UNIVERSIDAD DE LOS ANDES

DEPARTAMENTO DE INGENIERÍA DE

SISTEMAS Y COMPUTACIÓN

LABORATORIO 3: ANÁLISIS DE CAPA DE TRANSPORTE Y SOCKETS

ISIS 3204 – INFRAESTRUCTURA DE COMUNICACIONES

PROFESORES:
NATHALIA QUIROGA

GRUPO 9:

JUAN DAVID PEREZ DIAZ – 202210323

JUAN SEBASTIAN OJEDA ROMERO - 202110289

DANIEL FELIPE TORRES LOPEZ -202110365

Table of Contents

Repositorio

https://github.com/sebasojedar/TCP_UDP_publisher_subscriber_lab3redes.git

Pruebas y comparación

A la hora de probar, se utilizó un bróker, dos publicadores (que jugarán los roles de transmisores de dos partidos distintos) y dos suscriptores (que en este caso serán dos personas distintas que estarán viendo partidos distintos).

Para realizar pruebas en TCP, se utilizó el comando sudo tcpdump -i lo tcp port 8080 -w tcp_pubsub.pcap dado que para esta parte del laboratorio se había trabajado en un entorno virtual dentro de WSL. El orden en el que fue realizada la prueba fue primero se conectó el bróker, luego los dos publicadores cada uno estableciendo su tema y por último se conectaron los suscriptores (cada uno tenía que escoger por medio de texto a que tema suscribirse).

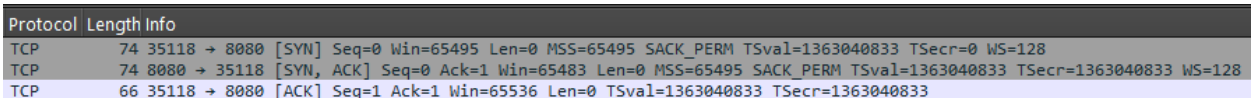

Figura 1, captura de frames del three way handshake del primer publicador en TCP.

En la anterior captura se observa el three way handshake realizado por el primer publicador, el proceso por el cual el protocolo de TCP se asegura de establecer una conexión estable entre un cliente y un servidor. Sabemos que es este proceso gracias a las características flags de SYN, SYN y ACK, y ACK. En las siguientes capturas se encontrará el three way handshake para el publicador faltante y los dos suscriptores correspondientes.

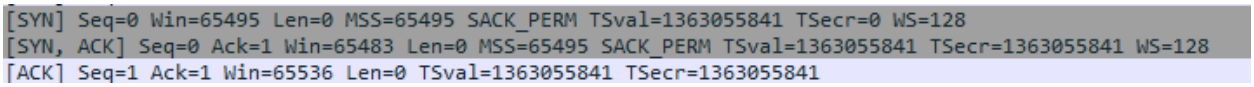

Figura 2, captura de frames del three way handshake del segundo publicador en TCP.

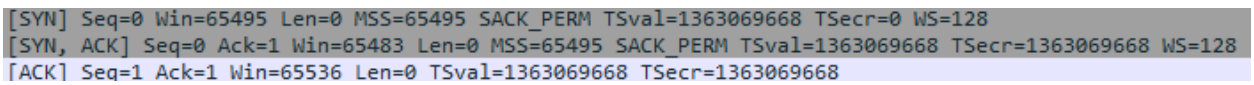

Figura 3, captura de frames del three way handshake del primer suscriptor en TCP.

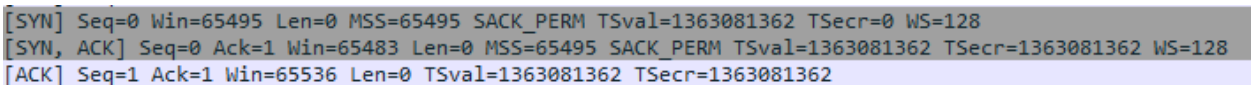

Figura 4, captura de frames del three way handshake del segundo suscriptor en TCP.

A partir de este punto se enviaron mensajes enumerados como gol# (donde # es el número del gol realizado) en grupos de a 3 turnándose entre publicadores de tal forma que podamos observar si algún mensaje se pierde, si llega en desorden o si llega distinto a lo esperado.

En base a los resultados del .pcap podemos observar que todos los archivos llegaron en el orden deseado, indicando que TCP es un protocolo apto para enviar mensajes ordenados y completos, sin riesgo de pérdida.

Capturas UDP

En la siguiente secuencia de imágenes se muestra el funcionamiento del sistema usando el protocolo UDP. A diferencia del caso con TCP, en este protocolo no existe un proceso de conexión como el *three-way handshake*, ya que los mensajes se envían directamente sin establecer una sesión.

Aquí se capturan los primeros datagramas enviados por los publicadores hacia el bróker.
En estos paquetes se ven los puertos de origen distintos (por ejemplo 45014, 60845) enviando al puerto 9000.
Representa el momento en que los publicadores comienzan a transmitir sus mensajes iniciales (“Gol 1”, “Gol 2”, etc.)

	Time	Source	Destination	Protocol	Lengtl	Info
1	0.000000	127.0.0.1	127.0.0.1	UDP	54	45359 → 9000 Len=12
2	0.000310	127.0.0.1	127.0.0.1	UDP	48	9000 → 45359 Len=6
3	9.021328	127.0.0.1	127.0.0.1	UDP	54	59375 → 9000 Len=12
4	9.021469	127.0.0.1	127.0.0.1	UDP	48	9000 → 59375 Len=6
5	19.274985	127.0.0.1	127.0.0.1	UDP	59	60845 → 9000 Len=17
6	19.275224	127.0.0.1	127.0.0.1	UDP	46	9000 → 45359 Len=4
7	20.262264	127.0.0.1	127.0.0.1	UDP	59	60845 → 9000 Len=17
8	20.262494	127.0.0.1	127.0.0.1	UDP	46	9000 → 45359 Len=4
9	21.248968	127.0.0.1	127.0.0.1	UDP	59	60845 → 9000 Len=17
10	21.249187	127.0.0.1	127.0.0.1	UDP	46	9000 → 45359 Len=4

Estos paquetes muestran cómo el bróker recibe los datagramas en el puerto 9000 y los reenvía a los puertos de los suscriptores (por ejemplo 43559, 59375).
Aquí se puede notar el cambio en el puerto destino, lo que indica el reenvío correcto hacia cada suscriptor.

11	23.002520	127.0.0.1	127.0.0.1	UDP	59 45014 → 9000 Len=17
12	23.002688	127.0.0.1	127.0.0.1	UDP	46 9000 → 59375 Len=4
13	24.496979	127.0.0.1	127.0.0.1	UDP	59 45014 → 9000 Len=17
14	24.497152	127.0.0.1	127.0.0.1	UDP	46 9000 → 59375 Len=4
15	25.812746	127.0.0.1	127.0.0.1	UDP	59 45014 → 9000 Len=17
16	25.812972	127.0.0.1	127.0.0.1	UDP	46 9000 → 59375 Len=4
17	27.725338	127.0.0.1	127.0.0.1	UDP	59 60845 → 9000 Len=17
18	27.725608	127.0.0.1	127.0.0.1	UDP	46 9000 → 45359 Len=4
19	29.123075	127.0.0.1	127.0.0.1	UDP	59 60845 → 9000 Len=17
20	29.123281	127.0.0.1	127.0.0.1	UDP	46 9000 → 45359 Len=4

En esta parte se observan los datagramas llegando a los suscriptores.

El orden se mantiene igual al de envío porque la prueba se hizo en localhost (127.0.0.1), lo que evita retrasos o pérdida de paquetes.

Corresponde a la recepción ordenada de los mensajes de goles intercalados entre los dos tem

21	30.041063	127.0.0.1	127.0.0.1	UDP	59 60845 → 9000 Len=17
22	30.041280	127.0.0.1	127.0.0.1	UDP	46 9000 → 45359 Len=4
23	31.979724	127.0.0.1	127.0.0.1	UDP	60 45014 → 9000 Len=18
24	31.980056	127.0.0.1	127.0.0.1	UDP	47 9000 → 59375 Len=5
25	33.262829	127.0.0.1	127.0.0.1	UDP	60 45014 → 9000 Len=18
26	33.263054	127.0.0.1	127.0.0.1	UDP	47 9000 → 59375 Len=5
27	34.426060	127.0.0.1	127.0.0.1	UDP	60 45014 → 9000 Len=18
28	34.426276	127.0.0.1	127.0.0.1	UDP	47 9000 → 59375 Len=5
29	36.328769	127.0.0.1	127.0.0.1	UDP	60 60845 → 9000 Len=18
30	36.328977	127.0.0.1	127.0.0.1	UDP	47 9000 → 45359 Len=5
31	37.713645	127.0.0.1	127.0.0.1	UDP	60 60845 → 9000 Len=18
32	37.713834	127.0.0.1	127.0.0.1	UDP	47 9000 → 45359 Len=5
33	38.954624	127.0.0.1	127.0.0.1	UDP	60 60845 → 9000 Len=18
34	38.954790	127.0.0.1	127.0.0.1	UDP	47 9000 → 45359 Len=5
35	41.411751	127.0.0.1	127.0.0.1	UDP	60 45014 → 9000 Len=18

Aquí se ven los últimos mensajes de los publicadores, que incluyen los cierres de cada tema.

El tráfico termina con la misma estructura, mostrando los últimos reenvíos del bróker a los dos suscriptores.

Esto confirma que todos los mensajes se entregaron y que la comunicación se mantuvo estable hasta el final.

36	41.412152	127.0.0.1	127.0.0.1	UDP	47 9000 → 59375 Len=5
37	43.001986	127.0.0.1	127.0.0.1	UDP	60 45014 → 9000 Len=18
38	43.002192	127.0.0.1	127.0.0.1	UDP	47 9000 → 59375 Len=5
39	44.488273	127.0.0.1	127.0.0.1	UDP	60 45014 → 9000 Len=18
40	44.488482	127.0.0.1	127.0.0.1	UDP	47 9000 → 59375 Len=5
41	46.357405	127.0.0.1	127.0.0.1	UDP	60 60845 → 9000 Len=18
42	46.357611	127.0.0.1	127.0.0.1	UDP	47 9000 → 45359 Len=5
43	47.858792	127.0.0.1	127.0.0.1	UDP	60 45014 → 9000 Len=18
44	47.859033	127.0.0.1	127.0.0.1	UDP	47 9000 → 59375 Len=5

En general, el sistema con UDP mostró un comportamiento estable, con baja carga en el bróker y entrega rápida de los mensajes. A diferencia de TCP, no se observa intercambio de control ni establecimiento de conexión, lo que hace el envío más ágil pero sin garantía de entrega.

Comparación del desempeño de TCP y UDP

	TCP	UDP
CONFIABILIDAD	TCP garantiza la entrega de los paquetes mediante el uso de confirmaciones (ACK), retransmisiones automáticas y control de errores.	UDP no garantiza que los mensajes lleguen. En caso de presentarse pérdidas, no hay retransmisión.
ORDEN ENTREGA	TCP usa un mecanismo de numeración para los segmentos siguiendo los números de secuencia, manteniendo el orden.	UDP no numera los datagramas y no existe control del orden de llegada. Por lo que los paquetes podrían llegar duplicados o desordenados.
PÉRDIDA DE MENSAJES	TCP reenvía automáticamente los paquetes perdidos si no recibe el ACK dentro de la ventana de tiempo establecida.	En UDP existe la posibilidad de que sus paquetes se pierdan sin que estos notifiquen al emisor o receptor, y sin retrasmisión de estos.
OVERHEAD CABECERAS	TCP usa una cabecera de mínimo 20 bytes (sin opciones), pero esta puede aumentar hasta 60 bytes (con opciones). Incluye control de flujo y congestión de paquetes.	UDP usa una cabecera de 8 bytes sin campos de control ni confirmación.

TCP: Mayor confiabilidad y orden, y sin pérdida de mensajes; sin embargo, es más tardado y su cabecera es mucho más grande.

UDP: Menor cabecera y latencia; sin embargo, podría haber una pérdida de paquetes ya que no garantiza confiabilidad ni orden en los paquetes.

Preguntas de análisis

- **¿Qué ocurriría si en lugar de dos publicadores (partidos transmitidos) hubiera cien partidos simultáneos? ¿Cómo impactaría esto en el desempeño del bróker bajo TCP y bajo UDP?**

Con TCP, el bróker mantendría 100 conexiones activas y aumentaría el consumo de CPU y memoria para mantener el manejo de los buffers, el control de flujo y las confirmaciones.

Con UDP, no existe overhead y la carga es menor, pero se pierde el orden y confiabilidad de los paquetes, sería más complejo reconstruir el flujo correcto de los mensajes.

- **Si un gol se envía como mensaje desde el publicador y un suscriptor no lo recibe en UDP, ¿qué implicaciones tendría para la aplicación real? ¿Por qué TCP maneja mejor este escenario?**

  Si el mensaje se pierde en un sistema usando el protocolo UDP no es posible recuperarlo y el suscriptor no podría revisarlo nunca. TCP maneja mejor este escenario porque si el suscriptor no recibe el mensaje reenviaría y, aunque más tardado, el suscriptor tendría la posibilidad de verlo.

- **En un escenario en vivo de partidos, ¿qué protocolo (TCP o UDP) resultaría más adecuado? Justifique con base en los resultados de la práctica.**

  UDP sería más adecuado. En la práctica se comprobó que transmite más rápido y con menor consumo de recursos, lo cual es clave en un evento en vivo. Aunque puede perder algunos mensajes, su baja latencia permite que las actualizaciones lleguen casi al instante a todos los usuarios.

- **Compare el overhead observado en las capturas de Wireshark entre TCP y UDP. ¿Cuál protocolo introduce más cabeceras por mensaje? ¿Cómo influye esto en la eficiencia?**
  TCP introduce más cabeceras por mensaje, ya que incluye control de conexión y confirmaciones. En las capturas de Wireshark se vio que UDP tiene cabeceras más pequeñas, lo que reduce el overhead y hace la transmisión más eficiente.

- **Si el marcador de un partido llega desordenado en UDP (por ejemplo, primero se recibe el 2-1 y luego el 1-1), ¿qué efectos tendría en la experiencia del usuario? ¿Cómo podría solucionarse este problema a nivel de aplicación?**

  El usuario vería información confusa, porque los eventos llegarían en otro orden. Para evitarlo, la aplicación puede numerar los mensajes o incluir la hora de envío, y así mostrar los datos según su secuencia real, aunque lleguen desordenados.

- **¿Cómo cambia el desempeño del sistema cuando aumenta el número de suscriptores interesados en un mismo partido? ¿Qué diferencias se observaron entre TCP y UDP en este aspecto?**
  Cuando aumentan los suscriptores, con UDP el bróker mantiene buen desempeño porque solo reenvía el mismo mensaje a varias direcciones sin crear conexiones. Con TCP el

rendimiento baja, ya que necesita manejar una conexión separada por cliente. En el laboratorio se notó que UDP soporta más usuarios con menos carga.

- **¿Qué sucede si el bróker se detiene inesperadamente? ¿Qué diferencia hay entre TCP y UDP en la capacidad de recuperación de la sesión?**

  Si el bróker se detiene, con UDP los mensajes se pierden y los clientes no se dan cuenta, porque no hay conexión establecida. Con TCP, las conexiones se cierran y los clientes pueden detectar la caída y reconectarse. Por eso, TCP permite recuperar la sesión más fácilmente que UDP.

- **¿Cómo garantizar que todos los suscriptores reciban en el mismo instante las actualizaciones críticas (por ejemplo, un gol)? ¿Qué protocolo facilita mejor esta sincronización y por qué?**

  Para que todos reciban la actualización al mismo tiempo se puede usar UDP, porque envía el mensaje a todos los suscriptores sin esperar confirmación. Es el más rápido para difundir información simultánea. TCP no sirve igual porque envía por turnos y depende de cada conexión, lo que genera retrasos distintos.

- **Analice el uso de CPU y memoria en el bróker cuando maneja múltiples conexiones TCP frente al manejo de datagramas UDP. ¿Qué diferencias encontró?**

  Con UDP el uso de CPU y memoria fue menor, porque el bróker no mantiene conexiones activas. Solo recibe y reenvía mensajes. En cambio, con TCP necesita abrir y sostener varias conexiones, lo que aumenta el consumo de recursos. Por eso, en el laboratorio el bróker con UDP resultó más liviano y rápido.

- **Si tuviera que diseñar un sistema real de transmisión de actualizaciones de partidos de fútbol para millones de usuarios, ¿elegiría TCP, UDP o una combinación de ambos? Justifique con base en lo observado en el laboratorio.**

  Usaría una combinación de ambos. En el laboratorio se vio que UDP es más rápido y adecuado para las actualizaciones en tiempo real, mientras que TCP ofrece fiabilidad cuando se necesita asegurar que la información llegue completa. Mezclar los dos permitiría mantener la velocidad sin perder datos importantes.