Informe sobre TP N°4

Juan Ignacio Massacesi

May 2025

1 Actividad 1

Este programa implementa un chat en red local utilizando el socket UDP y la técnica de multihilo (threading) para permitir la comunicación simultánea entre múltiples usuarios.

1.1 Importación de módulos

Se importan los módulos necesarios:

- socket: proporciona funciones para crear y manejar conexiones de red.
- threading: permite ejecutar funciones en paralelo (en diferentes hilos).
- time: se utiliza para pausar la ejecución brevemente (por ejemplo, al salir del chat).

1.2 Función recibir(sock, username)

Esta función se ejecuta en un hilo dedicado a recibir mensajes.
Comportamiento de la función:

- Escucha mensajes entrantes de la red.
- Ignora mensajes mal formateados.
- Detecta eventos especiales:
  - "nuevo": nuevo usuario se une.
  - "exit": usuario abandona el chat.
- Imprime mensajes normales en consola con el nombre del remitente. Ejemplo: "Juan:Hola a todos"

1.3 Función enviar(sock, username)

Esta función se ejecuta en otro hilo y se encarga de enviar los mensajes del usuario local.
Comportamiento de la función:

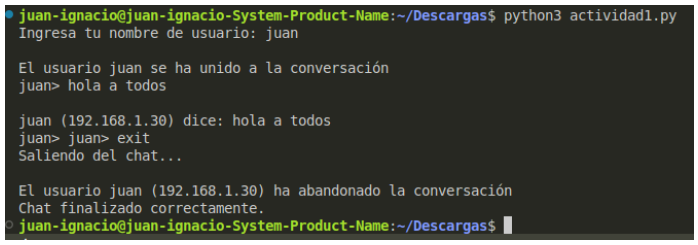
- Al iniciar, notifica al resto que un nuevo usuario se ha unido.
- Lee mensajes desde la entrada del usuario (input()).
- Si el mensaje es "exit", informa a los demás y termina el hilo.
- Envía el mensaje a través de la red usando broadcast, como "username:mensaje".

1.4 Función principal main()

Coordina la configuración del socket y el inicio de los hilos.
Comportamiento de la función:

- Pide al usuario su nombre.
- Crea un socket UDP.
  - Habilita la opción de broadcast (SO_BROADCAST).
  - Lo asocia al puerto 60000 y a todas las interfaces (0.0.0.0).
- Inicia dos hilos:
  - Uno para recibir (recibir).
  - Otro para enviar (enviar).
    Esto permite que el programa escuche mensajes mientras estás escribiendo en el chat, sin que una tarea bloquee a la otra.
- Espera a que el hilo de envío finalice (cuando el usuario escriba "exit").
- Cierra el socket y finaliza el programa.

1.5 Resultado por consola



2 Actividad 2

Este programa implementa un servidor TCP que acepta conexiones de clientes, permite el intercambio de mensajes, y utiliza hilos tanto para la recepción de mensajes del cliente como para la entrada del servidor desde consola.

2.1 Código Servidor

2.1.1 Importación de módulos

- socket: proporciona funciones para crear y manejar conexiones de red.
- threading: permite ejecutar funciones en paralelo (en diferentes hilos).

2.1.2 Función handle_client(conn, addr)

Esta función se ejecuta en un hilo y se encarga de recibir mensajes del cliente conectado.
Comportamiento de la función:

- Imprime un mensaje cuando un cliente se conecta.
- Recibe datos desde el cliente (hasta 1024 bytes).
- Si el mensaje es vacío o "exit", cierra la conexión.
- Muestra en consola cada mensaje recibido.

2.1.3 Función server_main()

Controla la lógica principal del servidor: configuración del socket, espera de conexiones y gestión de hilos.
Comportamiento de la función:

- Configuración del servidor:
  - '0.0.0.0' significa que el servidor escuchará en todas las interfaces de red (localhost, IPs públicas, etc.).
  - Se establece el puerto 12345, donde los clientes deberán conectarse.
- Creación y configuración del socket:
  - Se crea un socket TCP (SOCK_STREAM).
  - Se "enlaza" a la IP y puerto definidos.
  - listen(1) indica que el socket está esperando conexiones (puede atender una a la vez).
- Se notifica al operador del servidor que está activo y a la espera de clientes.

2.1.4 Función server_input()

Dentro de server_main(), se define esta función que corre en un hilo separado.
Comportamiento de la función:

- Permite escribir mensajes en consola para enviar al cliente.
- Si el mensaje es "exit":
  - Si hay un cliente conectado: no permite cerrar el servidor.
  - Si no hay conexión activa: cierra el servidor correctamente.
- Envío de mensajes al cliente mediante sendall().

2.2 Código Cliente

2.2.1 Importación de módulos

- socket: proporciona funciones para crear y manejar conexiones de red.
- threading: permite ejecutar funciones en paralelo (en diferentes hilos).
- sys: se usa para terminar el programa de forma segura (sys.exit()).

2.2.2 Función recibir_mensaje(sock)

Esta función se ejecuta en un hilo separado y se encarga de recibir mensajes del servidor mientras el cliente escribe por consola.
Comportamiento de la función:

- Escucha continuamente por nuevos mensajes desde el servidor.
- Si la conexión se cierra o hay un error, muestra un mensaje y termina el programa.
- Imprime los mensajes que llegan del servidor en pantalla.

2.2.3 Función client_main()

Esta es la función principal del cliente, que establece la conexión y permite enviar mensajes al servidor.
Comportamiento de la función:

- Solicita al usuario la dirección IP del servidor.
- Crea un socket TCP e intenta conectarse a ese servidor.
- Si se conecta con éxito:
  - Muestra instrucciones al usuario.
  - Inicia un hilo para recibir mensajes usando recibir_mensaje().

- En el hilo principal:
  - Lee mensajes desde consola.
  - Si el mensaje es "exit", cierra el socket y finaliza el programa.
  - Si es otro mensaje, lo envía al servidor.

3 Actividad 3

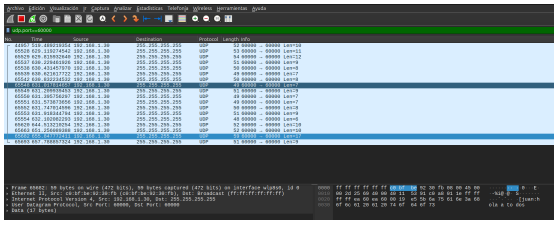


Figure 1: Análisis de tráfico de protocolo UDP

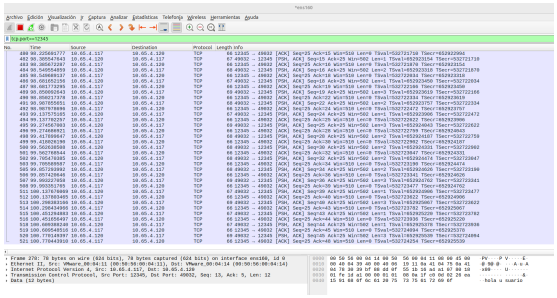


Figure 2: Análisis de tráfico de protocolo TCP

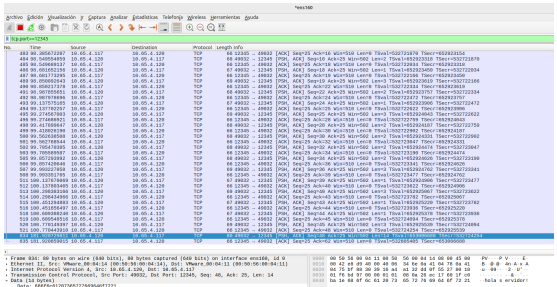
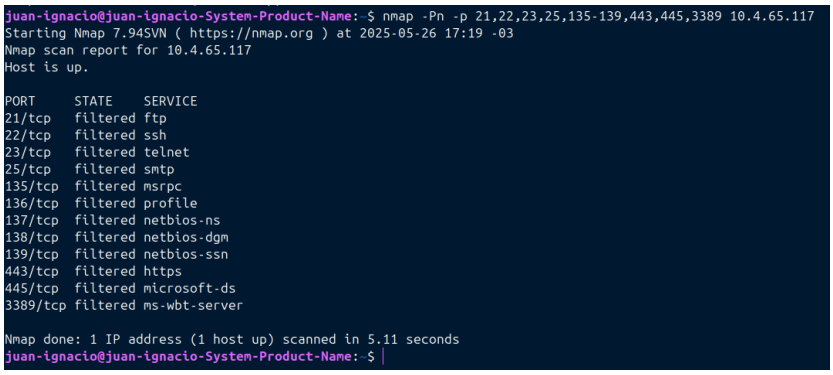


Figure 3: Análisis de tráfico de protocolo TCP

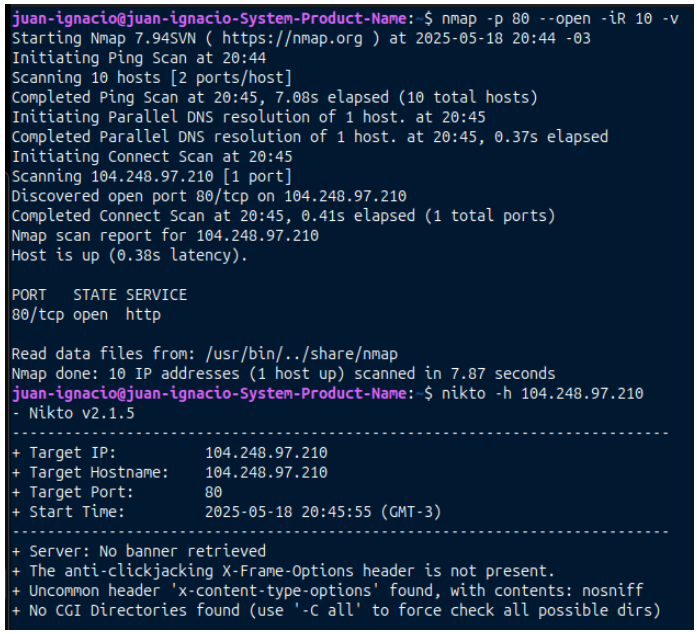
4 Actividad 4

4.1 Actividad 4.2



Todos los puertos consultados (21, 22, 23, 25, 135–139, 443, 445 y 3389) se encuentran en estado “filtered”. Esto significa que el host está activo, pero un firewall o dispositivo de red está bloqueando el acceso a esos puertos. Por lo tanto, no se puede determinar si están abiertos o cerrados, pero sí se puede concluir que están protegidos frente a accesos externos.

4.2 Actividad 4.3

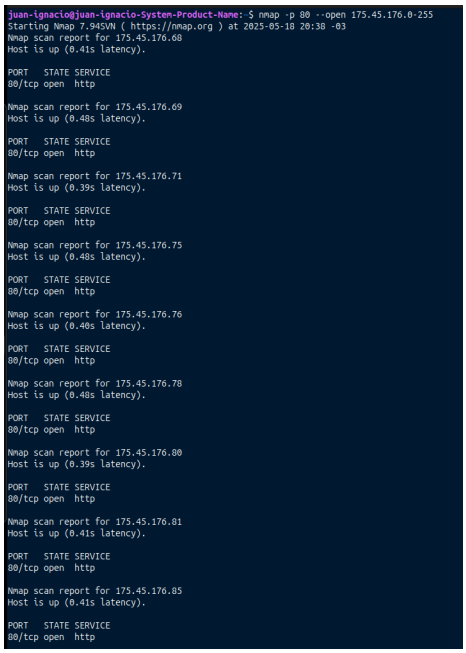


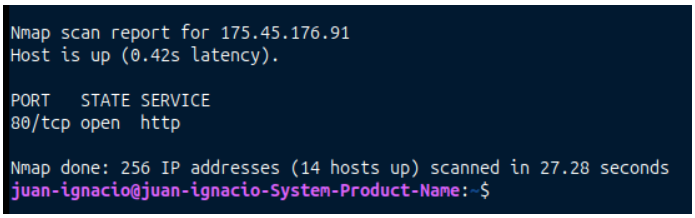
IP address	**104.248.97.210** (change)
Latitude	1.314
Longitude	103.6839
Country	Singapore
Region	
City	Singapore
Organization	Digital Ocean

Figure 4: Ubicación Geográfica de la IP

Falta la cabecera anti-clickjacking. El servidor no está protegido contra clickjacking, un tipo de ataque en el que un sitio malicioso carga este sitio dentro de un iframe invisible para engañar al usuario.

4.3 Actividad 4.4




Nmap scan report for 175.45.176.91
Host is up (0.42s latency).
PORT STATE SERVICE
80/tcp open http
Nmap done: 256 IP addresses (14 hosts up) scanned in 27.28 seconds
juan-ignacio@juan-ignacio-System-Product-Name:~$