Model Context Protocol (MCP): A Comprehensive Literature Review

1. Introduction and Definition

The **Model Context Protocol (MCP)** is an open standard developed by Anthropic, designed to revolutionize how large language models (LLMs) connect and interact with external data sources, tools, and services. Officially open-sourced in November 2024, MCP addresses the fragmentation and complexity of integrating AI with diverse data systems by providing a unified, standardized protocol for communication and data access (Anthropic, 2024a).

MCP can be conceptualized as a "USB-C port for AI applications" (Modelcontextprotocol.io, 2025), providing a universal, open protocol built on JSON-RPC 2.0 that standardizes how AI models invoke functions, fetch data, and utilize prompts from external systems. It replaces the need for custom, fragmented connectors for each data source, simplifying development and maintenance (Vaughan-Nichols, 2025).

The primary goal of MCP is to overcome the limitations of isolated models by facilitating secure, scalable, and standardized two-way communication between AI applications and diverse data environments. This enables AI systems to access real-time data, perform complex workflows, and maintain context across multiple interactions, significantly enhancing responsiveness and contextual accuracy.

2. Anthropic's MCP Work

2.1 Development and Open-Sourcing

Anthropic introduced and open-sourced the Model Context Protocol in November 2024, positioning it as a solution to the challenge of connecting AI assistants to systems where data resides, including content repositories, business tools, and development environments (Anthropic, 2024a). The company recognized that even the most sophisticated models were constrained by their isolation from data—trapped behind information silos and legacy systems—with each new data source requiring its own custom implementation.

2.2 Core Components Released by Anthropic

Anthropic's initial release included three major components (Anthropic, 2024a):

1. **The Model Context Protocol specification and SDKs**: Providing the technical foundation and development tools for implementing MCP.
2. **Local MCP server support in Claude Desktop apps**: Enabling direct integration with local data sources.
3. **An open-source repository of MCP servers**: Including pre-built servers for popular enterprise systems like Google Drive, Slack, GitHub, Git, Postgres, and Puppeteer.

2.3 Industry Partnerships and Early Adoption

Anthropic has fostered partnerships with several organizations to accelerate MCP adoption. Early adopters include Block and Apollo, who have integrated MCP into their systems. Development tools companies such as Zed, Replit, Codeium, and Sourcegraph are working with MCP to enhance their platforms, enabling AI agents to better retrieve relevant information for coding tasks (Anthropic, 2024a).

Microsoft has partnered with Anthropic to develop an official C# SDK, facilitating MCP integration into enterprise applications built on .NET, Visual Studio, and Azure. The SDK is open-source and available via GitHub, supporting rapid development of MCP connectors (Web Search Results, 2025).

3. Broader Academic Literature on Context Protocols

3.1 Research on Context Window Management

The development of MCP is situated within broader research on context window management in large language models. As models have evolved, the size of context windows has increased dramatically, from a few thousand tokens to potentially millions, but managing these windows effectively remains a significant challenge (Web Search Results, 2025).

Research has explored various approaches to context window management, including:

- **Transformer-XL**: Introduced a recurrence mechanism that caches hidden states from previous segments, effectively extending the context window beyond fixed limits.
- **Parallel Context Windows (PCW)**: Segments long texts into manageable chunks processed independently, then recombined, allowing off-the-shelf models to handle longer inputs without retraining.
- **Attention Sinks**: Use selective attention to focus on relevant parts of the input, dynamically managing the effective context length in streaming scenarios.

3.2 Security and Privacy Considerations

Academic research on model context protocols emphasizes the importance of security and privacy. Hou et al. (2025) discuss the security threats and future research directions for MCP, highlighting the need for:

- **User Consent and Control**: Users must explicitly authorize data sharing and tool invocation, with clear UI prompts.
- **Data Privacy**: Hosts must obtain explicit user consent before exposing data and must protect user data with access controls.
- **Tool Safety**: Tools, which may execute arbitrary code, require careful handling, with explicit user approval.

- **LLM Sampling Controls**: Users should approve sampling requests, controlling prompt visibility and server access.

4. Comparison with Other Context-Handling Approaches

4.1 MCP vs. RAG (Retrieval-Augmented Generation)

While both MCP and RAG aim to enhance LLM capabilities by connecting them to external information, they serve distinct purposes and have different functionalities (Web Search Results, 2025):

Aspect	RAG	MCP
Primary Focus	Data retrieval for improving response accuracy	Standardized interaction protocol for data access and action execution
Main Functionality	Fetches external information during inference	Connects models to external tools, APIs, and data sources for both read and write actions
Interaction Type	Read-only data retrieval	Bidirectional, enabling both read and write operations
Use Cases	Knowledge augmentation, real-time info, enterprise search	Tool integration, automation, agentic workflows, persistent memory
Limitations	Context window, cost, bespoke retrieval setup	Infrastructure complexity, need for protocol adoption

RAG is primarily about retrieving relevant external data to augment responses, while MCP is about standardizing how models interact with external systems, enabling both retrieval and action. The future may involve combining both paradigms to build more intelligent, flexible, and scalable systems.

4.2 MCP vs. Other Integration Approaches

MCP competes with several other approaches to AI integration (Vaughan-Nichols, 2025):

- **Google's Agent-to-Agent Protocol (A2A)**: Focuses on agent-to-agent communication rather than model-to-data source integration.

- **Workflow automation tools**: Such as Zapier and Pica, which provide specific integration capabilities but lack the comprehensive standardization of MCP.
- **Vendor-specific APIs and SDKs**: Offer tailored solutions but contribute to the fragmentation problem that MCP aims to solve.

MCP's advantage lies in its open, standardized approach and growing industry adoption, positioning it as a potential universal standard for AI integration.

5. Technical Implementation Details

5.1 Architecture and Components

MCP operates on a client-server model involving three main components (Modelcontextprotocol.io, 2025; Web Search Results, 2025):

- **Host (Application/Agent Environment)**: The AI application (e.g., Claude Desktop, IDEs) that initiates and manages connections.
- **MCP Clients**: Intermediaries within the host that handle communication with MCP servers, managing security, capability negotiation, and message routing.
- **MCP Servers**: External or internal services that expose resources, tools, and prompts via the MCP protocol.

5.2 Communication Protocol

The communication between clients and servers primarily uses JSON-RPC 2.0 messages, supporting features such as (Web Search Results, 2025):

- **Stateful Connections**: Maintaining context across multiple interactions.
- **Capability Negotiation**: Clients and servers agree on supported features.
- **Request-Response and Notifications**: For synchronous and asynchronous interactions.

MCP supports multiple transport layers:

- **Stdio**: For local, desktop-based applications.
- **HTTP with Streamable (or SSE)**: For remote, networked services, offering efficient, real-time communication.

5.3 Protocol Primitives

MCP defines several well-structured message types to facilitate interactions (Web Search Results, 2025):

- **InitializeRequest**: Establishes connections and negotiates capabilities.
- **ListToolsRequest**: Discovers available tools and functions.
- **CallToolRequest**: Invokes external tools or functions.
- **ReadResourceRequest**: Retrieves data from external sources.
- **CreateMessageRequest**: Generates or sends messages.

6. Performance Benefits and Limitations

6.1 Benefits

MCP offers several significant performance benefits (Vaughan-Nichols, 2025; Web Search Results, 2025):

- **Unified, standardized integration**: Enables developers to connect their services, APIs, and data sources to any AI client through a single, standardized interface.
- **Two-way communication and rich interactions**: Supports secure, real-time, two-way communication between AI models and external systems.
- **Scalability and ecosystem reuse**: Once implemented for a service, it becomes accessible to any MCP-compliant AI client, fostering an ecosystem of reusable connectors.
- **Consistency and interoperability**: Enforces a consistent JSON request/response format, making it easier to debug, maintain, and scale integrations.
- **Enhanced security and access control**: Supports encryption, granular access controls, and user approval for sensitive actions.
- **Reduced development time and maintenance**: Avoids fragmented, one-off integrations, saving time on setup and ongoing maintenance.

6.2 Limitations and Challenges

Despite its promise, MCP faces several challenges and limitations (Web Search Results, 2025):

- **Authentication**: MCP currently lacks a standardized authentication mechanism; implementers must develop their own solutions.
- **Ecosystem Maturity**: While growing rapidly, the ecosystem of ready-made servers is still developing.
- **Manual Setup**: Current implementations often require manual setup for MCP servers in applications like Claude Desktop.
- **Limited Support for Remote Servers**: Early implementations have focused on local servers, with plans to expand to cloud and remote environments.

7. Ethical Considerations and Implications

7.1 Privacy and Data Security

MCP's ability to connect AI models with external data sources raises important privacy and security considerations. The protocol emphasizes (Web Search Results, 2025):

- **User Consent**: Explicit authorization for data sharing and tool invocation.
- **Data Protection**: Safeguarding user data with robust access controls.
- **Secure Communication**: Encryption and secure transport mechanisms.

7.2 Transparency and Control

MCP aims to provide transparency and control over AI-data interactions by:

- **Clear Capability Negotiation**: Making explicit what data and tools are accessible.
- **User Approval Mechanisms**: Requiring explicit approval for sensitive operations.
- **Logging and Monitoring**: Enabling oversight of AI-data interactions.

7.3 Potential for Misuse

As with any powerful technology, MCP could potentially be misused. Concerns include:

- **Unauthorized Data Access**: If security measures are inadequate.
- **Excessive Automation**: Leading to unintended consequences if AI agents have too much autonomy.
- **Privacy Violations**: If user consent mechanisms are bypassed or inadequate.

8. Conclusion

The Model Context Protocol represents a significant advancement in the integration of large language models with external data sources and tools. By providing a standardized, secure, and flexible protocol for AI-data interaction, MCP addresses the fragmentation and complexity that has hindered the development of truly context-aware AI systems.

Anthropic's leadership in developing and open-sourcing MCP has catalyzed rapid industry adoption, with major players like OpenAI, Google DeepMind, and Microsoft embracing the standard. The growing ecosystem of MCP servers and clients suggests that MCP is well-positioned to become the universal standard for AI integration.

As the protocol matures, we can expect to see continued evolution in areas such as authentication, remote server support, and security enhancements. The combination of MCP with other approaches like RAG may lead to even more powerful hybrid systems that can retrieve, reason, and act seamlessly.

The ultimate promise of MCP is to enable a new generation of AI applications that are deeply integrated with the digital world, capable of accessing real-time data, performing complex actions, and maintaining context across diverse environments. This represents a paradigm shift from isolated, static AI to deeply integrated, context-aware, and action-capable systems that can better serve human needs.

9. References

Anthropic. (2024a). Introducing the Model Context Protocol. Retrieved from https://www.anthropic.com/news/model-context-protocol

Hou, X., et al. (2025). Model Context Protocol (MCP): Landscape, Security Threats, and Future Research Directions. arXiv.

Modelcontextprotocol.io. (2025). Introduction - Model Context Protocol. Retrieved from https://modelcontextprotocol.io/introduction

Vaughan-Nichols, S. (2025, April 25). What is Model Context Protocol? The emerging standard bridging AI and data, explained. ZDNET. Retrieved from https://www.zdnet.com/article/what-is-model-context-protocol-the-emerging-standard-bridging-ai-and-data-explained/

Web Search Results. (2025). Comprehensive findings on Model Context Protocol, including technical details, comparisons, and implementations.