MCP-Diag: A Deterministic, Protocol-Driven Architecture for AI-Native Network Diagnostics

Devansh Lodha
devansh.lodha@iitgn.ac.in

Mohit Panchal
mohit.panchal@iitgn.ac.in

Sameer G. Kulkarni
sameergk@iitgn.ac.in

DoT 5G Use Case Lab

IIT Gandhinagar

1. The Stochastic Grounding Problem

Context: Modern infrastructure (SDN, Microservices) requires **Intent-Based** Operations, but networks require deterministic inputs.

The Conflict: LLMs are probabilistic engines. They hallucinate when interfacing with rigid Network CLIs.

- **The Input Gap:** LLMs guess capabilities (e.g., inventing flags like --force or running Cisco commands on Linux).
- **The Output Gap:** Network CLIs are unstructured. If ping returns a non-standard error, LLMs often hallucinate "Success" because it fits the probability distribution.

2. The State of the Art

System	Grounding	Governance	Interaction	State
Langchain	High Risk (Raw Text)	Client-Side (Pass-through)	Blocking	Token-Window
Shell-GPT	Medium Risk (Suggestion)	User-Gated (Manual Exec)	Real-Time	Session Cache
Anthropic Computer Use	High Risk (Visual OCR)	Isolation Only (Observation)	Real-Time	Visual History

3. The Model Context Protocol (MCP)

JSON-RPC: Universality. Any model can talk to any system.

Client-Server Architecture: Decouples the Inference Engine (Client) from the Root Shell (Server).

Primitives:

- **Tools:** Enforce strict Input Contracts so the LLM never guesses arguments.
- **Resources:** Provide context (e.g., Man Pages) to ground the model.
- **Prompts:** Define orchestrated workflows.
- **Elicitation:** Enforces Human-in-the-Loop at the protocol level.

4. The MCP-Diag Architecture

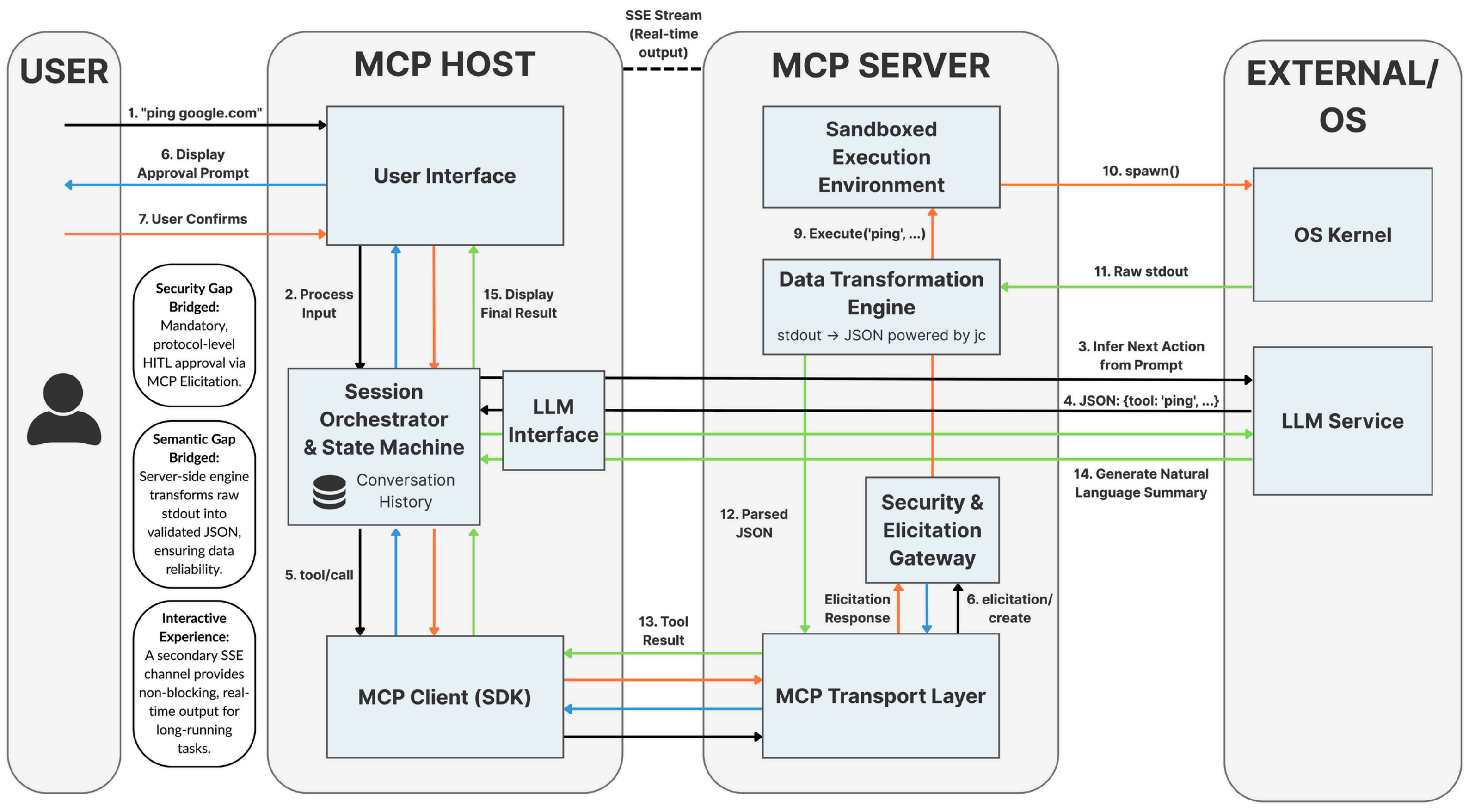
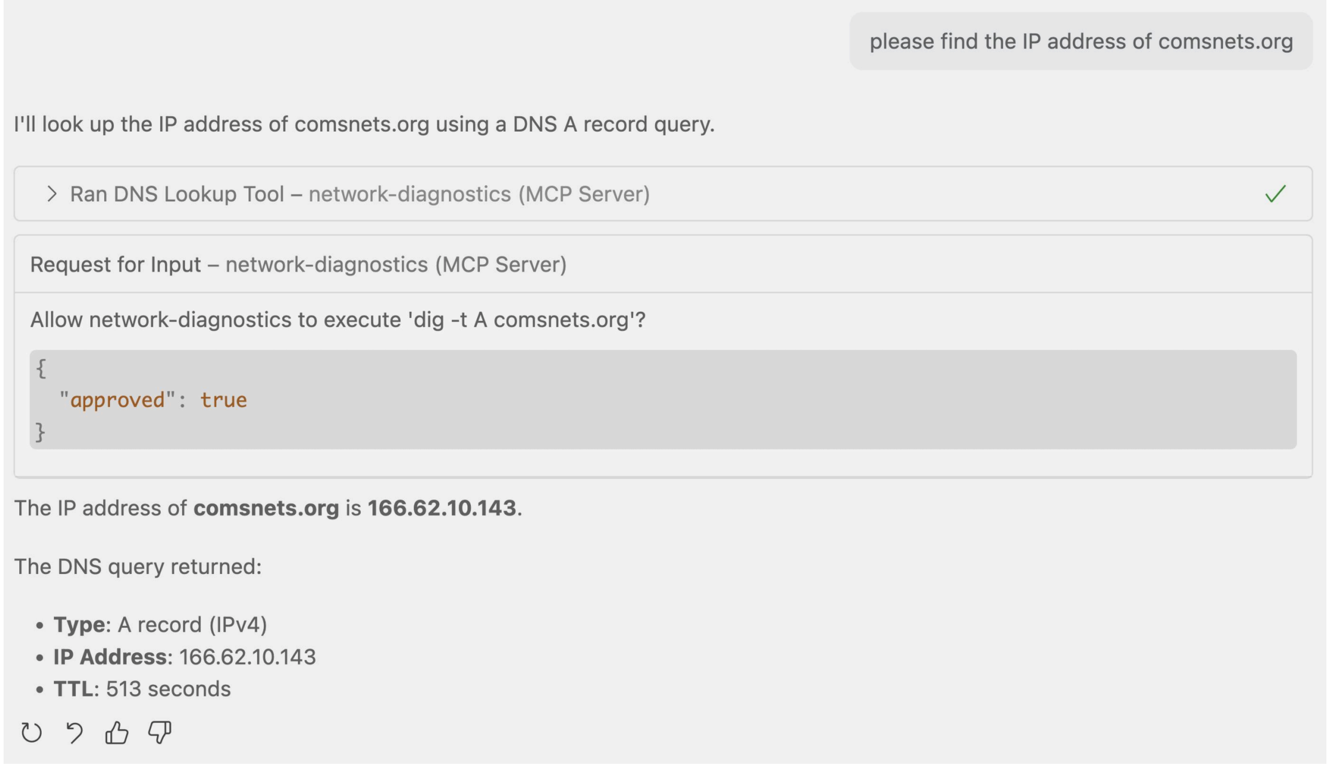

USER
MCP HOST
SSE Stream (Real-time output)
MCP SERVER
EXTERNAL/ OS
1. "ping google.com"
6. Display Approval Prompt
7. User Confirms
User Interface
2. Process Input
15. Display Final Result
Session Orchestrator & State Machine
Conversation History
LLM Interface
5. tool/call
MCP Client (SDK)
13. Tool Result
Security Gap Bridged: Mandatory, protocol-level HITL approval via MCP Elicitation.
Semantic Gap Bridged: Server-side engine transforms raw stdout into validated JSON, ensuring data reliability.
Interactive Experience: A secondary SSE channel provides non-blocking, real-time output for long-running tasks.
Sandboxed Execution Environment
9. Execute('ping', ...)
Data Transformation Engine
stdout → JSON powered by jc
12. Parsed JSON
Security & Elicitation Gateway
Elicitation Response
6. elicitation/ create
MCP Transport Layer
10. spawn()
OS Kernel
11. Raw stdout
3. Infer Next Action from Prompt
4. JSON: {tool: 'ping', ...}
LLM Service
14. Generate Natural Language Summary

Figure 1: The MCP-Diag Architecture. The Host (left) orchestrates the session, while the Server (right) executes tools. The architecture enforces **(1) Protocol-Level Governance** via a mandatory Elicitation loop, **(2) Deterministic Grounding** by transforming raw stdout into strict JSON schemas before AI ingestion, and **(3) Real-Time Telemetry** via a hybrid Stdio/SSE transport layer.

5. Validation



please find the IP address of comsnets.org

I'll look up the IP address of comsnets.org using a DNS A record query.

> Ran DNS Lookup Tool – network-diagnostics (MCP Server) ✓

Request for Input – network-diagnostics (MCP Server)

Allow network-diagnostics to execute 'dig -t A comsnets.org'?

```
{
  "approved": true
}
```

The IP address of **comsnets.org** is **166.62.10.143**.

The DNS query returned:

- **Type**: A record (IPv4)
- **IP Address**: 166.62.10.143
- **TTL**: 513 seconds

Figure 2: Validation via VS Code MCP Host. A single workflow demonstrating the architectural guarantees: **(1) Interoperability:** Running inside a standard third-party host. **(2) Reasoning:** The LLM inferred dig was required from a natural language query. **(3) Elicitation:** The protocol paused execution for user approval. **(4) Synthesis:** The agent received structured JSON, not raw text, enabling it to summarize the DNS record type and TTL accurately.

6. Performance Evaluation

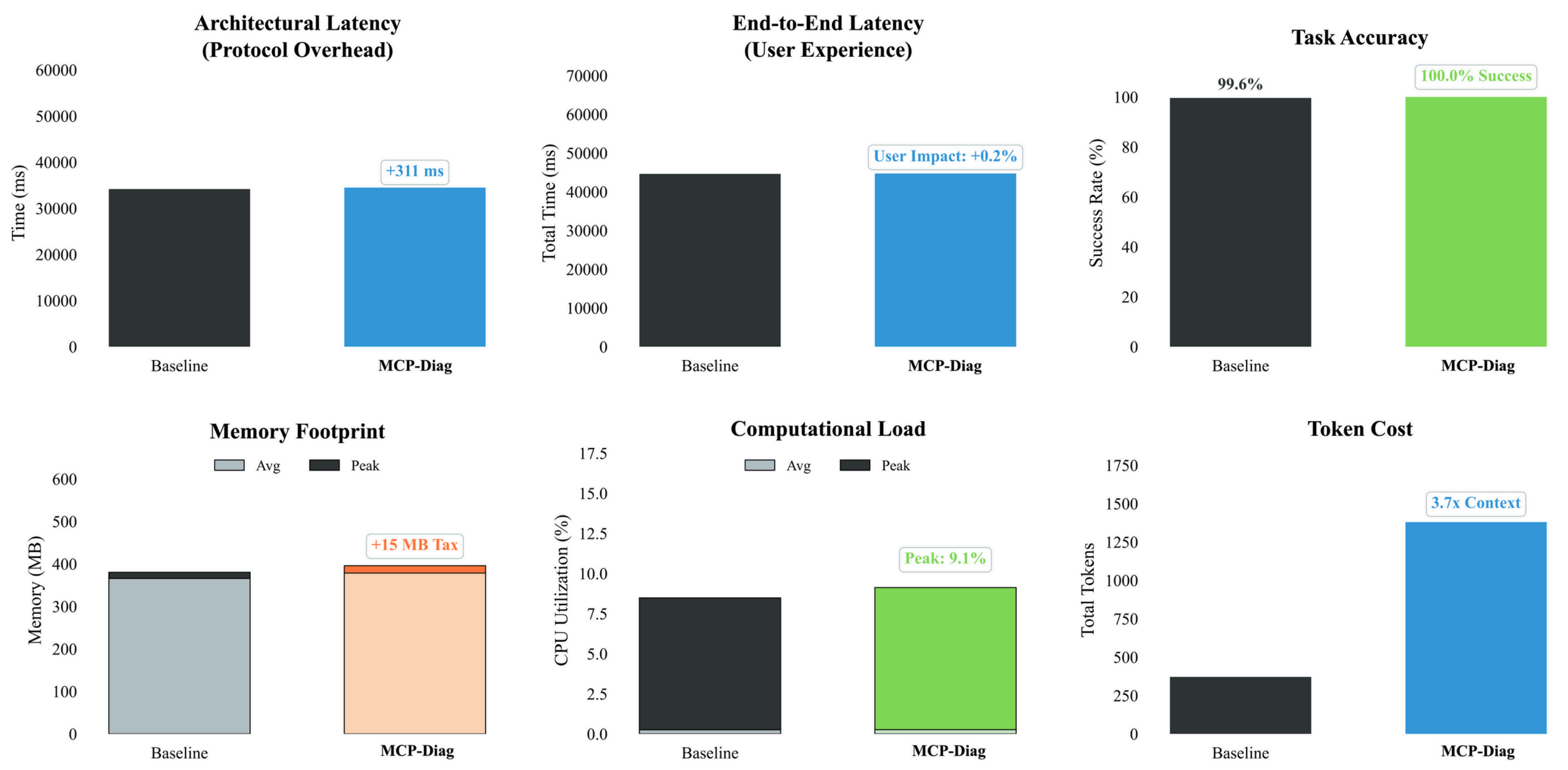


Figure 3: Performance Benchmarks (N=500). We compared MCP-Diag against a standard Baseline Agent on the Moz Top 500 domains. **Accuracy:** The Baseline failed in **0.4%** of cases due to formatting errors; MCP-Diag achieved **100% Grounding Accuracy** by enforcing schemas. **Token Cost:** We accept a **3.7x Token Overhead** to define strict contracts. **Latency:** The architectural overhead is **<0.9% (~300ms)** for long-running network tasks. **Resources:** The sidecar is lightweight, adding only **15MB** memory and **~1%** peak CPU overhead compared to the baseline.

7. Conclusion & Future Prospects

Conclusion: We solve the **Translation Gap** by stopping the LLM from guessing inputs and outputs. We address the **Governance Gap** by enforcing mandatory elicitation. Finally, we resolve the **Latency Gap** by implementing real-time streaming for long-running commands.

Future Directions:

- **Complexity:** Handling complex tools by leveraging growing LLM context windows and better parsers.
- **Autonomy:** Using MCP Prompts to define multi-step reasoning chains for autonomous root-cause analysis.
- **Automation:** Using LLMs to read man pages and automatically generate schema code, removing manual overhead.
- **Scalability:** Moving from a 1:1 topology to a Concurrent Architecture (single server, multiple clients) and eventually a distributed mesh of diagnostic agents.

Scan for the code
github.com/devansh-lodha/mcp-diag

COMSNETS

Best Graduate Forum Paper (Runner Up)
18th IEEE COMSNETS 2026
Bengaluru, India - January 2026

Acknowledgements: Supported by Govt. of India, SERB Core Research Grant (CRG/2023/008021), MeitY (4(3)/2024-ITEA), and DoT 5G Use Case Lab.