Proyecto 1 – Uso de un protocolo existente (MCP)

CC3067 Redes – Universidad del Valle de Guatemala

Estudiante: Eber Alexander Cuxé Tan **Carné:** 22648

Fecha: 23/09/2025

Índice

1. Antecedentes

Los Large Language Models (LLMs) requieren “herramientas” para interactuar con el mundo real. MCP (Model Context Protocol) propone un estándar abierto para integrar herramientas a agentes/LLMs, desacoplando la implementación de la herramienta del modelo. MCP utiliza **JSON-RPC** y define roles de **host**, **cliente** y **servidor** para interoperabilidad y reutilización. (Guía del curso y recursos MCP: ver Referencias).

2. Objetivos del proyecto

- Implementar un protocolo basado en estándares (MCP/JSON-RPC).
- Comprender la arquitectura/servicios de MCP.
- Implementar un **servidor MCP local** y un **servidor MCP remoto**, y consumirlos desde el host conversacional.
- Integrar un LLM por API con **modo planificador** (planner) que decide cuándo llamar herramientas.
- Capturar con **Wireshark** las interacciones host ↔ servidor remoto y **clasificar** los mensajes JSON-RPC (sincronización, solicitud, respuesta) explicando por **capas**.

3. Arquitectura y alcance

- **Host (chat por terminal):** LLM on/off, **modo planificador**, logging.
- **Cliente STDIO:** JSON-RPC por STDIO binario (framing `Content-Length` + `Content-Type`).
- **Servidor MCP Local (BearingPro-MCP):** selección/verificación de rodamientos con catálogo local, preparado para migración a **API SKF** sin tocar el host.
- **Servidor MCP Remoto (Cloud Run):** endpoint HTTP `/mcp` con `initialize` y `tools/call`.

Flujo (planner): Usuario → Host → LLM (plan JSON) → Host llama Tool MCP → Host devuelve observación → LLM responde.

4. Características implementadas

- Chat por terminal con LLM on/off.
- Planner: la LLM propone un plan JSON (`action: call_tool|answer`); el host llama MCP y realimenta resultados.
- MCP local (STDIO): tools `catalog_list`, `select_bearing`, `verify_point`.
- Catálogo enriquecible por `catalog.json`.
- MCP remoto (Cloud Run): tools demo `echo`, `time_now`, `add`.
- Logs de interacción.

5. Instalación y uso

Requisitos: Python 3.11+, `pip install anthropic requests` (LLM opcional).

Variables de entorno (Windows):

```
set BEARINGPRO_CMD=python local_servers/bearingpro/main.py
set REMOTE_MCP_URL=https://<service>.run.app/mcp
set ANTHROPIC_API_KEY=...   (opcional)
set ANTHROPIC_MODEL=...     (opcional)
```

Ejecutar host: `py -m host.chat`

Comandos:

- Planner `modo planner on/off`
- Local `catálogo`, `selección Fr=.. Fa=.. rpm=.. L10h=..`, `verificar <modelo> con Fr=.. rpm=..`
- Remoto `remoto init`, `remoto hora`, `remoto suma 3 4`
- LLM `modo llm on/off`, `modo llm on (por defecto).`

6. Servidores MCP desarrollados

6.1 MCP Local — BearingPro-MCP

Tools:

- `catalog_list()` → { ok, models:[...] }
- `select_bearing(Fr_N, Fa_N, rpm, L10h_target)` → { ok, candidates:[...], P_equiv_N }
- `verify_point(model, Fr_N, Fa_N, rpm, L10h_target)` → { ok, ..., L10h_pred, meets_target, margin_percent }

Cálculos: P ≈ Fr + 1.5·Fa; L10h = (C/P)^3 · 1e6 / (60·rpm).

6.2 MCP Remoto — Cloud Run (HTTP)

Endpoint: POST `/mcp` (JSON-RPC: `initialize`, `tools/call`). Tools: `echo`, `time_now`, `add`.

7. Pruebas y resultados

- **Local:** selección/validación con pocos datos, planner operativo.
- **Remoto:** `remoto init`; `remoto hora` y `remoto suma` responden correctamente.

8. Análisis de comunicación (Wireshark)

8.1 Procedimiento

Filtros: SNI `tls.handshake.extensions_server_name contains "remote-mcp"`, TCP 443 `tcp.port == 443`, Stream `tcp.stream == N`, QUIC/HTTP3 `quic || http3`. (Opcional) `SSLKEYLOGFILE` para descifrar TLS/QUIC.

8.2 Evidencias

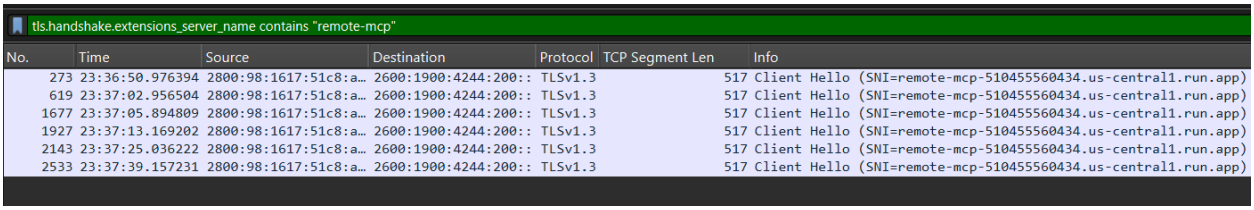


Figura 1 – Client Hello con SNI al servicio de Cloud Run.

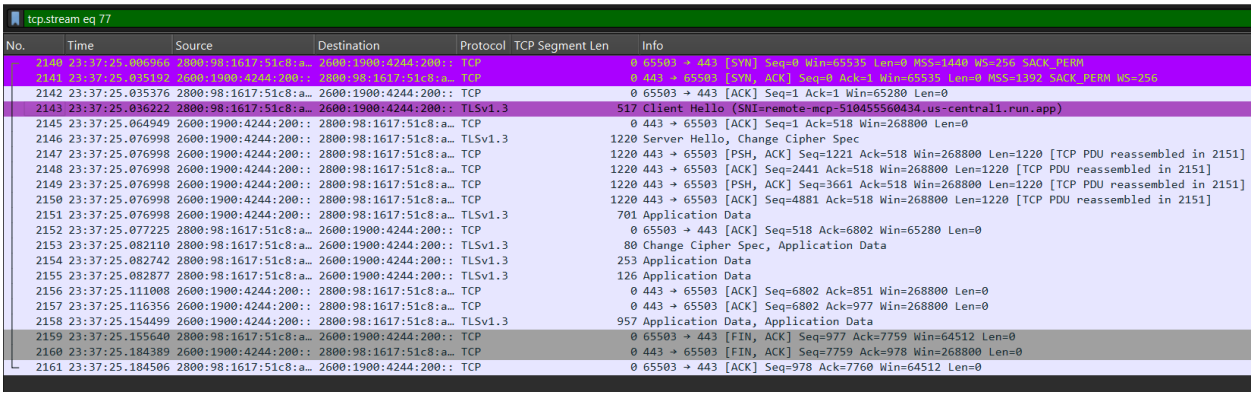


Figura 2 – Handshake TCP: SYN, SYN/ACK, ACK.

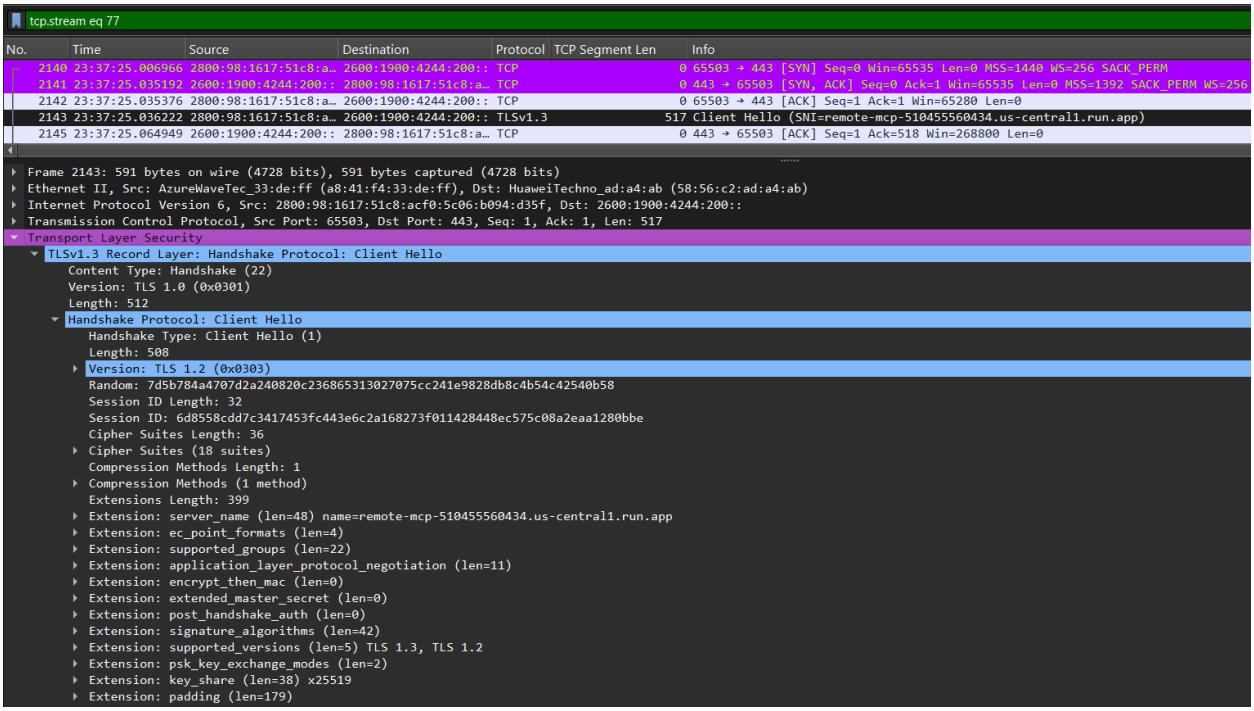


Figura 3a – TLS 1.3 Client Hello.

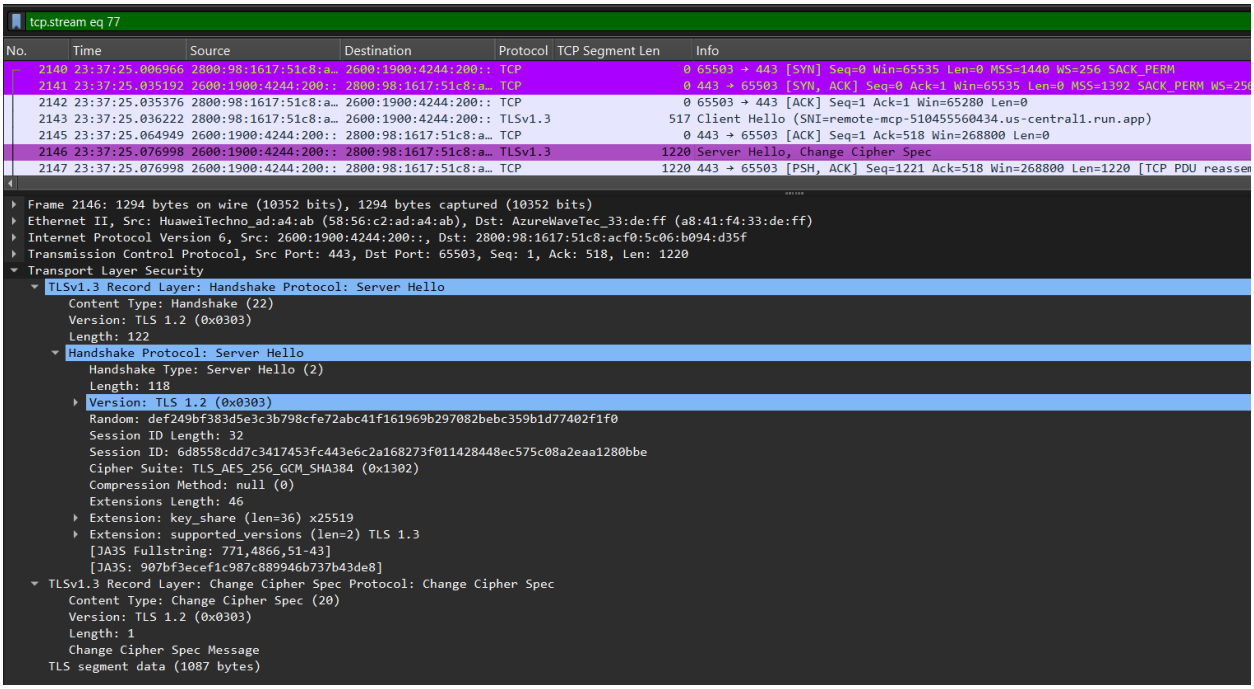
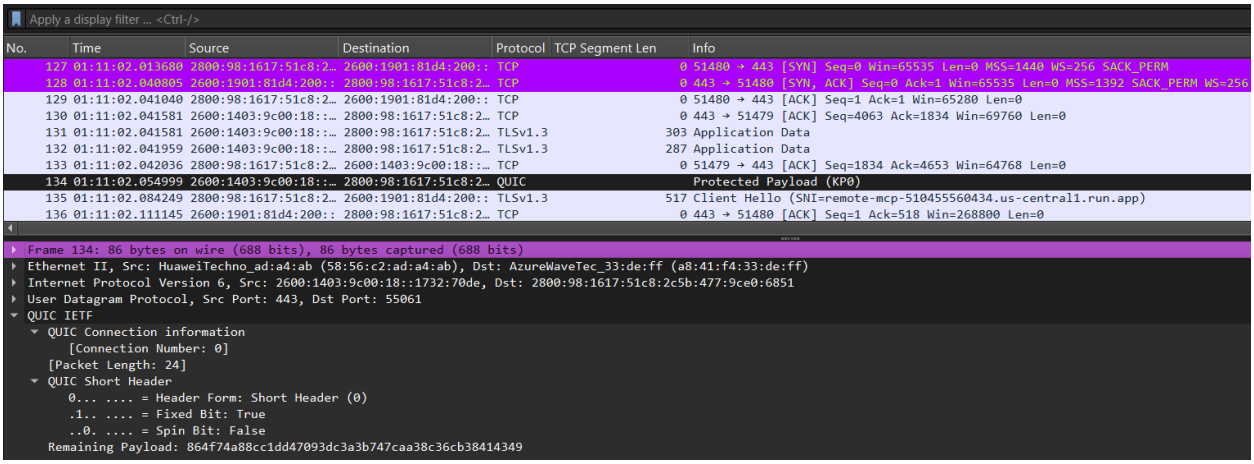


Figura 3b – TLS 1.3 Server Hello.



Figuras 4–5 – Tramas equivalentes a* initialize *(sincronización) y* tools/call *(solicitud/respuesta) en TLS/QUIC.

8.3 Clasificación JSON-RPC

- **Sincronización:** `initialize` (primer POST a `/mcp`).
- **Solicitud:** `tools/call` (p.ej., `time_now`, `add`).
- **Respuesta:** JSON con `result`/`error` del servidor.

8.4 Explicación por capas

- **Enlace:** tramas Ethernet/802.11 entre NIC y AP.
- **Red (IP):** IPv6 pública.
- **Transporte:** TCP/443 (3-way handshake, TLS 1.3) y/o QUIC/UDP/443 (Protected Payload).
- **Aplicación:** HTTP/2 o HTTP/3 sobre TLS 1.3; cuerpo con JSON-RPC (*initialize* / *tools/call* / *result*).

9. Conclusiones

- La arquitectura MCP modular permite que la LLM use datos locales y facilita agregar servidores locales/remotos.
- Se validó un servidor remoto en Cloud Run y su consumo desde el host.
- El modo planificador permitió que la LLM use datos locales sin acceso directo a archivos
- Las capturas Wireshark permiten clasificar sincronización/solicitud/respuesta y explicar el tráfico por capas.
- Al hablar JSON-RPC estándar (por STDIO y HTTP), el host pudo integrar tanto un servidor local como uno remoto sin cambios drásticos en la capa de aplicación.

10. Comentarios

- La migración en la obtención de datos de un archivo JSON a una API, como la de SKF se contempló, por plazos de validación con la empresa no se pudo implementar más que con parte de la información pública.
- Es un reto la implementación de los servidores oficiales de Anthropic. La documentación es extensa, confusa en algunos momentos, en mi caso, lo atribuyo a la falta de conocimiento del área. Mucha información nueva en mi caso.
- El proyecto es atractivo y explora un poco el mundo de los estándares digitales. Es fascinante el simple hecho de imaginar cómo está todo interconectado y cómo hacen para poder tener interoperabilidad cuando a veces no la hay.

11. Referencias

- JSON-RPC — https://www.jsonrpc.org/
- MCP Architecture — https://modelcontextprotocol.io/docs/learn/architecture
- Cloud Run tutorial — https://cloud.google.com/blog/topics/developers-practitioners/build-and-deploy-a-remote-mcp-server-to-google-cloud-run-in-under-10-minutes
- **Professor de Redes de Computadores (GPT-5 Thinking).** (2025). Asistencia conversacional durante el desarrollo del proyecto MCP (sesiones de chat). OpenAI.