Tarea 1 Redes – Uso y análisis de Wireshark

Integrantes:

-Sofía Parada

-Benjamín Gutiérrez

1.

Para este caso se realizo un juego en 4 turnos en donde se jugo en la columna 1, 2, 3 y 4, ganando el cliente.

476	47.586693	192.168.1.100	192.168.1.100	TCP	45 50087 → 8000 [PSH, ACK] Seq=1 Ack=1 Win=2619648 Len=1
477	47.586711	192.168.1.100	192.168.1.100	TCP	44 8000 → 50087 [ACK] Seq=1 Ack=2 Win=2619648 Len=0
478	47.586835	127.0.0.1	127.0.0.1	UDP	33 58316 → 1234 Len=1
479	47.587006	127.0.0.1	127.0.0.1	UDP	34 1234 → 58316 Len=2
480	47.587108	127.0.0.1	127.0.0.1	UDP	37 1234 → 58316 Len=5

En la primera conexión debería existir dos mensaje tcp, uno del cliente al servidor mandando la confirmación del juego, y la del servidor al cliente confirmando recepción, luego del intermediario al go el código de confirmación que es 1, y el go recibe confirmo mediante un OK, para luego mandar el puerto dinámico en el que se va a comunicar después.

487	49.224977	192.168.1.100	192.168.1.100	TCP	45 50087 → 8000 [PSH, ACK] Seq=2 Ack=1 Win=2619648 Len=1
488	49.225003	192.168.1.100	192.168.1.100	TCP	44 8000 → 50087 [ACK] Seq=1 Ack=3 Win=2619648 Len=0
489	49.225412	127.0.0.1	127.0.0.1	UDP	33 58317 → 64456 Len=1
490	49.225671	127.0.0.1	127.0.0.1	UDP	33 64456 → 58317 Len=1
491	49.225771	127.0.0.1	127.0.0.1	UDP	37 1234 → 58316 Len=5
492	49.226457	192.168.1.100	192.168.1.100	TCP	331 8000 → 50087 [PSH, ACK] Seq=1 Ack=3 Win=2619648 Len=287
493	49.226471	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=3 Ack=288 Win=2619392 Len=0
494	49.226689	192.168.1.100	192.168.1.100	TCP	45 8000 → 50087 [PSH, ACK] Seq=288 Ack=3 Win=2619648 Len=1
495	49.226700	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=3 Ack=289 Win=2619392 Len=0

Aquí es donde empiezan a jugar, el cliente manda al intermediario la columna a jugar, como se ve en los primeras dos filas, luego el intermediario manda esa columna al puerto dinámico, el go manda de vuelta la columna aleatoria y además el siguiente puerto dinámico.

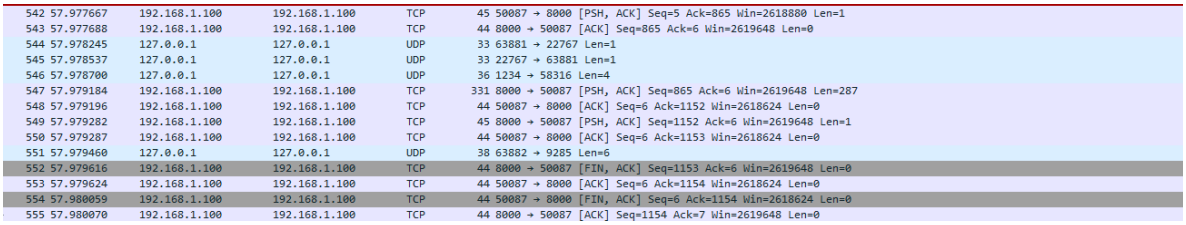
Luego en la conexión tcp el intermediario le manda la tabla actualizada al cliente, el cliente confirma recibo. Y además le manda el digito 5 indicando que el juego todavía no termina.

506	52.600769	192.168.1.100	192.168.1.100	TCP	45 50087 → 8000 [PSH, ACK] Seq=3 Ack=289 Win=2619392 Len=1
507	52.600796	192.168.1.100	192.168.1.100	TCP	44 8000 → 50087 [ACK] Seq=289 Ack=4 Win=2619648 Len=0
508	52.601282	127.0.0.1	127.0.0.1	UDP	33 63879 → 31470 Len=1
509	52.601543	127.0.0.1	127.0.0.1	UDP	33 31470 → 63879 Len=1
510	52.601710	127.0.0.1	127.0.0.1	UDP	37 1234 → 58316 Len=5
511	52.602352	192.168.1.100	192.168.1.100	TCP	331 8000 → 50087 [PSH, ACK] Seq=289 Ack=4 Win=2619648 Len=287
512	52.602364	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=4 Ack=576 Win=2619136 Len=0
513	52.602546	192.168.1.100	192.168.1.100	TCP	45 8000 → 50087 [PSH, ACK] Seq=576 Ack=4 Win=2619648 Len=1
514	52.602553	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=4 Ack=577 Win=2619136 Len=0

En esta iteración ocurre lo mismo que en la anterior, cambian los puertos se manda una columna distinta y se recibe una tabla actualizada junto con el estado del juego.

525	55.591800	192.168.1.100	192.168.1.100	TCP	45 50087 → 8000 [PSH, ACK] Seq=4 Ack=577 Win=2619136 Len=1
526	55.591816	192.168.1.100	192.168.1.100	TCP	44 8000 → 50087 [ACK] Seq=577 Ack=5 Win=2619648 Len=0
527	55.592326	127.0.0.1	127.0.0.1	UDP	33 63880 → 61567 Len=1
528	55.592672	127.0.0.1	127.0.0.1	UDP	33 61567 → 63880 Len=1
529	55.592881	127.0.0.1	127.0.0.1	UDP	37 1234 → 58316 Len=5
530	55.593333	192.168.1.100	192.168.1.100	TCP	331 8000 → 50087 [PSH, ACK] Seq=577 Ack=5 Win=2619648 Len=287
531	55.593344	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=5 Ack=864 Win=2618880 Len=0
532	55.593494	192.168.1.100	192.168.1.100	TCP	45 8000 → 50087 [PSH, ACK] Seq=864 Ack=5 Win=2619648 Len=1
533	55.593500	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=5 Ack=865 Win=2618880 Len=0

Ocurre una tercera iteración con la misma descripción anterior.



542	57.977667	192.168.1.100	192.168.1.100	TCP	45 50087 → 8000 [PSH, ACK] Seq=5 Ack=865 Win=2618880 Len=1
543	57.977688	192.168.1.100	192.168.1.100	TCP	44 8000 → 50087 [ACK] Seq=865 Ack=6 Win=2619648 Len=0
544	57.978245	127.0.0.1	127.0.0.1	UDP	33 63881 → 22767 Len=1
545	57.978537	127.0.0.1	127.0.0.1	UDP	33 22767 → 63881 Len=1
546	57.978700	127.0.0.1	127.0.0.1	UDP	36 1234 → 58316 Len=4
547	57.979184	192.168.1.100	192.168.1.100	TCP	331 8000 → 50087 [PSH, ACK] Seq=865 Ack=6 Win=2619648 Len=287
548	57.979196	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=6 Ack=1152 Win=2618624 Len=0
549	57.979282	192.168.1.100	192.168.1.100	TCP	45 8000 → 50087 [PSH, ACK] Seq=1152 Ack=6 Win=2619648 Len=1
550	57.979287	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=6 Ack=1153 Win=2618624 Len=0
551	57.979460	127.0.0.1	127.0.0.1	UDP	38 63882 → 9285 Len=6
552	57.979616	192.168.1.100	192.168.1.100	TCP	44 8000 → 50087 [FIN, ACK] Seq=1153 Ack=6 Win=2619648 Len=0
553	57.979624	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [ACK] Seq=6 Ack=1154 Win=2618624 Len=0
554	57.980059	192.168.1.100	192.168.1.100	TCP	44 50087 → 8000 [FIN, ACK] Seq=6 Ack=1154 Win=2618624 Len=0
555	57.980070	192.168.1.100	192.168.1.100	TCP	44 8000 → 50087 [ACK] Seq=1154 Ack=7 Win=2619648 Len=0

Por último, en la cuarta iteración gana el jugador, por lo que las primeras 9 filas ocurre lo mismo que antes, solo que ahora entrega el código para terminar el juego. En la conexión UDP que se encuentra sola se observa al intermediario mandando la orden de terminar la ejecución al go, y en las siguientes filas se observan las ejecuciones terminando.

Con estos análisis, podemos determinar que cada ejecución planteada en el código se observa en el wireshark de manera correcta.

2.

Los protocolos que se debían observar a la hora de revisar el intercambio de mensajes eran TCP entre el cliente y el servidor intermediario, y UDP entre el servidor intermediario y el servidor en go. Fueron estos los que encontramos, para cada conexión TCP había dos filas mientras que por cada UDP solo 1, realizando correctamente sus conexiones y cumpliendo sus protocolos.

3.

El contenido dentro de Wireshark si es legible, cada fila entrega un source y un destino, además del protocolo. En la columna info salen los puertos y la longitud de los paquetes. Si uno hace click en una fila en las conexiones que existe traspaso de data se logra apreciar el contenido enviado, traducido de hexadecimal. Con todo esto se logra concluir que, si uno sabe que mensajes deben realizarse y con que contenido, uno puede analizar las filas y determinar que pasa en cada una.

```
02 00 00 00 45 00 00 20  36 d6 00 00 80 11 00 00   ····E··  6·····
7f 00 00 01 7f 00 00 01  04 d2 e3 cc 00 0c a7 cd   ········ ·····
39 32 38 35                                        9285
```

Ejemplo de puerto enviado por protocolo UDP traducido de hexadecimal