LLMs and Tools
Part-3: Agentic Workflow

Advanced Large Language Models

ELL8299 · AIL861 · ELL881

Dinesh Raghu
Senior Researcher & Manager, IBM Research

LLMs and Tools

Part 1: Incorporating Tools during Fine-tuning (Tool Augmentation)

Part 2: Teaching LLMs to Use External APIs (Function Calling)

Part 3: Automating Complex Tasks (AI Agents)

i. ReAct, Self-Refine and Reflexion
ii. MemGPT and SWEET-RL
iii. Agentic Protocols

The Need for Agent Protocols

- Just as the internet needed protocols like HTTP and DNS to let websites and servers communicate, AI systems need agent protocols to enable seamless interaction

- Today's agents are built using diverse frameworks (such as LangGraph, LangFlow, CrewAI, AutoGen, and AutoGPT). Each framework defines and manages "tools" differently, leading to fragmentation and poor interoperability.

- Agent protocols provide a common language shared rules and formats that let agents, tools, and external resources communicate reliably.

Overview of Protocols

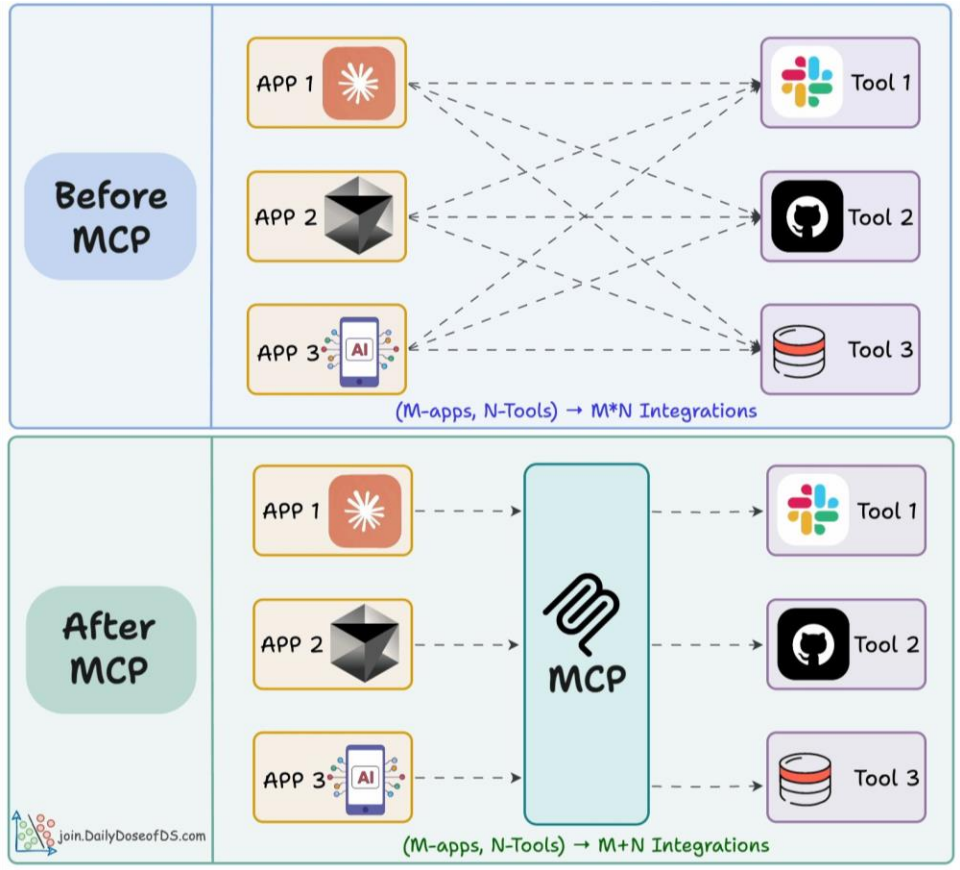
Entity	Scenarios	Protocol	Proposer	Application Scenarios	Key Techniques	Development Stage
Context-Oriented	**General-Purpose**	MCP Anthropic (2024)	Anthropic	Connecting agents and resources	RPC, OAuth	Factual Standard
	Domain-Specific	agent.json WildCardAI (2025)	Wildcard AI	Offering website information to agents	/.well-known	Drafting
Inter-Agent	**Genreal-Purpose**	A2A Google (2025)	Google	Inter-agent communication	RPC, OAuth	Landing
		ANP Chang (2024)	ANP Community	Inter-agent communication	JSON-LD, DID	Landing
		AITP NEAR (2025)	NEAR Foundation	Inter-agent communication	Blockchain, HTTP	Drafting
		AComP AI and Data (2025)	IBM	Multi agent system communication	OpenAPI	Drafting
		AConP Cisco (2025)	Langchain	Multi agent system communication	OpenAPI, JSON	Drafting
		Coral citeagentcoralprotocol	The Coral Community	Multi agent system communication	-	Drafting
		Agora Marro et al. (2024)	University of Oxford	Meta protocol between agents	Protocol Document	Concept
	Domain-Specidic	LMOS Eclipse (2025)	Eclipse Foundation	Internet of things and agents	WOT, DID	Landing
		Agent Protocol AlEngineerFoundation (2025)	AI Engineer Foundation	Controller-agent interaction	RESTful API	Landing
		LOKA Ranjan et al. (2025)	CMU	Decentralized agent system	DECP	Concept
		PXP Srinivasan et al. (2024)	BITS Pilani	Human-agent interaction	-	Concept
		CrowdES Bae et al. (2025)	GIST.KR	Robot-agent interaction	-	Concept
		SPPs Gąsieniec et al. (2024)	University of Liverpool	Robot-agent interaction	-	Concept

Credits: A Survey of AI Agent Protocols, Yang et al., June 2025

Outline

- Model Context Protocol (MCP) — Factual Standard
- Agent2Agent (A2A) Protocol — Landing
- agents.json — Drafting

Model Context Protocol (MCP)



- MCP is an open-source standard for connecting AI applications (e.g. ChatGPT or custom enterprise-chatbots) to data sources (e.g. local files, databases), tools (e.g. search engines, calculators) and prompts (e.g. specialized prompts)

- MCP is often described as USB-C port for AI applications. Just as USB-C provides a standardized way to connect electronic devices, MCP provides a standardized way to connect AI applications to external systems

Image Credits: https://x.com/akshay_pachaar/status/1934591700799996303

Model Context Protocol (MCP)

MCP is built on a client-server architecture and it consists of three primary components:

1. Host
   - AI application that end-users interact with directly
   - Examples include ChatGPT, Claude Desktop, and Cursor

2. Server
   - an external program or service that exposes capabilities to AI models via the MCP protocol

3. Client
   - a component within the Host application that manages communication with a specific MCP Server

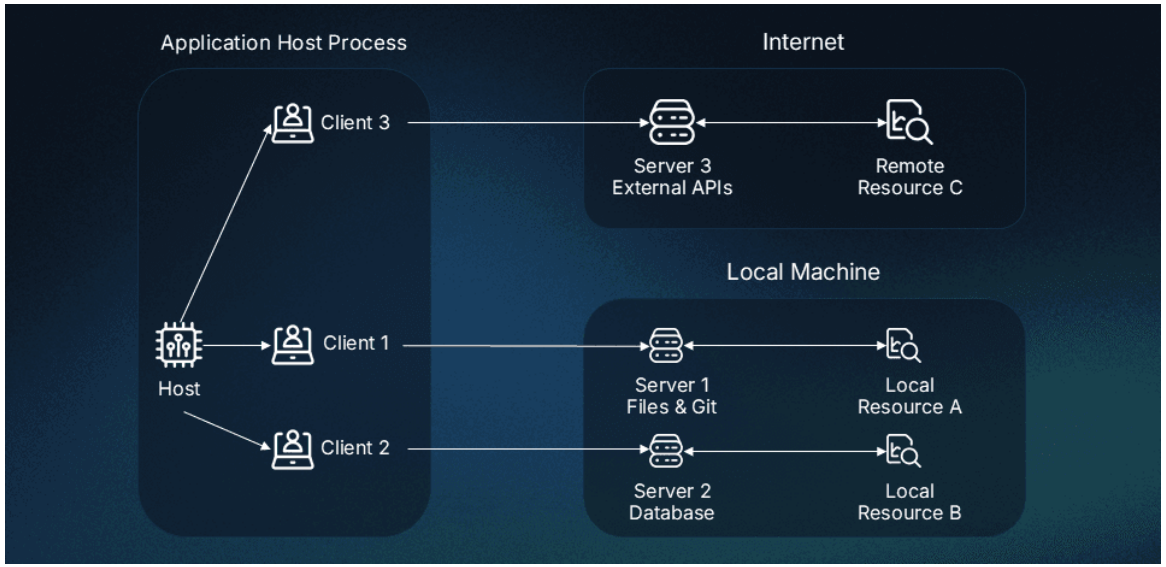


Image Credits: https://www.descope.com/learn/post/mcp

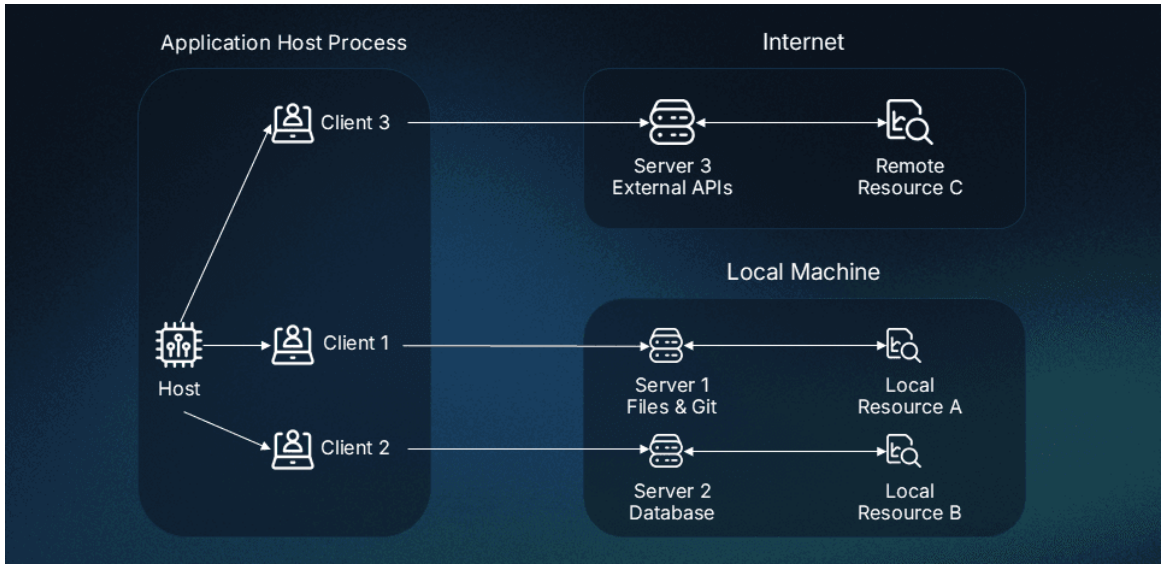
Model Context Protocol (MCP)

Transport layer

The communication mechanism between clients and servers. MCP supports two primary transport methods:

1. STDIO (Standard Input/Output): Mainly local integrations where the server runs in the same environment as the client.

2. HTTP+SSE (Server-Sent Events): Remote connections, with HTTP for client requests and SSE for server responses and streaming.



All communication in MCP uses JSON-RPC 2.0 as the underlying message standard, providing a standardized structure for requests, responses, and notifications.

Image Credits: https://www.descope.com/learn/post/mcp

Model Context Protocol (MCP)

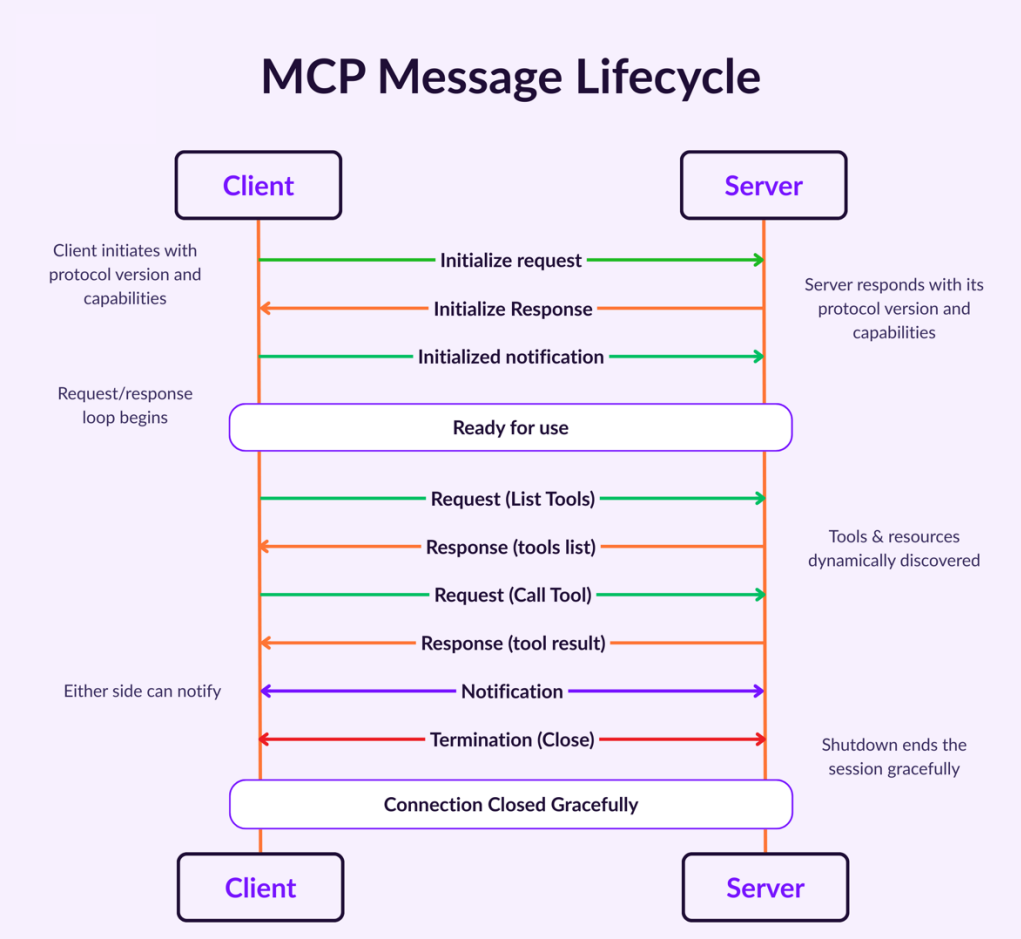


Image Credits: https://blog.neosage.io/p/why-every-ai-builder-needs-to-understand

Model Context Protocol (MCP)

MCP defines three core primitives that servers can expose:

1. **Tools**: Executable functions that model can decide to invoke to perform actions

2. **Resources**: Data sources that provide read-only contextual information
   - e.g., file contents, database records, and documentation

3. **Prompts**: Reusable templates and workflows that help structure interactions with language models
   - e.g., prompt templates

Model Context Protocol (MCP)

Tools:

- **Control**: Tools are typically model-controlled – LLM decides when to call them based on the user's request and context.
- **Safety**: Due to their ability to perform actions with side effects, tool execution can be dangerous. Therefore, they typically require explicit user approval.
- **Use Cases**: Sending messages, creating tickets, querying APIs, performing calculations.

```
{
  name: "searchFlights",
  description: "Search for available flights",
  inputSchema: {
    type: "object",
    properties: {
      origin: { type: "string", description: "Departure city" },
      destination: { type: "string", description: "Arrival city" },
      date: { type: "string", format: "date", description: "Travel date" }
    },
    required: ["origin", "destination", "date"]
  }
}
```

Method	Purpose	Returns
`tools/list`	Discover available tools	Array of tool definitions with schemas
`tools/call`	Execute a specific tool	Tool execution result

Model Context Protocol (MCP)

Resources:

- **Control**: Resources are application-controlled - Host application typically decides when to access them.
- **Nature**: They are designed for data retrieval with minimal computation, similar to GET endpoints in REST APIs.
- **Safety**: Since they are read-only, they typically present lower security risks than Tools.
- **Use Cases**: Accessing file contents, retrieving database records, reading configuration information.

Model Context Protocol (MCP)

Resources:

Method	Purpose	Returns
`resources/list`	List available direct resources	Array of resource descriptors
`resources/templates/list`	Discover resource templates	Array of resource template definitions
`resources/read`	Retrieve resource contents	Resource data with metadata
`resources/subscribe`	Monitor resource changes	Subscription confirmation

Direct Resources - fixed URIs that point to specific data.

Example: `calendar://events/2024` - returns calendar availability for 2024

Resource Templates - dynamic URIs with parameters for flexible queries.

Example: `travel://activities/{city}/{category}` - returns activities by city and category

Model Context Protocol (MCP)

Prompts:

- **Control**: Prompts are **user-controlled**, often presented as options in the Host application's UI.
- **Purpose**: They structure interactions for optimal use of available Tools and Resources.
- **Selection**: Users typically select a prompt before the AI model begins processing, setting context for the interaction.
- **Use Cases**: Common workflows, specialized task templates, guided interactions.

```python
def code_review(code: str, language: str) -> list:
    """Generate a code review for the provided code snippet."""
    return [
        {
            "role": "system",
            "content": f"You are a code reviewer examining {language} code. Provide a detailed review highlighting best practices, potential issues, and suggestions for improvement."
        },
        {
            "role": "user",
            "content": f"Please review this {language} code:\n\n```{language}\n{code}\n```"
        }
    ]
```

Model Context Protocol (MCP)

- When tool call requires private data (e.g., credentials, tokens), the LLM may include it in the call.
- For cloud-hosted LLMs, this means private data is uploaded, creating major security and privacy risks.
- MCP decouples tool invocations from LLM responses

```
[
  {
    "name": "GitHub",
    "transport": "stdio",
    "command": "npx",
    "args": [
      "-y",
      "@modelcontextprotocol/server-github"
    ],
    "env": {
      "GITHUB_TOKEN": "ghp_yourpersonalaccesstokenhere",
      "GITHUB_DEFAULT_REPO": "your-org/your-repo",
      "MCP_SERVER_LOG_LEVEL": "info"
    },
    "description": "MCP GitHub server for accessing issues, PRs, and repos"
  }
]
```

Client Config

Image Credits: https://www.descope.com/learn/post/mcp

Model Context Protocol (MCP)

What happens during Server-Side API Modifications?

- Non-MCP Approach
  - LLM function-call fails API specs have changed
  - Causes downtime – requires spec change or code change
- MCP Approach
  - MCP server will notify the MCP client about modifications
  - MCP client will pull the tool-list again without causing downtime or failures

Image Credits: https://www.descope.com/learn/post/mcp

Outline

- Model Context Protocol (MCP) — Factual Standard
- Agent2Agent (A2A) Protocol — Landing
- agents.json — Drafting

Agent2Agent Protocol (A2A)

Core Actors:

- User
- A2A Client (Client Agent)
- A2A Server (Remote Agent)

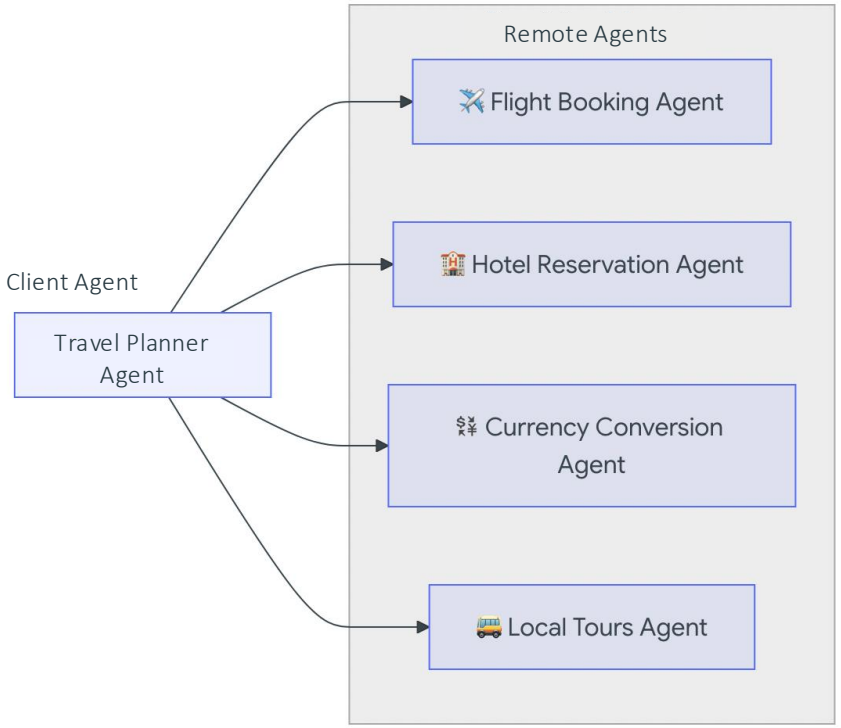
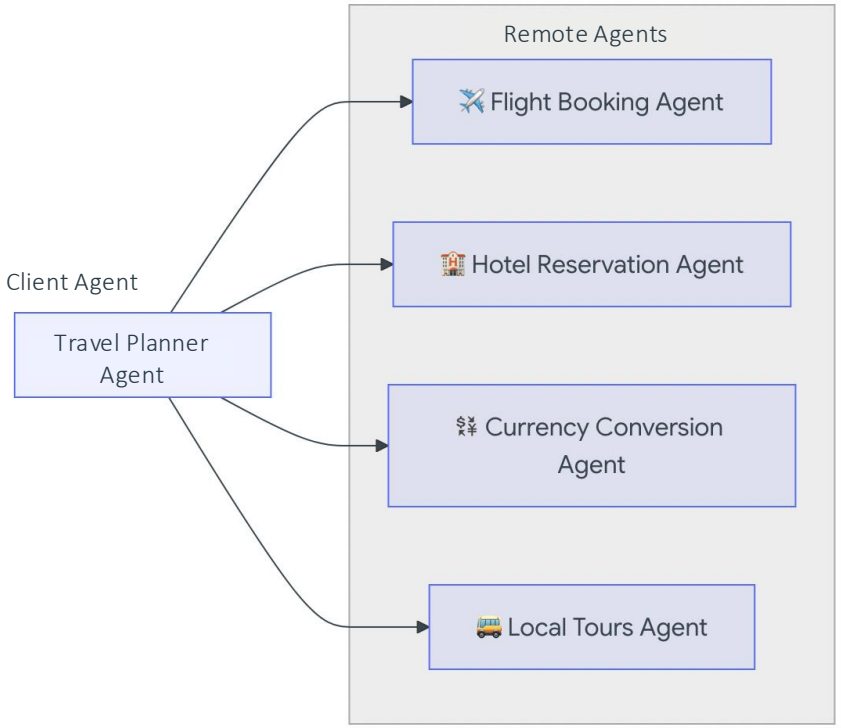


Image Credits: https://a2a-protocol.org/latest/topics/what-is-a2a/

Agent2Agent Protocol (A2A)

Core Actors:

- User
- A2A Client (Client Agent)
- A2A Server (Remote Agent)



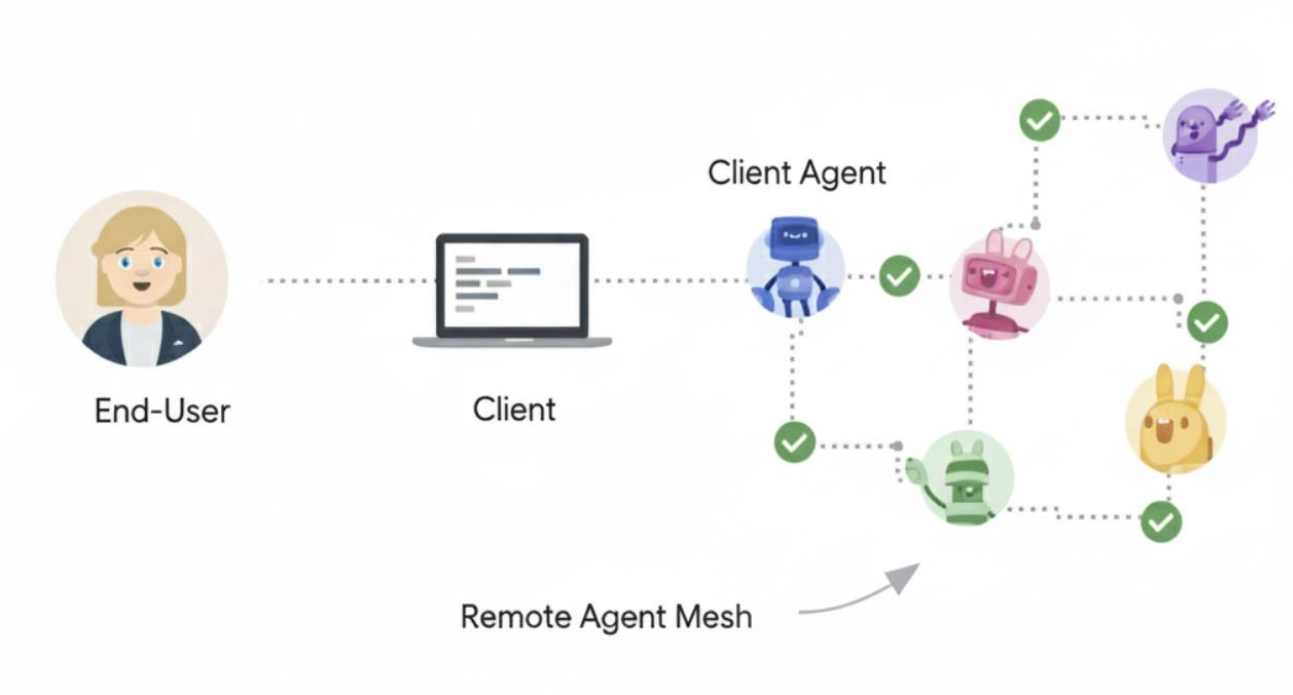


Image Credits: https://a2a-protocol.org/latest/topics/what-is-a2a/

Agent2Agent Protocol (A2A)

Element	Description	Key Purpose
Agent Card	A JSON metadata document describing an agent's identity, capabilities, endpoint, skills, and authentication requirements.	Enables clients to discover agents and understand how to interact with them securely and effectively.
Task	A stateful unit of work initiated by an agent, with a unique ID and defined lifecycle.	Facilitates tracking of long-running operations and enables multi-turn interactions and collaboration.
Message	A single turn of communication between a client and an agent, containing content and a role ("user" or "agent").	Conveys instructions, context, questions, answers, or status updates that are not necessarily formal artifacts.
Part	The fundamental content container (for example, TextPart, FilePart, DataPart) used within Messages and Artifacts.	Provides flexibility for agents to exchange various content types within messages and artifacts.
Artifact	A tangible output generated by an agent during a task (for example, a document, image, or structured data).	Delivers the concrete results of an agent's work, ensuring structured and retrievable outputs.

Image Credits: https://a2a-protocol.org/latest/topics/what-is-a2a/

MCP vs A2A

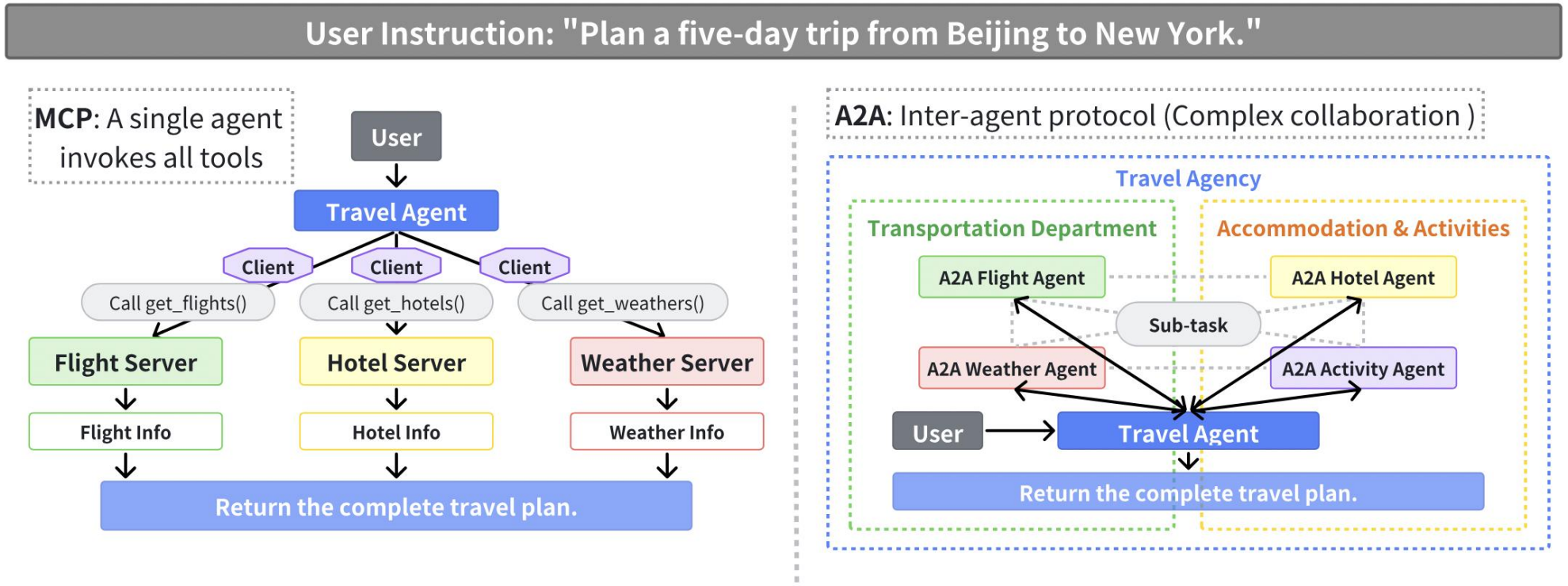


Image Credits: A Survey of AI Agent Protocols, Yang et al., June 2025

Outline

- Model Context Protocol (MCP) — Factual Standard
- Agent2Agent (A2A) Protocol — Landing
- agents.json — Drafting

agents.json

- Enabling AI agents to interact with APIs is not straight forward
  - APIs are designed for developers and not LLMs
  - When a user describes a task at a abstract level, it is on the LLM to stitch together necessary tools
  - For example, when a user wants to *reply to latest email from Bob*, the LLM has to stitch 3 Gmail API endpoints - (1) search for threads, (2) list the emails in a thread, and (3) reply with an email given base64 RFC 822 content

- Current Solution: Use trial-and-error to iteratively refine the system instructions, API description, and parameter documentation until they function correctly.

- `agents.json` is a JSON schema of structured contracts designed for AI agents.
  - introduces flows and links. Flows are contracts with a series of 1 or more API calls that describe an outcome. Links describe how two actions are stitched together

Discuss Example

Summary

Entity	Scenarios	Protocol	Proposer	Application Scenarios	Key Techniques	Development Stage
Context-Oriented	**General-Purpose**	MCP Anthropic (2024)	Anthropic	Connecting agents and resources	RPC, OAuth	Factual Standard
	Domain-Specific	agent.json WildCardAI (2025)	Wildcard AI	Offering website information to agents	/.well-known	Drafting
Inter-Agent	**Genreal-Purpose**	A2A Google (2025)	Google	Inter-agent communication	RPC, OAuth	Landing
		ANP Chang (2024)	ANP Community	Inter-agent communication	JSON-LD, DID	Landing
		AITP NEAR (2025)	NEAR Foundation	Inter-agent communication	Blockchain, HTTP	Drafting
		AComP AI and Data (2025)	IBM	Multi agent system communication	OpenAPI	Drafting
		AConP Cisco (2025)	Langchain	Multi agent system communication	OpenAPI, JSON	Drafting
		Coral citeagentcoralprotocol	The Coral Community	Multi agent system communication	-	Drafting
		Agora Marro et al. (2024)	University of Oxford	Meta protocol between agents	Protocol Document	Concept
	Domain-Specidic	LMOS Eclipse (2025)	Eclipse Foundation	Internet of things and agents	WOT, DID	Landing
		Agent Protocol AIEngineerFoundation (2025)	AI Engineer Foundation	Controller-agent interaction	RESTful API	Landing
		LOKA Ranjan et al. (2025)	CMU	Decentralized agent system	DECP	Concept
		PXP Srinivasan et al. (2024)	BITS Pilani	Human-agent interaction	-	Concept
		CrowdES Bae et al. (2025)	GIST.KR	Robot-agent interaction	-	Concept
		SPPs Gąsieniec et al. (2024)	University of Liverpool	Robot-agent interaction	-	Concept

Credits: A Survey of AI Agent Protocols, Yang et al., June 2025