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).

1. 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:

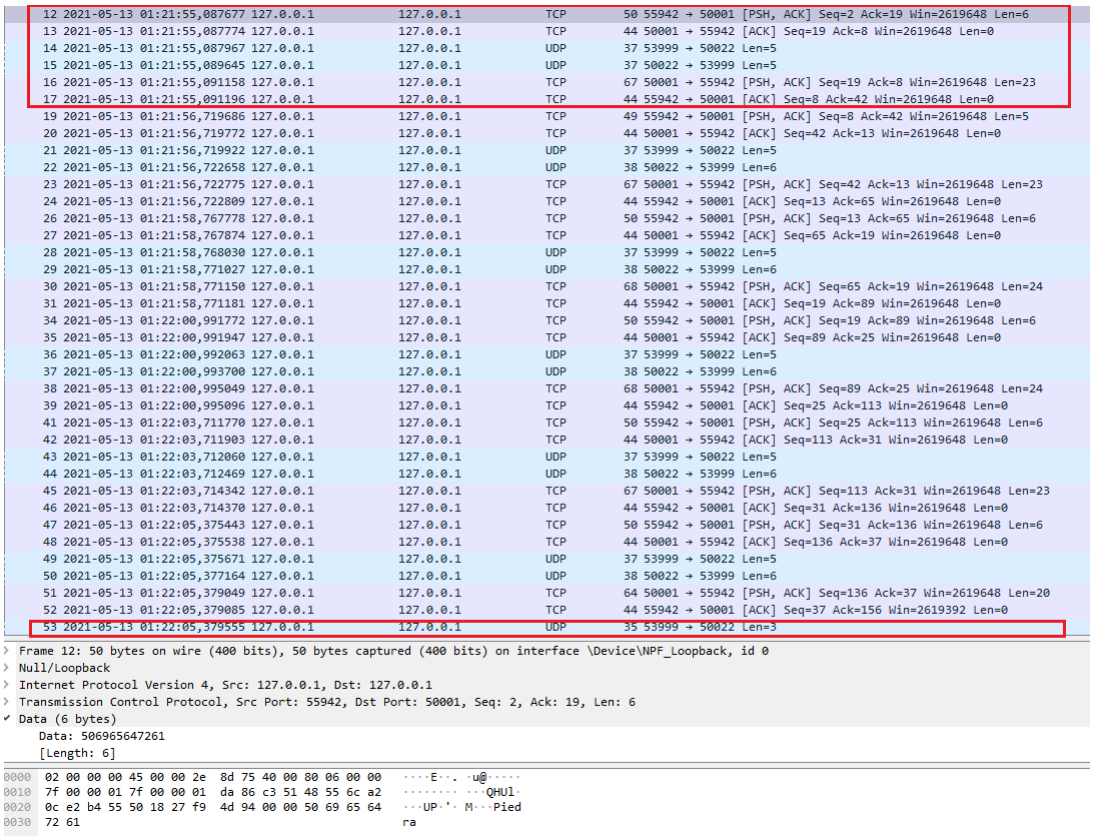

Figure 1: Tráfico durante una partida

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.

2. 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.

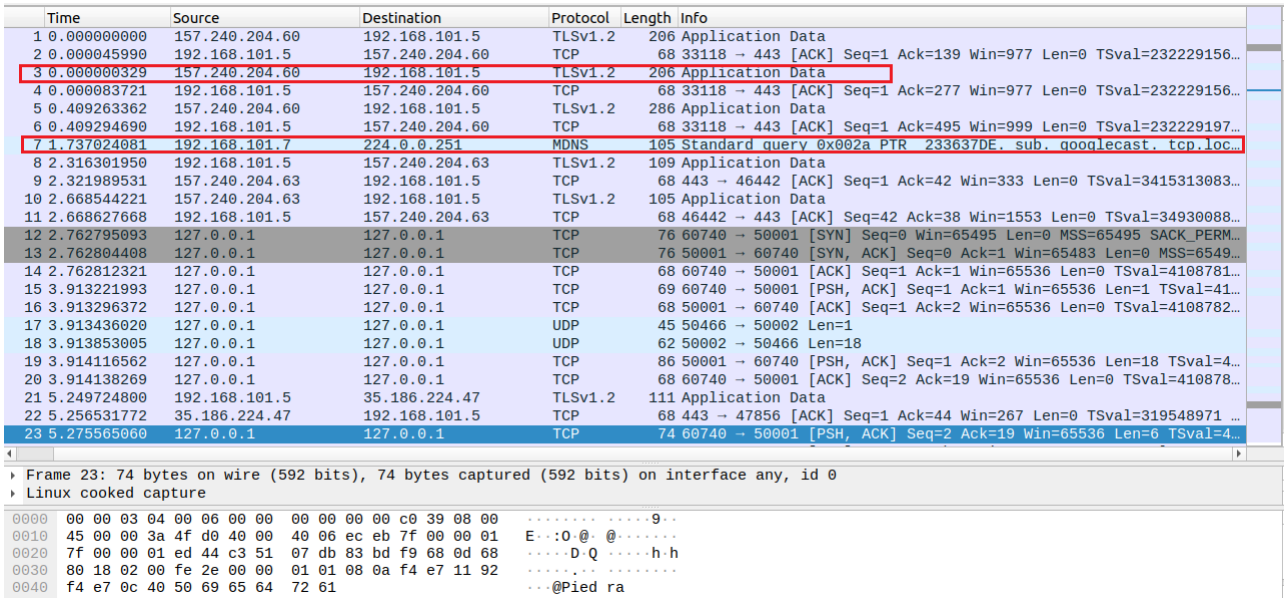

Figure 2: Mensajes encontrados

3. 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.

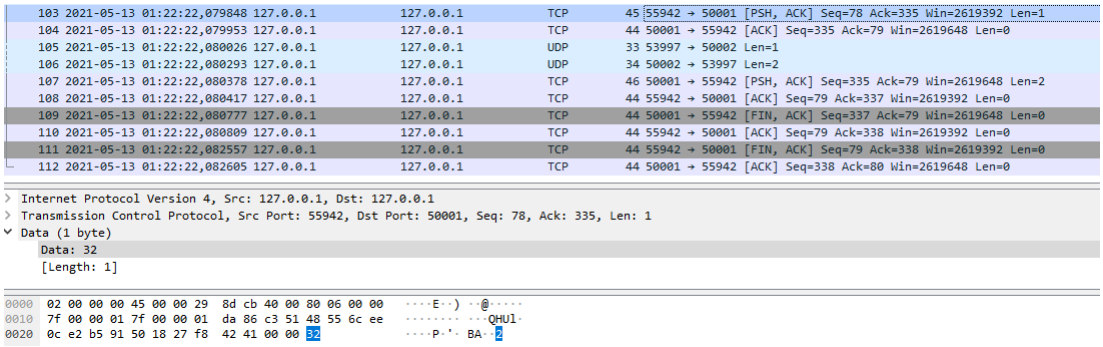

Figure 3: Tráfico al seleccionar la opción 2 y salir del programa

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 ejecutaran 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.

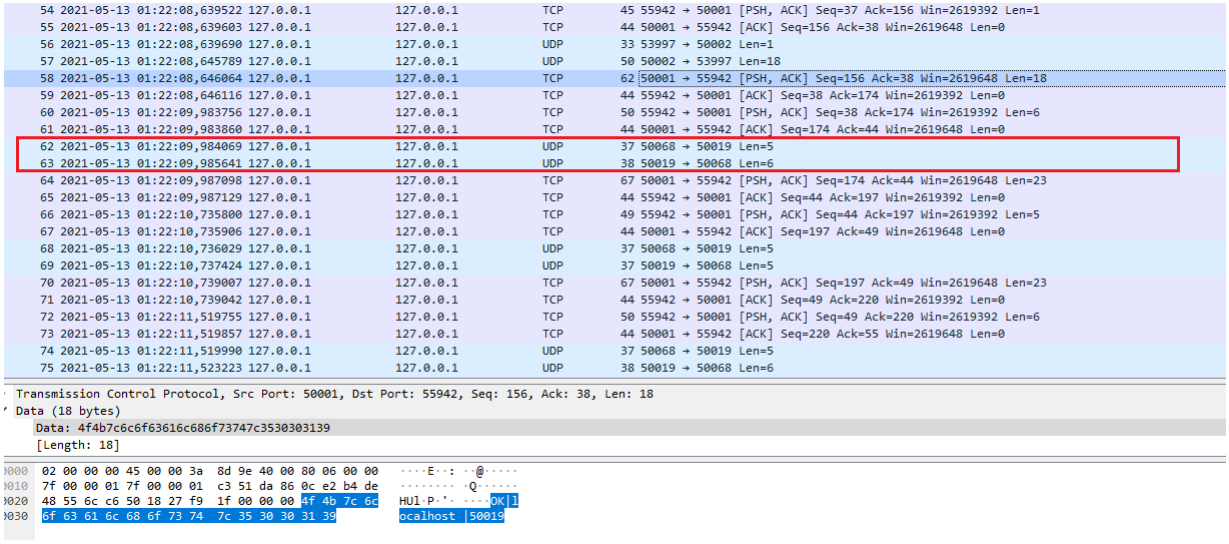

Figure 4: Tráfico durante una partida en el puerto 50019

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.

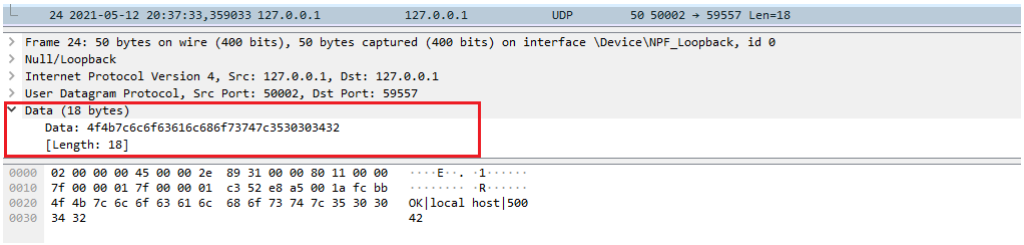

Figure 5: Data del mensaje

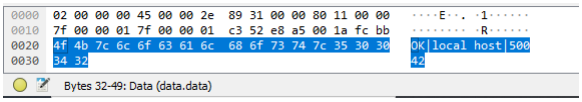

Figure 6: Data descodificada