

Proyecto 1

Handshake:

228 13.617076	192.168.86.63	54.243.238.66	TLSv1.2	571 Client Hello (SNI=one-piece-mcp-server-e56917b5ada1.herokuapp.com)
266 14.569083	192.168.86.63	54.243.238.66	TLSv1.2	571 Client Hello (SNI=one-piece-mcp-server-e56917b5ada1.herokuapp.com)

Session ID: 0bd078e2cb542e2c28fb84b76c2171155c3a5272119676f77aeada25bc08f1eb	00a0 00 9e 00 6b 00 67 00 ff 01 00 01 8f 00 00 34	... k g4
Cipher Suites Length: 36	00b0 00 32 00 00 2f 6f 6e 65 2d 70 69 65 63 65 2d 6d	2- /one-piece-m
Cipher Suites (18 suites)	00c0 63 70 2d 73 65 72 76 65 72 2d 65 35 36 39 31 37	cp-serve r-e56917
Compression Methods Length: 1	00d0 62 35 61 64 61 31 2e 68 65 72 6f 6b 75 61 70 70	b5ada1.h erokuapp
Compression Methods (1 method)	00e0 2e 63 6f 6d 00 00 00 04 03 00 01 02 00 0a 00 16	.com
Extensions Length: 399	00f0 00 14 00 1d 00 17 00 1e 00 10 00 18 01 00 01 01
Extension: server_name (len=52) name=one-piece-mcp-server-e56917b5ada1.herokuapp.com	0100 01 02 01 03 01 04 00 23 00 00 00 10 00 0b 00 09	... # ...
Type: server_name (0)	0110 08 68 74 74 70 2f 31 2e 31 00 16 00 00 00 17 00	http/1.1 ...
Length: 52	0120 00 00 0d 00 2a 00 28 04 03 05 03 06 03 08 07 08	* (...
Server Name Indication extension	0130 00 08 09 08 0a 08 0b 08 04 08 05 08 06 04 01 05
Server Name list length: 50	0140 01 06 01 03 03 03 01 03 02 04 02 05 02 06 02 00
Server Name Type: host_name (0)	0150 2b 00 05 04 03 04 03 03 00 2d 00 02 01 01 00 33	+3
Server Name length: 47	0160 00 26 00 24 00 1d 00 20 e5 3a 63 2d 6c 6d b5 19	& \$... c-lm
Server Name: one-piece-mcp-server-e56917b5ada1.herokuapp.com	0170 ff 14 48 d8 9a 3d 51 81 78 1c 7f 3c e6 aa e2 23	-H =0 x... #
Extension: ec_point_formats (len=4)	0180 df a4 ec ee e9 2c 30 5f 00 15 00 af 00 00 00 00
Type: ec_point_formats (11)	0190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Length: 4	01a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EC point formats Length: 3	01b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Elliptic curves point formats (3)	01c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Extension: supported_groups (len=22)	01d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Type: supported_groups (10)	01e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
	01f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Sync:

9007 557.736152	192.168.86.63	54.243.238.66	HTTP/JSON	149 POST /mcp HTTP/1.1 , JSON (application/json)
-----------------	---------------	---------------	-----------	--

Ethernet II, Src: HP_81:2e:8c (bc:e9:2f:81:2e:8c), Dst: Google_d5:a3:a3 (e4:5e:1b:d5:a3:a3)	0000 50 4f 53 54 20 2f 6d 63 70 20 48 54 50 2f 31	POST /mc p HTTP/1
Internet Protocol Version 4, Src: 192.168.86.63, Dst: 54.243.238.66	0010 2e 31 0d 0a 48 6f 73 74 3a 20 6f 6e 65 2d 70 69	.1 Host : one-pl
Transmission Control Protocol, Src Port: 56491, Dst Port: 443, Seq: 986, Ack: 4460, Len: 95	0020 65 63 65 2d 6d 63 70 2d 73 65 72 76 65 72 2d 65	eece-mcp-serve-r
Transport Layer Security	0030 35 36 39 31 37 62 35 61 64 61 31 2e 68 65 72 6f	56917b5a da1.hero
[2 Reassembled TLS segments (379 bytes): #9006(313), #9007(66)]	0040 6b 75 61 70 70 2e 63 6f 6d 0d 0a 41 63 63 65 70	kuapp.co m Accep
Hypertext Transfer Protocol	0050 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70	t-encodi ng: gcp
POST /mcp HTTP/1.1\r\n	0060 63 7a 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76	, deflat e Conne
Host: one-piece-mcp-server-e56917b5ada1.herokuapp.com\r\n	0070 65 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 70	ction: k eep-aliv
Accept-Encoding: gzip, deflate\r\n	0080 79 74 68 6f 6e 2d 68 74 74 70 78 2f 30 2e 32 38	e -User- Agent: p
Connection: keep-alive\r\n	0090 2e 31 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65	ython-ht tpx/0.28
User-Agent: python-https/0.28.1\r\n	00a0 6f 6e 0d 0a 41 63 63 65 70 74 3a 20 61 70 70 6c	.1 Cont ent-Type
Content-Type: application/json\r\n	00b0 69 63 61 74 69 6f 6e 2f 6a 73 6f 6e 2c 20 74 65	: applic ation/js
Accept: application/json, text/event-stream\r\n	00c0 78 74 2f 65 76 65 6e 74 2d 73 74 72 65 61 6d 0d	on Acce pt: appl
mcp-session-id: 817ae71b9da247d9b26e08e09f6a70bb\r\n	00d0 0a 6d 63 70 2d 73 65 73 73 69 6f 6e 2d 69 64 3a	ication/ json, te
Content-Length: 66\r\n	00e0 20 38 31 37 61 65 37 31 62 39 64 61 32 34 37 64	xt/event -stream
[Content length: 66]	00f0 39 62 32 36 65 30 38 65 30 39 66 36 61 37 30 62	-mcp-ses sion-id:
\r\n	0100 62 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74	817ae71 b9da247d
[Request in frame: 9010]	0110 68 3a 20 36 36 0d 0a 0d 0a 7b 22 6a 73 6f 6e 72	9b26e08e 09f6a70b
[Full request URI: https://one-piece-mcp-server-e56917b5ada1.herokuapp.com/mcp]	0120 70 63 22 3a 22 32 2e 30 22 2c 22 69 64 22 3a 22	b. Conte nt-length
File Data: 66 bytes	0130	h: 66; ("json
	0140	pc":2.0 , "id":

9010 557.820425	54.243.238.66	192.168.86.63	HTTP	802 HTTP/1.1 200 OK (text/event-stream)
-----------------	---------------	---------------	------	---

Cache-Control: no-cache, no-transform\r\n	02d0 68 65 72 6f 6b 75 2d 72 6f 75 74 65 72 0d 0a 58	heroku-r outer-X
Content-Type: text/event-stream\r\n	02e0 2d 41 63 63 65 6c 2d 42 75 66 66 65 72 69 6e 67	-Accel-B offering
Date: Tue, 23 Sep 2025 23:09:31 GMT\r\n	02f0 3a 20 6e 6f 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65	: no Co ntent-Le
Mcp-Session-Id: 817ae71b9da247d9b26e08e09f6a70bb\r\n	0300 6e 67 74 68 3a 20 31 33 39 37 0d 0a 0d 0a 65 76	ngth: 13 97... ev
Nel: {"report_to": "heroku-nel", "response_headers": ["Via"], "max_age": 3600, "success_fraction": 0.01, "f	0310 65 6e 74 3a 20 6d 65 73 73 61 6f 65 0d 0a 64 61	mits mes sage-da
Report-To: {"group": "heroku-nel", "endpoints": [{"url": "https://nel.heroku.com/reports?s=j%2FFlPxs21%2FbQkUICP1Cn13aNOYMD	0320 74 61 3a 20 7b 22 6a 73 6f 6e 72 70 63 22 3a 22	tp: ("js oncp"
Servers: Heroku\r\n	0330 32 2e 30 22 2c 22 69 64 22 3a 22 74 6f 6f 6e 73	2.0"; "id "tools
Via: 1.1 heroku-router\r\n	0340 2d 31 22 2c 22 72 65 73 75 6c 74 22 3a 7b 22 74	-1", "res ult": ("t
X-Accel-Buffering: no\r\n	0350 6f 6f 6e 73 22 3a 5b 7b 22 6e 61 6d 65 22 3a 22	ools":{"name": "
Content-Length: 1397\r\n	0360 6f 70 5f 67 65 74 5f 63 68 61 72 61 63 74 65 72	op_get_c haracter
[Content length: 1397]	0370 73 22 2c 22 64 65 73 63 72 69 70 74 69 6f 6e 72	s", desc ription"
\r\n	0380 3a 22 47 65 74 20 61 6c 6c 20 63 68 61 72 61 63	: "Get all l charac
[Request in frame: 9007]	0390 74 65 72 73 20 6e 72 6f 6d 20 74 68 65 20 4f 6e	ters fro m the On
[Time since request: 0.084273000 seconds]	03a0 65 20 50 69 65 63 65 20 41 50 49 2e 5c 6e 52 65	e Piece API, nke
[Request URI: /mcp]	03b0 74 75 72 6e 73 3a 5c 6e 20 20 20 20 6c 69 73 74	turns:\n list
[Full request URI: https://one-piece-mcp-server-e56917b5ada1.herokuapp.com/mcp]	03c0 3a 20 41 20 6c 69 73 74 20 6f 66 20 63 68 61 72	: A list of char
File Data: 1397 bytes	03d0 61 63 74 65 72 73 2e 22 2c 22 69 6e 70 75 74 53	acters": "inputS
Media Type	03e0 63 68 65 6d 61 22 3a 7b 22 70 72 6f 70 65 72 74	chema":{"propert
Media type: text/event-stream (1397 bytes)	03f0 60 65 73 22 3a 7b 7d 2c 22 74 79 70 65 22 3a 22	ies":{"type": "
	0400 6f 62 6a 65 63 74 22 7d 2c 22 5f 6d 65 74 61 22	object", "meta
	0410 3a 7b 22 5f 66 61 73 74 6d 63 70 22 3a 7b 22 74	:"fast mcp": ("t

Mensaje de solicitud:

1126 108.915663	192.168.86.55	23.22.130.173	HTTP/JSON	225 POST /mcp HTTP/1.1 , JSON (application/json)
-----------------	---------------	---------------	-----------	--

Mensaje de respuesta:

1145 109.449993 23.22.130.173 192.168.86.55 HTTP 84 HTTP/1.1 200 OK (text/event-stream)

¿Qué sucede a nivel de la capa de enlace, red, transporte y aplicación?

Durante la interacción entre mi chatbot y el MCP remoto, la capa de enlace (L2) transporta cada unidad de datos como una trama Ethernet II. En cada trama se observan las direcciones físicas locales: la MAC de destino corresponde al siguiente salto (típicamente el gateway o el switch del segmento) y la MAC de origen es la de mi NIC. El campo Type = 0x0800 indica que la carga útil entregada a la capa superior es IPv4.

Esta comunicación es local al enlace: las tramas solo viajan entre mi equipo y el dispositivo adyacente (switch/AP/gateway). En los saltos siguientes, cada equipo vuelve a re-encapsular el paquete con sus propias MAC de origen y destino; por eso, en la captura no aparece la MAC del servidor de Heroku, sino la del primer salto de mi red.

Durante la capa de red (L3) el tráfico viaja encapsulado en datagramas IPv4 desde mi host hacia la infraestructura de Heroku. En cada paquete se observan la IP de origen (mi equipo dentro de la LAN) y la IP de destino (un balanceador/edge de Heroku). Al salir de mi red, el router realiza NAT: la IP privada de mi equipo se traduce a una IP pública, por eso en Internet los paquetes aparecen con la IP pública del router y no con la privada de mi PC. A lo largo del trayecto, cada router decrementa el TTL (time-to-

live); en los paquetes recibidos desde la nube el TTL llega con un valor menor, consistente con el número de saltos recorridos.

En la capa de transporte, TCP establece la sesión (3-Way Handshake), aplica fiabilidad, control de flujo y de congestión, segmenta/reensambla datos y cierra con FIN/RST. Sobre ese flujo estable y cifrado (tras TLS) circulan los mensajes de aplicación que tú capturaste: el SYNC (tools/list), los REQUEST (tools/call) y sus RESPONSE (mismo id). La latencia de cada llamada puede medirse en L4/L7 como el cambio entre el paquete del request y el de su response dentro del mismo tcp.stream.

A nivel de aplicación se observa (1) un SYNC inicial tools/list para descubrir capacidades, requests tools/call con sus argumentos, y responses con result/error, siempre correlacionados por id. Las cabeceras Accept, Content-Type y mcp-session-id sostienen la sesión y el formato

Servidor Soccer

Tool	Endpoint (método y ruta relativa)	Parámetros (path / query)	Especificación / Descripción	Requisitos (headers / env)
get_competitions	GET /competitions	—	Lista todas las competiciones disponibles.	Header: X-Auth-Token = API_KEY. Env: API_KEY, FOOTBALL_BASE, HTTP_TIMEOUT (opcional).
get_teams_competitions	GET /competitions/{competition_id}/teams	competition_id (path, p. ej. PL, CL)	Lista los equipos de una competición.	Header: X-Auth-Token = API_KEY. Env: API_KEY, FOOTBALL_BASE, HTTP_TIMEOUT.
get_teams_by_competition	GET /competitions/{competition_id}/teams	competition_id (path)	(Funcionalidad duplicada de la anterior) Lista	Igual que arriba.

			equipos por competición.	
get_matches_by_competition	GET /competitions/{competition_id}/matches	competition_id (path)	Lista los partidos de una competición.	Header/env como arriba.
get_team_by_id	GET /teams/{team_id}	team_id (path)	Devuelve el detalle de un equipo.	Header/env como arriba.
get_top_scorers_by_competition	GET /competitions/{competition_id}/scorers	competition_id (path)	Goleadores top de una competición.	Header/env como arriba.
get_player_by_id	GET /persons/{player_id}	player_id (path)	Devuelve el detalle de un jugador/persona.	Header/env como arriba.
get_info_matches_of_a_player	GET /persons/{player_id}/matches	player_id (path)	Lista los partidos de una persona/jugador.	Header/env como arriba.

One piece server

Tool	Endpoint (método y ruta relativa)	Parámetros (path / query)	Especificación / Descripción	Requisitos (headers / env)
op_get_characters	GET /characters/en	—	Devuelve la lista de personajes.	Env: ONE_PIECE_BASE, HTTPX_TIMEOUT (opcional). Maneja errores y response.json().

op_get	GET			Env:
_chara	/character	character	Devuelve el detalle de un	ONE_PIECE_BAS
cter_b	s/en/{char	_id (path,	personaje por ID.	E,
y_id	acter_id}	int)		HTTPX_TIMEOUT.
		query		Env:
op_se	GET	opcional	Busca personajes con filtros.	ONE_PIECE_BAS
arch_c	/character	es:	Si no envías ningún filtro,	E,
haract	s/en/searc	name,	devuelve error útil. Devuelve	HTTPX_TIMEOUT.
ers	h/	job,	{url, status, result} o raw si el	Usa httpx y
		bounty,	server no manda JSON.	maneja
		age, size		redirecciones/CT.

Conclusiones y comentario sobre el proyecto

Este proyecto me permitió cerrar el ciclo completo de una integración cliente–servidor moderna: diseñé y expuse herramientas mediante MCP (Model Context Protocol), las consumí desde un chatbot, desplegué un servidor remoto y verifiqué el tráfico con Wireshark hasta clasificar los mensajes JSON-RPC en sync, request y response.

Lo más retador fue establecer la comunicación correcta con el MCP remoto. Tuve que entender cómo formatear correctamente JSON-RPC 2.0 (campos jsonrpc, id, method, params), qué cabeceras HTTP exige el servidor (Accept: application/json, text/event-stream y Content-Type: application/json) y cómo manejar el estado de sesión vía mcp-session-id.

En términos de implementación, MCP Fast hizo muy directa la creación de tools: el patrón “defino una función → la publico como tool → la invoco con tools/call” facilita pensar en la capa de aplicación sin pelear. La separación de transportes (stdio para local y streamable-http para remoto) me obligó a ser explícito al configurar el chatbot y me ayudó a entender por qué a veces “no había tráfico” (estaba usando stdio en lugar de HTTP).

Repositorios:

[Javilejoo/chatbot-soccerMCP](#)

[Javilejoo/Soccer-MCP-Server](#)