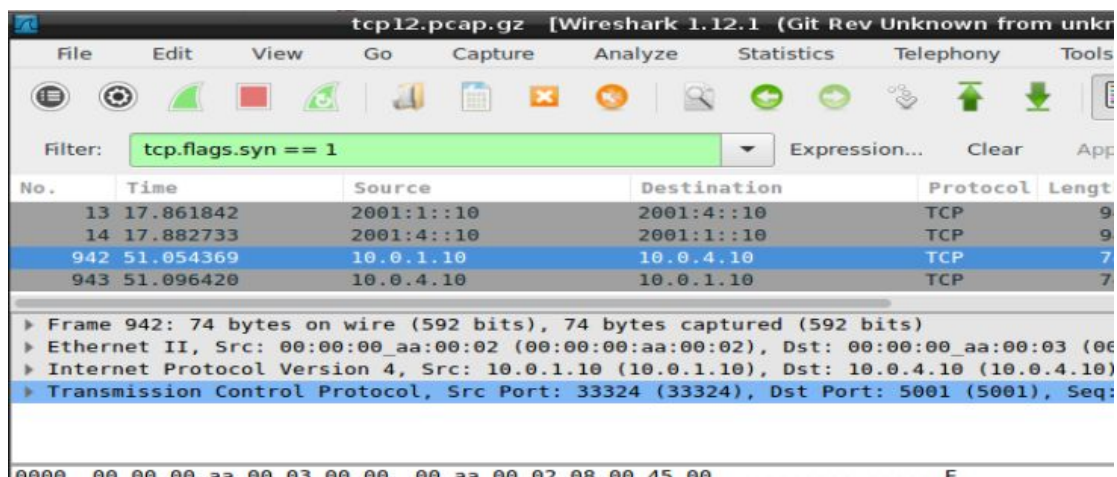
En el primer intento de conexión la dirección IP de la fuente es: 2001:1::10 con puerto 44076 y la dirección IP del destino es 2001:4::10 con puerto 5001.



No.	Time	Source	Destination	Protocol	Length	Info
13	17.861842	2001:1::10	2001:4::10	TCP	94	44076->5001
14	17.882733	2001:4::10	2001:1::10	TCP	94	5001->44076
942	51.054369	10.0.1.10	10.0.4.10	TCP	74	33324->5001
943	51.096420	10.0.4.10	10.0.1.10	TCP	74	5001->33324

Frame 13: 94 bytes on wire (752 bits), 94 bytes captured (752 bits)
Ethernet II, Src: 00:00:00_aa:00:02 (00:00:00:aa:00:02), Dst: 00:00:00_aa:00:03 (00:00:00:aa:00:03)
Internet Protocol Version 6, Src: 2001:1::10 (2001:1::10), Dst: 2001:4::10 (2001:4::10)
Transmission Control Protocol, Src Port: 44076 (44076), Dst Port: 5001 (5001), Seq: 0, Len: 0

En el segundo intento de conexión la dirección IP de la fuente es: 10.0.1.10 con puerto 33324 y la dirección IP del destino es 10.0.4.10 con puerto 5001.



No.	Time	Source	Destination	Protocol	Length	Info
13	17.861842	2001:1::10	2001:4::10	TCP	94	44076->5001
14	17.882733	2001:4::10	2001:1::10	TCP	94	5001->44076
942	51.054369	10.0.1.10	10.0.4.10	TCP	74	33324->5001
943	51.096420	10.0.4.10	10.0.1.10	TCP	74	5001->33324

Frame 942: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
Ethernet II, Src: 00:00:00_aa:00:02 (00:00:00:aa:00:02), Dst: 00:00:00_aa:00:03 (00:00:00:aa:00:03)
Internet Protocol Version 4, Src: 10.0.1.10 (10.0.1.10), Dst: 10.0.4.10 (10.0.4.10)
Transmission Control Protocol, Src Port: 33324 (33324), Dst Port: 5001 (5001), Seq: 0, Len: 0

- 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?

Ambas conexiones son exitosas, ya que se completa el 3 way handshake, en la imagen se puede apreciar los 2 primeros mensajes del saludo, en la captura se ve también el tercero que completa el saludo.

No.	Time	Source	Destination	Protocol	Length	In
13	17.861842	2001:1::10	2001:4::10	TCP	94	440
14	17.882733	2001:4::10	2001:1::10	TCP	94	500
942	51.054369	10.0.1.10	10.0.4.10	TCP	74	333
943	51.096420	10.0.4.10	10.0.1.10	TCP	74	500

En caso de que las conexiones no sean exitosas el servidor nos va a responder un mensaje con el flag RST (reset) prendido. Como se puede apreciar en la imagen de abajo, no hay ningún mensaje con este flag:

No.	Time	Source	Destination
-----	------	--------	-------------

- d. Dada la primera conexión exitosa responder:

- ¿Quién inicia la conexión?
 - ¿Quién es el servidor y quién el cliente?
 - ¿En qué segmentos se ve el 3-way handshake?
 - ¿Cuáles ISNs se intercambian?
 - ¿Cuál MSS se negoció?
 - ¿Cuál de los dos hosts envía la mayor cantidad de datos (IP:port)?
- i. La primera conexión exitosa la inicia el source IP 2001:1::10

No.	Time	Source	Destination	Protocol	Length
13	17.861842	2001:1::10	2001:4::10	TCP	9
14	17.882733	2001:4::10	2001:1::10	TCP	9
942	51.054369	10.0.1.10	10.0.4.10	TCP	7
943	51.096420	10.0.4.10	10.0.1.10	TCP	7

Frame 942: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)

Ethernet II, Src: 00:00:00_aa:00:02 (00:00:00:aa:00:02), Dst: 00:00:00_aa:00:03 (00:00:00:aa:00:03)

Internet Protocol Version 4, Src: 10.0.1.10 (10.0.1.10), Dst: 10.0.4.10 (10.0.4.10)

Transmission Control Protocol, Src Port: 33324 (33324), Dst Port: 5001 (5001), Seq: 1000000000

- ii. El cliente es quien envía la petición, en este caso el IP 2001:1::10, y el servidor es quien recibe la petición, el IP 2001:4::10
- iii. Se ve en los segmentos con número 3454136311, 2842952321 y 3454136312.

13	17.861842	2001:1::10	2001:4::10	TCP	94	44076-5001 [SYN] Seq=3454136311 Win=14
14	17.882733	2001:4::10	2001:1::10	TCP	94	5001-44076 [SYN, ACK] Seq=2042952321 A
15	17.882816	2001:1::10	2001:4::10	TCP	86	44076-5001 [ACK] Seq=3454136312 Ack=20

Esto se puede reconocer ya que el primer segmento se envía con el flag de SYN prendido (enviado por el cliente), luego obtiene una respuesta con el flag SYN y ACK prendido (enviada por el servidor), el cual confirma la llegada del primer segmento de la conversación, y finaliza con un tercer segmento solo con un ACK prendido (enviado por el cliente) que confirma la recepción del segundo segmento de la conversación.

- iv. El ISN (Initial Sequence Number) es el primer número de secuencia que se envía al establecer una conexión, el cual es determinado por el sistema operativo. Los que se intercambian son 3454136311 de parte del cliente y 2042952321 de parte del servidor.
- v. El MSS (Maximum Segment Size) es el tamaño máximo de segmento que puede recibir un host. Quien determina el MSS es el receiver, como en TCP las comunicaciones son bidireccionales, ambos lados determinan cual es el tamaño máximo de segmento que van a recibir.

En el primer segmento el cliente avisa su MSS es 1440 bytes

Filter:	<div>tcp</div>	▼	Expression...	Clear	Ap
io.	Time	Source	Destination	Protocol	Length
13	17.861842	2001:1::10	2001:4::10	TCP	
14	17.882733	2001:4::10	2001:1::10	TCP	
15	17.882816	2001:1::10	2001:4::10	TCP	
16	17.883759	2001:1::10	2001:4::10	TCP	1
17	17.884172	2001:1::10	2001:4::10	TCP	15
18	17.884353	2001:1::10	2001:4::10	TCP	15

[Good Checksum: False]

[Bad Checksum: False]

Urgent pointer: 0

▼ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Opera

▶ Maximum segment size: 1440 bytes

▶ TCP SACK Permitted Option: True

▼ Timestamps: TSval 350082 TSer 0

000	00	00	00	00	aa	00	03	00	00	00	aa	00	02	86	dd	60	00
010	00	00	00	00	28	06	40	20	01	00	01	00	00	00	00	00	00	...(.@.

En el segundo segmento el servidor avisa su MSS también es 1440 bytes (no necesariamente tiene que ser igual que el del cliente)

Iter:	tcp		Expression...	Clear	Apply	
	Time	Source	Destination	Protocol	Length	In
13	17.861842	2001:1::10	2001:4::10	TCP	94	44076-5001
14	17.882733	2001:4::10	2001:1::10	TCP	94	5001-44076
15	17.882816	2001:1::10	2001:4::10	TCP	86	44076-5001
16	17.883759	2001:1::10	2001:4::10	TCP	110	44076-5001
17	17.884172	2001:1::10	2001:4::10	TCP	1514	44076-5001
18	17.884353	2001:1::10	2001:4::10	TCP	1514	44076-5001
Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-operation						
▶ Maximum segment size: 1440 bytes						
▶ TCP SACK Permitted Option: True						
▼ Timestamps: TSval 350085, TSecr 350082						
Kind: Time Stamp Option (8)						
Length: 10						
Timestamp value: 350085						
00 00 00 00 aa 00 02 00 00 00 aa 00 03 86 dd 60 00^						

vi. El host que envía la mayor cantidad de datos es 2001:1::10:44076

Wireshark Summary

File: /home/redes/Descargas/tcp12.pcap.gz
 Length: 1003674 bytes
 Format: Wireshark/tcpdump/... - pcap (gzip con...
 Encapsulation: Ethernet
 Packet size limit: 65535 bytes

Time
 First packet: 2020-04-20 15:08:31
 Last packet: 2020-04-20 15:09:29
 Elapsed: 00:00:58

Capture
 Capture file comments:

Interface: unknown
 Dropped Packets: unknown
 Capture Filter: unknown
 Link type: Ethernet
 Packet size limit: 65535 bytes

Display
 Display filter: ipv6.src == 2001:1::10 and tcp
 Ignored packets: 0 (0,000%)

Traffic	Captured	Displayed	Displayed %
Packets	1263	463	36,659%
Between first and last packet	58,450 sec	21,758 sec	
Avg. packets/sec	21,608	21,280	
Avg. packet size	779 bytes	1502 bytes	
Bytes	983458	695210	70,690%
Avg. bytes/sec	16825,514	31952,259	
Avg. MBit/sec	0,135	0,256	

Epoch time when this frame was captured (frame.time_epoch)

En Wireshark, filtramos los mensajes TCP que envía el cliente, y nos dice que la cantidad de datos que envía es el 70.690% dentro de la comunicación. Notar que aunque no sea el que mayor cantidad de paquetes envía sí es el que mayor cantidad de datos envía.

- Identificar primer segmento de datos (origen, destino, tiempo, número de fila y número de secuencia TCP).
 - ¿Cuántos datos lleva?
 - ¿Cuándo es confirmado (tiempo, número de fila y número de secuencia TCP)?
 - La confirmación, ¿qué cantidad de bytes confirma?

En la imagen se puede apreciar el primer segmento y todos los datos del mismo:

Filter: tcp

No.	Time	Source	Destination	Protocol	Length	Info
13	17.861842	2001:1::10	2001:4::10	TCP	94	44076-5001 [SYN] Seq=3454136311
14	17.882733	2001:4::10	2001:1::10	TCP	94	5001-44076 [SYN, ACK] Seq=3454136311
15	17.882816	2001:1::10	2001:4::10	TCP	86	44076-5001 [ACK] Seq=3454136311
16	17.883759	2001:1::10	2001:4::10	TCP	110	44076-5001 [PSH, ACK] Seq=3454136311
17	17.884172	2001:1::10	2001:4::10	TCP	1514	44076-5001 [ACK] Seq=3454136311
18	17.884252	2001:1::10	2001:4::10	TCP	1514	44076-5001 [ACK] Seq=3454136311
19	17.884314	2001:1::10	2001:4::10	TCP	1514	44076-5001 [ACK] Seq=3454136311
20	17.884370	2001:1::10	2001:4::10	TCP	1514	44076-5001 [ACK] Seq=3454136311

Frame 13: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface 0
 Ethernet II, Src: 00:00:00:aa:00:02 (00:00:00:aa:00:02), Dst: 00:00:00:aa:00:03 (00:00:00:aa:00:03)
 Internet Protocol Version 6, Src: 2001:1::10 (2001:1::10), Dst: 2001:4::10 (2001:4::10)
 Transmission Control Protocol, Src Port: 44076 (44076), Dst Port: 5001 (5001), Seq: 3454136311, Len: 0
 Source Port: 44076 (44076)
 Destination Port: 5001 (5001)
 [Stream index: 0]
 [TCP Segment Len: 0]
 Sequence number: 3454136311
 Acknowledgment number: 0
 Header Length: 40 bytes
 ... 0000 0000 0010 = Flags: 0x002 (SYN)
 Window size value: 14400
 [Calculated window size: 14400]
 Checksum: 0xfa99 [validation disabled]
 Urgent pointer: 0
 Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale

- i. Este pedido no lleva datos de información, solo datos del protocolo para sincronizar la conexión TCP (flags, MSS, etc). Se puede apreciar que no tiene una sección "Data" y que tiene el valor "Len: 0".
 - ii. Es confirmado en el segundo 17.882733, en la fila 14 y con número de secuencia 3454136311
 - iii. Confirma la cantidad de bytes del paquete 3454136311, ya que el ACK es 3454136312.
- 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)?

Filter:		tcp.flags.fin == 1			Expression...	Clear	Apply	Guardar
No.	Time	Source	Destination	Protocol	Length	Info		
795	32.969238	2001:1::10	2001:4::10	TCP	1422	44076->5001 [FIN, PSH, ACK] Seq=3454790360		
936	39.619464	2001:4::10	2001:1::10	TCP	86	5001->44076 [FIN, ACK] Seq=2042952322 Ack=		

Lo inicia el cliente con IP 2001:1::10. Para esto manda un segmento con el flag de FIN prendido, el cual tiene el número de fila 795, al segundo 32.969238, cuyo número de secuencia es 3454790360.

Ejercicio 10

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

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

El que lo activa es el receptor porque conoce el tamaño de su buffer y es quien le informa al sender que vaya reduciendo (o no) la ventana de datos, variando el valor en el campo windows size. Se considera activo cuando el windows size se reduce a un valor menor al máximo del sender, es decir, cuando se lo empieza a limitar..

b. ¿Qué problema resuelve?

El control de flujo tiene en cuenta la cantidad de datos que puede almacenar el receiver y cuando la aplicación va procesando los datos. Esto nos evita que el receiver descarte mensajes porque está saturado, en su lugar va achicando la ventana del sender a medida que su espacio disminuye, de esta forma se evita que el sender envíe mensajes que el receiver no puede recibir (los cuales sería descartados) y evitamos un tráfico innecesario en la red.

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

El receiver está constantemente informando del espacio que tiene para recibir, pero se considera que se desactiva cuando el tamaño de lo que puede recibir es mayor o igual a lo que el sender puede enviar, es decir, que no se limita al sender.

Ejercicio 11

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

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

Al control de congestión lo activa el sender, cuando detecta situaciones (disparadores) como 3 ACK duplicados o 4 para el mismo segmento, o un timeout, cuando expira el RTO, con los cuales interpreta que hay una congestión en la red.

b. ¿Qué problema resuelve?

Si no tendríamos control de congestión, cuando la red se satura, va a descartar muchos de los paquetes que envía el sender, y se va a seguir congestionando porque se van a mandar paquetes constantemente, haciendo un mal uso de la red.

c. Diferencie slow start de congestion-avoidance.

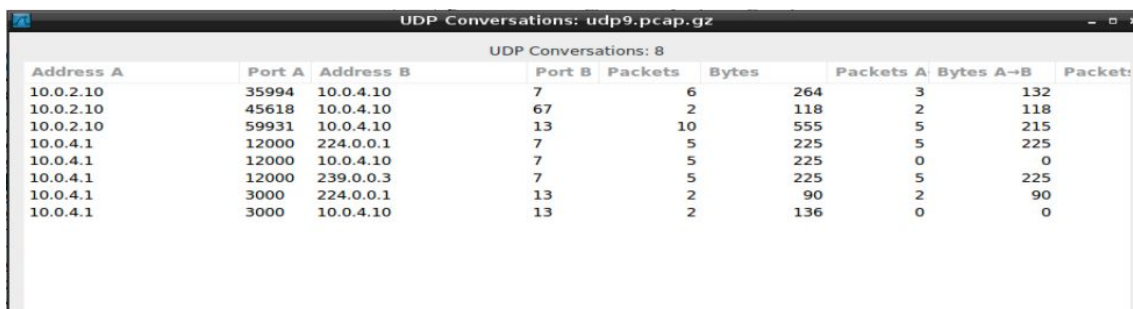
Ambos son métodos que dictan una forma de incrementar el tamaño de la ventana de transmisión en base a la congestión, congestion-avoidance plantea un incremento lineal, de a 1 unidad, en cambio, slow start plantea un incremento exponencial, multiplicando x2 al valor actual. Un ejemplo del uso de estas técnicas es Tahoe, el cual plantea en la primera medida usar slow start, y para las próximas usar slow start hasta un umbral y luego seguir con congestion-avoidance.

Ejercicio 12

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

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

Hay un total de 8 comunicaciones UDP.



Address A	Port A	Address B	Port B	Packets	Bytes	Packets A	Bytes A→B	Packet:
10.0.2.10	35994	10.0.4.10	7	6	264	3	132	
10.0.2.10	45618	10.0.4.10	67	2	118	2	118	
10.0.2.10	59931	10.0.4.10	13	10	555	5	215	
10.0.4.1	12000	224.0.0.1	7	5	225	5	225	
10.0.4.1	12000	10.0.4.10	7	5	225	0	0	
10.0.4.1	12000	239.0.0.3	7	5	225	5	225	
10.0.4.1	3000	224.0.0.1	13	2	90	2	90	
10.0.4.1	3000	10.0.4.10	13	2	136	0	0	

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

Se pueden identificar algunos mensajes fallidos o exitosos, interpretando los mensajes de otros protocolos como ICMP que nos muestra, por ejemplo, un mensaje cuando el puerto al que intentamos comunicarnos no está disponible, pero no tenemos manera de identificar todas las comunicaciones exitosas y fallidas, como nombramos antes, solo algunas.

c. ¿UDP sigue el modelo cliente/servidor?

Si, tiene todos los requisitos para utilizarlo.

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

Este protocolo tiene varias utilidades, algunos servicios como los que se ven en la captura (ECHO, DAYTIME), también se utiliza para:

- Protocolo de Transferencia de Ficheros Trivial (TFTP)
- Usado por el Sistema de Ficheros en Red (NFS)
- Protocolo de Gestión Simple de Redes (SNMP)
- Aplicaciones multicast
- Aplicaciones en tiempo real (real time applications) para transmisión de audio o video.

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

La única medida que toma el protocolo para el control de errores es la utilización del campo checksum, el cual no es muy efectivo.

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

En la captura no se aprecia nada diferente en particular, pero con respecto a los puertos TCP y UDP hay que tener en cuenta que el puerto 0 no se debería usar en TCP, cuando en UDP se podría.

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

- ¿Quién envía el primer datagrama (srcIP,srcPort)?
- ¿Cuántos datos se envían en un sentido y en el otro?

i. El primer datagrama es enviado por la IP 10.0.2.10 desde el puerto 35994 a la IP 10.0.4.10, puerto 7.

No.	Time	Source	Destination	Protocol	Length	Info
16	11.706563	10.0.2.10	10.0.4.10	ECHO	44	Request
17	11.706743	10.0.4.10	10.0.2.10	ECHO	44	Response
18	12.867682	10.0.2.10	10.0.4.10	ECHO	44	Request
19	12.867846	10.0.4.10	10.0.2.10	ECHO	44	Response
20	13.600944	10.0.2.10	10.0.4.10	ECHO	44	Request
21	13.601093	10.0.4.10	10.0.2.10	ECHO	44	Response

UDP Conversations: udp9.pcap.gz							
UDP Conversations: 8							
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A	Bytes A→B
10.0.2.10	35994	10.0.4.10	7	6	264	3	132
10.0.2.10	45618	10.0.4.10	67	2	118	2	118

ii. El cliente, con IP 10.0.2.10, envía 132 bytes, y el servidor, con IP 10.0.4.10, también envía 132 bytes.

Packets A	Bytes A→B	Packets A	Bytes A→B
3	132	3	132

h. ¿Se puede calcular un RTT?

No, para calcular un RTT es necesario tener una confirmación de llegada de un paquete, un ACK, del cual UDP no dispone, por lo que no se puede calcular un RTT.