UNIVERSIDAD DEL VALLE DE GUATEMALA

Redes, Sección 20

Jorge Yass

Proyecto 1

Angela García, 22869

Ciudad de Guatemala, Septiembre de 2024

Repositorio local y agente: angelargd8/proyecto1-redes

Repositorio del servidor remoto: angelargd8/RemoteServerMCP

Vídeo: https://youtu.be/c710uRN_phY

Introducción

Los LLM son un tipo de modelo de lenguaje de aprendizaje profundo, los cuales se usan para crear modelos de inteligencia artificial y solo pueden responder a base de su conocimiento. Los chatbots y agentes, le dan herramientas a los LLMs para aumentar su capacidad. En este proyecto se usa el protocolo Model Context Protocol para proveer herramientas a los agentes y chatbots, independientes del LLM. Siendo, implementados cuatro MCPs independientes en un solo chatbot.

Cuerpo

MCP locales:

- **MCP de FileSystem**
  - Herramienta que permite a los agentes interactuar de manera segura con los sistemas de archivos locales, proporcionando operaciones de escritura, lectura, actualización y gestión de directorios
  - Basado en el servidor oficial: @modelcontextprotocol/server-filesystem
  - Parámetros:
    - Path: ruta en donde se desea editar
    - Content: contenido de texto
  - Endpoints:
    - filesystem:create_directory
    - filesystem:write_file
    - filesystem:read_text_file
    - filesystem:list_directory

- **MCP de Git**
  - Es un servidor que permite la integración de herraminetas de inteligencia artificial con otras herramientas y servicios para mejorar la experiencia del desarrollo al proporcionar asistencia. En este caso para crear commits, crear repositorios y hacer un push.
  - Basado en el servidor oficial: @cyanheads/git-mcp-server
  - Párametros:
    - Path

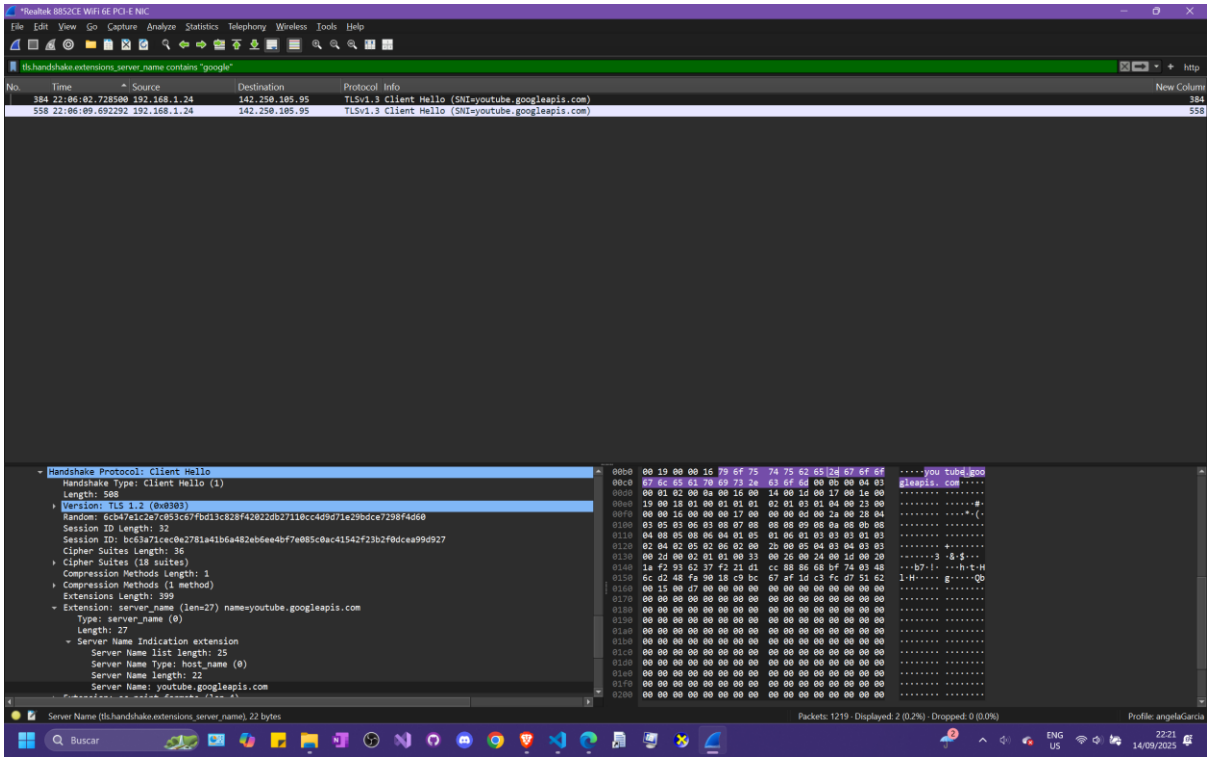
        - Message
        - Remote_url
        - branch
    - Endpoints:
        - git_set_working_dir
        - git_init
        - git_add
        - git_commit
        - git_log
        - git_status

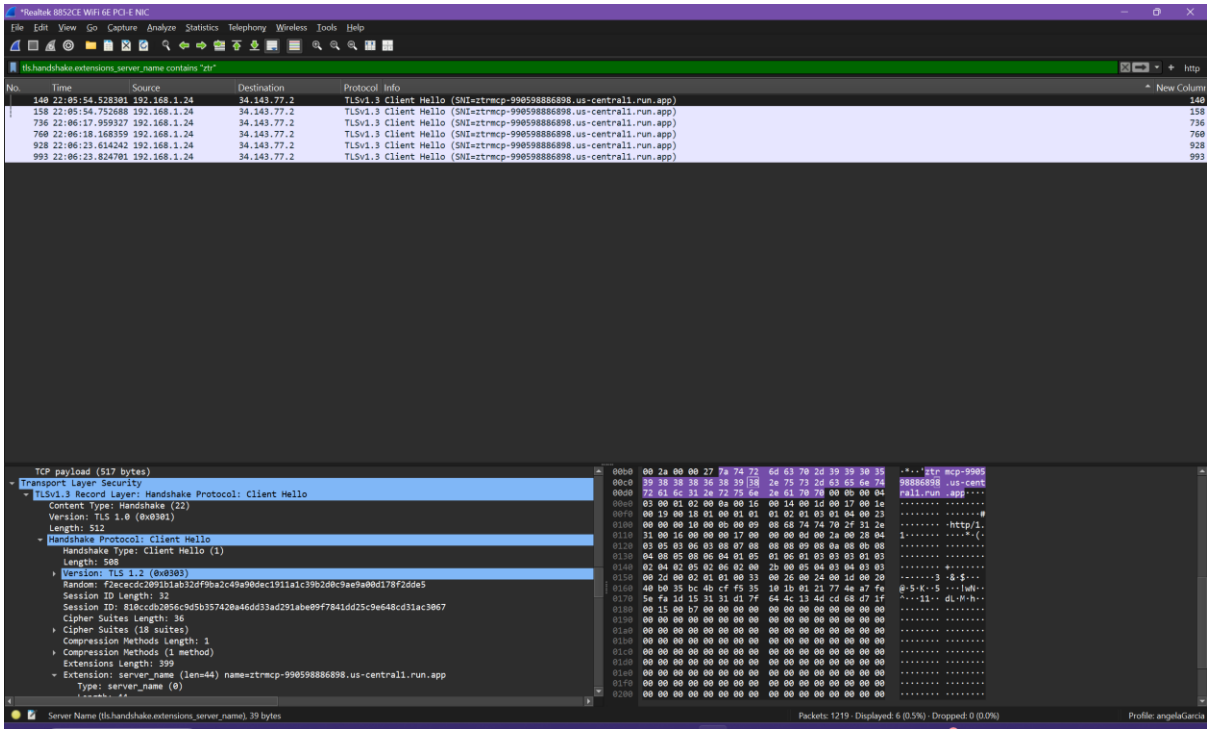
- **MCP “explore youtube trends”**
    - Tiene el proposito de detectar y explorar tendencias de YouTube.
    - Párametros:
        - Region
        - Limit
        - Keywords
    - Endpoints:
        - Yt_init
        - Yt_list_regions
        - Yt_list_Categories
        - Yt_fetch_most_popular
        - Yt_register_keywords
        - Yt_search_recent
        - Yt_calc_trends
        - Yt_trend_details
        - Yt_export_report

MCP remoto:

- **MCP de zotero**
    - Crea referencias por medio de un enlace de internet.
    - Servidor remoto desplegado en Google Cloud Run, por vía HTTP/SSE
    - Párametros:
        - Url
        - Style
        - Locale
    - Enpoints:
        - Apa_from_url

Análisis de la comunicación entre servidor remoto y cliente usando WireShark, capturar las interacciones entre el anfitrión y el servidor. Indicar mensajes JSON-RPC corresponden a los mensajes de sincronización, cuáles a solicitud o petición y a cuáles las respuestas





Capas para Youtube MCP y Zotero MCP cuando hay comunicación hacia internet:

Capa enlace: Ethernet II

Capa red: IP v4

Capa trasporte: TCP

Capa de sesión: TLS

En este caso JSON-RPC/HTTP la capa de aplicación no es visible en Wireshark porque está cifrado y acerca de TLS en la capa de transporte si es visible. Mientras que TCP/IP y Ethernet son capas inferiores.

Para los mensajes así es como se ven en los distintos MCPs:

Tipo de mensaje	YouTube MCP con STDIO	Zotero MCP con HTTP/SSE en cloud run
Sincronización	Cliente inicia sesión con el servidor local: `json { "jsonrpc": "2.0", "id": 1, "method": "initialize", "params": { "protocolVersion": "2024-11-05" } }` Respuesta: `json { "jsonrpc": "2.0", "id": 1, "result": { "capabilities": {}, "protocolVersion": "2024-11-05" } }`	Cliente inicia conexión HTTP con servidor remoto: `json { "jsonrpc": "2.0", "id": 1, "method": "initialize", "params": { "protocolVersion": "2024-11-05" } }` Respuesta: `json { "jsonrpc": "2.0", "id": 1, "result": { "capabilities": {}, "protocolVersion": "2024-11-05" } }`
Solicitud	Usuario pide listar regiones en YouTube: `json { "jsonrpc": "2.0", "id": 10, "method": "tools/call", "params": { "name": "yt_list_regions", "arguments": {} } }`	Usuario pide cita en APA de una URL: `json { "jsonrpc": "2.0", "id": 22, "method": "tools/call", "params": { "name": "apa_from_url", "arguments": { "url": "https://academia-lab.com/enciclopedia/modelo-basado-en-agentes/", "style":`

		"apa", "locale": "es-ES" } } }
Respuesta	Servidor local devuelve lista de regiones: json { "jsonrpc": "2.0", "id": 10, "result": { "regions": [{ "code": "AE", "name": "United Arab Emirates" }, { "code": "BH", "name": "Bahrain" }, { "code": "DZ", "name": "Algeria" }] } }	Servidor remoto devuelve la referencia en APA: json { "jsonrpc": "2.0", "id": 22, "result": { "references": ["Modelo basado en agentes _ AcademiaLab. (s. f.). Modelo basado en agentes _ AcademiaLab. https://academia-lab.com/enciclopedia/modelo-basado-en-agentes/"] } }

Los dos MCPs cumplen con la pila de capas que pasa de Ethernet II, IPv4, TCP hasta TLS, pero a diferencia en que el servidor de YouTube MCP el STDIO no es visible en WireShark mientras que el MCP de Zotero todo el JSON-RPC del cliente al servidor si es visible en Wireshark, pero es flujo cifrado.

Conclusiones

Se comprendió que el protocolo MCP es un estándar abierto que permite a las aplicaciones de IA acceder de manera uniforme a diversas fuentes de datos y herramientas, esto lo convierte en un estándar que facilita a la interoperabilidad y reduce la complejidad de desarrollo. En la práctica se logró implementar 4 tipos de MCPs distindos, tanto de modalidad local y remota, lo que permitió comparar las diferencias en la comunicación y observar en Wireshark la pila OSI hasta la capa de sesión TLS. Además, se identificaron los tipos de mensajes JSON-RPC, sincronización, solicitud y respuesta. Se comprendió como es que los LLM interactúan a nivel de API con estos servidores. Este proyecto, permitió implementar y probar MCPs y profundizar en la relación entre los protocolos de red y modelos de lenguaje, conociendo bases útiles para integrar nuevas herramientas en arquitecturas de IA más robustas.

Comentarios del proyecto

El proyecto es interesante, sin embargo, en un principio me costó entender bien acerca de lo que se tenía que hacer (más que nada la parte programada, porque entenderlo desde claude es fácil). Al igual que al no tener una idea más atractiva que “Explore youtube trends”. Considero que es más interesante y útil el de crear referencias, que es el remoto y me hubiera gustado realizar algo más complejo y útil en mi servidor local. Al igual, al implementarlas me encontré con varias funciones y documentación que ya no

funcionaban porque ya están deprecadas. Además, al estar desarrollando el código me di cuenta de que hay muchas formas de implementar los MCPs e incluso hay algo que me gusto que vi, pero por el tiempo ya no lo implemente y es acerca de políticas con probabilidades y si el agente no tiene un porcentaje de seguridad del 75% que caiga al fallback. Otra cosa, algo que me confundió, es que en el documento dice que Anthropic regala a sus usuarios $5 de prueba, lo cual al probarlo no fue verdad :(. Entonces, al usar OpenIA no cobra los $5, sino que uno coloca un budget y cobran por llamadas y al comparar precios con Anthropic es más barato dependiendo de los modelos que se usen (en mi caso use GPT4-oMini), además de que da la opción de compartir el proyecto y la api key con más personas desde la misma plataforma. Y por último, en lo personal creo que me hubiera gustado hacer el proyecto en otro lenguaje porque no me gusta usar python para muchos archivos, el proyecto considero que en cuanto a código puede mejorar aún.

Referencias (Hechas con el mcp):

- Uso del servidor MCP de GitHub - Documentación de GitHub. (s. f.). Uso del servidor MCP de GitHub - Documentación de GitHub. https://docs-internal.github.com/es/copilot/how-tos/provide-context/use-mcp/use-the-github-mcp-server
- Servidor FileSystem MCP independiente de la plataforma para operaciones seguras de archivos. (2025). Servidor FileSystem MCP independiente de la plataforma para operaciones seguras de archivos. https://creati.ai/es/mcp/filesystem-mcp-server-7863/
- Build an MCP server. (s. f.). Build an MCP server. https://modelcontextprotocol.io/docs/develop/build-server