Universidad del Valle de Guatemala
Facultad de Ingeniería
Redes

“Protocolo Existente”

Davis Roldán 22672

14 de septiembre del 2025, Guatemala de la Asunción

Especificación de los servidores MCP desarrollados (HTTP y STDIO): endpoints, métodos, parámetros

- Estándar común: Model Context Protocol (MCP) sobre JSON-RPC 2.0.
- Descubrimiento de capacidades: initialize → tools/list → tools/call.
- Contrato de entrada/salida: JSON estricto; cada tool publica su inputSchema (campos required y properties).

Servidor MCP HTTP/HTTPS (remoto)

- Transporte: HTTP/HTTPS.
- Endpoint único: POST /mcp
- Headers: Content-Type: application/json (y Accept: application/json)
- Negociación de versión: protocolVersion (ej. 2024-09 o 2025-06-18)

Initialize

```
{
 "jsonrpc": "2.0",
 "id": 1,
 "method": "initialize",
 "params": {
  "protocolVersion": "2024-09",
  "capabilities": { "tools": {}, "resources": {}, "prompts": {} },
  "clientInfo": { "name": "mcp-chatbot", "version": "0.1.0" }
 }
}
```

Respuesta típica de éxito:


```
{
"jsonrpc":"2.0","id":3,"result":{"ok":true,"data":{"valid":true,"reason":"syntax+mx_ok"}} }
```


Servidores MCP STDIO (locales: Filesystem y Git)

- Transporte: tuberías estándar del proceso hijo (no hay TCP/TLS).
- Lanzamiento: comando del servidor; el cliente envía/recibe líneas JSON por STDIN/STDOUT.
- Ciclo: initialize → (algunos requieren) notifications/initialized (sin id) → tools/list → tools/call.

Handshake típico por STDIO (líneas JSON):

1. Cliente → initialize (igual que HTTP)
2. (Para Git MCP y similares) Cliente → notifications/initialized (notificación sin id; no se espera respuesta, pero desbloquea tools/list)
3. Cliente → tools/list
4. Cliente → tools/call (según inputSchema de cada tool)

análisis de enlace, red, transporte y aplicación

1. **Para HTTP/HTTPS (servidor remoto):**

   Capa de Enlace (L2): transmisión local (Wi-Fi/Ethernet), tramas 802.11/802.3, direcciones MAC, FCS y posibles retransmisiones locales.

   Capa de Red (L3): IP enruta paquetes desde la LAN del cliente hacia la IP pública del servicio (Render); NAT en el borde del cliente; TTL, posible fragmentación.

   Capa de Transporte (L4): TCP establece conexión fiable (3-way handshake), control de flujo y congestión; si es HTTPS, se superpone TLS (negociación de cifrados, autenticación del servidor, canal cifrado e íntegro).

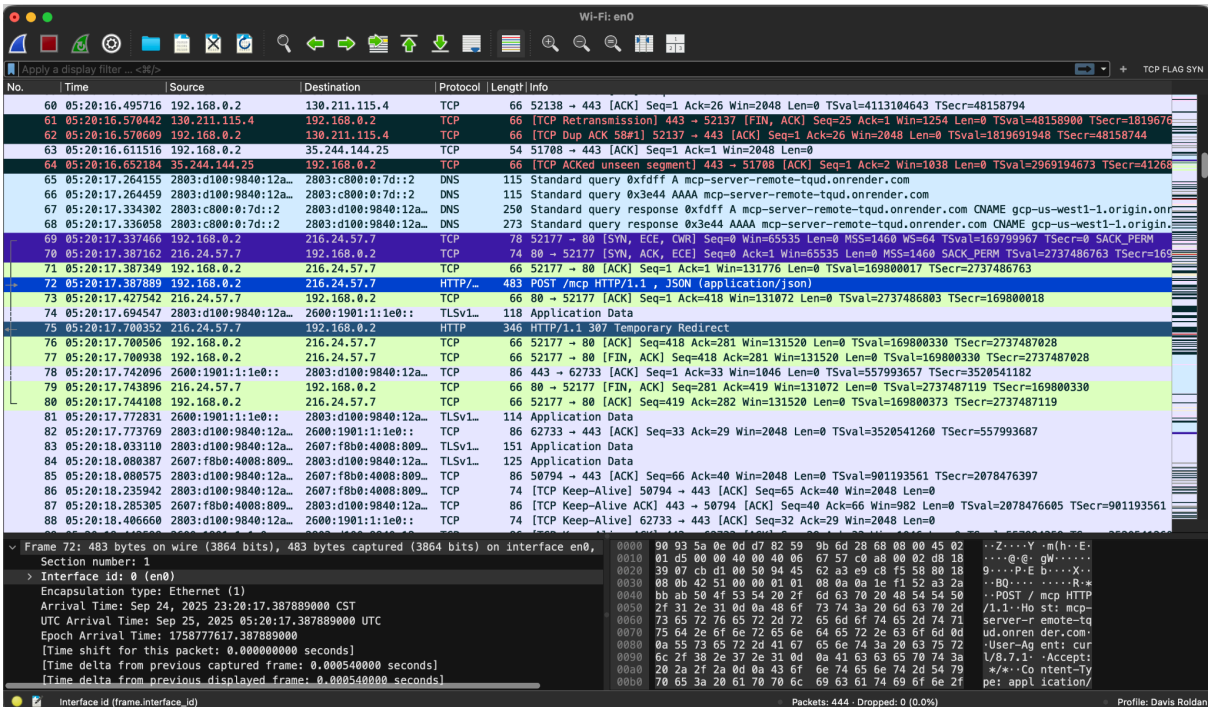
   Capa de Aplicación (L7): HTTP POST a /mcp con JSON-RPC 2.0. El cliente negocia protocolVersion en initialize, descubre tools con tools/list y ejecuta tools/call con arguments que cumplen el inputSchema. El servidor valida, ejecuta y responde con result o error.

2. **Para STDIO (servidores locales en el mismo host):**

   Capa de Enlace / Red / Transporte: no hay tránsito en red; el intercambio se realiza por STDIN/STDOUT del proceso (pipes del SO). No existe TCP/TLS ni direccionamiento IP (se evita latencia de red).

   Capa de Aplicación (L7): idéntica semántica MCP/JSON-RPC que en HTTP: initialize, (opcional notifications/initialized), tools/list, tools/call. El cliente y el proceso hijo se comunican con líneas JSON. El Filesystem MCP aplica un directorio raíz permitido para seguridad.

Análisis Wireshark



Entorno observado

- Cliente (LAN): 192.168.0.2
- Servidor remoto (Render): 216.24.57.7
- Servicio: MCP sobre HTTP→HTTPS (redirect a 443/TLS)
- Interfaz capturada: Wi-Fi en0

TCP 3-way handshake (puerto 80)

- 192.168.0.2 → 216.24.57.7 :80 SYN (MSS=1460, WS=64, SACK_PERM)
- 216.24.57.7 → 192.168.0.2 :80 SYN, ACK
- 192.168.0.2 → 216.24.57.7 :80 ACK

Aplicación (HTTP en claro, puerto 80)

- Request: POST /mcp HTTP/1.1 con Content-Type: application/json ⇒ JSON-RPC (Solicitud). Es el primer intercambio MCP; en este punto, típicamente es el initialize (sincronización de protocolo).
- Response: HTTP/1.1 307 Temporary Redirect ⇒ el backend redirige a HTTPS (puerto 443).
- Nuevo TCP 3-way handshake (puerto 443)
- 192.168.0.2 → 216.24.57.7 :443 SYN (parámetros similares: MSS, WS, SACK)
- 216.24.57.7 → 192.168.0.2 :443 SYN, ACK
- 192.168.0.2 → 216.24.57.7 :443 ACK

TLS handshake (puerto 443)

- Client Hello (con SNI al host del servicio)
- Server Hello (+ negociación de cifrado)

A partir de aquí, Application Data (cifrado TLS) — el contenido HTTP/JSON-RPC ya no es visible.

Tráfico de aplicación (cifrado)

- Ráfagas Client→Server y Server→Client como TLS Application Data.
  En este tramo viajan:
  JSON-RPC (Solicitudes) como tools/list y tools/call (POST /mcp sobre HTTPS).
  JSON-RPC (Respuestas) emparejadas por el mismo id de JSON-RPC.

No se ven los cuerpos por estar cifrados; para verlos se necesita SSLKEYLOGFILE o capturar contra HTTP local.

Paso	Capa	Dirección	Puerto	Detalle	Rol JSON-RPC
1	L4	192.168.0.2 → 216.24.57.7	80	TCP SYN, MSS=1460, WS=64, SACK	—
2	L4	216.24.57.7 → 192.168.0.2	80	TCP SYN, ACK	—
3	L4	192.168.0.2 → 216.24.57.7	80	TCP ACK	—
4	L7	192.168.0.2 → 216.24.57.7	80	HTTP POST /mcp Content-Type: application/json	Solicitud (p.ej. initialize)
5	L7	216.24.57.7 → 192.168.0.2	80	HTTP/1.1 307 Temporary Redirect	—
6	L4	192.168.0.2 →	443	TCP SYN	—

		216.24.57.7			
7	L4	216.24.57.7 → 192.168.0.2	443	TCP SYN, ACK	—
8	L4	192.168.0.2 → 216.24.57.7	443	TCP ACK	—
9	L7	192.168.0.2 ↔ 216.24.57.7	443	TLS handshake (ClientHello/ServerHello/Finished)	—
10	L7	192.168.0.2 ↔ 216.24.57.7	443	TLS Application Data (HTTP POST /mcp + JSON-RPC)	Solicitudes/Respuestas

Conclusión

El proyecto demuestra que MCP, sobre un mismo contrato JSON-RPC, permite operar tanto en HTTP (remoto, escalable y seguro con TLS) como en STDIO (local, rápido y con acceso controlado al entorno). Esta convergencia reduce el acoplamiento: el cliente negocia initialize, descubre tools/list y ejecuta tools/call sin importar el transporte. Además, los normalizadores y guardas en el cliente (validación de argumentos, mapeo server→tool y notificación initialized donde aplica) vuelven la ejecución robusta y predecible. En seguridad, HTTPS y la restricción de directorios en Filesystem MCP ofrecen un equilibrio práctico entre exposición y protección. En conjunto, la arquitectura es simple de extender (añadir nuevas tools), fácil de mantener y lista para endurecer con observabilidad (métricas, trazabilidad) y controles de acceso más finos según crezcan las necesidades.