Memoria 2 ~ 2024-25:

FUNCIONAMIENTO ECO

PRIMER CUATRIMESTRE

Arquitectura de Redes – Jaime García Reinoso 7/10/2024

Luis Miguel Herrá Alpuente – luis.herra@edu.uah.es – 06612001F

1. Al arrancar el servidor, éste imprime una línea que comienza por "Conectado con". ¿Qué dos datos aparecen? ¿Qué significan?

Nos muestra la dirección a la cual nos hemos conectado (cliente), en este caso concreto nos estamos conectando **a nuestro propio sistema**. Por ello, nos muestra nuestra **IP local** 127.0.0.1 y **nuestro puerto** 53775.

Nos hemos **conectado desde** una instancia creada bajo la **IP local** 127.0.0.1, por ello HOST = "", y bajo el **puerto 9999**. Es decir, **nuestro** sistema está aceptando una solicitud de conexión de IP = 127.0.0.1 PORT = 9999.

2. Haga un análisis de los datos más importantes que pueda extraer de los paquetes capturados con Wireshark

Podemos observar todo lo mencionado en el ejercicio 1, las direcciones destino y origen así como sus puertos (Origen: 127.0.0.1 Puerto: 53775 a Destino: 127.0.0.1 Puerto: 9999). El texto enviado y como el eco lo devuelve, con su longitud. Los paquetes SYN, ACK, FIN y sus respectivos tamaños; muestran el proceso de inicio de transferencia, confirmación de llegada y fin del proceso.

3. ¿Podemos conectar dos clientes al mismo servidor? Para comprobarlo, reinicie el servidor de forma que no haya ningún cliente conectado. Después, abrimos dos terminales diferentes, y nos conectamos desde ambas terminales al mismo servidor.

a. ¿Aparecen dos líneas de "Conectado con" en el servidor?

No, únicamente se ha conectado la primera instancia. A partir de esta premisa, podemos concluir que la conexión es única; no permite conectarse a más de un cliente.

b. Intente mandar texto desde ambos clientes. ¿Recibe eco de ambos, solamente de uno, o de ninguno?

Únicamente puedo mandar información desde la primera instancia, por tanto, solo recibe el eco de la primera instancia.

c. Ahora cierre la conexión desde el cliente que SI está conectado (como se describía antes, mandando el mensaje EXIT). ¿Qué sucede ahora? ¿Hay algún cambio en el servidor? ¿Hay algún cambio en el otro cliente?

Se conecta la segunda instancia, será ahora está la principal y aquella que ocupe la conexión.

4. Haga un análisis de los datos más importantes que pueda extraer de los paquetes capturados con Wireshark.

La información relativa a la transferencia del eco se mantiene igual a la del ejercicio 2 pues, la instancia está conectada; podemos concluir el puerto, IP, tamaño de paquete, cabeceras, ACK, SYN, FIN. En este caso, la única novedad, es que la solicitud enviada por la segunda instancia, al estar el canal ocupado, es rechazada por el servidor; concretamente el SYN.

5. Tras reiniciar al servidor, intente acceder al servidor de eco desde el navegador web. Para ello, la URL sería http://<direccion>:<puerto>, en nuestro caso http://127.0.0.1:9999, dependiendo del puerto configurado. Es posible que el navegador web se quede cargando y tenga que interrumpir la carga manualmente para ver la respuesta. ¿Qué se ve en el servidor, y qué se ve en el navegador web?

Efectivamente, el navegador web se mantiene cargando y al interrumpir la carga observamos que muestra: 127.0.0.1 ha enviado una respuesta no válida (ERR_INVALID_HTTP_RESPONSE). Por parte del servidor, podemos observar la siguiente información:

```
Conectado con 127.0.0.1:51353
Recibido: GET / HTTP/1.1
Host: 127.0.0.1:9999
Connection: keep-alive
sec-ch-ua: "Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: es-ES,es;q=0.9
```

A partir de lo anteriormente mencionado podemos concluir.

El navegador se intenta conectar al servidor pero no lo consigue, debido a que este envía el formato HTTP y no simplemente texto como se espera. Como el servidor no envía una respuesta válida, el navegador se mantendrá a la espera de esta; permanece cargando. En el lado del servidor, vemos que se realiza la conexión, nos muestra las características del navegador y el paquete recibido (como no es el esperado, la respuesta tampoco lo será).

6. Haga un análisis de los datos más importantes que pueda extraer de los paquetes capturados con Wireshark.

El navegador envía GET /HTTP/1.1 como indica el servidor. El servidor no sabe cómo responder ante este formato no envía repuesta válida hasta que el tiempo de espera del navegador se agote. Podemos ver las cabeceras, ACK, SYN y fin de los intentos.

7. Haga un código simple de un cliente TCP que se conecte al servidor y envíe una cadena de texto, imprimiendo el texto que devuelve el servidor de eco.

```
import socket

HOST = '' #local host
PORT = 9999
BUFFER = 1024

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(10)

while True:
    cliente, addr = s.accept()  # Acepta la conexión del cliente
    print(f"Conectado con {addr[0]}:{addr[1]}")

    # Envía un mensaje al cliente
    mensaje = input("Introduce el mensaje para el cliente: ")
    cliente.sendall(mensaje.encode())  # Usa 'cliente' para enviar datos
    cliente.close()
```

La versión inferior envía los ecos, separando con un espacio, hasta que recibe EXIT.

```
import socket

HOST = ''
PORT = 9999
BUFFER = 1024

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(10)

while True:
    cliente, addr = s.accept()  # Acepta la conexión del cliente
    print(f"Conectado con {addr[0]}:{addr[1]}")

    # Envía un mensaje al cliente
    mensaje = ""
    while (mensaje != "EXIT"): #verifica que sea diferente a EXIT
        cliente.sendall(mensaje.encode())
        cliente.sendall(" ".encode())
        mensaje = input("Introduce el mensaje para el cliente ==> ")

    cliente.close()
```