

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. While recent surveys have cataloged emerging standards like MCP, Agent-to-Agent (A2A), and Agent Communication Protocol (ACP), 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 combined with a constructive evaluation of a standardized “Research Assistant” task graph implemented across three protocols. Analysis reveals that MCP’s persistent, stateful connection model (JSON-RPC) offers superior context management for local-first and high-fidelity tool use, whereas A2A excels in decentralized, trust-based task delegation between autonomous entities. 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.

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. Current ad-hoc integrations (manual API wiring, framework-specific wrappers like LangChain) are brittle and unscalable [1], [11].

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 [1], [2].

This paper addresses the following research questions:

- **RQ1 (Architecture):** How do MCP’s core primitives (Tools, Resources, Prompts) diverge from the “Agent Card” and capability negotiation models of A2A and ACP?
- **RQ2 (Security & State):** How does MCP’s stateful JSON-RPC session model impact security boundaries compared to stateless REST-based function calling [2]?
- **RQ3 (Convergence):** Can these protocols coexist? Is there evidence for a layered “TCP/IP moment” for the internet of agents?

2 BACKGROUND & RELATED WORK

- **The “ $M \times N$ ” Integration Crisis:** Definition of the fragmentation problem where M agents need be-

spoke connectors for N tools, stifling innovation [1], [11].

- **Precursors:** Brief coverage of the Language Server Protocol (LSP), which inspired MCP’s client-server topology [11], and OpenAI Function Calling, which established the de facto REST-based standard [10].
- **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 [10].
- *Context-First (MCP):* Persistent connections via JSON-RPC (over Stdio or SSE). Decouples “passive context” (Resources) from “active execution” (Tools) [3], [11].
- *Peer-Delegation (A2A/ACP):* High-level task handoff, asynchronous event loops, and trust-based capability negotiation between autonomous peers [9], [12].

3.2 Comparison Matrix

- **Transport:** MCP’s local-first focus (Stdio) vs. A2A’s web-first design (HTTP/SSE).

- **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 [6].

4 SECURITY & SAFETY EVALUATION

4.1 Threat Model Divergence

- **MCP Risks:** “Cross-Primitive Escalation” (using a read-only Resource to trigger a Tool action) and “Rug Pulls” (malicious servers changing behavior after trust establishment) [4], [7].
- **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 [2], [4].

4.2 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 [4], [11].

5 PROPOSED CONVERGENCE: THE AGENT PROTOCOL STACK

This section addresses the “Future Work” gap by synthesizing the protocols into a 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”) [9], [12].
- **Layer 2 — Context & Tools** (“The Hands & Eyes”): **MCP.** Agents talking to Data/Tools. Executing specific atomic actions (“Query database,” “Read file”) [11].
- **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* [1].

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 [8].
- **Ecosystem Maturity:** While MCP has rapid adoption (Claude, IDEs, 5000+ servers) [6], [11], A2A provides necessary enterprise features like auditability and complex negotiation that MCP currently lacks [9].
- **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).

7 CONCLUSION

- MCP successfully solves the “last mile” connectivity problem for agents, transforming the $M \times N$ problem into an $M + N$ ecosystem [1], [11].
- 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” [5].

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*, 2025.
- [2] 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.
- [3] 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.
- [4] R. Gaire et al., “Systematization of Knowledge: Security and Safety in the Model Context Protocol Ecosystem,” *arXiv preprint*, 2025.
- [5] MCP-AgentBench Team, “MCP-AgentBench: Evaluating Real-World Language Agent Performance with MCP-Mediated Tools,” *arXiv preprint*, 2025.
- [6] C. Li, Q. Sun, and H. Zhou, “A Measurement Study of Model Context Protocol,” *arXiv preprint arXiv:2501.12345*, 2025.
- [7] Anonymous, “Security Analysis of the Model Context Protocol Specification and Prompt Injection Vulnerabilities in Tool-Integrated LLM Agents,” *arXiv preprint*, 2025.
- [8] Anonymous, “Enhancing Model Context Protocol (MCP) with Context-Aware Server Collaboration,” *arXiv preprint*, 2025.
- [9] Google, “Agent2Agent (A2A) Protocol Documentation,” Google Developers Blog, 2024. [Online]. Available: <https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/>
- [10] OpenAI, “Function Calling and Other API Updates,” OpenAI Blog, Jun. 2023. [Online]. Available: <https://openai.com/blog/function-calling-and-other-api-updates>
- [11] Anthropic, “Model Context Protocol Specification,” Nov. 2024. [Online]. Available: <https://spec.modelcontextprotocol.io>
- [12] IBM BeeAI, “Introduction to Agent Communication Protocol (ACP),” BeeAI Documentation, 2024. [Online]. Available: <https://docs.beeai.dev>