Model Context Protocol (MCP): A Literature Review

Technical Research Team

May 9, 2025

Contents

1 Model Context Protocol (MCP): A Literature Review

1.1 Table of Contents

1.2 Executive Summary

The Model Context Protocol (MCP) represents a significant advancement in the integration of large language models (LLMs) with external data sources, tools, and services. Developed and open-sourced by Anthropic in November 2024, MCP addresses the fragmentation and complexity inherent in AI system integration by providing a standardized framework for connecting AI models with external systems.

This literature review synthesizes current research and technical documentation on MCP, examining its conceptual foundations, technical architecture, practical applications, and future directions. Key findings include:

1. **Standardization Benefits**: MCP transforms the complex "N×M integration problem" into a simpler "N+M" scenario, significantly reducing development overhead and complexity.
2. **Technical Architecture**: Built on a client-server model using JSON-RPC 2.0, MCP provides three core primitives (tools, resources, and prompts) that enable structured interactions between AI models and external systems.
3. **Competitive Advantage**: Compared to alternatives like OpenAI Function Calling and LangChain, MCP offers greater openness, standardization, and model-agnostic flexibility.
4. **Current Applications**: MCP has found adoption in AI-assisted development, enterprise data access, AI-driven data querying, desktop applications, and workflow orchestration.
5. **Challenges**: Key challenges include infrastructure scalability, security and privacy risks, ecosystem maturity, and integration complexity with legacy systems.
6. **Future Directions**: Promising research areas include protocol enhancements for distributed deployment, security advancements, ecosystem development, and advanced use cases like multi-agent collaboration.

As the AI landscape continues to evolve, MCP stands poised to play a pivotal role in creating a more connected, interoperable ecosystem of AI tools and services. This review provides stakeholders with a comprehensive understanding of MCP's current state and future potential.

1.3 1. Introduction

1.3.1 1.1 Background and Context

The rapid evolution of large language models (LLMs) has transformed the artificial intelligence landscape, enabling increasingly sophisticated natural language understanding and generation capabilities. However, these models face a fundamental limitation: they operate within the confines of their training data, disconnected from real-time information and external systems. This isolation significantly constrains their practical utility in real-world applications.

Traditional approaches to LLM integration have involved creating bespoke connections for each data source or tool, resulting in fragmented architectures that are difficult to scale and maintain. As the number of AI applications and external systems grows, this "N×M integration problem" becomes increasingly unmanageable, creating a bottleneck in AI development and deployment.

1.3.2 1.2 Purpose and Scope of the Review

This literature review examines the Model Context Protocol (MCP) as a solution to these integration challenges. Focusing primarily on Anthropic's implementation, the review synthesizes information from authoritative sources to provide a comprehensive understanding of MCP's conceptual foundations, technical architecture, practical applications, and future directions.

The review addresses several key questions: - What are the core principles and objectives of MCP? - How does MCP's technical architecture enable standardized integration? - How does MCP compare with alternative approaches to LLM integration? - What are the current applications and use cases of MCP? - What technical challenges and limitations does MCP face? - What future research directions might enhance MCP's capabilities?

By addressing these questions, this review aims to provide stakeholders—including developers, researchers, and decision-makers—with a comprehensive understanding of MCP's role in advancing AI capabilities and integration.

1.4 2. Definition and Conceptual Framework

1.4.1 2.1 Core Definition

The Model Context Protocol (MCP) is an open standard designed to facilitate seamless, secure, and standardized interaction between large language models and external data sources, tools, and systems [1]. Officially announced and open-sourced by Anthropic in November 2024, MCP addresses the fragmentation and complexity inherent in integrating AI systems with diverse external resources [2].

1.4.2 2.2 Conceptual Foundations

At its conceptual core, MCP serves as a universal connector—analogous to USB-C for hardware—that enables AI models to "plug into" different data sources and tools in a consistent manner [3]. This standardization eliminates the need for developers to create custom integrations for every new tool or dataset they want their AI application to interact with.

The fundamental problem MCP addresses is what industry experts call the "N×M integration problem" [4]. As illustrated in Figure 1, this problem arises when the number of AI applications (N) and the variety of tools and data sources (M) increase, making the complexity of creating custom integrations for each combination unmanageable. MCP transforms this into a much simpler N+M setup where each AI model and each tool only needs to conform to the MCP standard once.

```
Figure 1: The N×M Integration Problem vs. MCP Solution

Traditional Integration (N×M):
AI App 1 ---- Custom Integration ---- Tool 1
AI App 1 ---- Custom Integration ---- Tool 2
AI App 2 ---- Custom Integration ---- Tool 1
AI App 2 ---- Custom Integration ---- Tool 2
...and so on for each combination

MCP Solution (N+M):
AI App 1 ----\              /---- Tool 1
AI App 2 --------- MCP --------- Tool 2
...                          ...
```

1.4.3 2.3 Key Objectives

The primary objectives of MCP include:

1. **Standardization**: Providing a uniform interface for diverse tools and data sources, reducing integration complexity
2. **Breaking down data silos**: Enabling AI models to access relevant external data dynamically, enhancing their contextual understanding

3. **Enhancing interoperability**: Supporting seamless interaction between different AI systems and tools through a common protocol
4. **Facilitating two-way communication**: Enabling bidirectional exchanges between models and external systems for more interactive capabilities
5. **Fostering an open ecosystem**: Encouraging collaborative development of connectors and servers through open standards and community contribution [5]

These objectives collectively aim to create a more connected, efficient AI ecosystem where models can leverage external capabilities without the overhead of custom integration work.

1.5 3. Anthropic's Implementation and Approach

1.5.1 3.1 Development History

Anthropic developed and open-sourced the Model Context Protocol in November 2024 [6]. The company's approach was driven by the recognition that even the most sophisticated models are constrained by their isolation from data—trapped behind information silos and legacy systems [7]. By creating an open standard, Anthropic aimed to replace fragmented integrations with a single protocol that could serve as the foundation for a more connected AI ecosystem.

1.5.2 3.2 Anthropic's MCP Components

Anthropic's implementation of MCP includes three major components:

1. **The Model Context Protocol specification and SDKs**: Detailed documentation and software development kits in multiple programming languages (TypeScript, Python, Java, C#, Kotlin) to facilitate implementation across diverse technology stacks [8]
2. **Local MCP server support in Claude Desktop apps**: Integration of MCP capabilities directly into Anthropic's consumer-facing applications, enabling secure access to local files and resources [9]
3. **An open-source repository of MCP servers**: Pre-built connectors for popular enterprise systems like Google Drive, Slack, GitHub, Git, Postgres, and Puppeteer, reducing the implementation burden for developers [10]

1.5.3 3.3 Anthropic's Design Philosophy

Anthropic's approach to MCP emphasizes several key principles:

1. **Openness**: MCP is designed as an open standard to encourage widespread adoption and community contribution, avoiding vendor lock-in
2. **Security**: The architecture incorporates robust security measures, including sandboxing and access control, to protect sensitive data and systems
3. **Modularity**: The client-server design ensures a clear separation of concerns, allowing different components to be developed and scaled independently
4. **Flexibility**: Support for multiple transport mechanisms accommodates various deployment scenarios, from local development to enterprise environments [11]

1.5.4 3.4 Integration with Claude Models

Anthropic has specifically optimized Claude 3.5 Sonnet to work effectively with MCP, making it adept at quickly building MCP server implementations [12]. This integration enables Claude to rapidly connect with important datasets

and tools, enhancing its contextual understanding and response capabilities. The tight coupling between Claude models and MCP demonstrates Anthropic's commitment to creating an AI system that can seamlessly interact with external resources.

1.6 4. Technical Architecture and Components

1.6.1 4.1 Client-Server Architecture

MCP employs a client-server architecture comprising five key components, as illustrated in Figure 2:

```
Figure 2: MCP Client-Server Architecture

[MCP Host (AI Application)] <--> [MCP Client] <--> [Transport Layer] <--> [MCP Server] <-->
```

1. **MCP Host**: The AI application or agent (e.g., Claude, GPT-based systems) that interacts with external data via MCP
2. **MCP Client**: The intermediary component within the host that manages communication with MCP servers
3. **MCP Server**: Lightweight programs exposing specific tools, resources, or functionalities
4. **Transport Layer**: The communication mechanism between clients and servers
5. **External Data/Tools**: Data repositories, APIs, or services accessible via MCP servers [13]

This architecture provides a clear separation of concerns, allowing each component to evolve independently while maintaining interoperability through the standardized protocol.

1.6.2 4.2 Communication Protocol

MCP is built on JSON-RPC 2.0, a lightweight remote procedure call protocol, enabling structured request-response interactions [14]. The protocol supports multiple transport mechanisms:

1. **Stdio (Standard Input/Output)**: For local, in-process communication, offering low latency and simplicity
2. **HTTP over SSE (Server-Sent Events)**: For remote, web-based interactions, enabling distributed deployment [15]

The communication flow typically includes:

- **Discovery**: The host detects available MCP servers and their capabilities
- **Capability Negotiation**: The host retrieves the tools, resources, and prompts offered by servers
- **Execution**: The host invokes tools or accesses resources, receiving results in a standardized format
- **Termination**: The connection is gracefully closed when interactions conclude [16]

1.6.3 4.3 Core Primitives

MCP exposes three fundamental primitives:

1. **Tools**: Executable functions (e.g., send email, query database) that can be invoked by the AI to perform specific actions
2. **Resources**: Read-only data (e.g., files, logs, knowledge bases) that the AI can access to enhance its contextual understanding
3. **Prompts**: Predefined templates guiding AI interactions with specific tools or resources [17]

These primitives provide a comprehensive framework for AI-external system interaction, covering both active (tools) and passive (resources) integration patterns.

1.6.4 4.4 Security Architecture

Security is integral to MCP's design:

1. **Authentication**: Ensures only authorized hosts and servers communicate, often via OAuth 2.1 or token-based mechanisms
2. **Authorization**: Servers enforce permissions, limiting access to resources or actions based on user identity and roles
3. **Sandboxing**: Hosts and servers operate within controlled environments to prevent unauthorized data access or execution
4. **Auditing**: All interactions can be logged for compliance and security monitoring [18]

This multi-layered security approach addresses the significant concerns associated with granting AI systems access to sensitive data and systems.

1.7 5. Comparison with Other Protocols and Frameworks

1.7.1 5.1 MCP vs. OpenAI Function Calling

OpenAI's function calling is primarily focused on enabling LLMs to translate user prompts into structured API calls. It allows the model to determine which function to call and with what parameters, facilitating interactions with external services. In contrast, MCP is a broader protocol that standardizes the entire execution and interaction process between LLM applications and external systems [19].

Table 1 provides a comparative analysis of MCP and OpenAI Function Calling:

Aspect	MCP	OpenAI Function Calling
Openness	Open standard	Proprietary (OpenAI-specific)
Compatibility	Multi-LLM, multi-tool	Limited to OpenAI models (GPT-4, GPT-3.5)
Security	Built-in, customizable	Basic, depends on API controls
Standardization	Unified, extensible	Specific, less flexible
Ecosystem	Growing, community-driven	Closed, vendor-specific
Communication	Interactive, bidirectional	Request-response API calls
Scope	Standardized, model-agnostic protocol	Vendor-specific API invocation

1.7.2 5.2 MCP vs. LangChain

LangChain is a popular framework for building applications with LLMs, focusing on chaining together different components to create complex workflows. While MCP provides a foundational standard for communication, LangChain offers a higher-level abstraction for building complex workflows. The two can be complementary, with LangChain potentially leveraging MCP for standardized tool access [20].

Table 2 compares MCP and LangChain:

Aspect	MCP	LangChain
Level	Low-level protocol	High-level framework

Aspect	MCP	LangChain
Focus	Standardized interaction protocol	Tool chaining, orchestration
Integration	Direct, protocol-based	Abstraction over multiple LLMs and tools
Complexity	Lower entry barrier	Higher, with more features
Flexibility	High, protocol-based	High, with templating and pipelines

1.7.3 5.3 MCP vs. AI Agent Protocols

AI agent protocols, such as Agent 2 Agent, focus on enabling communication and collaboration between autonomous AI agents. These protocols typically emphasize agent coordination rather than external system integration.

Table 3 compares MCP with AI agent protocols:

Aspect	MCP	AI Agent Protocols (e.g., Agent 2 Agent)
Purpose	Standardized tool/resource access	Inter-agent communication and collaboration
Scope	External system integration	Multi-agent coordination and task execution
Security	Emphasized	Varies, often less focus on external data access
Ecosystem	Growing, open	Specialized, often proprietary

As the AI ecosystem evolves, these approaches may converge or become complementary, with MCP potentially serving as a foundation for more complex agent interactions [21].

1.8 6. Current Applications and Use Cases

1.8.1 6.1 AI-Assisted Development

MCP has found significant adoption in development environments, enhancing the capabilities of AI coding assistants:

1. **Sourcegraph Cody**: Utilizes MCP to access extensive codebases and documentation, providing developers with accurate code suggestions based on their specific codebase context [22]
2. **Zed Editor**: Incorporates MCP to enable AI features to interact seamlessly with development tools, enhancing the coding experience with contextual assistance [23]
3. **Replit**: Leverages MCP to enhance code retrieval and generation capabilities, making its AI features more aware of the user's development environment [24]
4. **Codeium**: Uses MCP to improve context-aware coding assistance, enabling more accurate and relevant suggestions [25]

These implementations demonstrate MCP's value in bridging the gap between AI coding assistants and the development environments they operate within.

1.8.2 6.2 Enterprise Data Access

Organizations are using MCP to connect AI systems with internal data, enabling more contextually aware AI applications:

1. **Block (formerly Square)**: Adopted MCP to securely connect AI systems with internal data repositories, enhancing the contextual understanding of their AI applications [26]
2. **Apollo**: Utilizes MCP to link AI tools with customer relationship management systems, enabling more personalized customer interactions [27]
3. **Enterprise knowledge bases**: MCP enables secure access to company documents, policies, and procedures, allowing AI systems to provide more accurate and organization-specific responses [28]

These enterprise applications highlight MCP's role in breaking down data silos while maintaining security and compliance requirements.

1.8.3 6.3 AI-Driven Data Querying

MCP facilitates natural language interfaces to structured data, making database interactions more accessible:

1. **AI2SQL**: Leverages MCP to enable natural language generation of SQL queries, allowing users to interact with databases using conversational language [29]

2. **Database connectors**: Pre-built MCP servers for Postgres and other databases enable AI to query structured data directly, enhancing data analysis capabilities [30]

These applications demonstrate MCP's potential to democratize data access by providing natural language interfaces to structured data repositories.

1.8.4 6.4 Desktop AI Applications

MCP enhances local AI assistants by providing secure access to local resources:

1. **Claude Desktop**: Integrates MCP to allow secure access to local files and applications, enhancing the assistant's ability to work with user data [31]
2. **Personal assistants**: MCP enables deep integration with local data and applications, creating more personalized and contextually aware AI experiences [32]

These implementations showcase MCP's ability to enhance AI assistants with local context while maintaining security boundaries.

1.8.5 6.5 Workflow Orchestration

MCP supports complex, multi-step workflows that span multiple tools and data sources:

1. **GitHub PR review**: MCP servers can fetch PR details, analyze code changes, and generate summaries, streamlining the code review process [33]
2. **Multi-tool workflows**: Combining APIs, databases, and file systems seamlessly to accomplish complex tasks that require multiple data sources and actions [34]

These orchestration capabilities highlight MCP's potential to enable more complex, multi-step AI workflows that more closely mirror human work patterns.

1.9 7. Technical Challenges and Limitations

1.9.1 7.1 Infrastructure and Scalability Challenges

Despite its promising architecture, MCP faces several infrastructure and scalability challenges:

1. **Limited support for distributed deployment**: Most MCP services rely on the traditional Stdio communication model, which hampers elastic scaling and remote deployment in cloud environments [35]
2. **Support for multi-tenancy**: Current MCP implementations are primarily local and single-user, with a need for scalable multi-tenant architectures to support enterprise-scale deployments [36]
3. **Performance optimization**: Ensuring robust performance across complex networks of interconnected systems requires careful optimization, particularly for latency-sensitive applications [37]

These challenges highlight the need for continued development of MCP's infrastructure capabilities to support enterprise-scale deployments.

1.9.2 7.2 Security and Privacy Risks

MCP's integration capabilities introduce several security and privacy considerations:

1. **Token theft and account compromise**: MCP's reliance on OAuth tokens introduces risks of token theft, potentially compromising connected services [38]
2. **Server compromise**: Breaching MCP servers can grant attackers access to connected services, creating a potential security vulnerability [39]
3. **Prompt injection**: The natural language interface opens avenues for prompt injection attacks, where malicious inputs could manipulate the AI's behavior [40]
4. **Data privacy concerns**: Broad permission scopes and centralized data aggregation pose privacy challenges, particularly in regulated industries [41]

Addressing these security and privacy risks is essential for MCP's adoption in sensitive or regulated environments.

1.9.3 7.3 Ecosystem and Tooling Maturity

The MCP ecosystem is still evolving, with several areas requiring further development:

1. **Discovery mechanisms**: Currently, MCP server discovery is manual, limiting dynamic tool discovery and adaptation [42]

2. **Standardized client experience**: There is a lack of unified interfaces for tool invocation and debugging, creating inconsistent developer experiences [43]
3. **Workflow management**: Managing multi-step, resumable workflows remains an open problem, particularly for complex enterprise processes [44]

These ecosystem challenges highlight the need for continued community development and standardization efforts.

1.9.4 7.4 Integration Complexity

Integrating MCP with existing systems presents several challenges:

1. **Legacy system integration**: Connecting MCP with outdated systems that lack modern APIs presents challenges, potentially limiting its applicability in certain environments [45]
2. **Data format inconsistencies**: Varying data formats across different systems can complicate information exchange, requiring additional transformation layers [46]
3. **Cross-department coordination**: MCP implementation typically spans multiple departments, requiring alignment across technical and organizational boundaries [47]

These integration challenges underscore the need for robust implementation patterns and organizational strategies to maximize MCP's value.

1.10 8. Future Directions and Research Opportunities

1.10.1 8.1 Protocol Enhancements

Several protocol enhancements could address current limitations and expand MCP's capabilities:

1. **Support for remote and distributed MCP deployment**: Developing protocols that enable elastic scaling, multi-tenancy, and multi-user support would enhance MCP's enterprise readiness [48]
2. **Streamable HTTP and real-time communication**: Transitioning from SSE to more scalable, streamable protocols could improve performance and reliability in distributed environments [49]
3. **Standardized discovery mechanisms**: Creating registries and discovery APIs to facilitate dynamic tool discovery would enhance the flexibility and adaptability of MCP-enabled systems [50]

These protocol enhancements would address key infrastructure limitations and expand MCP's applicability across diverse deployment scenarios.

1.10.2 8.2 Security and Privacy Advancements

Advancing MCP's security and privacy capabilities is essential for broader adoption:

1. **Robust authentication frameworks**: Developing standardized, fine-grained access control mechanisms would enhance security in multi-user environments [51]
2. **Secure token management**: Creating mechanisms for secure storage, rotation, and revocation of OAuth tokens would mitigate the risks of token theft and misuse [52]
3. **Prompt injection mitigation**: Developing techniques for input validation and anomaly detection would protect against manipulation through malicious prompts [53]

These security advancements would address key concerns associated with granting AI systems access to sensitive data and systems.

1.10.3 8.3 Ecosystem Development

Building a robust ecosystem around MCP could accelerate its adoption and impact:

1. **Marketplace development**: Creating platforms for MCP server sharing, rating, and monetization would foster a vibrant developer community [54]

2. **Revenue-sharing models**: Designing fair, transparent models to motivate high-quality MCP service development would encourage innovation and quality [55]

3. **Industry-wide standardization**: Collaborating with industry consortia to formalize MCP as a standard would enhance interoperability and adoption [56]

These ecosystem developments would create a sustainable community around MCP, driving continued innovation and adoption.

1.10.4 8.4 Advanced Use Cases

Exploring advanced use cases could unlock new value from MCP:

1. **Multi-agent collaboration**: Enabling multiple AI agents to work together through MCP would support more complex problem-solving scenarios [57]

2. **Cross-modal integration**: Extending MCP to support non-text modalities like images and audio would enhance its applicability across diverse domains [58]

3. **Federated learning integration**: Using MCP to facilitate privacy-preserving model training would address key concerns in sensitive domains [59]

These advanced use cases represent the frontier of MCP's potential, pointing toward a future where AI systems can collaborate and learn in increasingly sophisticated ways.

1.11 9. Conclusion

The Model Context Protocol represents a significant advancement in addressing the integration challenges that have constrained the practical utility of large language models. By providing a standardized framework for connecting AI models with external data sources, tools, and systems, MCP transforms the complex "N×M integration problem" into a more manageable "N+M" scenario, reducing development overhead and complexity.

This literature review has examined MCP's conceptual foundations, technical architecture, practical applications, and future directions, revealing several key insights:

1. **Architectural Soundness**: MCP's client-server architecture, built on JSON-RPC 2.0, provides a robust foundation for standardized integration, with clear primitives (tools, resources, and prompts) that enable structured interactions.
2. **Competitive Advantage**: Compared to alternatives like OpenAI Function Calling and LangChain, MCP offers greater openness, standardization, and model-agnostic flexibility, positioning it as a potentially unifying standard in the fragmented AI integration landscape.
3. **Practical Utility**: MCP's adoption across diverse domains—from development environments to enterprise data access—demonstrates its practical value in enhancing AI capabilities through contextual integration.
4. **Evolving Maturity**: While MCP faces challenges in areas like infrastructure scalability, security, and ecosystem maturity, active research and development efforts are addressing these limitations.
5. **Future Potential**: The protocol's future directions—including distributed deployment, enhanced security, ecosystem development, and advanced use cases—point toward an increasingly capable and versatile integration standard.

As the AI landscape continues to evolve, MCP stands poised to play a pivotal role in creating a more connected, interoperable ecosystem of AI tools and services. By enabling AI models to seamlessly access external data and capabilities, MCP helps bridge the gap between AI's theoretical capabilities and its practical utility in real-world applications.

For stakeholders across the AI ecosystem—from developers and researchers to enterprise decision-makers—MCP offers a promising path toward more contextually aware, capable, and integrated AI systems. As the protocol matures and its ecosystem expands, its impact on AI development and deployment is likely to grow, potentially establishing it as a foundational standard for AI integration in the years to come.

1.12 10. Bibliography

[1] Anthropic. (2024, November 25). Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol

[2] Hou, X., Zhao, Y., Wang, S., & Wang, H. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv:2503.23278 [cs.CR]. https://arxiv.org/abs/2503.23278

[3] Amanatullah. (2025, March 21). Anthropic's Model Context Protocol (MCP): A Deep Dive for Developers. Medium. https://medium.com/@amanatulla1606/anthropics-model-context-protocol-mcp-a-deep-dive-for-developers-1d3db39c9fdc

[4] Bowdoin, J. (2025, March 13). The Unstoppable Rise of MCP Servers: How This Open Protocol Is Reshaping the AI. https://jeffreybowdoin.com/blog/mcp-server-protocol-ai/

[5] Anthropic. (2025). Model Context Protocol (MCP) - Anthropic Documentation. https://docs.anthropic.com/en/docs/agents-and-tools/mcp

[6] Anthropic. (2024, November 25). Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol

[7] Ibid.

[8] Amanatullah. (2025, March 21). Anthropic's Model Context Protocol (MCP): A Deep Dive for Developers. Medium. https://medium.com/@amanatulla1606/anthropics-model-context-protocol-mcp-a-deep-dive-for-developers-1d3db39c9fdc

[9] Anthropic. (2024, November 25). Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol

[10] Ibid.

[11] Hou, X., Zhao, Y., Wang, S., & Wang, H. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv:2503.23278 [cs.CR]. https://arxiv.org/abs/2503.23278

[12] Anthropic. (2024, November 25). Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol

[13] Amanatullah. (2025, March 21). Anthropic's Model Context Protocol (MCP): A Deep Dive for Developers. Medium. https://medium.com/@amanatulla1606/anthropics-model-context-protocol-mcp-a-deep-dive-for-developers-1d3db39c9fdc

[14] Ibid.

[15] Ibid.

[16] Hou, X., Zhao, Y., Wang, S., & Wang, H. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv:2503.23278 [cs.CR]. https://arxiv.org/abs/2503.23278

[17] Amanatullah. (2025, March 21). Anthropic's Model Context Protocol (MCP): A Deep Dive for Developers. Medium. https://medium.com/@amanatulla1606/anthropics-model-context-protocol-mcp-a-deep-dive-for-developers-1d3db39c9fdc

[18] Hou, X., Zhao, Y., Wang, S., & Wang, H. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv:2503.23278 [cs.CR]. https://arxiv.org/abs/2503.23278

[19] Amanatullah. (2025, March 21). Anthropic's Model Context Protocol (MCP): A Deep Dive for Developers. Medium. https://medium.com/@amanatulla1606/anthropics-model-context-protocol-mcp-a-deep-dive-for-developers-1d3db39c9fdc

[20] Web search results. (2025). Model Context Protocol LLM comparison. Retrieved from search query.

[21] Ibid.

[22] Bowdoin, J. (2025, March 13). The Unstoppable Rise of MCP Servers: How This Open Protocol Is Reshaping the AI. https://jeffreybowdoin.com/blog/mcp-server-protocol-ai/

[23] Ibid.

[24] Anthropic. (2024, November 25). Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol

[25] Ibid.

[26] Ibid.

[27] Bowdoin, J. (2025, March 13). The Unstoppable Rise of MCP Servers: How This Open Protocol Is Reshaping the AI. https://jeffreybowdoin.com/blog/mcp-server-protocol-ai/

[28] Web search results. (2025). MCP applications challenges future research. Retrieved from search query.

[29] Bowdoin, J. (2025, March 13). The Unstoppable Rise of MCP Servers: How This Open Protocol Is Reshaping the AI. https://jeffreybowdoin.com/blog/mcp-server-protocol-ai/

[30] Anthropic. (2024, November 25). Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol

[31] Ibid.

[32] Amanatullah. (2025, March 21). Anthropic's Model Context Protocol (MCP): A Deep Dive for Developers. Medium. https://medium.com/@amanatulla1606/anthropics-model-context-protocol-mcp-a-deep-dive-for-developers-1d3db39c9fdc

[33] Ibid.

[34] Web search results. (2025). Model Context Protocol LLM comparison. Retrieved from search query.

[35] Web search results. (2025). MCP applications challenges future research. Retrieved from search query.

[36] Ibid.

[37] Bowdoin, J. (2025, March 13). The Unstoppable Rise of MCP Servers: How This Open Protocol Is Reshaping the AI. https://jeffreybowdoin.com/blog/mcp-server-protocol-ai/

[38] Web search results. (2025). MCP applications challenges future research. Retrieved from search query.

[39] Ibid.

[40] Ibid.

[41] Hou, X., Zhao, Y., Wang, S., & Wang, H. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv:2503.23278 [cs.CR]. https://arxiv.org/abs/2503.23278

[42] Web search results. (2025). MCP applications challenges future research. Retrieved from search query.

[43] Ibid.

[44] Hou, X., Zhao, Y., Wang, S., & Wang, H. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv:2503.23278 [cs.CR]. https://arxiv.org/abs/2503.23278

[45] Bowdoin, J. (2025, March 13). The Unstoppable Rise of MCP Servers: How This Open Protocol Is Reshaping the AI. https://jeffreybowdoin.com/blog/mcp-server-protocol-ai/

[46] Ibid.

[47] Ibid.

[48] Web search results. (2025). MCP applications challenges future research. Retrieved from search query.

[49] Ibid.

[50] Ibid.

[51] Ibid.

[52] Ibid.

[53] Ibid.

[54] Ibid.

[55] Ibid.

[56] Ibid.

[57] Web search results. (2025). Model Context Protocol LLM comparison. Retrieved from search query.

[58] Web search results. (2025). MCP applications challenges future research. Retrieved from search query.

[59] Ibid.