Model Context Protocol & Agent-to-Agent

29 Maggio 2025

Michele Ferro
michele.ferro@aitho.it

Data Scientist & BE Developer

Agenda

- **Model Context Protocol**
  Just like USB-C, built for Agents
- **Agent-to-Agent**
  Google's answer to MCP
- **Example Code**

Model Context Protocol

Just like USB-C, but for Agents

Introduction to MCP

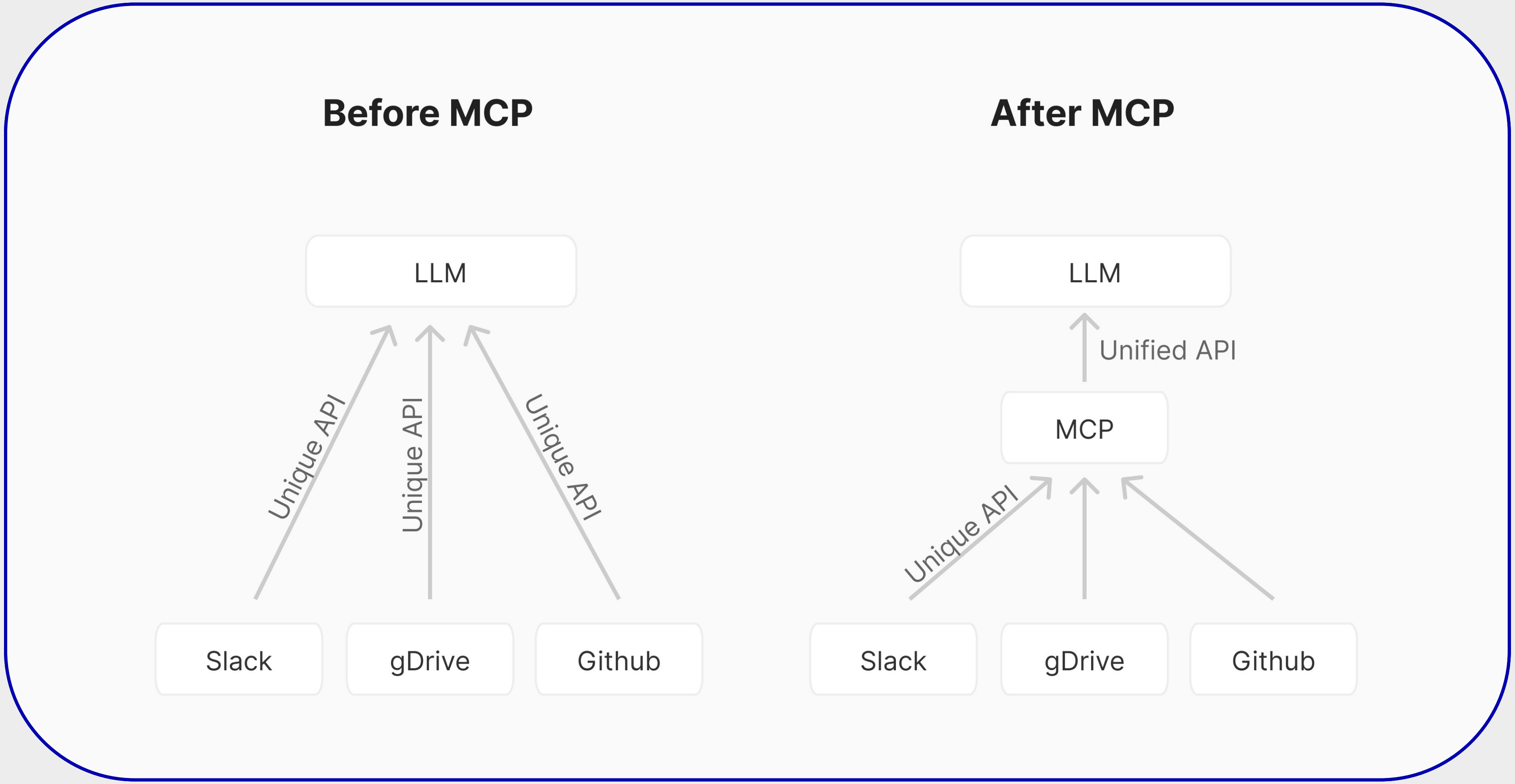
What is the Model Context Protocol (MCP)?

- **Open standard** introduced by ***Anthropic*** (Nov 2024) for AI interoperability
- Connects **LLMs** with external **tools**, **data sources** and **APIs**
- Provides a **reusable context layer** so every AI agent can "plug in" to the same set of services
- Think of it as "**USB-C for AI**": a single, universal interface for many peripherals which reduces custom connector code

Why MCP matters?

- **Standardizes** how AI models request and consume external data, avoiding ad-hoc connectors
- **Eliminates point-to-point wiring**: one protocol works for any service, instead of N custom integrations
- **Accelerates development** and **time-to-market** by offering ready-made SDKs and templates
- **Ensures consistency** and reduces security gaps via a shared schema
- Lets teams focus on **model quality**, not plumbing

MCP: Before VS After



MCP Core Components

MCP Client	MCP Server	Guardian Layer
– Runs inside an AI application (chatbot, IDE plugin, autonomous agent) – Formats requests for context (“*Give me user’s GitHub repos*”) and parses server replies	– Hosts “tool descriptors” that map logical names (e.g. **`list_repos`**) to actual endpoints – Manages **authentication**, **rate-limiting**, and **detailed logging** of every call	– Central enforcement of **access policies** (who can call what, and when – Provides a **Web Application Firewall (WAF)** and continuous **audit trails**

aitho

Protocol Design

- **JSON-based messages** with schemas for **inputs**, **outputs**, **errors**
- **HTTPS transport** supporting sync **request/response** and **streaming**
- **Stateless**: each call is self-contained for easy scaling
- **Plug-in descriptors** let you add new tool types without code changes
- **Standardized error codes** and **retry semantics**

aitho

SDKs & Tooling

- **Python SDK**
  Flask/Django helpers for MCP server and client
- **Node.js SDK**
  Express/Koa middleware to expose REST APIs as MCP tools
- **C# SDK**
  .NET library for enterprise backends (ASP.NET, Azure)
- **CLI Utility:**
  - Spin up a **mock MCP** server locally
  - **Validate** JSON descriptors against the spec
- **IDE Plugins**
  VS Code extension for browsing tools and auto-generating stubs

aitho

MCP Server Capabilities

An **MCP server** acts as a bridge between AI agents and external functionalities, offering standardized interfaces to enhance AI interactions

Tools	Resources	Prompts
– **Executable functions** that LLMs can invoke to perform actions, such as API calls or computations – Designed to be **model-controlled**, allowing AI models to automatically utilize them with appropriate permissions	– **Data endpoints** offering access to information like files, DB records or API responses – Tipically **read-only** and application controlled, enambling AI models to retrieve data without side effects	– **Reusable prompt templates** that guide AI interactions, ensuring consistency and efficiency – **User-controlled**, allowing users to select and customize prompts as needed

This modular architecture enables AI agents to seamlessly **act**, **access data**, and **communicate** within a unified framework

Example Use Cases

- **Automated Code Reviews**
  - AI calls **`list_pull_requests`**
  - Runs **`listeners`**
  - Then **`post_github_comment`**
- **Seamless Collaboration**
  From chat:
  - **`create_jira_issue`**
  - **`assign_ticket`**
  - **`send_slack_message`**
- **On-Demand Data Insights**
  Use **`execute_query`** on SQL/NoSQL DBs, then **summarize** results in natural language

Security & Organizational Benefits

- **Threat Vectors**
  Command injection, credential exposure
- **Mitigations**
  Strict **schema validation**, **least-privilege** access, **real-time logging**
- **Guardian Features**:
  Integrated **WAF**, **rate-limiting**, **audit trails**
- **Benefits**:
  - **Reduced engineering overhead** (one integration, everywhere)
  - **Vendor-agnostic portability** (*Claude*, *GPT*, *Gemini*, etc.)
  - **Centralized maintenance** of tool descriptors

Challenges & Future Roadmap

- **Adoption Challenges**
  Legacy stacks, governance overhead, learning curve
- **Vendor Coordination**
  Alignment of multiple LLM and service providers
- **Roadmap**
  – **Official support** in SDKs by OpenAI, Google, Microsoft
  – **IDE deep-links** (IntelliJ, PyCharm, VS Code)
  – **Managed cloud services** (AWS/GCP/Azure–hosted MCP)
  – Foundation for **agentic AI** that auto-discovers and orchestrates tools

Agent-to-Agent

Google's answer to MCP

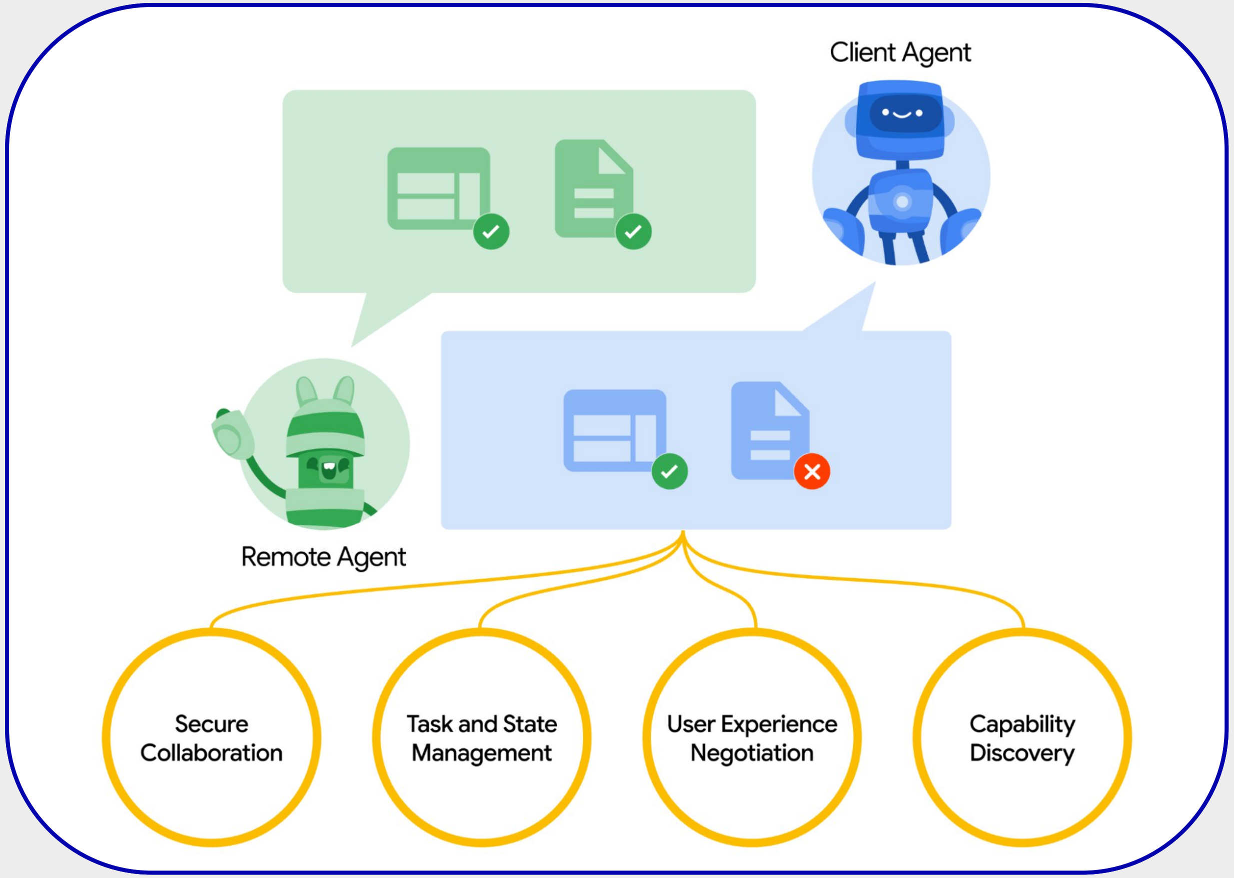
Introduction to A2A

What is the Agent-to-Agent (A2A) Protocol?

- **Open standard** introduced by ***Google*** (April 2025) for direct agent collaboration
- Enables **peer-to-peer messaging** between heterogeneous AI agents
- Complements MCP by covering **agent-level orchestration**
- Built on **HTTP**, **Server-Sent Events (SSE)**, **JSON-RPC**
- **Eliminates** bespoke middleware-agents "speak" natively

Purpose & Vision

- **Empower** agents to:
  - discover each other's capabilities at runtime
  - delegate tasks dynamically based on skill sets
  - coordinate multi-step workflows autonomously
- Shift from "model as tool" to "**agent as autonomous collaborator**"
- Seed an **Agent Economy** of micro-agents composing larger services



aitho

A2A Key Building Blocks

Agent Cards	Communication Channels	Security Layer
– **JSON documents** advertising metadata, endpoints, permissions – Enable dynamic **discovery** and **self-description**	– **JSON-RPC over HTTP** for sync calls – **SSE** for streaming updates and event notifications	– **OAuth2/OpenID Connect** for authentication – **JWT-based claims** for fine-grained authorization

Agent Cards in Detail

Metadata	Capabilities	Permissions	Health Checks	Extensibility
– Name – Version – Description – Maintainer contact	– Methods (e.g. `translate( text, lang )`) – Input/output schemas	– Who can call which method – Rate/quota limits	Endpoints reporting: – Status – Up-time – Metrics	Custom fields for domain-specific needs

Communication Patterns

- **Request/Response** for quick, transactional tasks
- **Streaming Updates** (SSE) for long-running or progressive outputs
- **Event Subscription:** agents subscribe to peers' event streams
- **Error Handling:** standardized error objects, retry/back-off logic

SDKs & Sample Integrations

- **Python SDK**
  AsyncIO + FastAPI framework support
- **JavaScript SDK**
  Works in **Node.js** and browser contexts
- **CLI Tool**
  Scaffolds new agents; validates Agent Card schemas
- **Sample Projects:**
  - **Semantic Kernel** and **LangChain** integration examples
  - **MongoDB/Redis** discovery store implementations
- **Observability**
  Grafana & Prometheus plugins for monitoring

Future Directions

- **Integration with Agent Network Protocol (ANP)**
  Decentralized discovery via the ANP
- **Vendor Coordination**
  Unify tool-level (MCP) and agent-level (A2A) orchestration
- **Roadmap**
  Marketplaces to publish, license and compose agents
- **Standards Governance**
  Working groups evolving specs and security guidelines
- **Toward AI Autonomy**
  Agents that self-coordinate, self-heal and self-optimize

A2A VS MCP

Comparison, Pros & Cons

aitho

Model Context Protocol

- **Purpose**
  Connect LLMs to **tools**, **APIs**, and **data sources**
- **Scope**
  Agent-to-environment interaction
- **Pros**
  - **Standardized integration** with external systems
  - Easy to **instrument**, **log** and **secure**
  - Enables **tool reuse** across multiple agents
- **Cons**
  - No native agent-to-agent collaboration
  - Requires predefined tool schemas and descriptors

Agent-to-Agent Protocol

- **Purpose**
  Facilitate **communication between agents**
- **Scope**
  Agent-to-agent coordination and orchestration
- **Pros**
  - **Enables dynamic delegation and agent discovery**
  - Supports **peer-to-peer**, asynchronous workflows
  - Promotes **modularity** and **composable AI** systems
- **Cons**
  - Higher complexity (discovery, permissions, fallbacks)
  - Security and trust between agents is harder to enforce

MCP & A2A: working together in a new agentic era

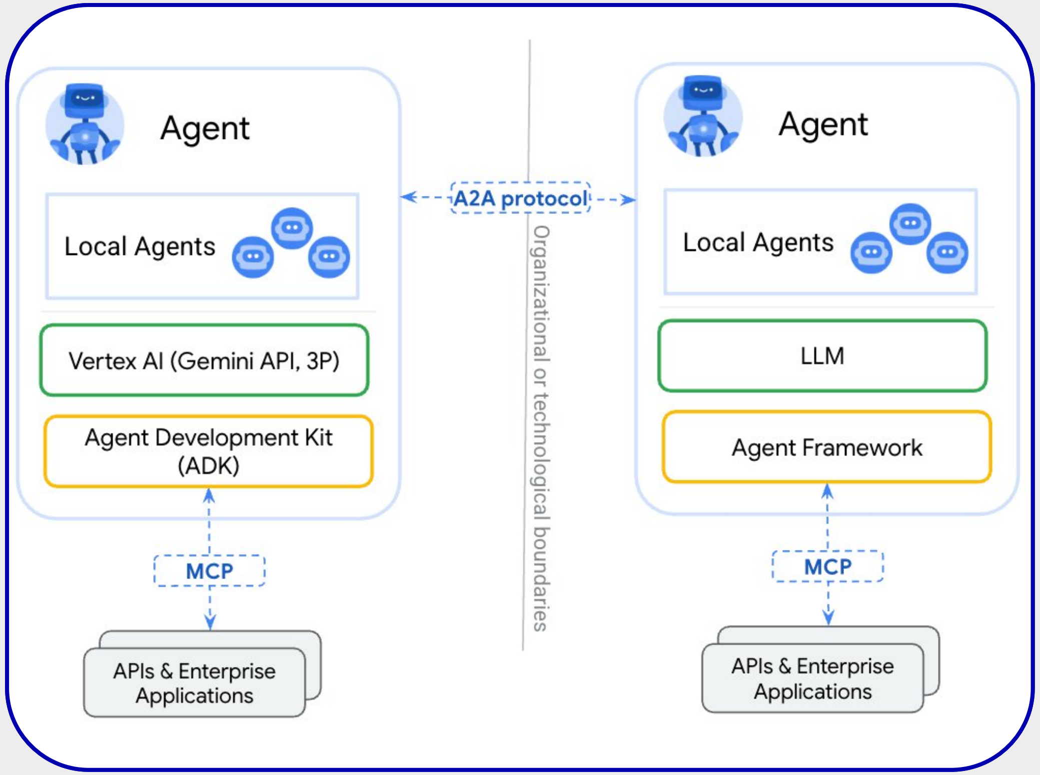
The future of AI systems lies in **agentic collaboration**, where multiple agents **coordinate**, **delegate** and **act** in dynamic environments

MCP & A2A create a composable ecosystem where agents use:

- **MCP** to **act**
- **A2A** to **think** and **delegate**

This synergy allows for:

- **Dynamic multi-agent orchestration**
- **Scalable**, **decentralized intelligence**
- The rise of a true **agent economy**


Agent
Local Agents
Vertex AI (Gemini API, 3P)
Agent Development Kit (ADK)
MCP
APIs & Enterprise Applications
A2A protocol
Organizational or technological boundaries
Agent
Local Agents
LLM
Agent Framework
MCP
APIs & Enterprise Applications

Let's look at the code!

Thank you!

aitho
welcome home developers

🌐 www.aitho.it

✉ partners@aitho.it

Any question?

aitho