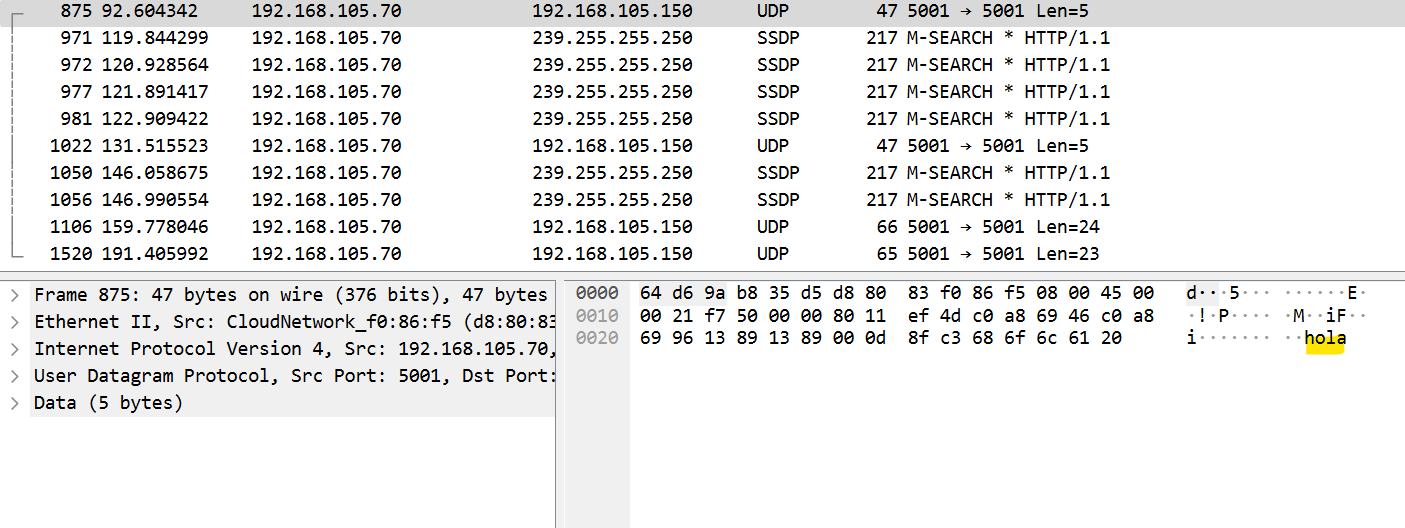
# Capturas:

Como se puede ver, si aparecen los mensajes y su contenido

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Calendario

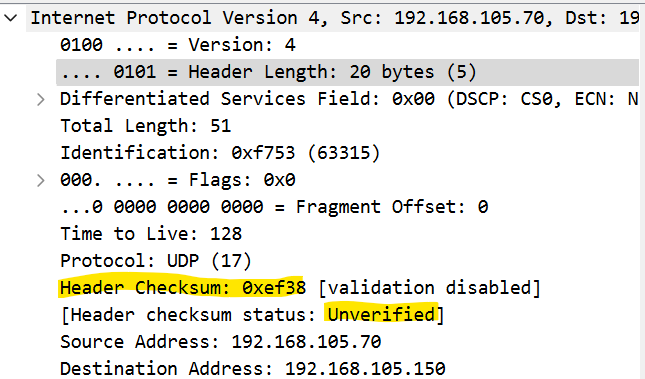
Descripción generada automáticamente con confianza media

Tabla

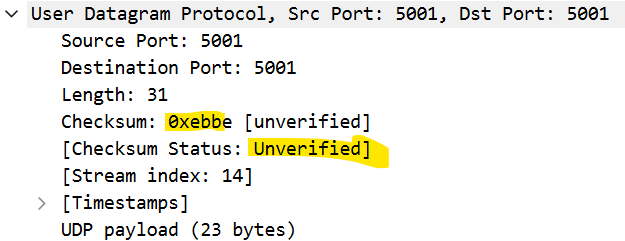
Descripción generada automáticamente

# El checksum:

Del IP



Del UDP:



Ambos sirven para la integridad de los archivos.

# ¿Qué patrones de diseño/arquitectura aplicaría al desarrollo de un programa basado en red como este?

**Patrón Observer (Observador):**

Facilita la actualización de la interfaz de usuario cuando se reciben nuevos mensajes de la red sin acoplar estrechamente la lógica de red con la presentación.

**Patrón Singleton:**

Útil para manejar conexiones de red compartidas, asegurando que solo exista una instancia de la conexión que gestione todas las operaciones de envío y recepción.

**Patrón Factory (Fábrica):**

Permite crear diferentes tipos de conexiones o mensajes (por ejemplo, TCP o UDP) sin cambiar el código que utiliza estas clases.

**Arquitectura MVC (Modelo-Vista-Controlador):**

Ayuda a mantener separadas la lógica de negocio y la presentación, facilitando el mantenimiento y la escalabilidad del código.

**Patrón Strategy (Estrategia):**

Permite cambiar dinámicamente el protocolo de comunicación o el método de manejo de datos sin modificar las clases que los utilizan.

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

TCP es un protocolo orientado a la conexión, lo que implica que debe establecerse una conexión estable entre el cliente y el servidor antes de que se pueda transferir cualquier dato. A diferencia de UDP, que es sin conexión, TCP asegura la transmisión confiable de los datos mediante confirmaciones, retransmisiones y control de flujo. Las modificaciones incluirían el uso de las clases ServerSocket y Socket para manejar las conexiones, y el uso de flujos de entrada y salida (InputStream y OutputStream) para enviar y recibir datos de manera continua. Además, sería necesario manejar el cierre adecuado de las conexiones una vez que la comunicación termine

# ¿Qué utilidades de codificación o seguridad agregaría al código?:

* Encriptación: Para la privacidad
* Sesiones: Que cierren automáticamente
* Limitar el numero de intentos fallidos