

Redes de Computadores: Laboratorio 1

Jorge Salas (201773625-9)
jorge.salasl@sansano.usm.cl

Javiera Villarroel (201773580-5)
javiera.villarroel@sansano.usm.cl

13/05/2021

Análisis de Tráfico

Se utilizó "Adapter for loopback traffic capture" para obtener el tráfico entre localhost (Windows).

- Respecto al número de mensajes que se envían durante la ejecución del programa, basándose en el esquema de la Figura 1, ¿Cuántos mensajes se deberían enviar durante una partida? Considere ahora su código, ¿Cuántos mensajes se enviaron en una partida?

Durante una partida, según el esquema deberían ocurrir los siguientes mensajes:

1. Cliente envía jugada a servidor Intermedio
2. Servidor Intermedio solicita jugada al servidor Cachipún
3. Servidor Cachipún envía jugada a servidor Intermedio
4. Servidor Intermedio envía resultados

Esto ocurre tantas veces según se hace una jugada. En nuestro tráfico de Wireshark se encontró lo siguiente para una partida completa:

12	2021-05-13	01:21:55,087677	127.0.0.1	127.0.0.1	TCP	50	55942 → 50001 [PSH, ACK] Seq=2 Ack=19 Win=2619648 Len=6
13	2021-05-13	01:21:55,087774	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=19 Ack=8 Win=2619648 Len=0
14	2021-05-13	01:21:55,087967	127.0.0.1	127.0.0.1	UDP	37	53999 → 50022 Len=5
15	2021-05-13	01:21:55,089645	127.0.0.1	127.0.0.1	UDP	37	50022 → 53999 Len=5
16	2021-05-13	01:21:55,091158	127.0.0.1	127.0.0.1	TCP	67	50001 → 55942 [PSH, ACK] Seq=19 Ack=8 Win=2619648 Len=23
17	2021-05-13	01:21:55,091196	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=8 Ack=42 Win=2619648 Len=0
19	2021-05-13	01:21:56,719686	127.0.0.1	127.0.0.1	TCP	49	55942 → 50001 [PSH, ACK] Seq=8 Ack=42 Win=2619648 Len=5
20	2021-05-13	01:21:56,719772	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=42 Ack=13 Win=2619648 Len=0
21	2021-05-13	01:21:56,719922	127.0.0.1	127.0.0.1	UDP	37	53999 → 50022 Len=5
22	2021-05-13	01:21:56,722658	127.0.0.1	127.0.0.1	UDP	38	50022 → 53999 Len=6
23	2021-05-13	01:21:56,722775	127.0.0.1	127.0.0.1	TCP	67	50001 → 55942 [PSH, ACK] Seq=42 Ack=13 Win=2619648 Len=23
24	2021-05-13	01:21:56,722809	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=13 Ack=65 Win=2619648 Len=0
26	2021-05-13	01:21:58,767778	127.0.0.1	127.0.0.1	TCP	50	55942 → 50001 [PSH, ACK] Seq=13 Ack=65 Win=2619648 Len=6
27	2021-05-13	01:21:58,767874	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=65 Ack=19 Win=2619648 Len=0
28	2021-05-13	01:21:58,768830	127.0.0.1	127.0.0.1	UDP	37	53999 → 50022 Len=5
29	2021-05-13	01:21:58,771827	127.0.0.1	127.0.0.1	UDP	38	50022 → 53999 Len=6
30	2021-05-13	01:21:58,771150	127.0.0.1	127.0.0.1	TCP	68	50001 → 55942 [PSH, ACK] Seq=65 Ack=19 Win=2619648 Len=24
31	2021-05-13	01:21:58,771181	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=19 Ack=89 Win=2619648 Len=0
34	2021-05-13	01:22:00,991772	127.0.0.1	127.0.0.1	TCP	50	55942 → 50001 [PSH, ACK] Seq=19 Ack=89 Win=2619648 Len=6
35	2021-05-13	01:22:00,991947	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=89 Ack=25 Win=2619648 Len=0
36	2021-05-13	01:22:00,992863	127.0.0.1	127.0.0.1	UDP	37	53999 → 50022 Len=5
37	2021-05-13	01:22:00,993700	127.0.0.1	127.0.0.1	UDP	38	50022 → 53999 Len=6
38	2021-05-13	01:22:00,995049	127.0.0.1	127.0.0.1	TCP	68	50001 → 55942 [PSH, ACK] Seq=89 Ack=25 Win=2619648 Len=24
39	2021-05-13	01:22:00,995096	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=25 Ack=113 Win=2619648 Len=0
41	2021-05-13	01:22:03,711770	127.0.0.1	127.0.0.1	TCP	50	55942 → 50001 [PSH, ACK] Seq=25 Ack=113 Win=2619648 Len=6
42	2021-05-13	01:22:03,711903	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=113 Ack=31 Win=2619648 Len=0
43	2021-05-13	01:22:03,712060	127.0.0.1	127.0.0.1	UDP	37	53999 → 50022 Len=5
44	2021-05-13	01:22:03,712469	127.0.0.1	127.0.0.1	UDP	38	50022 → 53999 Len=6
45	2021-05-13	01:22:03,714342	127.0.0.1	127.0.0.1	TCP	67	50001 → 55942 [PSH, ACK] Seq=113 Ack=31 Win=2619648 Len=23
46	2021-05-13	01:22:03,714370	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=31 Ack=136 Win=2619648 Len=0
47	2021-05-13	01:22:05,375443	127.0.0.1	127.0.0.1	TCP	50	55942 → 50001 [PSH, ACK] Seq=31 Ack=136 Win=2619648 Len=6
48	2021-05-13	01:22:05,375538	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=136 Ack=37 Win=2619648 Len=0
49	2021-05-13	01:22:05,375671	127.0.0.1	127.0.0.1	UDP	37	53999 → 50022 Len=5
50	2021-05-13	01:22:05,377164	127.0.0.1	127.0.0.1	UDP	38	50022 → 53999 Len=6
51	2021-05-13	01:22:05,379049	127.0.0.1	127.0.0.1	TCP	64	50001 → 55942 [PSH, ACK] Seq=136 Ack=37 Win=2619648 Len=20
52	2021-05-13	01:22:05,379085	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=37 Ack=156 Win=2619392 Len=0
53	2021-05-13	01:22:05,379555	127.0.0.1	127.0.0.1	UDP	35	53999 → 50022 Len=3

> Frame 12: 50 bytes on wire (400 bits), 50 bytes captured (400 bits) on interface \Device\NPF_{...}_id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 55942, Dst Port: 50001, Seq: 2, Ack: 19, Len: 6

✓ Data (6 bytes)

Data: 50695647261

[Length: 6]

```

0000  02 00 00 00 45 00 00 2e 8d 75 40 00 80 06 00 00  ....E...u@....
0010  7f 00 00 01 7f 00 00 01 da 86 c3 51 48 55 6c a2  .........QHUL-
0020  0c e2 b4 55 50 18 27 f9 4d 94 00 00 50 69 65 64  ...UP...M...Pied
0030  72 61                                     ra

```

Figure 1: Tráfico durante una partida

En rojo están los mensajes que ocurren para una ronda, donde se observa lo siguiente:

55942→50001: Se envía jugada al servidor Intermedio que corre en el puerto 50001

50001→55942: Mensaje ACK (Indica que el servidor intermedio ha reconocido el mensaje de sincronización (SYN) que envió el cliente)

53999→50022: Servidor Intermedio (actuando como cliente) solicita jugada al servidor Cachipún (en el puerto donde está corriendo la partida)

50022→53999: Servidor Cachipún envía jugada al servidor Intermedio

50001→55942: Servidor intermedio envía resultados al Cliente

55942→50001: Mensaje ACK

Además, al terminar la partida ocurre el último mensaje marcado el que corresponde a avisarle del fin de la partida al servidor 5022 donde estaba abierta la partida. Considerando esto, por cada ronda ocurren dos mensajes más, y uno más para finalizar la partida.

- Referente a los mensajes realizados por las aplicaciones: ¿Qué tipos de protocolo espera ver? ¿Cuáles encontró? Justifique sus expectativas y las diferencias que encuentre. Se esperaba encontrar los protocolos TCP y UDP que son propios de la naturaleza del problema, así como también se esperaba encontrar packets relacionados con métodos HTTP. Sin embargo, este último no apareció, pero en su lugar, se encontraron enlistados protocolos como TLS 1.2 y mDNS.

Time	Source	Destination	Protocol	Length	Info
1 0.000000000	157.240.204.60	192.168.101.5	TLSv1.2	206	Application Data
2 0.000045990	192.168.101.5	157.240.204.60	TCP	68	33118 → 443 [ACK] Seq=1 Ack=139 Win=977 Len=0 TSval=232229156...
3 0.000000329	157.240.204.60	192.168.101.5	TLSv1.2	206	Application Data
4 0.000083721	192.168.101.5	157.240.204.60	TCP	68	33118 → 443 [ACK] Seq=1 Ack=277 Win=977 Len=0 TSval=232229156...
5 0.409263362	157.240.204.60	192.168.101.5	TLSv1.2	286	Application Data
6 0.409294690	192.168.101.5	157.240.204.60	TCP	68	33118 → 443 [ACK] Seq=1 Ack=495 Win=999 Len=0 TSval=232229197...
7 1.737024081	192.168.101.7	224.0.0.251	MDNS	105	Standard query 0x002a PTR 233637DE, sub. googlecacst. tcp.loc...
8 2.316301950	192.168.101.5	157.240.204.63	TLSv1.2	109	Application Data
9 2.321989531	157.240.204.63	192.168.101.5	TCP	68	443 → 46442 [ACK] Seq=1 Ack=42 Win=333 Len=0 TSval=3415313083...
10 2.668544221	157.240.204.63	192.168.101.5	TLSv1.2	105	Application Data
11 2.668627668	192.168.101.5	157.240.204.63	TCP	68	46442 → 443 [ACK] Seq=42 Ack=38 Win=1553 Len=0 TSval=34930088...
12 2.762795093	127.0.0.1	127.0.0.1	TCP	76	60740 → 50001 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM...
13 2.762804408	127.0.0.1	127.0.0.1	TCP	76	50001 → 60740 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=6549...
14 2.762812321	127.0.0.1	127.0.0.1	TCP	68	60740 → 50001 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=4108781...
15 3.913221993	127.0.0.1	127.0.0.1	TCP	69	60740 → 50001 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=1 TSval=41...
16 3.913296372	127.0.0.1	127.0.0.1	TCP	68	50001 → 60740 [ACK] Seq=1 Ack=2 Win=65536 Len=0 TSval=4108782...
17 3.913436020	127.0.0.1	127.0.0.1	UDP	45	50466 → 50002 Len=1
18 3.913853005	127.0.0.1	127.0.0.1	UDP	62	50002 → 50466 Len=18
19 3.914116562	127.0.0.1	127.0.0.1	TCP	86	50001 → 60740 [PSH, ACK] Seq=1 Ack=2 Win=65536 Len=18 TSval=4...
20 3.914138269	127.0.0.1	127.0.0.1	TCP	68	60740 → 50001 [ACK] Seq=2 Ack=19 Win=65536 Len=0 TSval=410878...
21 5.249724800	192.168.101.5	35.186.224.47	TLSv1.2	111	Application Data
22 5.256531772	35.186.224.47	192.168.101.5	TCP	68	443 → 47856 [ACK] Seq=1 Ack=44 Win=267 Len=0 TSval=319548971 ...
23 5.275565060	127.0.0.1	127.0.0.1	TCP	74	60740 → 50001 [PSH, ACK] Seq=2 Ack=19 Win=65536 Len=6 TSval=4...

Frame 23: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface any, id 0
Linux cooked capture

```

0000 00 00 03 04 00 06 00 00 00 00 00 00 c0 39 08 00 .....9..
0010 45 00 00 3a 4f d0 40 00 40 06 ec eb 7f 00 00 01 E..:0.@.
0020 7f 00 00 01 ed 44 c3 51 07 db 83 bd f9 68 0d 68 ....D.Q.....h.h
0030 80 18 02 00 fe 2e 00 00 01 01 08 0a f4 e7 11 92 .....
0040 f4 e7 0c 40 50 69 65 64 72 61 ...@Pied ra

```

Figure 2: Mensajes encontrados

- Las interacciones vía TCP entre el Cliente y el Servidor Intermediario, ¿deben ocupan los mismos puertos a lo largo del tiempo? ¿Coincide con lo visto en Wireshark? Fundamente.

Sí, debido a que el servidor Intermediario se ejecuta en el puerto 50001. El Cliente se comunica a través de este puerto, y este puerto se cierra solo al finalizar el juego. Lo anterior coincide con lo visto en Wireshark, pero además se observa que el Cliente se ejecuta en un puerto automático no definido en la tarea, en este caso en el puerto 55942, el cual se mantiene hasta el final del programa, de mismo modo que el 50001. A continuación se presenta captura del final del programa: Se muestra desde que el Cliente envía la opción 2 (para cerrar el programa) y se puede observar que se mantiene los mismos puertos que en la Figure 1. Además se observa el cierre (FIN) de la conexión TCP.

103	2021-05-13	01:22:22,079848	127.0.0.1	127.0.0.1	TCP	45	55942 → 50001 [PSH, ACK] Seq=78 Ack=335 Win=2619392 Len=1
104	2021-05-13	01:22:22,079953	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=335 Ack=79 Win=2619648 Len=0
105	2021-05-13	01:22:22,080026	127.0.0.1	127.0.0.1	UDP	33	53997 → 50002 Len=1
106	2021-05-13	01:22:22,080293	127.0.0.1	127.0.0.1	UDP	34	50002 → 53997 Len=2
107	2021-05-13	01:22:22,080378	127.0.0.1	127.0.0.1	TCP	46	50001 → 55942 [PSH, ACK] Seq=335 Ack=79 Win=2619648 Len=2
108	2021-05-13	01:22:22,080417	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=79 Ack=337 Win=2619392 Len=0
109	2021-05-13	01:22:22,080777	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [FIN, ACK] Seq=337 Ack=79 Win=2619648 Len=0
110	2021-05-13	01:22:22,080809	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=79 Ack=338 Win=2619392 Len=0
111	2021-05-13	01:22:22,082557	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [FIN, ACK] Seq=79 Ack=338 Win=2619392 Len=0
112	2021-05-13	01:22:22,082605	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=338 Ack=80 Win=2619648 Len=0

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 > Transmission Control Protocol, Src Port: 55942, Dst Port: 50001, Seq: 78, Ack: 335, Len: 1
 > Data (1 byte)
 Data: 32
 [Length: 1]

0000 02 00 00 00 45 00 00 29 8d cb 40 00 00 06 00 00E...@.....
 0010 7f 00 00 01 7f 00 00 01 da 86 c3 51 48 55 6c eeQHUL.
 0020 0c e2 b5 91 50 18 27 f8 42 41 00 00 30P...BA..

Figure 3: Tráfico al seleccionar la opción 2 y salir del programa

4. Las interacciones vía UDP entre el Servidor Intermediario y el Servidor Cachipún, ¿Deben ocupar los mismos puertos a lo largo del tiempo?, ¿Qué cambia al ejecutar varias partidas seguidas sin cerrar el terminal? ¿Coincide con lo visto en Wireshark? Fundamente.

En este caso no ocupan los mismos puertos, debido a que cada partida se ejecuta en un puerto aleatorio (puede ocurrir que sea el mismo aleatoriamente). El Servidor Cachipún principal se ejecuta en el puerto 50002, con el cual el servidor Intermediario puede comunicarse, pero al comenzar una partida abre una conexión en un puerto aleatorio con el cual también se comunica el servidor Intermediario para obtener la jugada aleatoria del bot. Por lo tanto, al ejecutar varias partidas seguidas sin cerrar el terminal, las partidas se ejecutarán en puertos aleatorios, por lo que no se mantiene el mismo a lo largo del tiempo. En este caso, se presenta una partida que se ejecuta en el puerto 50019, a diferencia de la partida de Figure 1, la cual se ejecutaba en el puerto 50022.

54	2021-05-13	01:22:08,639522	127.0.0.1	127.0.0.1	TCP	45	55942 → 50001 [PSH, ACK] Seq=37 Ack=156 Win=2619392 Len=1
55	2021-05-13	01:22:08,639603	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=156 Ack=38 Win=2619648 Len=0
56	2021-05-13	01:22:08,639690	127.0.0.1	127.0.0.1	UDP	33	53997 → 50002 Len=1
57	2021-05-13	01:22:08,645789	127.0.0.1	127.0.0.1	UDP	50	50002 → 53997 Len=18
58	2021-05-13	01:22:08,646064	127.0.0.1	127.0.0.1	TCP	62	50001 → 55942 [PSH, ACK] Seq=156 Ack=38 Win=2619648 Len=18
59	2021-05-13	01:22:08,646116	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=38 Ack=174 Win=2619392 Len=0
60	2021-05-13	01:22:09,983756	127.0.0.1	127.0.0.1	TCP	50	55942 → 50001 [PSH, ACK] Seq=38 Ack=174 Win=2619392 Len=6
61	2021-05-13	01:22:09,983860	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=174 Ack=44 Win=2619648 Len=0
62	2021-05-13	01:22:09,984069	127.0.0.1	127.0.0.1	UDP	37	50068 → 50019 Len=5
63	2021-05-13	01:22:09,985641	127.0.0.1	127.0.0.1	UDP	38	50019 → 50068 Len=6
64	2021-05-13	01:22:09,987098	127.0.0.1	127.0.0.1	TCP	67	50001 → 55942 [PSH, ACK] Seq=174 Ack=44 Win=2619648 Len=23
65	2021-05-13	01:22:09,987129	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=44 Ack=197 Win=2619392 Len=0
66	2021-05-13	01:22:10,735800	127.0.0.1	127.0.0.1	TCP	49	55942 → 50001 [PSH, ACK] Seq=44 Ack=197 Win=2619392 Len=5
67	2021-05-13	01:22:10,735906	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=197 Ack=49 Win=2619648 Len=0
68	2021-05-13	01:22:10,736029	127.0.0.1	127.0.0.1	UDP	37	50068 → 50019 Len=5
69	2021-05-13	01:22:10,737424	127.0.0.1	127.0.0.1	UDP	37	50019 → 50068 Len=5
70	2021-05-13	01:22:10,739007	127.0.0.1	127.0.0.1	TCP	67	50001 → 55942 [PSH, ACK] Seq=197 Ack=49 Win=2619648 Len=23
71	2021-05-13	01:22:10,739042	127.0.0.1	127.0.0.1	TCP	44	55942 → 50001 [ACK] Seq=49 Ack=220 Win=2619392 Len=0
72	2021-05-13	01:22:11,519755	127.0.0.1	127.0.0.1	TCP	50	55942 → 50001 [PSH, ACK] Seq=49 Ack=220 Win=2619392 Len=6
73	2021-05-13	01:22:11,519857	127.0.0.1	127.0.0.1	TCP	44	50001 → 55942 [ACK] Seq=220 Ack=55 Win=2619648 Len=0
74	2021-05-13	01:22:11,519990	127.0.0.1	127.0.0.1	UDP	37	50068 → 50019 Len=5
75	2021-05-13	01:22:11,523223	127.0.0.1	127.0.0.1	UDP	38	50019 → 50068 Len=6

> Transmission Control Protocol, Src Port: 50001, Dst Port: 55942, Seq: 156, Ack: 38, Len: 18
 > Data (18 bytes)
 Data: 4f4b7c6c6f63616c686f73747c3530303139
 [Length: 18]

0000 02 00 00 00 45 00 00 3a 8d 9e 40 00 00 06 00 00E...@.....
 0010 7f 00 00 01 7f 00 00 01 c3 51 da 86 0c e2 b4 deQHUL.
 0020 48 55 6c c6 50 18 27 f9 1f 00 00 00 4f 4b 7c 6cHUL.P...OK|
 0030 6f 63 61 6c 68 6f 73 74 7c 35 30 30 31 39ocalhost|50019

Figure 4: Tráfico durante una partida en el puerto 50019

5. Respecto al contenido de los mensajes enviados entre cada par de entidades, ¿son legibles? ¿Por qué es o no legible el contenido del mensaje?

Los mensajes enviados contienen caracteres que no son legibles debido a que se envían codificados (lo cual se puede observar en el código al utilizar encode en Python o []byte en Golang), como en el caso de la foto se muestran los bytes y su formato hexadecimal. El mensaje codificado se observa en la foto (recuadro rojo), donde además se entrega el largo del mensaje, en este caso "OK|localhost|50042" con un Len=18. Wireshark separa los distintos parámetros entregados en el mensaje. Bajo este recuadro se muestra el men-

saje decodificado, donde Wireshark separa el mensaje por distintos parámetros, siendo el final "Data", es decir, el mensaje enviado.

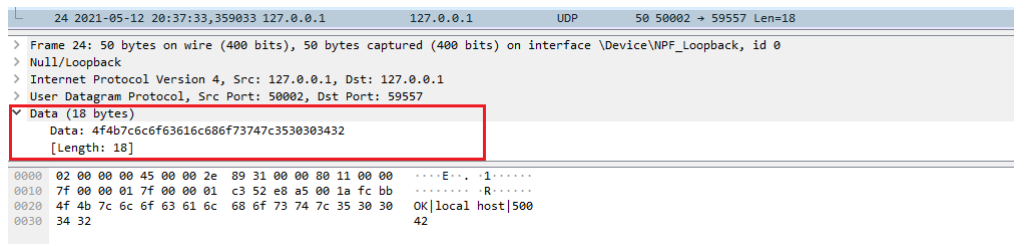


Figure 5: Data del mensaje

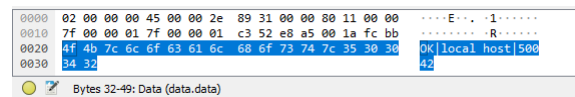


Figure 6: Data descodificada