

**UNIVERSIDAD DE LOS ANDES DEPARTAMENTO DE INGENIERIA DE SISTEMAS Y
COMPUTACIÓN**



LABORATORIO: INTRODUCCIÓN A REDES DE DATOS

ISIS 3204 – INFRESTRUCTURA DE COMUNICACIONES

Nathalia Alexandra Quiroga Alfaro

Grupo 2

Jerónimo Franco 202222204

Sebastian Palma 202222498

Miguel Castillo 201633992

2025-10

Contenido

Análisis	3
Comparar el desempeño de TCP y UDP	6
Evidencias	7
1. TCP.....	7
2. UDP.....	12
Preguntas	17

Análisis

1. En TCP: ¿los mensajes llegan completos y en orden? ¿Cómo maneja TCP la confiabilidad y el control de flujo?

Primero que todo como nos podemos dar cuenta los mensajes si llegan en orden. El tiempo que se muestra en cada uno de ellos demuestran él los mensajes le llegaron en orden al subscriptor.

```
msanti@PortatilSanti:/mnt/c/Users/msant/Downloads/LabTresRedes/TCP$ make run-subscriber
./subscriber 1
[SUBSCRIBER] Conectado al broker en 127.0.0.1:9000
[SUBSCRIBER] Suscrito al topic '1'
[SUBSCRIBER] Mensaje recibido: [18:20:23] 1: Inicio del partido: Equipo A vs Equipo B

[SUBSCRIBER] Mensaje recibido: [18:20:25] 1: Gol de Equipo A al minuto 5

[SUBSCRIBER] Mensaje recibido: [18:20:27] 1: Tiro libre a favor de Equipo B

[SUBSCRIBER] Mensaje recibido: [18:20:29] 1: Atajada del arquero de Equipo A

[SUBSCRIBER] Mensaje recibido: [18:20:34] 1: Tarjeta amarilla para jugador 8 de Equipo B

[SUBSCRIBER] Mensaje recibido: [18:20:36] 1: Gol de cabeza de Equipo B al minuto 30

[SUBSCRIBER] Mensaje recibido: [18:20:38] 1: Cambio en Equipo A: entra jugador 7 por jugador 11

[SUBSCRIBER] Mensaje recibido: [18:20:40] 1: Disparo desviado del delantero de Equipo A

[SUBSCRIBER] Mensaje recibido: [18:20:42] 1: Penalti a favor de Equipo B

[SUBSCRIBER] Mensaje recibido: [18:20:44] 1: Gol de penalti del jugador 9 de Equipo B

[SUBSCRIBER] Mensaje recibido: [18:20:46] 1: Lesión del defensa central de Equipo A

[SUBSCRIBER] Mensaje recibido: [18:20:48] 1: Gol de tiro libre de Equipo A al minuto 70

[SUBSCRIBER] Mensaje recibido: [18:20:50] 1: Tarjeta roja para el numero 5 de Equipo B
```

TCP es confiable porque usa el método de three way hand check para establecer una conexión segura. Como nos podemos dar cuenta en la imagen se solicita una conexión con [SYN] y con el servidor responde con [SYN ACK] el cliente responde con un [ACK] y con esto se aseguran de una red confiable

126	1.170441	kubernetes.docker.i...	kubernetes.docker.i...	TCP	56	irdmi(8000) → 57498	[SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
129	1.170948	kubernetes.docker.i...	kubernetes.docker.i...	TCP	44	irdmi(8000) → 57498	[ACK] Seq=1 Ack=14 Win=65280 Len=0
291	4.201396	kubernetes.docker.i...	kubernetes.docker.i...	TCP	56	irdmi(8000) → 57499	[SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
294	4.202114	kubernetes.docker.i...	kubernetes.docker.i...	TCP	44	irdmi(8000) → 57499	[ACK] Seq=1 Ack=14 Win=65280 Len=0
340	5.695606	kubernetes.docker.i...	kubernetes.docker.i...	TCP	56	irdmi(8000) → 57500	[SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
343	5.696317	kubernetes.docker.i...	kubernetes.docker.i...	TCP	44	irdmi(8000) → 57500	[ACK] Seq=1 Ack=13 Win=65280 Len=0
345	5.700430	kubernetes.docker.i...	kubernetes.docker.i...	TCP	44	irdmi(8000) → 57500	[ACK] Seq=1 Ack=75 Win=65280 Len=0
346	5.700732	kubernetes.docker.i...	kubernetes.docker.i...	TCP	100	irdmi(8000) → 57498	[PSH, ACK] Seq=1 Ack=14 Win=65280 Len=56
457	6.881790	kubernetes.docker.i...	kubernetes.docker.i...	TCP	56	irdmi(8000) → 57501	[SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
460	6.882464	kubernetes.docker.i...	kubernetes.docker.i...	TCP	44	irdmi(8000) → 57501	[ACK] Seq=1 Ack=13 Win=65280 Len=0

Para el control de flujo el cliente le informa al servidor cuanto espacio tiene en su búfer o más bien conocido como el tamaño de la ventana del respeto, en este caso es de 255 se ve en window.

```

Source Port: irdmi (8000)
Destination Port: 57500 (57500)
[Stream index: 11]
[Stream Packet Number: 11]
> [Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 746618375
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 176 (relative ack number)
Acknowledgment number (raw): 685549176
0101 .... = Header Length: 20 bytes (5)
> Flags: 0x010 (ACK)
Window: 255
[Calculated window size: 65280]
[Window size scaling factor: 256]
Checksum: 0x371a [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [Timestamps]
> [SEQ/ACK analysis]

```

2. En UDP: ¿qué evidencias hay de pérdida o desorden en los mensajes?

Al revisar los mensajes enviados por los publicadores (Izquierda) y recibidos por los suscriptores (Derecha) en las terminales, se puede observar que no hay

pérdida ni desorden en los mensajes transmitidos.

```
Símbolo del sistema x + - □ x
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Tarjeta amarilla para jugador 8 de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Gol de cabeza de Equipo B al minuto 30
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Cambio en Equipo A: entra jugador 7 por jugador 11
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Disparo desviado del delantero de Equipo A
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Penalti a favor de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Gol de penalti del jugador 9 de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Lesion del defensa central de Equipo A
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Gol de tiro libre de Equipo A al minuto 70
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Tarjeta roja para el numero 5 de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|17:50:23|Final del partido: Equipo A 2 / 2 Equipo B
[PUBLISHER] Fin del archivo Partido1.txt.

Símbolo del sistema x + - □ x
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Gol de tiro libre de Equipo D al minuto 25
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Cambio en Equipo C: entra jugador 12 por jugador 6
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Centro de Equipo D despejado por la defensa
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Gol de cabeza de Equipo D al minuto 50
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Empate de Equipo C con gol al minuto 55
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Tarjeta amarilla para el arquero de Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Lesion del jugador 8 de Equipo C
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Penalti a favor de Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Gol de penalti de jugador 9 de Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|17:50:24|Final del partido: Equipo C 2 / 3 Equipo D
[PUBLISHER] Fin del archivo Partido2.txt.

Símbolo del sistema - .\subco x + - □ x
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\subscriber_udp.exe 127.0.0.1 5000 "Equipo A vs Equipo B"
[SUBSCRIBER] Suscrito al partido Equipo A vs Equipo B
[SUBSCRIBER] Mensaje recibido: Inicio del partido: Equipo A vs Equipo B
[SUBSCRIBER] Mensaje recibido: Gol de Equipo A al minuto 5
[SUBSCRIBER] Mensaje recibido: Tiro libre a favor de Equipo B
[SUBSCRIBER] Mensaje recibido: Atajada del arquero de Equipo A
[SUBSCRIBER] Mensaje recibido: Tarjeta amarilla para jugador 8 de Equipo B
[SUBSCRIBER] Mensaje recibido: Gol de cabeza de Equipo B al minuto 30
[SUBSCRIBER] Mensaje recibido: Cambio en Equipo A: entra jugador 7 por jugador 11
[SUBSCRIBER] Mensaje recibido: Disparo desviado del delantero de Equipo A
[SUBSCRIBER] Mensaje recibido: Penalti a favor de Equipo B
[SUBSCRIBER] Mensaje recibido: Gol de penalti del jugador 9 de Equipo B
[SUBSCRIBER] Mensaje recibido: Lesion del defensa central de Equipo A
[SUBSCRIBER] Mensaje recibido: Gol de tiro libre de Equipo A al minuto 70
[SUBSCRIBER] Mensaje recibido: Tarjeta roja para el numero 5 de Equipo B
[SUBSCRIBER] Mensaje recibido: Final del partido: Equipo A 2 / 2 Equipo B

Símbolo del sistema - .\subco x + - □ x
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\subscriber_udp.exe 127.0.0.1 5000 "Equipo C vs Equipo D"
[SUBSCRIBER] Suscrito al partido Equipo C vs Equipo D
[SUBSCRIBER] Mensaje recibido: Comienza el partido: Equipo C vs Equipo D
[SUBSCRIBER] Mensaje recibido: Gol del Equipo C al minuto 3
[SUBSCRIBER] Mensaje recibido: Remate desviado del delantero de Equipo D
[SUBSCRIBER] Mensaje recibido: Tarjeta amarilla para jugador 14 de Equipo C
[SUBSCRIBER] Mensaje recibido: Gol de tiro libre de Equipo D al minuto 25
[SUBSCRIBER] Mensaje recibido: Cambio en Equipo C: entra jugador 12 por jugador 6
[SUBSCRIBER] Mensaje recibido: Centro de Equipo D despejado por la defensa
[SUBSCRIBER] Mensaje recibido: Gol de cabeza de Equipo D al minuto 50
[SUBSCRIBER] Mensaje recibido: Empate de Equipo C con gol al minuto 55
[SUBSCRIBER] Mensaje recibido: Tarjeta amarilla para el arquero de Equipo D
[SUBSCRIBER] Mensaje recibido: Lesion del jugador 8 de Equipo C
[SUBSCRIBER] Mensaje recibido: Penalti a favor de Equipo D
[SUBSCRIBER] Mensaje recibido: Gol de penalti de jugador 9 de Equipo D
[SUBSCRIBER] Mensaje recibido: Final del partido: Equipo C 2 / 3 Equipo D
```

Ahora, si miramos las capturas de Wireshark podemos detallar el tráfico generado por el protocolo podemos observar cada momento de la transmisión. Aquí observamos como se conectan los suscriptores al partido notificándole al bróker.

udp.port == 5000						
No.	Time	Source	Destination	Protocol	Length	Info
109	2.177947	localhost.localdomain	localhost.localdomain	UDP	63	56747 → complex-main(5000) Len=31
130	3.527980	localhost.localdomain	localhost.localdomain	UDP	63	56749 → complex-main(5000) Len=31
Frame 109: Packet, 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface 0						
Internet Protocol Version 4, Src: localhost.localdomain (127.0.0.1), Dst: localhost.localdomain (127.0.0.1)						
User Datagram Protocol, Src Port: 56747 (56747), Dst Port: complex-main (5000)						
Data (31 bytes)						
Data: 535542534352494245527c45717569706f20412076732045717569706f2042 [Length: 31]						

Luego vemos como el publicador envía el mensaje del inicio del partido y el bróker lo reenvía al suscriptor interesado.

udp.port == 5000						
No.	Time	Source	Destination	Protocol	Length	Info
109	2.177947	localhost.localdomain	localhost.localdomain	UDP	63	56747 → complex-main(5000) Len=31
130	3.527980	localhost.localdomain	localhost.localdomain	UDP	63	56749 → complex-main(5000) Len=31
187	5.123053	localhost.localdomain	localhost.localdomain	UDP	112	56750 → complex-main(5000) Len=80
188	5.123768	localhost.localdomain	localhost.localdomain	UDP	72	complex-main(5000) → 56747 Len=40
Frame 188: Packet, 72 bytes on wire (576 bits), 72 bytes captured (576 bits) on interface 0						
Internet Protocol Version 4, Src: localhost.localdomain (127.0.0.1), Dst: localhost.localdomain (127.0.0.1)						
User Datagram Protocol, Src Port: complex-main (5000), Dst Port: 56747 (56747)						
Data (40 bytes)						
Data: 4966963696f2064656c207061272469646f3a2045717569706f20412076732045717569706f20 [Length: 40]						

Posterior a esto, observamos como el publicador empieza a mandar cada mensaje realizado con el partido hasta su final y el bróker reenvía todo esto al suscriptor interesado.

udp.port == 5000						
No.	Time	Source	Destination	Protocol	Length	Info
109	2.177947	localhost.localdomain	localhost.localdomain	UDP	63	56747 → complex-main(5000) Len=31
130	3.527980	localhost.localdomain	localhost.localdomain	UDP	63	56749 → complex-main(5000) Len=31
187	5.123053	localhost.localdomain	localhost.localdomain	UDP	112	56750 → complex-main(5000) Len=80
188	5.123768	localhost.localdomain	localhost.localdomain	UDP	72	complex-main(5000) → 56747 Len=40
189	5.123784	localhost.localdomain	localhost.localdomain	UDP	99	56750 → complex-main(5000) Len=67
190	5.124073	localhost.localdomain	localhost.localdomain	UDP	102	56750 → complex-main(5000) Len=70
191	5.124254	localhost.localdomain	localhost.localdomain	UDP	59	complex-main(5000) → 56747 Len=27
192	5.124504	localhost.localdomain	localhost.localdomain	UDP	103	56750 → complex-main(5000) Len=71
193	5.124678	localhost.localdomain	localhost.localdomain	UDP	115	56750 → complex-main(5000) Len=83
194	5.124759	localhost.localdomain	localhost.localdomain	UDP	62	complex-main(5000) → 56747 Len=30
195	5.124913	localhost.localdomain	localhost.localdomain	UDP	110	56750 → complex-main(5000) Len=78
196	5.125039	localhost.localdomain	localhost.localdomain	UDP	63	complex-main(5000) → 56747 Len=31
197	5.125100	localhost.localdomain	localhost.localdomain	UDP	122	56750 → complex-main(5000) Len=90
198	5.125222	localhost.localdomain	localhost.localdomain	UDP	75	complex-main(5000) → 56747 Len=43
199	5.125316	localhost.localdomain	localhost.localdomain	UDP	114	56750 → complex-main(5000) Len=82
200	5.125356	localhost.localdomain	localhost.localdomain	UDP	70	complex-main(5000) → 56747 Len=38
201	5.125532	localhost.localdomain	localhost.localdomain	UDP	82	complex-main(5000) → 56747 Len=50
202	5.125577	localhost.localdomain	localhost.localdomain	UDP	74	complex-main(5000) → 56747 Len=42
203	5.125685	localhost.localdomain	localhost.localdomain	UDP	99	56750 → complex-main(5000) Len=67
204	5.125845	localhost.localdomain	localhost.localdomain	UDP	59	complex-main(5000) → 56747 Len=27
205	5.125896	localhost.localdomain	localhost.localdomain	UDP	112	56750 → complex-main(5000) Len=80
206	5.126050	localhost.localdomain	localhost.localdomain	UDP	72	complex-main(5000) → 56747 Len=40
207	5.126090	localhost.localdomain	localhost.localdomain	UDP	110	56750 → complex-main(5000) Len=78
208	5.126241	localhost.localdomain	localhost.localdomain	UDP	70	complex-main(5000) → 56747 Len=38
209	5.126282	localhost.localdomain	localhost.localdomain	UDP	114	56750 → complex-main(5000) Len=82
210	5.126449	localhost.localdomain	localhost.localdomain	UDP	74	complex-main(5000) → 56747 Len=42
211	5.126480	localhost.localdomain	localhost.localdomain	UDP	113	56750 → complex-main(5000) Len=81
212	5.126597	localhost.localdomain	localhost.localdomain	UDP	114	56750 → complex-main(5000) Len=82

Frame 214: Packet, 74 bytes on wire (592 bits), 74 bytes captured (592 bits)	0000	02 00 00 00 45 00 00 46 f4 7e 00 00 80 11 00 00	... E F ...
Null/Loopback	0010	7f 00 00 01 7f 00 00 01 13 88 dd ab 00 32 d9 15	... 2 ...
Internet Protocol Version 4, Src: localhost.localdomain (127.0.0.1), Dst: localhost.localdomain (127.0.0.1)	0020	46 69 6e 61 6c 20 64 65 6c 20 70 61 72 74 69 64	Final de 1 partid
User Datagram Protocol, Src Port: complex-main (5000), Dst Port: 56747 (56747)	0030	6f 3a 20 45 71 75 69 70 6f 20 41 20 32 20 2f 28	o: Equip o A 2 /
Data (42 bytes)	0040	32 20 45 71 75 69 70 6f 20 42	2 Equipo B

Analizando las capturas de Wireshark y las capturas de los mensajes enviados, podemos notar que ningún mensaje se perdió en la transmisión. Sin embargo, si existían ligeros retrasos temporales entre que el publicador enviaba la información al bróker y este reenviaba al suscriptor, pero nada lo suficientemente significativo como para generar desorden en los mensajes. Esto tiene sentido, puesto que el sistema no cuenta con suficientes publicadores o suscriptores enviando o recibiendo información al mismo tiempo para sobrecargar el bróker.

3. ¿Qué diferencias observa en el manejo de la conexión entre ambos protocolos?

Comparar el desempeño de TCP y UDP

	UDP	TCP
Confiabilidad	No es confiable	Muy confiable
Orden de entrega	No, le importa el orden	Entrega en orden
Pérdida de mensajes	Puede perder mensajes dependiendo del camino que tomen los archivos	Si el mensaje se pierde se vuelve a enviar
Overhead de cabeceras/protocolo	Es pequeña	Es grande

Evidencias

1. TCP

a. Uso de librerías

En la implementación se utilizan las librerías winsock2.h y ws2tcpip.h que forman parte del API nativo de Windows (WinSock). Estas librerías son nativas del sistema operativo Windows, diseñados específicamente para permitir el uso de sockets en entornos C/C++. WinSock provee las funciones necesarias para la creación, envío y recepción de datagramas mediante sockets en el protocolo TCP, así como la gestión de direcciones IP y puertos.

b. Implementación

Lo primer que hacemos tanto en el publicador, borker y subcriptor es crear un socket TCP. Lo que se hace es crear un socket donde definimos los parámetros “AF_INET” para usar IPv4, “SOCK_STREAM” que define que es un socket orientado a conexión y después el protocolo que en este caso es TCP y se define como “IPPROTO_TCP”

```
// Crear socket TCP
sock_fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (sock_fd < 0) {
    perror("Error al crear socket");
    return EXIT_FAILURE;
}
```

Luego vemos como el publicador y el subcriptor se conecta al bróker antes de empezar a enviar o recibir datos, esto por la naturaleza de TCP. Esto se hace con la función “connect” dándole como parámetros el socket que se creo anteriormente, la ip y el puerto en el cual está trabajando el bróker. Después el tamaño de ip más el puerto. Si no se pueden conectar, manada un error y termina de correr el publicador o suscriptor.

```
// Conectar al broker
if (connect(sock_fd, (struct sockaddr *)&broker_addr, sizeof(broker_addr)) == SOCKET_ERROR) {
    perror("Error al conectar con el broker");
    closesocket(sock_fd);
    WSACleanup();
    return EXIT_FAILURE;
}
```

Cuando el bróker se enciende, este está escuchando constantemente para saber cuándo se quiere conectar un nuevo suscriptor o publicador. Cuando se solicita una nueva conexión el bróker escucha esta solicitud

o sea `FD_ISSET()` es verdadero. Cuando este paso acepta la conexión y empieza ya se pueden hacer envío de datos. Entre este paso y el anterior se hace el three way hand check para tener una conexión confiable y estable. Además, en este paso se identifica si es un cliente o un suscriptor.

```
if (FD_ISSET(server_fd, &read_fds)) {
    int new_socket = accept(server_fd, (struct sockaddr *)&client_addr, &addrlen);
    if (new_socket >= 0) {
        registrar_cliente(new_socket);
    }
}
```

Luego vemos como el publicador lee los mensajes de cada partido que serán enviados al bróker. El uso de `fopen()` permite abrir el archivo de texto que contiene los mensajes en modo lectura, de esta manera se leen los archivos `Partido1.txt` y `Partido2.txt` para él envío de información y como es que es enviada al bróker.

```
while (fgets(mensaje, sizeof(mensaje), file)) {
    mensaje[strcspn(mensaje, "\n")] = '\0';

    time_t t = time(NULL);
    struct tm *tm_info = localtime(&t);
    char hora[9];
    strftime(hora, sizeof(hora), "%H:%M:%S", tm_info);

    char mensaje_limpio[900];
    strncpy(mensaje_limpio, mensaje, sizeof(mensaje_limpio) - 1);
    mensaje_limpio[sizeof(mensaje_limpio) - 1] = '\0';

    snprintf(buffer_envio, sizeof(buffer_envio), "PUBLISHER%s%s%s\n", partido, hora, mensaje_limpio);

    // Enviar mensaje
    if (write(sock_fd, buffer_envio, strlen(buffer_envio)) < 0) {
        perror("Error al enviar mensaje");
        break;
    }

    printf("[PUBLISHER] Mensaje enviado: %s\n", buffer_envio);
    sleep(5);
}

printf("[PUBLISHER] Fin del archivo '%s'. Cerrando conexión.\n", archivo);
```

Luego visualizamos la parte más fundamental del broker. Se usa la función `red()` para recibir los paquetes de cada una de las partes. Al mismo tiempo, se recibe el mensaje y se imprime en el broker. La función `renviar a subscribers()` en el `if` envía los mensajes únicamente a los suscriptores interesados sin establecer una conexión por cada cliente.


```

for (int i = 0; i < MAX_CLIENTS; i++) {
    int fd = publisher_sockets[i];
    if (fd > 0 && FD_ISSET(fd, &read_fds)) {
        char buffer[BUFFER_SIZE];
        int bytes = read(fd, buffer, BUFFER_SIZE - 1);
        if (bytes <= 0) {
            close(fd);
            publisher_sockets[i] = 0;
            printf("[BROKER] Publisher desconectado\n");
        } else {
            buffer[bytes] = '\0';
            printf("[BROKER] Mensaje recibido: %s\n", buffer);

            char *tipo = strtok(buffer, "|");
            char *topic = strtok(NULL, "|");
            char *hora = strtok(NULL, "|");
            char *mensaje = strtok(NULL, "");
            if (tipo && topic && hora && mensaje) {
                char mensaje_final[BUFFER_SIZE];
                snprintf(mensaje_final, sizeof(mensaje_final), "[%s] %s: %s\n", hora, topic, mensaje);
                reenviar_a_subscribers(topic, mensaje_final);
            }
        }
    }
}
}

```

Finalmente es suscriptor está escuchando constantemente a todos los mensajes que el broker envía y los imprime por consola esto se hace con las funciones read y printf.

```

// Escuchar mensajes del broker
while (1) {
    int bytes = read(sock_fd, buffer, BUFFER_SIZE - 1);
    if (bytes <= 0) {
        printf("Conexión cerrada por el broker\n");
        break;
    }

    buffer[bytes] = '\0';
    printf("[SUBSCRIBER] Mensaje recibido: %s\n", buffer);
}

```

c. Evidencia de funcionamiento

Lo primero que vamos a hacer es correr el bróker.

```

C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\TCP>.\broker_tcp.exe
[BROKER] Escuchando en puerto 8000...

```

Después el suscriptor y vemos que en el bróker sale un mensaje que se ha conectado correctamente.

```

C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\TCP>.\broker_tcp.exe
[BROKER] Escuchando en puerto 8000...
[BROKER] Subscriber conectado: socket 244, topic '1'

```

```

C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\TCP>.\subscriber_tcp.exe 1
[SUBSCRIBER] Conectado al broker en 127.0.0.1:8000
[SUBSCRIBER] Suscrito al topic '1'

```

Finalmente, podemos empezar a correr el publicador y podemos enviar todos los datos.

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\TCP>.\publisher_tcp.exe Partido2.txt 2
[PUBLISHER] Conectado al broker en 127.0.0.1:8000
[PUBLISHER] Enviando mensajes del archivo 'Partido2.txt' para el partido '2'
[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:08|Comienza el partido: Equipo C vs Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:10|Gol del Equipo C al minuto 3

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:12|Remate desviado del delantero de Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:14|Tarjeta amarilla para jugador 14 de Equipo C

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:16|Gol de tiro libre de Equipo D al minuto 25

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:18|Cambio en Equipo C: entra jugador 12 por jugador 6

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:20|Centro de Equipo D despejado por la defensa

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:22|Gol de cabeza de Equipo D al minuto 50

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:24|Empate de Equipo C con gol al minuto 55

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:26|Tarjeta amarilla para el arquero de Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:28|Lesion del jugador 8 de Equipo C

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:30|Penalti a favor de Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:32|Gol de penalti de jugador 9 de Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:34|Final del partido: Equipo C 2 / 3 Equipo D

[PUBLISHER] Fin del archivo 'Partido2.txt'. Cerrando conexi|n.
```

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\TCP>.\publisher_tcp.exe Partido2.txt 2
[PUBLISHER] Conectado al broker en 127.0.0.1:8000
[PUBLISHER] Enviando mensajes del archivo 'Partido2.txt' para el partido '2'
[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:08|Comienza el partido: Equipo C vs Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:10|Gol del Equipo C al minuto 3

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:12|Remate desviado del delantero de Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:14|Tarjeta amarilla para jugador 14 de Equipo C

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:16|Gol de tiro libre de Equipo D al minuto 25

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:18|Cambio en Equipo C: entra jugador 12 por jugador 6

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:20|Centro de Equipo D despejado por la defensa

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:22|Gol de cabeza de Equipo D al minuto 50

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:24|Empate de Equipo C con gol al minuto 55

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:26|Tarjeta amarilla para el arquero de Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:28|Lesion del jugador 8 de Equipo C

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:30|Penalti a favor de Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:32|Gol de penalti de jugador 9 de Equipo D

[PUBLISHER] Mensaje enviado: PUBLISHER|2|20:10:34|Final del partido: Equipo C 2 / 3 Equipo D

[PUBLISHER] Fin del archivo 'Partido2.txt'. Cerrando conexi|n.
```

Finalmente, se puede ver que los mensajes llegan al suscriptor.

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\TCP>.\subscriber_tcp.exe 1
[SUBSCRIBER] Conectado al broker en 127.0.0.1:8000
[SUBSCRIBER] Suscrito al topic '1'
[SUBSCRIBER] Mensaje recibido: [20:10:07] 1: Inicio del partido: Equipo A vs Equipo B

[SUBSCRIBER] Mensaje recibido: [20:10:09] 1: Gol de Equipo A al minuto 5

[SUBSCRIBER] Mensaje recibido: [20:10:11] 1: Tiro libre a favor de Equipo B

[SUBSCRIBER] Mensaje recibido: [20:10:13] 1: Atajada del arquero de Equipo A

[SUBSCRIBER] Mensaje recibido: [20:10:15] 1: Tarjeta amarilla para jugador 8 de Equipo B

[SUBSCRIBER] Mensaje recibido: [20:10:17] 1: Gol de cabeza de Equipo B al minuto 30

[SUBSCRIBER] Mensaje recibido: [20:10:19] 1: Cambio en Equipo A: entra jugador 7 por jugador 11

[SUBSCRIBER] Mensaje recibido: [20:10:21] 1: Disparo desviado del delantero de Equipo A

[SUBSCRIBER] Mensaje recibido: [20:10:23] 1: Penalti a favor de Equipo B

[SUBSCRIBER] Mensaje recibido: [20:10:25] 1: Gol de penalti del jugador 9 de Equipo B

[SUBSCRIBER] Mensaje recibido: [20:10:27] 1: Lesi|n del defensa central de Equipo A

[SUBSCRIBER] Mensaje recibido: [20:10:29] 1: Gol de tiro libre de Equipo A al minuto 70

[SUBSCRIBER] Mensaje recibido: [20:10:31] 1: Tarjeta roja para el numero 5 de Equipo B

[SUBSCRIBER] Mensaje recibido: [20:10:33] 1: Final del partido: Equipo A 2 / 2 Equipo B

C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\TCP>.\subscriber_tcp.exe 2
[SUBSCRIBER] Conectado al broker en 127.0.0.1:8000
[SUBSCRIBER] Suscrito al topic '2'
[SUBSCRIBER] Mensaje recibido: [20:10:08] 2: Comienza el partido: Equipo C vs Equipo D

[SUBSCRIBER] Mensaje recibido: [20:10:10] 2: Gol del Equipo C al minuto 3

[SUBSCRIBER] Mensaje recibido: [20:10:12] 2: Remate desviado del delantero de Equipo D

[SUBSCRIBER] Mensaje recibido: [20:10:14] 2: Tarjeta amarilla para jugador 14 de Equipo C

[SUBSCRIBER] Mensaje recibido: [20:10:16] 2: Gol de tiro libre de Equipo D al minuto 25

[SUBSCRIBER] Mensaje recibido: [20:10:18] 2: Cambio en Equipo C: entra jugador 12 por jugador 6

[SUBSCRIBER] Mensaje recibido: [20:10:20] 2: Centro de Equipo D despejado por la defensa

[SUBSCRIBER] Mensaje recibido: [20:10:22] 2: Gol de cabeza de Equipo D al minuto 50

[SUBSCRIBER] Mensaje recibido: [20:10:24] 2: Empate de Equipo C con gol al minuto 55

[SUBSCRIBER] Mensaje recibido: [20:10:26] 2: Tarjeta amarilla para el arquero de Equipo D

[SUBSCRIBER] Mensaje recibido: [20:10:28] 2: Lesion del jugador 8 de Equipo C

[SUBSCRIBER] Mensaje recibido: [20:10:30] 2: Penalti a favor de Equipo D

[SUBSCRIBER] Mensaje recibido: [20:10:32] 2: Gol de penalti de jugador 9 de Equipo D

[SUBSCRIBER] Mensaje recibido: [20:10:34] 2: Final del partido: Equipo C 2 / 3 Equipo D
```

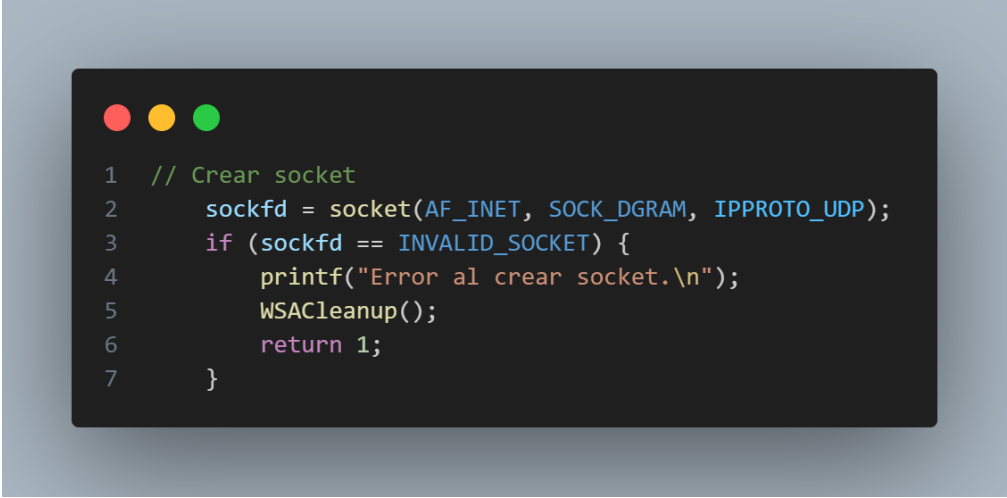
2. UDP

a. Uso de librerías

En la implementación se utilizan las librerías `winsock2.h` y `ws2tcpip.h` que forman parte del API nativo de Windows (WinSock). Estas librerías son nativas del sistema operativo Windows, diseñados específicamente para permitir el uso de sockets en entornos C/C++. WinSock provee las funciones necesarias para la creación, envío y recepción de datagramas mediante sockets en el protocolo UDP, así como la gestión de direcciones IP y puertos.

b. Implementación

Inicialmente podemos ver cómo es la creación del socket en todas las partes del modelo, es la parte fundamental permite la comunicación entre el publicador, el bróker y el suscriptor. La llamada a `socket ()` define el tipo de protocolo (`SOCK_DGRAM`) y el uso de UDP (`IPPROTO_UDP`).



```
1 // Crear socket
2 sockfd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
3 if (sockfd == INVALID_SOCKET) {
4     printf("Error al crear socket.\n");
5     WSACleanup();
6     return 1;
7 }
```

Luego vemos como el publicador lee los mensajes de cada partido que serán enviados al bróker. El uso de `fopen ()` permite abrir el archivo de texto que contiene los mensajes en modo lectura, de esta manera se leen los archivos `Partido1.txt` y `Partido2.txt` para el envío de información.

```

1 // Abrir archivo de mensajes
2 FILE *file = fopen(archivo, "r");
3 if (!file) {
4     printf("Error al abrir el archivo %s\n", archivo);
5     closesocket(sockfd);
6     WSACleanup();
7     return 1;
8 }

```

Inmediatamente podemos observar cómo se construye el mensaje en el publicador que será enviado al bróker. El formato "PUBLISHER|<topic>|<hora>|<mensaje>" permite tener una estructura detallada de los mensajes que se están enviando en el publicador. La función `sendto ()` envía el datagrama al destino sin necesidad de establecer una conexión, evidenciando la naturaleza no orientada a conexión de UDP.

```

1 snprintf(buffer_envio, sizeof(buffer_envio), "PUBLISHER|%s|%s|%s", topic, hora, mensaje);
2 sendto(sockfd, buffer_envio, strlen(buffer_envio), 0, (struct sockaddr*)&broker_addr, sizeof(broker_addr));

```

Luego visualizamos la parte más fundamental del broker. Se usa la función `recvfrom ()` para recibir los datagramas de cada una de las partes. Al mismo tiempo, se usa la función `strcmp ()` para distinguir el tipo de mensaje recibido (suscripción o publicación). La función `sendto ()` en el `else if` envía los mensajes únicamente a los suscriptores interesados sin establecer una conexión por cada cliente.

```

1  memset(buffer, 0, sizeof(buffer));
2  int n = recvfrom(sockfd, buffer, MAX_MSG_LEN, 0, (struct sockaddr*)&client_addr, &addr_len);
3  buffer[n] = '\0';
4
5  if (strncmp(buffer, "SUBSCRIBER", 11) == 0) {
6      char *topic = buffer + 11;
7
8      if (subscriber_count < MAX_SUBS) {
9          strcpy(subscribers[subscriber_count].topic, topic);
10         subscribers[subscriber_count].addr = client_addr;
11         subscriber_count++;
12         printf("[BROKER] Nuevo suscriptor a 'Partido %s'\n", topic);
13     }
14 }
15
16 else if (strncmp(buffer, "PUBLISHER", 10) == 0) {
17     char *topic = strtok(buffer + 10, "|");
18     char *hora = strtok(NULL, "|");
19     char *mensaje = strtok(NULL, "");
20
21     if (topic && hora && mensaje) {
22         printf("[BROKER] Publicacion recibida del partido '%s': %s\n", topic, hora, mensaje);
23
24         // Reenviar a los suscriptores interesados
25         for (int i = 0; i < subscriber_count; i++) {
26             if (strcmp(subscribers[i].topic, topic) == 0) {
27                 sendto(sockfd, mensaje, strlen(mensaje), 0, (struct sockaddr*)&subscribers[i].addr, sizeof(subscribers[i].addr));
28             }
29         }
30     }
31 }
32 }

```

Finalmente podemos ver cómo el suscriptor recibe los mensajes reenviados por el broker. La función `recvfrom()` realiza una espera pasivamente datagramas y los procesa conforme llegan, evidenciando la naturaleza asíncrona de UDP.

```

1  memset(buffer, 0, sizeof(buffer));
2  int n = recvfrom(sockfd, buffer, MAX_MSG_LEN, 0, NULL, NULL);
3  if (n > 0) {
4      buffer[n] = '\0';
5      printf("[SUBSCRIBER] Mensaje recibido: %s\n", buffer);
6  }

```

c. Evidencia de funcionamiento

Lo primero que realizamos es inicializar el bróker. Aquí podemos ver como nos indica que esta escuchando el puerto 5000 (Usado para la implementación de UDP) a la espera de suscriptores

```

C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\broker_udp.exe 5000
[BROKER] Escuchando en puerto 5000...

```

Lo siguiente que realizamos es inicializar los suscriptores. Aquí observamos como ambos suscriptores se suscribieron a diferentes partidos. Al mismo tiempo, un mensaje de confirmación del bróker de que se suscribieron correctamente

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\subscriber_udp.exe 127.0.0.1 5000 "Equipo A vs Equipo B"
[SUBSCRIBER] Suscrito al partido Equipo A vs Equipo B
```

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\subscriber_udp.exe 127.0.0.1 5000 "Equipo C vs Equipo D"
[SUBSCRIBER] Suscrito al partido Equipo C vs Equipo D
```

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\broker_udp.exe 5000
[BROKER] Escuchando en puerto 5000...
[BROKER] Nuevo subscriptor a 'Partido Equipo A vs Equipo B'
[BROKER] Nuevo subscriptor a 'Partido Equipo C vs Equipo D'
```

Por último, inicializamos los publicadores. Podemos visualizar como cada uno de los publicadores lee el archivo de cada partido y envía toda la información de este. Además, podemos ver como toda esta información pasa por el bróker y llega finalmente a cada suscriptor.

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\publisher_from_fil
e.exe 127.0.0.1 5000 "Equipo A vs Equipo B" Partido1.txt
[PUBLISHER] Enviando a 127.0.0.1:5000 informacion sobre el Partido Equipo A vs Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Inicio del partido: Equipo A vs Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Gol de Equipo A al minuto 5
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Tiro libre a favor de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Atajada del arquero de Equipo A
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Tarjeta amarilla para jugador 8 de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Gol de cabeza de Equipo B al minuto 30
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Cambio en Equipo A: entra jugador 7 por jugador 11
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Disparo desviado del delantero de Equipo A
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Penalti a favor de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Gol de penalti del jugador 9 de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Lesion del defensa central de Equipo A
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Gol de tiro libre de Equipo A al minuto 70
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Tarjeta roja para el numero 5 de Equipo B
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo A vs Equipo B|22:43:22|Final del partido: Equipo A 2 / 2 Equipo B
[PUBLISHER] Fin del archivo Partido1.txt.
```

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\publisher_from_fil
e.exe 127.0.0.1 5000 "Equipo C vs Equipo D" Partido2.txt
[PUBLISHER] Enviando a 127.0.0.1:5000 informacion sobre el Partido Equipo C vs Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Comienza el partido: Equipo C vs Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Gol del Equipo C al minuto 3
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Remate desviado del delantero de Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Tarjeta amarilla para jugador 14 de Equipo C
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Gol de tiro libre de Equipo D al minuto 25
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Cambio en Equipo C: entra jugador 12 por jugador 6
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Centro de Equipo D despejado por la defensa
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Gol de cabeza de Equipo D al minuto 50
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Empate de Equipo C con gol al minuto 55
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Tarjeta amarilla para el arquero de Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Lesion del jugador 8 de Equipo C
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Penalti a favor de Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Gol de penalti de jugador 9 de Equipo D
[PUBLISHER] Mensaje enviado: PUBLISHER|Equipo C vs Equipo D|22:38:52|Final del partido: Equipo C 2 / 3 Equipo D
[PUBLISHER] Fin del archivo Partido2.txt.
```

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTresRedes\UDP>.\broker_udp.exe 5000
[BROKER] Escuchando en puerto 5000...
[BROKER] Nuevo subscriptor a 'Partido Equipo A vs Equipo B'
[BROKER] Nuevo subscriptor a 'Partido Equipo C vs Equipo D'
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Inicio del partido: Equipo A vs Equipo B
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Gol de Equipo A al minuto 5
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Tiro libre a favor de Equipo B
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Atajada del arquero de Equipo A
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Tarjeta amarilla para jugador 8 de Equipo B
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Gol de cabeza de Equipo B al minuto 30
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Cambio en Equipo A: entra jugador 7 por jugador 11
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Disparo desviado del delantero de Equipo A
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Penalti a favor de Equipo B
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Gol de penalti del jugador 9 de Equipo B
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Lesi|n del defensa central de Equipo A
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Gol de tiro libre de Equipo A al minuto 70
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Tarjeta roja para el numero 5 de Equipo B
[BROKER] Publicacion recibida del partido 'Equipo A vs Equipo B': 22:38:52|Final del partido: Equipo A 2 / 2 Equipo B
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Comienza el partido: Equipo C vs Equipo D
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Gol del Equipo C al minuto 3
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Remate desviado del delantero de Equipo D
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Tarjeta amarilla para jugador 14 de Equipo C
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Gol de tiro libre de Equipo D al minuto 25
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Cambio en Equipo C: entra jugador 12 por jugador 6
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Centro de Equipo D despejado por la defensa
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Gol de cabeza de Equipo D al minuto 50
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Empate de Equipo C con gol al minuto 55
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Tarjeta amarilla para el arquero de Equipo D
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Lesion del jugador 8 de Equipo C
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Penalti a favor de Equipo D
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Gol de penalti de jugador 9 de Equipo D
[BROKER] Publicacion recibida del partido 'Equipo C vs Equipo D': 22:38:52|Final del partido: Equipo C 2 / 3 Equipo D
```



```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTre
sRedes\UDP>.\subscriber_udp.exe 127.0.0.1 5000 "Equipo A vs Equipo B"
[SUBSCRIBER] Suscrito al partido Equipo A vs Equipo B
[SUBSCRIBER] Mensaje recibido: Inicio del partido: Equipo A vs Equipo B
[SUBSCRIBER] Mensaje recibido: Gol de Equipo A al minuto 5
[SUBSCRIBER] Mensaje recibido: Tiro libre a favor de Equipo B
[SUBSCRIBER] Mensaje recibido: Atajada del arquero de Equipo A
[SUBSCRIBER] Mensaje recibido: Tarjeta amarilla para jugador 8 de Equipo B
[SUBSCRIBER] Mensaje recibido: Gol de cabeza de Equipo B al minuto 30
[SUBSCRIBER] Mensaje recibido: Cambio en Equipo A: entra jugador 7 por jugador 11
[SUBSCRIBER] Mensaje recibido: Disparo desviado del delantero de Equipo A
[SUBSCRIBER] Mensaje recibido: Penalti a favor de Equipo B
[SUBSCRIBER] Mensaje recibido: Gol de penalti del jugador 9 de Equipo B
[SUBSCRIBER] Mensaje recibido: Lesion del defensa central de Equipo A
[SUBSCRIBER] Mensaje recibido: Gol de tiro libre de Equipo A al minuto 70
[SUBSCRIBER] Mensaje recibido: Tarjeta roja para el numero 5 de Equipo B
[SUBSCRIBER] Mensaje recibido: Final del partido: Equipo A 2 / 2 Equipo B
```

```
C:\Users\sebas\OneDrive\Documentos\Septimo semestre\Redes\Laboratorios\Laboratorio 3\LabTre
sRedes\UDP>.\subscriber_udp.exe 127.0.0.1 5000 "Equipo C vs Equipo D"
[SUBSCRIBER] Suscrito al partido Equipo C vs Equipo D
[SUBSCRIBER] Mensaje recibido: Comienza el partido: Equipo C vs Equipo D
[SUBSCRIBER] Mensaje recibido: Gol del Equipo C al minuto 3
[SUBSCRIBER] Mensaje recibido: Remate desviado del delantero de Equipo D
[SUBSCRIBER] Mensaje recibido: Tarjeta amarilla para jugador 14 de Equipo C
[SUBSCRIBER] Mensaje recibido: Gol de tiro libre de Equipo D al minuto 25
[SUBSCRIBER] Mensaje recibido: Cambio en Equipo C: entra jugador 12 por jugador 6
[SUBSCRIBER] Mensaje recibido: Centro de Equipo D despejado por la defensa
[SUBSCRIBER] Mensaje recibido: Gol de cabeza de Equipo D al minuto 50
[SUBSCRIBER] Mensaje recibido: Empate de Equipo C con gol al minuto 55
[SUBSCRIBER] Mensaje recibido: Tarjeta amarilla para el arquero de Equipo D
[SUBSCRIBER] Mensaje recibido: Lesion del jugador 8 de Equipo C
[SUBSCRIBER] Mensaje recibido: Penalti a favor de Equipo D
[SUBSCRIBER] Mensaje recibido: Gol de penalti de jugador 9 de Equipo D
[SUBSCRIBER] Mensaje recibido: Final del partido: Equipo C 2 / 3 Equipo D
```

Preguntas

1. ¿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?

En el caso de UDP lo que se vería es una pérdida de paquetes importante si la red se llegara a saturar y como UDP no es confiable esos paquetes nunca llegarían al receptor, además se sobresaturaría la red porque UDP no tiene control de congestión entonces todos los paquetes se mandarían de una. Por otro lado, en el caso de TCP es diferente, si alguno de los paquetes se llegara a perder este lo reenviaría, entonces el receptor eventualmente lo podía ver. Además, como TCP usa de una manera justa la red no sobresaturaría la red sino que enviaría menos paquetes a la vez, demorándose más, pero usando adecuadamente la red.

2. 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?

Dado que estamos usando UDP y el suscriptor no recibe un mensaje de gol, podemos decir que el usuario perdió información del partido puesto que el protocolo utilizado no garantiza que los mensajes lleguen ni retransmite mensajes en caso de pérdida. En la aplicación real, el usuario tendrá el marcador desactualizado o incorrecto del partido al que esta suscrito. Por otro lado, TCP maneja mejor este escenario ya que posee mecanismos de retransmisión automática, confirmación de bytes enviados (ACKs) y control de flujo.

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

Sería mejor UDP, claramente se perderían algunos paquetes como se habló anteriormente, pero se está priorizando la velocidad de transmisión en este caso. Al ser un partido en vivo, TCP retrasaría la información demorando más que lleguen los avisos importantes. No sería óptimo que se demora mucho la transmisión del partido porque si no esté ya dejaría de ser en vivo.

4. Compare el overhead observado en las capturas Wireshark entre TCP y UDP. ¿Cuál protocolo introduce más cabeceras por mensaje? ¿Cómo influye esto en la eficiencia?

Como podemos observar en las capturas de Wireshark, los mensajes enviados por TCP se les añade una cabecera de 20 bytes o más debido a campos adicionales como los números de secuencia, control de flujo y opciones de control de conexión. Por otro lado, UDP solo añade una cabecera de 8 bytes ya que solo contiene cosas básicas como puerto de origen y destino. Esta diferencia de bytes influye directamente en la eficiencia del envío de información, puesto que esto permite que UDP transmita una mayor cantidad de mensajes en menos tiempo y con menor carga de red a costa de sacrificar confiabilidad y control.

5. 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 se pondría disgustado porque le estarías dañando toda la emoción al partido este ya sabría lo que se viene y esa no es la idea. Para solucionarlo usaría otros protocolos como QUIC el cual garantiza el orden de los paquetes, pero también es rápido como UDP.

6. ¿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?

El aumento de número de suscriptores afecta el desempeño se comporta de manera diferente usando TCP y UDP. En TCP, se deben mantener una conexión individual con cada suscriptor por parte del bróker, por lo que se deben destinar una gran cantidad de recursos como CPU, memoria y sockets abiertos para mantener las conexiones funcionando. Sin embargo, en UDP no es necesario mantener conexiones persistentes solo mandar paquetes sin verificación de llegada a múltiples direcciones, lo cual es bastante eficiente cuando hay una gran cantidad de suscripciones.

7. ¿Qué sucede si el bróker se detiene inesperadamente? ¿Qué diferencias hay entre TCP y UDP en la capacidad de recuperación de la sesión?

Si el bróker se detiene inesperadamente, en ambos casos el servicio se caería ya que los dos dependen de este para funcionar correctamente. La diferencia es que en el caso de UDP los paquetes se perderían, no se enviarían al receptor. Por otro lado, en el caso del TCP se reenvían los paquetes que no llegaron al no llegaron al bróker para que los mande al suscriptor.

8. ¿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?

Si es necesario que todos los usuarios reciban la información al mismo tiempo de manera consistente, el mejor protocolo que se puede utilizar es TCP puesto que permite tener conexiones estables que garantizar el envío de la información de manera ordenada para que todos los usuarios reciban las actualizaciones críticas.

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

El manejo de múltiples conexiones TCP en un broker consume significativamente más CPU y memoria que el manejo de datagramas UDP, debido a la gestión de estado, buffers y control de flujo que TCP requiere. UDP tiene un encabezado mucho más pequeño, lo por lo que usa mucho menos buffer.

10. 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

En un sistema real que contenga millones de usuarios publicando y suscribiéndose, el mejor protocolo que se puede implementar es un modelo híbrido que combine TCP y UDP. Por el lado de TCP, este puede utilizarse para sincronizaciones críticas, reconexiones y envío de datos importantes que no pueden perderse, como lo son un gol o expulsiones en un partido. Ahora, el uso de UDP sería para la transmisión de eventos en tiempo real, donde la prioridad es la rapidez y la baja latencia sin importar si se pierden algunos paquetes, como en jugadas o estadísticas en vivo.