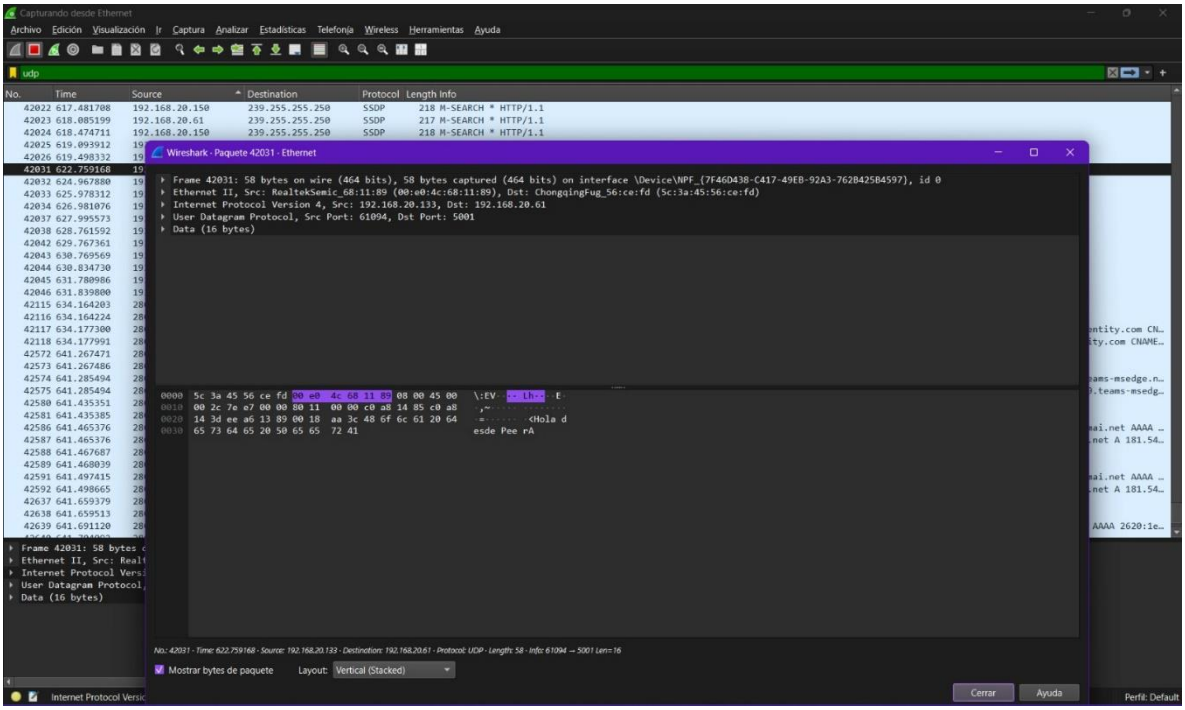
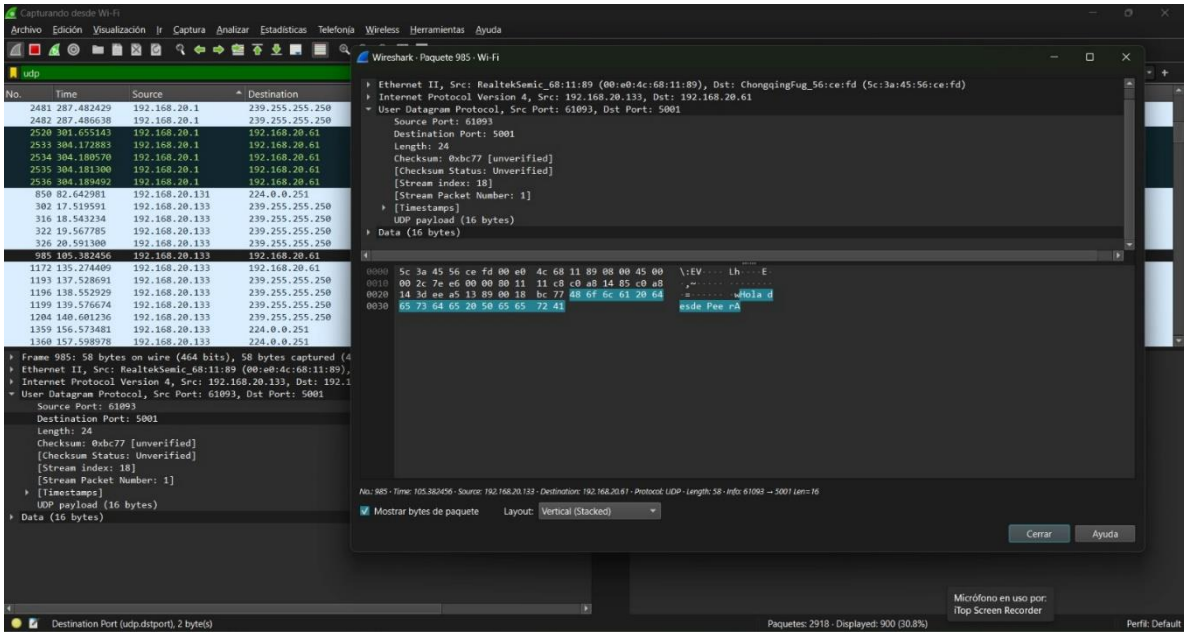
TALLER UDP

SAMUEL DOMINGUEZ – DAMY VILLEGAS

1. Capturas de WireShark





2. ¿Es posible ver en la captura de WireShark el contenido del mensaje enviado?

Si es posible ver en la captura de WireShark el contenido del mensaje que se envía desde el PeerA, hasta el PeerB.

3. **¿Cuál es el Checksum de la captura?¿Explique porque este Checksum?**

De acuerdo con la información proporcionada en la captura de Wireshark, el checksum calculado para el datagrama UDP es 0x8c77. Sin embargo, se indica que este valor está sin verificar (unverified).

El término "unverified" simplemente indica que Wireshark no ha podido confirmar de forma independiente que el valor del checksum calculado sea correcto. Esto no significa necesariamente que haya un error en los datos, sino que se requiere más información para realizar una verificación completa.

4. **¿Qué patrones de diseño o arquitectura aplicaría al desarrollo de un programa basado en red como este?**

Patrones de Diseño y Arquitectura para un Programa Basado en Red P2P con UDP

Al desarrollar un programa basado en red P2P que utiliza UDP, es fundamental seleccionar los patrones de diseño y arquitectura adecuados para garantizar la eficiencia, escalabilidad y robustez de la aplicación.

Patrones de Diseño Relevantes

- **Observer:** Ideal para notificar a los pares de la red sobre eventos, como la conexión o desconexión de otros pares, o la recepción de nuevos mensajes.
- **Strategy:** Permite cambiar el algoritmo de descubrimiento de pares o la forma de manejar los mensajes sin modificar el código principal.
- **Singleton:** Útil para garantizar que solo haya una instancia de ciertos componentes, como el gestor de conexiones o el sistema de mensajes.

Arquitecturas Comunes

- **Estructura Overlay:** Crea una capa de red virtual sobre la red física, permitiendo a los pares descubrirse y comunicarse de manera más eficiente.
- **Estructura Jerárquica:** Introduce nodos centrales que coordinan la comunicación entre los pares, mejorando la escalabilidad y la eficiencia.
- **Estructura Hibrida:** Combina elementos de las estructuras overlay y jerárquica para obtener los beneficios de ambas.

Consideraciones Específicas para UDP y P2P

UDP:

- **Sin garantía de entrega:** Diseña mecanismos de retransmisión o confirmación para garantizar la entrega de mensajes importantes.
- **Sin orden de entrega:** Implementa mecanismos de secuenciación si el orden de los mensajes es crítico.
- **Datagramas:** Utiliza datagramas de tamaño fijo o variable según las necesidades de la aplicación.

P2P:

- **Descubrimiento de pares:** Utiliza técnicas como el flooding, el bootstrapping o los servicios de directorio para permitir que los pares se encuentren.
- **Ruteo:** Implementa algoritmos de ruteo adecuados para encontrar la ruta más eficiente entre los pares.
- **Tolerancia a fallos:** Diseña la aplicación para que pueda manejar la desconexión de pares y la reconfiguración de la red.

5. **Investiguen que modificaciones son necesarias para implementar este mismo sistema, pero para la comunicación TCP en java.**

Para modificar este sistema para utilizar TCP en lugar de UDP en Java, se deben realizar algunos cambios, como por ejemplo:

Uso de Sockets TCP:

- En lugar de DatagramSocket (que es para UDP), utilizar Socket para el cliente y ServerSocket para el servidor.

Conexión persistente:

- TCP es un protocolo orientado a la conexión, por lo que se debera establecer una conexión entre el cliente y el servidor antes de enviar datos. Esto hace que el servidor acepte las conexiones entrantes con ServerSocket.accept(), mientras que el cliente deberá conectarse a la dirección del servidor usando un Socket.

Lectura y escritura de datos:

- A diferencia de UDP, donde se utilizan DatagramPacket para enviar y recibir datos, en TCP se utilizan flujos de entrada (InputStream) y de salida (OutputStream). Lo que implica que se deberá leer y escribir datos a través de estos flujos.

6. **¿Qué utilidades de codificación o seguridad agregaría al código?**

Para mejorar el código y su seguridad, agregaríamos las siguientes cosas:

Cifrado de los Mensajes:

Para asegurar que los mensajes intercambiados entre PeerA y PeerB sean seguros, puedes implementar cifrado. Por ejemplo, usando un cifrado simétrico como AES para cifrar y descifrar los datos antes de enviarlos o recibirlos.

Validación y Autenticación de Mensajes

Para asegurar de que el mensaje no haya sido alterado durante la transmisión, se puede usar un código de autenticación de mensajes (MAC). El cual permitirá verificar la integridad de los datos y que provengan de una fuente legítima.

Control de tamaño y validación de entrada:

Tener asegurado de que el tamaño de los paquetes y los mensajes no supere los límites esperados para evitar ataques de desbordamiento de búfer o enviar datos corruptos.

- Verificar la longitud de los datos antes de empaquetarlos.
- Validar la información recibida antes de procesarla.

Control de acceso y autenticación:

Se puede agregar un mecanismo básico de autenticación para asegurar que solo usuarios autorizados puedan enviar y recibir datos. Esto puede ser tan simple como un **token** o **contraseña compartida**.