AI AGENTS | PROTOCOL STANDARDS | AGENT FRAMEWORKS

What Every AI Engineer Should Know About A2A, MCP & ACP

How today's top AI protocols help agents talk, think, and work together

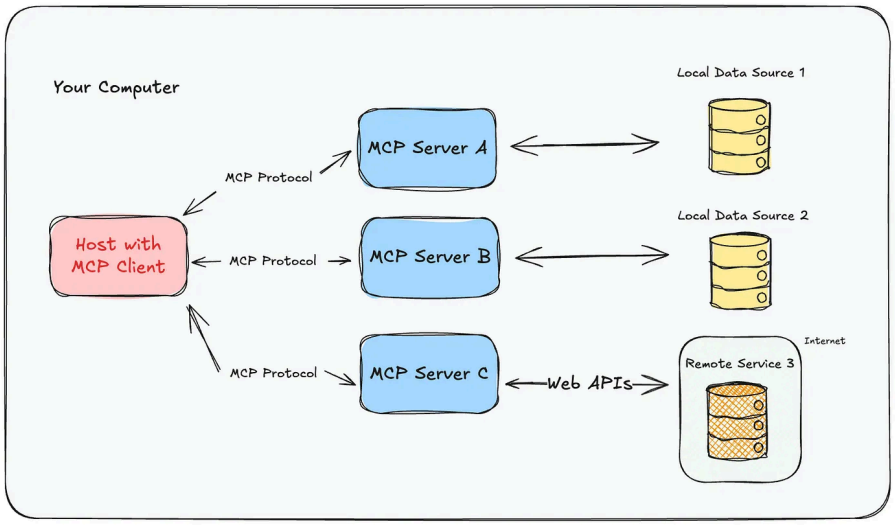
Edwin Lisowski Follow 7 min read · Apr 24, 2025

62 3

What is MCP (Model Context Protocol)?

The Model Context Protocol (MCP), introduced by Anthropic, defines a standardized interface for supplying structured, real-time context to large language models (LLMs).

💡 *Catch up with my previous article about MCP.* 💡



https://modelcontextprotocol.io/

Core Functionalities

Contextual Data Injection

MCP lets you pull in external resources — like files, database rows, or API responses — right into the prompt or working memory. All of it comes

through a standardized interface, so your LLM can stay lightweight and clean.

Function Routing & Invocation
MCP also lets models call tools dynamically. You can register capabilities like `searchCustomerData` or `generateReport`, and the LLM can invoke them on demand. It's like giving your AI access to a toolbox, but without hardwiring the tools into the model itself.

Prompt Orchestration
Rather than stuffing your prompt with every possible detail, MCP helps assemble just the context that matters. Think modular, on-the-fly prompt construction — smarter context, fewer tokens, better outputs.

Implementation Characteristics

- Operates over HTTP(S) with JSON-based capability descriptors

Open in app ↗

Medium Search Write

- Compatible with API gateways and enterprise authentication standards (e.g., OAuth2, mTLS)

Engineering Use Cases

① **LLM integrations for internal APIs:** Enable secure, read-only or interactive access to structured business data without exposing raw endpoints.

② **Enterprise agents:** Equip autonomous agents with runtime context from tools like Salesforce, SAP, or internal knowledge bases.

③ **Dynamic prompt construction:** Tailor prompts based on user session, system state, or task pipeline logic

What is ACP (Agent Communication Protocol)?

The Agent Communication Protocol (ACP) is an open standard originally proposed by BeeAI and IBM to enable structured communication, discovery, and coordination between AI agents operating in the same local or edge environment.

Unlike cloud-oriented protocols such as A2A or context-routing protocols like MCP, ACP is designed for local-first, real-time agent orchestration with minimal network overhead and tight integration across agents deployedc within a shared runtime.

Protocol Design & Architecture

ACP defines a decentralized agent environment in which:

- Each agent advertises its identity, capabilities, and state using a local broadcast/discovery layer.
- Agents communicate through event-driven messaging, often using a local bus or IPC (inter-process communication) system.
- A runtime controller (optional) can orchestrate agent behavior, aggregate telemetry, and enforce execution policies.

ACP agents typically operate as lightweight, stateless services or containers with a shared communication substrate.

Implementation Characteristics

- Designed for low-latency environments (e.g., local orchestration, robotics, offline edge AI)
- Can be implemented over gRPC, ZeroMQ, or custom runtime buses
- Emphasizes local sovereignty — no cloud dependency or external service registration required
- Supports capability typing and semantic descriptors for automated task routing

Engineering Use Cases

① **Multi-agent orchestration on edge devices** (e.g., drones, IoT clusters, or robotic fleets)

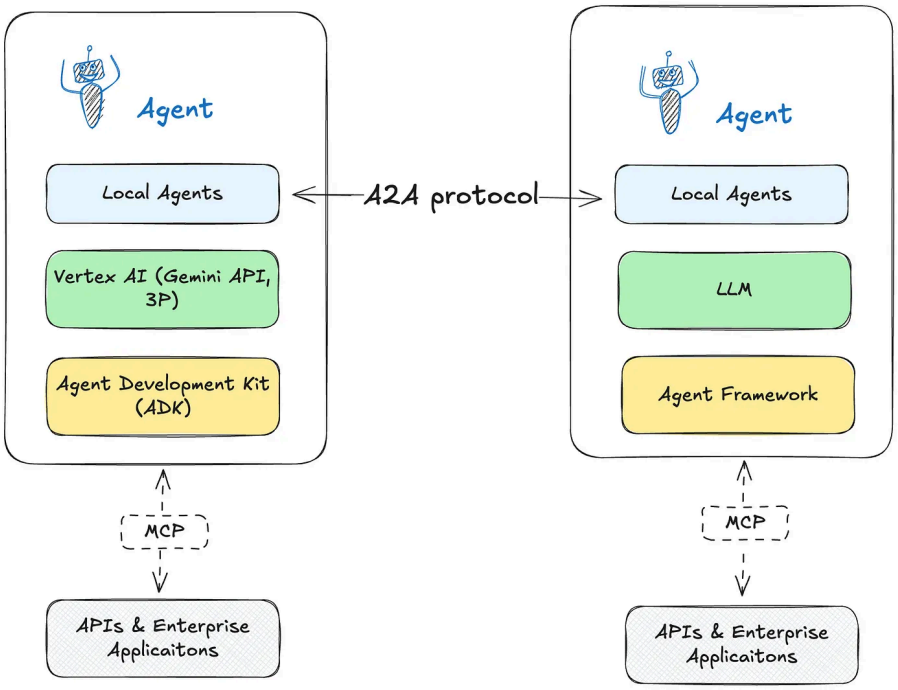
② **Local-first LLM systems** coordinating model invocations, sensor inputs, and action execution

③ **Autonomous runtime environments** where agents must coordinate without centralized cloud infrastructure

> *In short, ACP offers a runtime-local protocol layer for modular AI systems — prioritizing low-latency coordination, resilience, and composability. It's a natural fit for privacy-sensitive, autonomous, or edge-first deployments where cloud-first protocols are impractical.*

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

The Agent-to-Agent (A2A) Protocol, introduced by Google, is a cross-platform specification for enabling AI agents to communicate, collaborate, and delegate tasks across heterogeneous systems.



https://google.github.io/

Unlike ACP's local-first focus or MCP's tool integration layer, A2A addresses horizontal interoperability — standardizing how agents from different vendors or runtimes can exchange capabilities and coordinate workflows over the open web.

Protocol Overview

A2A defines a HTTP-based communication model where agents are treated as interoperable services. Each agent exposes an "Agent Card" — a machine-readable JSON descriptor detailing its identity, capabilities, endpoints, and authentication requirements.

Agents use this information to:

- Discover each other programmatically
- Negotiate tasks and roles
- Exchange messages, data, and streaming updates

A2A is transport-layer agnostic in principle, but currently specifies JSON-RPC 2.0 over HTTPS as its core mechanism for interaction.

Core Components

Agent Cards
JSON documents describing an agent's capabilities, endpoints, supported message types, auth methods, and runtime metadata.

A2A Client/Server Interface
Each agent may function as a client (task initiator), a server (task executor), or both, enabling dynamic task routing and negotiation.

Message & Artifact Exchange
Supports multipart tasks with context, streaming output (via SSE), and persistent artifacts (e.g., files, knowledge chunks).

User Experience Negotiation
Agents can adapt message format, content granularity, and visualization to match downstream agent capabilities.

Security Architecture

- OAuth 2.0 and API key-based authorization
- Capability-scoped endpoints — agents only expose functions required for declared interactions
- Agents can operate in "opaque" mode — hiding internal logic while revealing callable services

Implementation Characteristics

- Web-native by design: built on HTTP, JSON-RPC, and standard web security
- Model-agnostic: works with any agent system (LLM or otherwise) that implements the protocol
- Supports **task streaming** and multi-turn collaboration with lightweight payloads

Engineering Use Cases

① **Cross-platform agent ecosystems** where agents from different teams or vendors need to interoperate securely

② **Distributed agent orchestration** in cloud-native AI environments (e.g., Vertex AI, LangChain, HuggingFace Agents)

③ **Multi-agent collaboration frameworks**, such as enterprise AI workflows that span multiple systems (e.g., CRM, HR, IT agents)

Protocols Compared Side-by-Side

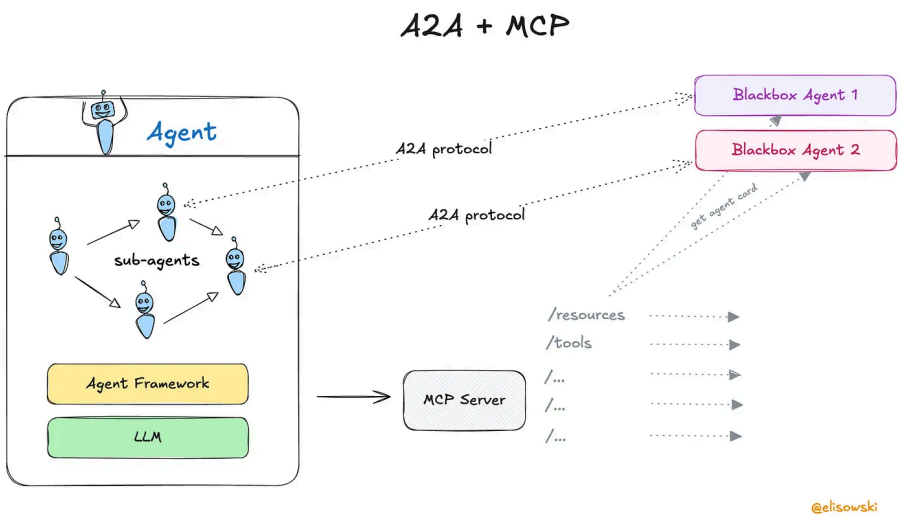
Feature	MCP	ACP	A2A
Primary focus	Context injection for LLMs	Local coordination of agents	Cross-platform agent communication
Architecture	Client-server (host/server model)	Decentralized, local runtime	HTTP-based client/server with Agent Cards
Scope	Vertical integration (tools → model)	Local-first agent runtime	Horizontal integration (agent ↔ agent)
Discovery Mechanism	Registry of tools on server	Local broadcast / runtime registration	Agent Card over HTTP(S)
Transport Protocol	HTTP(S), JSON	IPC, ZeroMQ, gRPC (flexible)	JSON-RPC 2.0 over HTTPS
Security Model	App-layer auth, OAuth2, scoped APIs	Runtime sandboxing, private network security	OAuth2, scoped endpoint exposure
Best for	LLM apps with external data/tool needs	Edge AI, embedded systems, offline agents	Multi-agent workflows across platforms
Example Use Case	Connecting an LLM to internal APIs	On-device coordination of multiple small agents	Distributed enterprise agents collaborating

@elisowski

Complementary or Competitive?

A2A + MCP

A2A and MCP aren't fighting each other — they're solving totally different parts of the agentic AI puzzle, and they actually fit together pretty nicely.



https://google.github.io/

Think of MCP as the protocol that lets AI agents plug into the world. It gives them access to files, APIs, databases — basically, all the structured context

they need to do something useful. Whether it's pulling real-time sales data or generating a custom report, MCP handles the connection to tools and data.

Now layer on A2A. This is where agents start collaborating. A2A gives them a shared language and set of rules to discover each other, delegate tasks, and negotiate how they'll work together — even if they're built by different vendors or running on different platforms.

So here's a simple way to think about it:
✧ MCP connects AI to tools.
✧ A2A connects AI to other AI.

> *Together, they form a strong modular base for building smart, collaborative systems.*

What About ACP?

Then there's ACP, which takes a different approach altogether. It's all about local-first agent coordination — no cloud required. Instead of using HTTP and web-based discovery, ACP enables agents to find and talk to each other right inside a shared runtime.

This is perfect for situations where:

- You have limited bandwidth or need low-latency (like in robotics or on-device assistants),
- Privacy matters and you want to keep everything offline,
- Or you're deploying in environments cut off from the internet (e.g., factory floors, edge nodes).

> *ACP isn't trying to compete with A2A — it just fills a different niche. But in some setups, especially in tightly controlled environments, ACP might replace A2A entirely, because it skips the overhead of web-native protocols and just gets the job done locally.*

Convergence or Fragmentation?

As more teams adopt these protocols, a few possible futures are taking shape.

Best case? We see convergence. Imagine a unified agent platform where A2A handles the back-and-forth between agents, MCP manages access to tools and data, and ACP-style runtimes plug in for edge or offline scenarios. Everything just works, and developers can build on top without worrying which protocol is doing what behind the scenes.

Worst case? Things fragment. Different vendors push their own flavors of A2A or MCP, and we end up with a mess — like the early days of web services, when nothing talked to anything else without a lot of glue code.

The middle ground? Open-source tools and middleware could save the day. These projects would sit between agents and protocols, abstracting the differences and giving devs a clean, unified API — while translating under the hood depending on where and how your agents run.

> *In short: we're early. But how we build and adopt these standards now will shape whether AI agents become a cohesive ecosystem — or a patchwork of silos.*

If you're wondering where to start or how to bring it all together, the team at Addepto can help. We've taken companies from "What do we even do with this?" to fully operational agent ecosystems — fast. You can also hit me up on Linkedin.

Mcp Protocol | A2a Protocol | Agent Protocol | Agent Ecosystem

Written by Edwin Lisowski

756 Followers · 29 Following

Co-Founder @Addepto (https://addepto.com) | Technology Expert & Social Science Enthusiast | AI Knowledge Base to Boost Your Business: https://context-clue.com/

Follow

Responses (3)

Thomas Modern

What are your thoughts?

Ryan Fattini

5 days ago (edited)

Very good summary, we argued for (ACP) pushing out further than MCP with event driven approaches for organizational alignment https://medium.com/p/92545cabe462

3 1 reply Reply

Soumya Yadav
1 day ago

That's quite an interesting take. Would invite you to check this: https://medium.com/@psoumyadav/mcp-and-a2a-how-ai-agents-are-redefining-collaboration-and-automation-c866dcf420b6

1 Reply

Mar Gomez
6 days ago

"Written by ChatGPT, edited by Edwin Lisowski" makes more sense to me.

Reply

More from the list: "LLM"

Curated by Thomas Modern

In Google Cloud... by Heik...
Getting Started with Google A2A: A Hands-o...
Apr 15

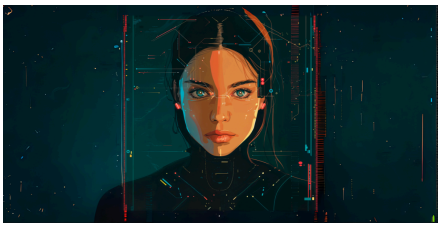
Minyang Chen
Google's A2A Protocol: Seamless Agent...
· Apr 18

CPlog
Introduction to LangGraph: A Beginner'...
Feb 14, 2024

AI Clou...
Bu Ap
Apr

View list

More from Edwin Lisowski

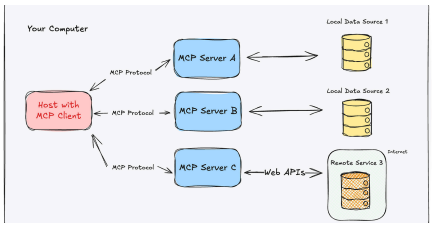


Edwin Lisowski

AI Agents vs Agentic AI: What's the Difference and Why Does It Matter?

If you've been keeping an eye on artificial intelligence (AI) lately, you've probably heard...

Dec 18, 2024 1.4K 36

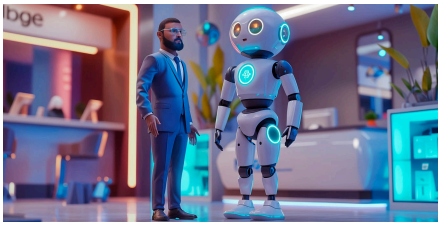


Edwin Lisowski

MCP Explained: The New Standard Connecting AI to Everything

How Model Context Protocol is making AI agents actually do things

Apr 15 241 1

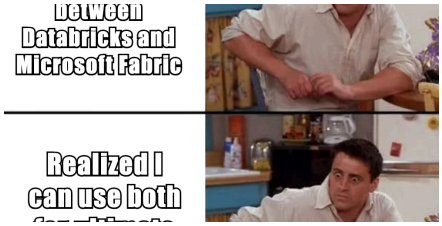


Edwin Lisowski

A List of AI Agents Set to Dominate in 2025

From streamlining workflows to performing human-like tasks, the AI agents of 2025...

Dec 9, 2024 330 13



Edwin Lisowski

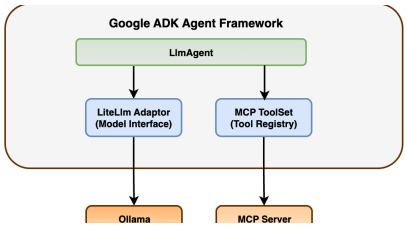
Databricks vs Microsoft Fabric: Which One is Best for Your Data...

Choosing the right data platform can feel overwhelming, especially with so many...

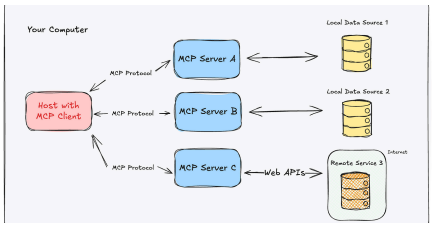
Mar 5 183 4

See all from Edwin Lisowski

Recommended from Medium



In Google Cloud - Community by Arjun Prabhulal



Edwin Lisowski

Building AI Agents with Google ADK, Gemma 3, and MCP Tools

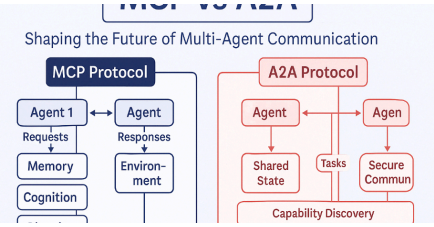
AI Agents using ADK + Gemma3 + MCP

Apr 24 251 4

MCP Explained: The New Standard Connecting AI to Everything

How Model Context Protocol is making AI agents actually do things

Apr 15 241 1

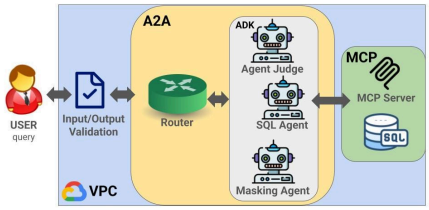


Lekha Priya

MCP vs A2A: What the Future of AI Agent Communication Looks Like

It started with a question that kept resurfacing in nearly every conversation I've had around...

Apr 20 6



Rubens Zimbres

Agent Development Kit: Enhancing Multi-Agents Systems with A2A...

Lately we've been flooded with new innovations and product launches. I was at...

Apr 18 248 2

In Level Up Coding by Anmol Baranwal

30+ MCP Ideas with Complete Source Code

MCP is going viral. AI agents can now talk to real tools & apps and actually get stuff done.

Apr 21 1.2K 16

Manoj Desai

Smart Routing: The Hidden Secret Behind 10x More Powerful AI...

Transform AI systems with smart routing for dramatic performance gains and cost...

Apr 20 180 5

See more recommendations