# Model Context Protocol: Evolving to Work Together with Real-Time Media Plumbing

## Introduction

At SignalWire, we've spent years designing robust, real-time communications infrastructures, notably FreeSWITCH and the SignalWire Cloud. Before the introduction of the Model Context Protocol (MCP), we developed the SignalWire Markup Language (SWML)-a structured, YAML- and JSON-based static markup language enabling dynamic interactions across voice calls, video conferences, messaging, and AI-driven conversational agents. Additionally, we created RELAY, a WebSocket-based RPC call control protocol specifically designed to manage live calls remotely, ensuring low latency, state synchronization, and efficient error recovery.

For Conversational AI IVRs, SWML combined with our SignalWire AI Gateway (SWAIG) allows applications to intuitively interact with REST APIs and execute structured RPC-like calls. Our infrastructure explicitly supports robust, stateful real-time WebSocket connections, dynamic AI-driven interactions, clear error handling mechanisms, explicit session and state management, and well-defined message framing and flow control.

## Evaluating the MCP Specification

When reviewing the MCP specification, we recognized and appreciated its ambition to standardize interactions between Large Language Models (LLMs) and external services. The schema demonstrates comprehensive thinking around defining prompts, resources, and interactions. However, our extensive experience with protocols like SIP, WebRTC, RTP, and high-volume real-time call control highlighted potential challenges in MCP's transport layer that could limit practical scalability and reliability in demanding production environments.

### Limitations of MCP’s Current Transport Approach

MCP currently uses "streamable HTTP," implemented via long-lived HTTP POST requests with chunked JSON responses. This approach simplifies initial integration and is well-suited for prototypes or demonstrations. Nonetheless, we identified several critical scalability limitations:

- **Connection fragility:** Persistent HTTP streams can become unstable, causing reliability issues at scale.
- **Fan-out difficulties:** Managing multiple simultaneous streaming consumers becomes increasingly complex.
- **Lack of multiplexing:** Limits the efficiency and scalability of concurrent interactions.

- **Insufficient error handling and recovery:** Current methods lack comprehensive strategies for session resumption and connection recovery.
- **Framing ambiguity:** Absence of standardized framing complicates parsing and can degrade reliability under heavy usage.

### Real-World Lessons from SIP, RTP, WebRTC, and High-Volume Call Control

Deployments involving SIP, RTP, WebRTC, and our high-volume call control protocols demonstrate that large-scale, real-time systems significantly benefit from robust transport technologies. Lessons learned include:

- **Multiplexing:** Essential for efficient handling of thousands of simultaneous streams.
- **Explicit session management:** Robust session and connection resumption strategies are critical for maintaining high availability.
- **Built-in flow control:** Prevents overload and ensures stable system performance.
- **Standardized message framing:** Simplifies parsing, reduces errors, and ensures reliability at scale.

These proven transport mechanisms inherently provide the necessary reliability and scalability required by demanding real-world scenarios.

## Recommendations for Transport Enhancements

To address these identified limitations, we recommend future MCP iterations explicitly support or clearly define alternative transports, keeping practical considerations in mind:

- **WebSockets:** While a logical option due to their real-time capabilities, scalability considerations should be taken into account.
- **Standard REST:** A stateless REST-based approach could offer simplified client access and reduce server-side complexity, greatly enhancing scalability.
- **HTTP/2 or QUIC:** Forward-thinking options providing efficient multiplexing, built-in flow control, and significantly reduced latency, essential for quick response times in LLM-based applications.
- **Explicit session management:** Strategies for robust connection establishment, error recovery, and resumption.
- **Clear framing standards:** Binary or line-delimited JSON framing to enhance parsing efficiency and reliability.

Quick response time is paramount when providing tools to LLM-based applications. Adopting these enhancements would elevate MCP from its current prototype-friendly form to a fully production-ready protocol capable of supporting enterprise-level scalability, performance, and reliability.

## Future Compatibility with SignalWire’s Infrastructure

The structured format of SWML and the extensibility of our SWAIG functions could naturally integrate as an MCP client if the transport and scalability concerns were adequately addressed. SignalWire's voice AI kernel operates as a central hub, seamlessly connecting telecom features with AI capabilities, coordinating voice interactions, listening, and cognitive processing via an integrated LLM. Developers can define conversational behavior using prompts and callbacks that map to SWAIG functions, enabling dynamic modifications of bot behavior, context management, and feature expansions without implementing each functionality individually.

Given this existing architecture, integrating MCP would be straightforward and highly beneficial. MCP could effectively extend SignalWire's robust conversational engine by providing additional remote services that could be effortlessly plugged into the platform. Such integration would amplify interoperability, further enhancing the adaptability and scalability of SignalWire's AI-driven conversational experiences.

SignalWire's existing infrastructure is well-equipped to support MCP's JSON-RPC schema, offering broader interoperability while preserving our native protocols' robustness and performance.

# Conclusion

MCP's vision aligns closely with ours-enabling powerful, standardized interactions between AI-driven agents and external resources. By incorporating transport-layer enhancements informed by our extensive experience with SIP, RTP, WebRTC, and high-volume call control, MCP can mature into a reliable, scalable protocol suitable for demanding real-time applications at enterprise scale. We look forward to MCP's evolution and strongly encourage proactive incorporation of these recommendations to ensure broad industry adoption and effective deployment.