

Beyond the $M \times N$ Problem: A Comparative Architectural Analysis of Model Context Protocol (MCP) and Agent-to-Agent (A2A) Interaction Patterns

Ömer Furkan Tercan

Abstract—The transition from conversational AI to agentic systems has created an “ $M \times N$ integration problem,” where M models require bespoke connectors for N data sources, leading to fragmented infrastructure [2], [5]. While recent surveys have cataloged emerging standards like MCP, Agent-to-Agent (A2A), and Agent Communication Protocol (ACP) [1], there remains a lack of rigorous comparison regarding their architectural trade-offs. This paper analyzes MCP against its primary contenders—Google’s A2A protocol and OpenAI’s REST-based function calling—to determine the specific utility of stateful versus stateless interaction patterns. We utilize a “Systematization of Knowledge” (SoK) approach [3] combined with a constructive evaluation of a standardized “Research Assistant” task graph [6] implemented across three protocols. Empirical validation demonstrates that MCP’s persistent connection model reduces marginal execution latency in multi-turn workflows compared to stateless REST, though this efficiency introduces an “implicit trust propagation” vulnerability that amplifies security risks [3], [7]. Analysis reveals that MCP’s persistent, stateful connection model (JSON-RPC) offers superior context management for local-first and high-fidelity tool use [3], [7], whereas A2A excels in decentralized, trust-based task delegation between autonomous entities [1], [3]. We propose a unified “Agent Protocol Stack,” arguing that MCP and A2A are complementary layers—MCP as the standard for *tool execution* and A2A for *agent collaboration*—rather than mutually exclusive competitors [4].

Index Terms—Model Context Protocol, MCP, Agent-to-Agent, A2A, ACP, agent interoperability, JSON-RPC, stateful protocols, Systematization of Knowledge.



1 INTRODUCTION

THE rapid proliferation of Large Language Models (LLMs) has necessitated standardized interfaces for tools and memory [3]. Current ad-hoc integrations (manual API wiring, framework-specific wrappers like LangChain) are brittle and unscalable [3].

Previous works, such as “A Survey of Agent Interoperability Protocols” [1], provided a necessary taxonomy of the landscape. However, developers currently lack a decision framework for choosing between *tool-centric* protocols (MCP) and *agent-centric* protocols (A2A) based on architectural constraints like latency, security, and state handling [3], [7].

This paper addresses the following research questions:

- **RQ1 (Architecture):** How do MCP’s core primitives (Tools, Resources, Prompts) [1] diverge from the “Agent Card” and capability negotiation models of A2A and ACP [1]?
- **RQ2 (Security & State):** How does MCP’s stateful JSON-RPC session model impact security boundaries compared to stateless REST-based function calling [7]?
- **RQ3 (Convergence):** Can these protocols coexist? Is there evidence for a layered “TCP/IP moment” for the internet of agents [4]?

2 BACKGROUND & RELATED WORK

- **The “ $M \times N$ ” Integration Crisis:** Definition of the fragmentation problem where M agents need bespoke connectors for N tools, stifling innovation [2], [5].
- **Precursors:** Brief coverage of the Language Server Protocol (LSP), which inspired MCP’s client-host-server topology [3], [8], and OpenAI Function Calling, which established the de facto REST-based standard [1], [3].
- **Existing Surveys:** Cite “A Survey of Agent Interoperability Protocols” [1] as the foundational text. This paper distinguishes itself by moving from *description* (what exists) to *architectural evaluation* (how it behaves under load/attack).

3 ARCHITECTURAL ANALYSIS

3.1 Taxonomy of Interaction Patterns

- *Direct Tooling (OpenAI/LangChain):* Ephemeral, request-response, stateless. The context must be re-injected every turn [3], [7].
- *Context-First (MCP):* Persistent connections via JSON-RPC (over Stdio or SSE). Decouples “passive context” (Resources) from “active execution” (Tools) [1], [3].

- *Peer-Delegation (A2A/ACP)*: High-level task handoff, asynchronous event loops, and trust-based capability negotiation between autonomous peers [1], [3].

3.2 Comparison Matrix

- **Transport**: MCP’s local-first focus (Stdio) vs. A2A’s web-first design (HTTP/SSE) [7].
- **Discovery**: MCP’s `initialize` handshake and capability declaration vs. A2A’s “Agent Card” lookup vs. OpenAI’s static schema definition.
- **State Handling**: MCP’s server-driven resource updates (subscriptions) vs. REST’s client-driven polling [3].

4 EMPIRICAL VALIDATION & SECURITY EVALUATION

4.1 Protocol Overhead Analysis

To quantify the architectural trade-offs discussed in Section 3, we conducted a lightweight comparative experiment measuring the latency and payload efficiency of MCP against stateless REST patterns. We implemented a standardized “Search-and-Summarize” workflow (listing, reading, and summarizing files) across two conditions: (1) Persistent JSON-RPC over Stdio (MCP) and (2) Stateless HTTP (representing standard REST/A2A patterns).

4.2 Threat Model Divergence

- **MCP Risks**: “Cross-Primitive Escalation” (using a read-only Resource to trigger a Tool action) [2] and “Rug Pulls” (malicious servers changing behavior after trust establishment) [3].
- **Injection Vectors**: Analysis of “Indirect Prompt Injection” where malicious content in an MCP Resource (e.g., a GitHub issue) hijacks the agent’s control flow [3], [9].

4.3 The Human-in-the-Loop Problem

MCP’s sampling primitive allows servers to request LLM completion, creating a bidirectional control flow that complicates permission boundaries compared to unidirectional REST APIs [10].

5 PROPOSED CONVERGENCE: THE AGENT PROTOCOL STACK

This section addresses the “Future Work” gap by synthesizing the protocols [4]. The layered model (the “TCP/IP” analogy):

- **Layer 3 — Collaboration (“The Social Layer”): A2A / ACP**. Agents talking to Agents. Delegating high-level goals (“Plan a trip”) [1], [3].
- **Layer 2 — Context & Tools (“The Hands & Eyes”): MCP**. Agents talking to Data/Tools. Executing specific atomic actions (“Query database,” “Read file”) [2], [4].
- **Layer 1 — Transport**: HTTP / SSE / JSON-RPC.

MCP and A2A are complementary. An A2A agent (the high-level planner) effectively acts as an MCP *Host* to execute specific sub-tasks using MCP *Servers* [4].

6 DISCUSSION & FUTURE DIRECTIONS

- **The “Context-Aware” Shift**: Discuss the move toward “Context-Aware MCP” (CA-MCP) where servers share a global state store to reduce context window bloating, addressing the limitations of “dumb” pipes [11].
- **Ecosystem Maturity**: While MCP has rapid adoption (Claude, IDEs, 5000+ servers) [2], A2A provides necessary enterprise features like auditability and complex negotiation that MCP currently lacks [3].
- **Recommendation**: Developers should use MCP for *vertical* integration (connecting an agent to a database) and A2A for *horizontal* integration (connecting a travel agent to a booking agent) [4].

7 CONCLUSION

- MCP successfully solves the “last mile” connectivity problem for agents, transforming the $M \times N$ problem into an $M + N$ ecosystem [2], [5].
- However, it is not a complete agent orchestration framework. The future of agentic infrastructure lies in the composition of these protocols, where MCP provides the standardized I/O layer for the “Internet of Agents” [4].

ACKNOWLEDGMENTS

The author would like to thank...

REFERENCES

- [1] A. Ehtesham, A. Singh, G. K. Gupta, and S. Kumar, “A Survey of Agent Interoperability Protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP),” *arXiv preprint arXiv:2505.02279*, May 2025.
- [2] J. A. Oribe, “The Model Context Protocol (MCP): Emergence, Technical Architecture, and the Future of Agentic AI Infrastructure,” Zenodo, 2025, doi: 10.5281/zenodo.17390299.
- [3] X. Hou, Y. Zhao, S. Wang, and H. Wang, “Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions,” *arXiv preprint arXiv:2503.23278*, 2025.
- [4] S. Mitra, “The Agent Protocol Stack: Why MCP + A2A + A2UI Is the TCP/IP Moment for Agentic AI,” *subhadipmitra.com*, Jan. 2026. [Online]. Available: <https://subhadipmitra.com/blog/2026/agent-protocol-stack/>
- [5] V. Ayyagari, “Model Context Protocol for Agentic AI: Enabling Contextual Interoperability Across Systems,” *International Journal of Computational and Experimental Science and Engineering*, vol. 11, no. 3, pp. 6072–6082, 2025, doi: 10.22399/ijcesen.3678.
- [6] K. Ganesh, “Anthropic’s Model Context Protocol (MCP) for AI Applications and Agents,” Bluetick Consultants Blog, Mar. 2025. [Online]. Available: <https://www.bluetickconsultants.com/implementing-anthropics-model-context-protocol-mcp-for-ai-applications-and-agents/>
- [7] N. Maloyan and D. Namiot, “Breaking the Protocol: Security Analysis of the Model Context Protocol Specification and Prompt Injection Vulnerabilities in Tool-Integrated LLM Agents,” *arXiv preprint arXiv:2601.17549*, Jan. 2026.
- [8] Online Discussion, “LSP vs MCP: The One True Story to Rule Them All,” r/mcp, Reddit, 2025. [Online]. Available: https://www.reddit.com/r/mcp/comments/1joqzpz/lsp_vs_mcp_the_one_true_story_to_rule_them_all/
- [9] P. R. B. Satya, A. Guyyala, V. Putta, and K. T. Areti, “Robustness of Automated AI Agents Against Adversarial Context Injection in MCP,” *International Journal of Computer Applications*, vol. 187, no. 56, Nov. 2025, doi: 10.5120/ijca2025925957.
- [10] Emergent Mind, “Prompt-Based Context Injection Mechanism,” 2025. [Online]. Available: <https://www.emergentmind.com/topics/prompt-based-context-injection-mechanism>

- [11] M. A. Jayanti and X. Y. Han, “Enhancing Model Context Protocol (MCP) with Context-Aware Server Collaboration,” *arXiv preprint arXiv:2601.11595*, Jan. 2026.