Implementación de un protocolo ya existente

Gerardo Pineda 22880

Repositorios

Cliente y servidor remoto: Gerax5/Redes_Proyecto1

Servidor local: Gerax5/MCP_Server

MCPs

Local:

El MCP local implementado permite consultar sobre como es la comunidad de un canal de youtube. Este mcp extrae con la ayuda del api de youtube comentarios de videos y a estos les aplica un análisis de sentimientos para conocer mas a su comunidad. También, esta manda las palabras con más frecuencia. La herramienta es:

youtube_analysis:

- **descripción:**
  - Analiza la comunidad, audiencia, seguidores y comentarios de canales de YouTube para entender su demografía y comportamiento
- **parámetros:**
  - Channel* (Esta es obligatoria): El nombre del canal de youtube
  - max_comments: cuantos comentarios solicita.
  - top_words: Cuantas palabras con mayor frecuencia solicita.

Remoto:

El mcp remoto realiza una llamada a la api de Pokémon para obtener las estadísticas del Pokémon solicitado. La herramienta es:

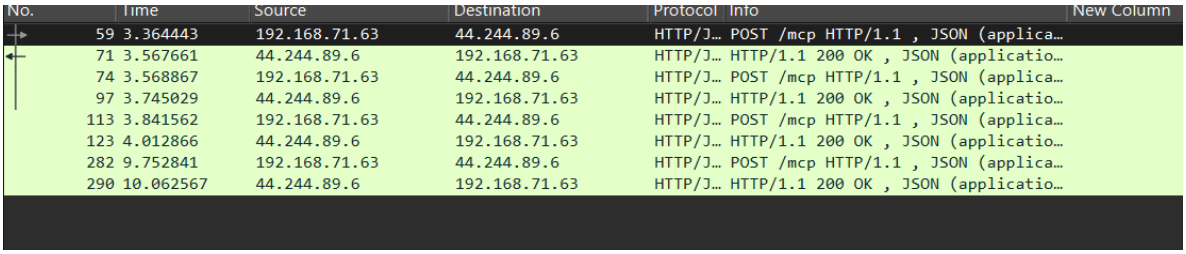
get_pokemon

- **descripción:**
  - Permite consultar estadísticas, tipos y otra información detallada de un Pokémon
- **Parámetros:**
  - Name: Nombre del Pokémon.

Analisis de Wireshark

Desgloce de lo que se ve en Wireshack

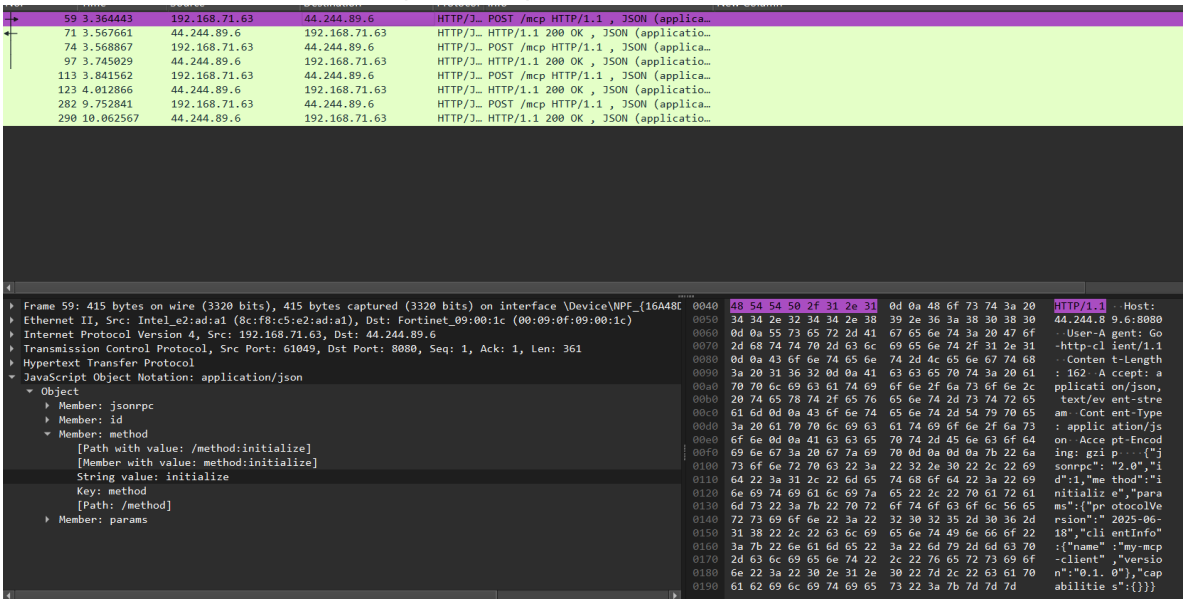
Estos fueron los registros que se encontraron en una simple interacción entre el chatbot y el mcp remoto.



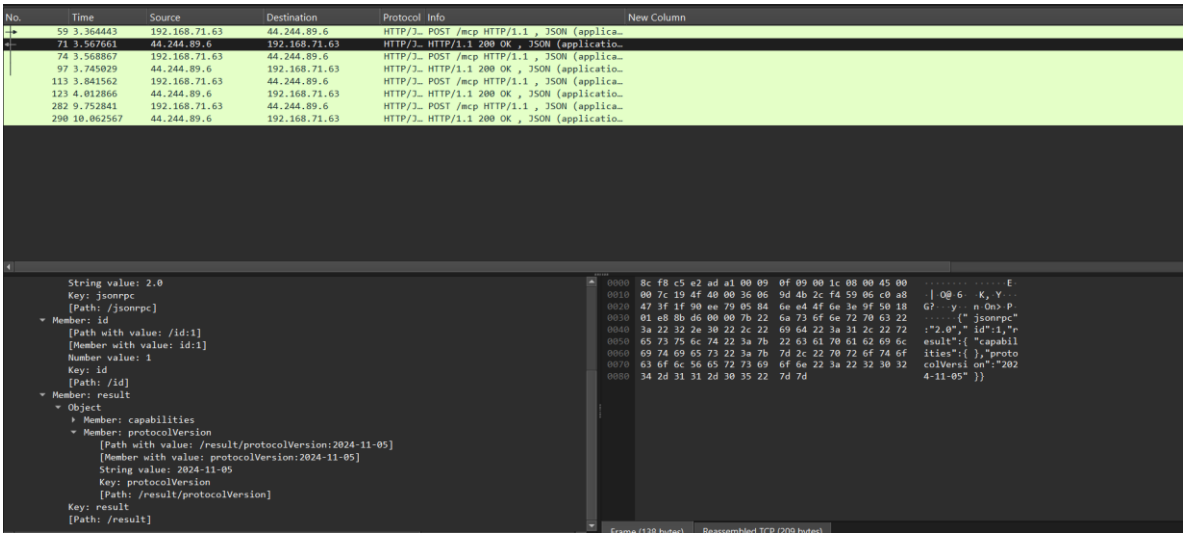
No.	Time	Source	Destination	Protocol	Info	New Column
59	3.364443	192.168.71.63	44.244.89.6	HTTP/J…	POST /mcp HTTP/1.1 , JSON (applica…	
71	3.567661	44.244.89.6	192.168.71.63	HTTP/J…	HTTP/1.1 200 OK , JSON (applicatio…	
74	3.568867	192.168.71.63	44.244.89.6	HTTP/J…	POST /mcp HTTP/1.1 , JSON (applica…	
97	3.745029	44.244.89.6	192.168.71.63	HTTP/J…	HTTP/1.1 200 OK , JSON (applicatio…	
113	3.841562	192.168.71.63	44.244.89.6	HTTP/J…	POST /mcp HTTP/1.1 , JSON (applica…	
123	4.012866	44.244.89.6	192.168.71.63	HTTP/J…	HTTP/1.1 200 OK , JSON (applicatio…	
282	9.752841	192.168.71.63	44.244.89.6	HTTP/J…	POST /mcp HTTP/1.1 , JSON (applica…	
290	10.062567	44.244.89.6	192.168.71.63	HTTP/J…	HTTP/1.1 200 OK , JSON (applicatio…	

Inicializacion

Primero el cliente manda al endpoint /mcp el método inicializar

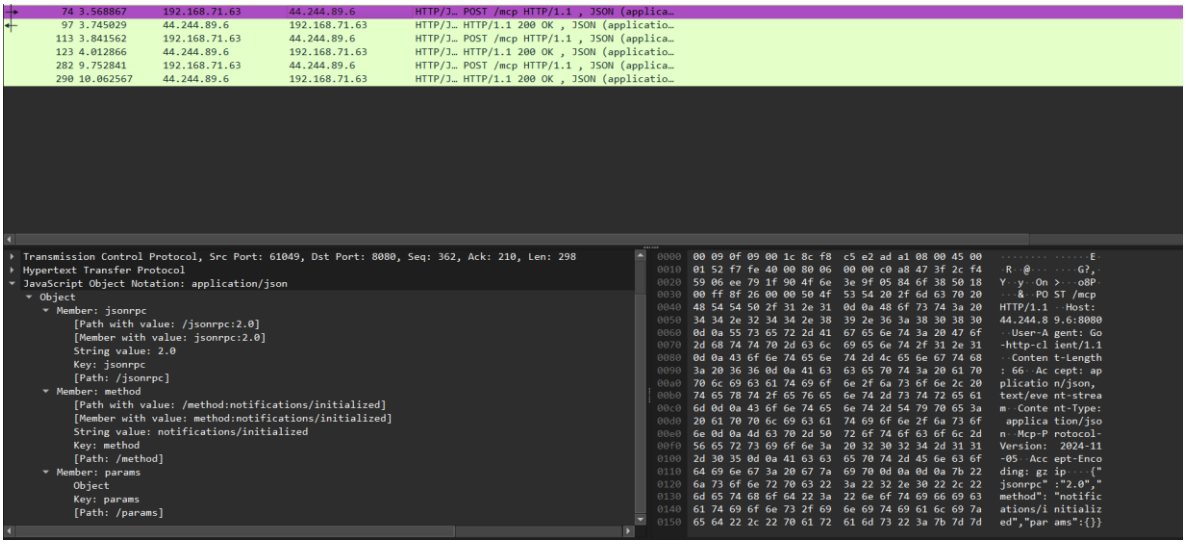


El servidor le responde con un 200 ok y le manda la versión del protocolo y capabilities.

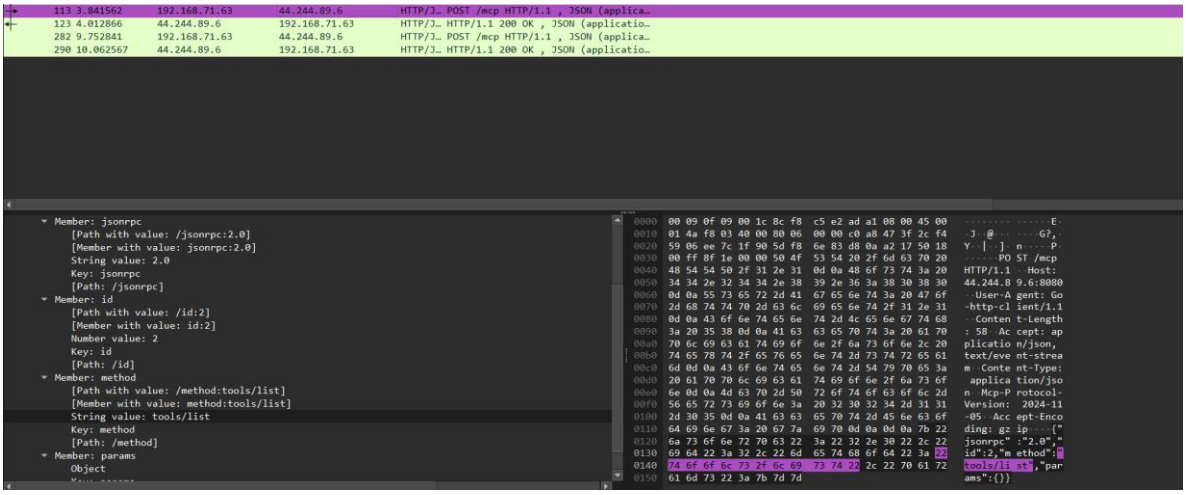


Luego el cliente le manda una notificación al mcp diciendo que ya se conectó y lo único que hace el servir es obtener este mensaje. En la imagen se puede ver como el servidor le contesta, pero esta interacción únicamente fue porque el servidor devuelve que el método

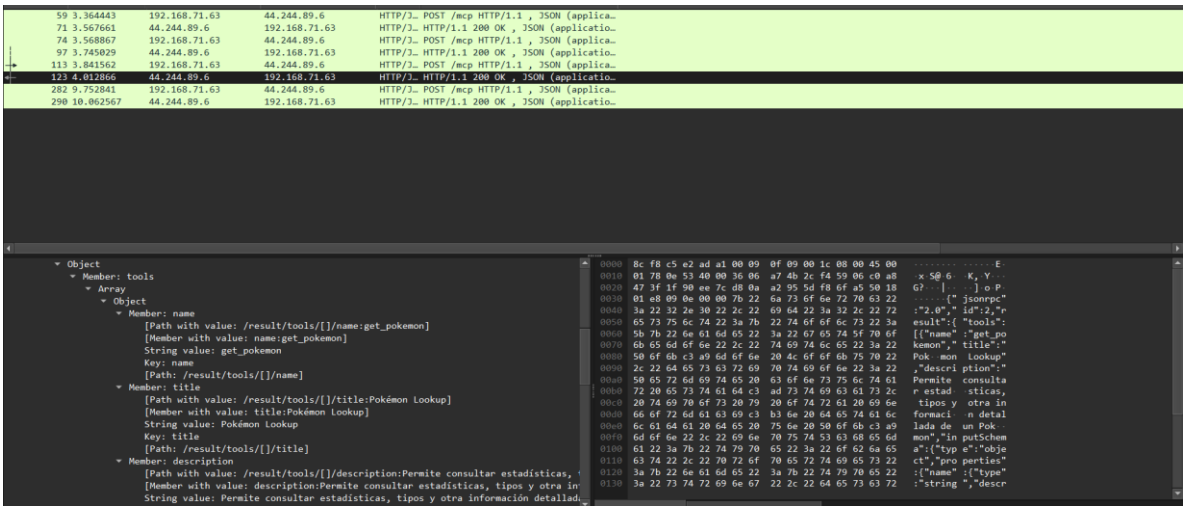
notification/initialized no esta implementado, pero en si debería de solo quedarse en el endpoint no devolver nada,



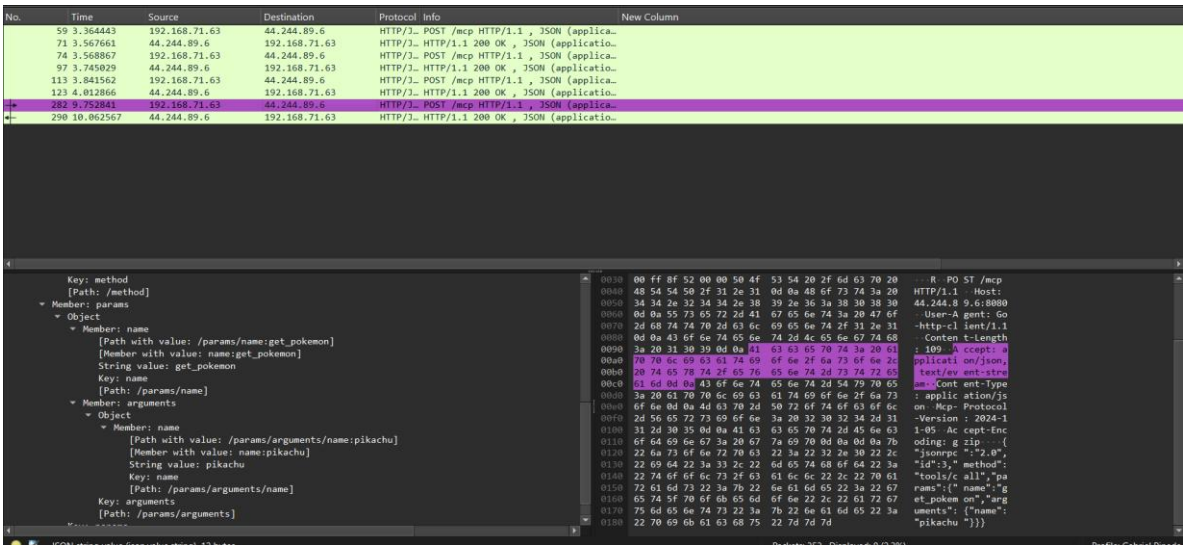
Luego el mcp pide el tool/list



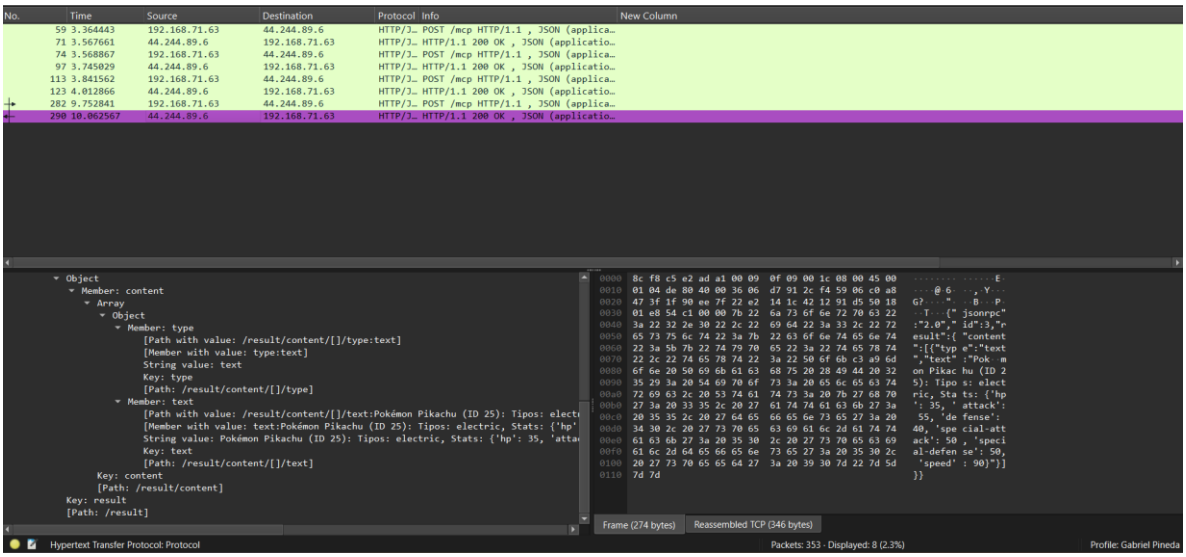
Y el servidor le responde las herramientas que tienen con los parámetros requeridos. Para que el cliente tenga noción de las herramientas que va a tener.



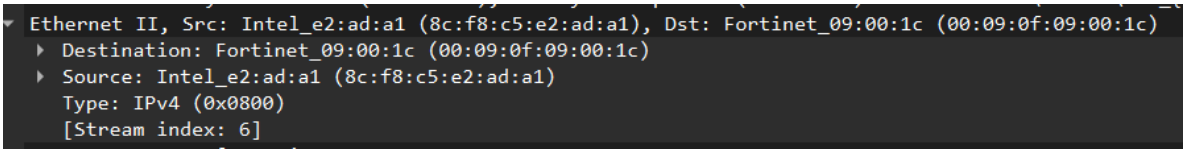
Posteriormente el cliente solicita al servidor usar la herramienta get_pokemon, esta manda el nombre del pokemon



Por último el servidor responde lo obtenido

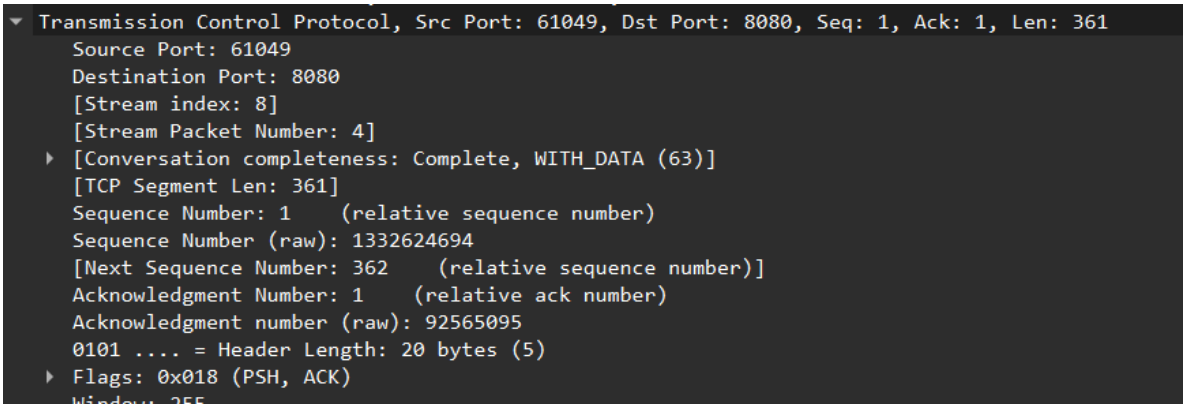


Capa de enlace

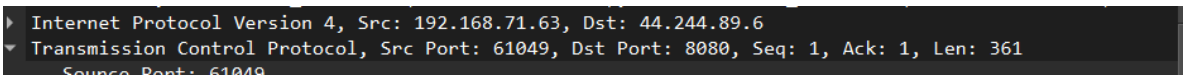


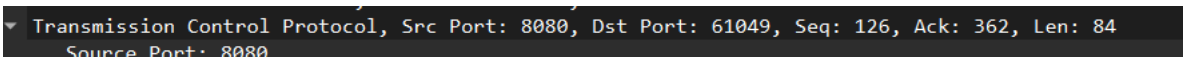
Cada mensaje se envia por internet, pasa por el mac de origen que es la tarjeta de red y el destino que es el Gateway route.

Capa de red es mi fuente destino que se ven en las imágenes anteriores la ip 192.168.71.63 a la ip destino de aws 44.244.89.6 se usa un enrutamiento a través de red. La capa de transporte sucede en con TCP en el puerto 8080 del servidor y el puerto 61049 de la computadora



Se puede observar cómo los ack y seq van aumentando en cada respuesta





Y la capa de aplicación es la que mencione con anterioridad en el desgloce de lo que mostraba wireshack con el protocolo http.

Dificultades

El apartado que representó mayor desafío fue lograr que el chatbot se comunicara correctamente con los servidores MCP. Mientras que implementar un chatbot con capacidad de mantener el contexto resultó relativamente sencillo, la verdadera complejidad apareció al integrar el uso de herramientas dentro de la comunicación con el protocolo. La dificultad se debió, en gran parte, a que era primera vez que usaba el lenguaje de programación y que no hay mucha documentación para implementar un chatbot con servidores mcp. Otra cosa que puedo mencionar, en mis MCP creados hice una implementación pura del protocolo sin recurrir a librerías como *fastmcp*. Finalmente, este reto se superó con recursos en línea y el análisis de ejemplos existentes, lo que permitió comprender mejor la estructura del protocolo e implementar con éxito la comunicación requerida.

Comentarios

Considero que es una buena forma de aplicar los conocimientos de la clase, agregándolo algo novedoso como usar un api de IA e integrarlo para hacer un propio chatbot. El cual pueda consultar a los servidores mcp. Buena forma de lograr hacer un servidor el cual pueda exponerse en la nube para que las demás personas la usen y consulten.

Conclusiones

- Los servidores MCP son útiles al integrarse con la inteligencia artificial, debido a que permiten automatizar procesos y reutilizar recursos que pueden ser desarrollador por nosotros mismos. Tendríamos una IA la cual es capaz de responder en base a su entrenamiento y herramientas externas que amplíen sus capacidades.
- Se logró implementar y analizar el protocolo MCP tanto en servidores locales como remotos, evidenciando cómo se establece la comunicación cliente-servidor a través de HTTP y confirmando la compatibilidad con los estándares definidos.
- Se evidenció el proceso de inicialización de la conexión, la notificación de estado, la consulta de herramientas disponibles y el uso de una herramienta específica. Cada una de estas etapas mostró el rol del protocolo en garantizar que el cliente entienda qué servicios puede consumir y cómo hacerlo.