

Leveraging A2A Protocol to Build Framework-Agnostic Multi-Agent Systems

Less Glue, More Agents • Google ADK, LangGraph, CrewAI, MCP & A2A

ABOUT THE SPEAKER

Sohil Vinod Shah

Staff Software Engineer @ PayPal | Google Developer Expert

in [linkedin.com/in/sohil-vinod-shah](https://www.linkedin.com/in/sohil-vinod-shah)

11+ years of Backend Software Engineering industry experience

Building agentic AI for Site Reliability Engineering at PayPal

Google Developer Expert — AI/ML

Father to a 5-month-old son



MCP Protocol

Agent ↔ Tools & Data

A2A Protocol

Agent ↔ Agent Mesh

Less Glue, More Agents: Two Protocols Matter

The difference is what the caller can assume when the call returns.

MCP (Model Context Protocol)

Connects: An agent to tools and data.

Time-Boxed Action: Structured input → structured output.

3 Outcomes Only: working, completed, or error.

Why It Works: Bounded contract—easy to trace, test, and govern.



 MCP Docs: modelcontextprotocol.io

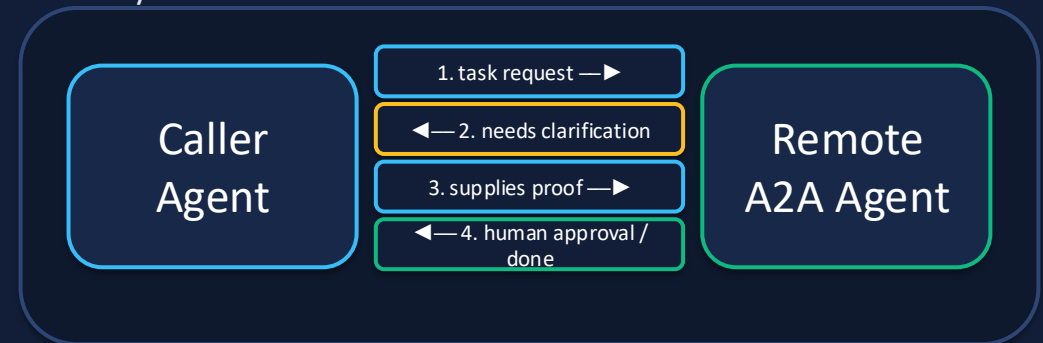
A2A (Agent2Agent Protocol)

Connects: Autonomous agents to each other.

Problem Solver: Asks for missing info, verifies, or pauses for human approval.

Open-Ended: Publishes /.well-known/agent-card.json + JSON-RPC.

Framework-Agnostic: ADK, LangGraph, and CrewAI interoperate natively.



 A2A Spec: google.github.io/A2A

Tool over MCP vs. Agent over A2A

Why modelling another team's agent as an MCP tool discards multi-turn reasoning and approval gates.

Dimension	Tool over MCP	Agent over A2A
Interaction	One request, one response	Multi-turn negotiation
Contract	Bounded, schema-defined	Open-ended (Agent Card)
Completion	Completes or errors	May pause, resume, or wait on human
On Failure	Error code → retry with new input	Asks for what is missing → continues
Use For	Actions, lookups, transformations	Delegation across teams & domains

 Rule of Thumb

- Bounded contract that always completes or errors → MCP as a Tool
- Needs to negotiate, ask questions, wait on a human, or run long → A2A as an Agent

 Google Developers Blog: developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability

Why A2A Instead of a Tool? Two Questions Decide It

Even when a happy-path exchange looks like one question and one answer, production is rarely that tidy.

1. Could this interaction ever need more than one turn?

Happy Path vs. Reality: Single-turn only works when the caller has 100% of the context upfront.

Partial Context: Callers often have half the details—requiring the expert agent to ask follow-up questions.

Human Confirmation: High-stakes actions require mid-flight approval gates.

Tools Break on Ambiguity: A tool fails on incomplete input; an A2A agent stays alive and asks for what is missing.

2. Does the other side belong to another team or domain?

Domain Autonomy: Prompts, internal tools, and domain rules evolve continuously.

Zero-Coupling Deploys: Behind A2A, the owning team ships updates without callers redeploying.

Polyglot Freedom: One team can build in Google ADK, another in LangGraph, another in CrewAI.

Stable Contract: The Agent Card (`/.well-known/agent-card.json`) is the only shared boundary.

💡 If YES to either question → Expose it over A2A, even if today's call looks simple.

How A2A Works: The 4-Step Agent Handshake

MCP carries bounded data lookups. A2A carries cross-domain specialist delegation.

Orchestrator Agent

Caller (e.g., Google ADK)

Domain Specialist Agent

Callee (ADK / LangGraph / CrewAI)

① Discover & Request Agent Card (/well-known/agent-card.json) —▶

◀— ② Agent Card Returned & Cached (Skills, Auth, JSON-RPC URL)

③ Send Task over A2A JSON-RPC (SendMessage + Context) —▶

◀— ④ Specialist Result Returned (When Complete)

⚡ Multi-Turn Continuation: ◀— Needs one more detail | Caller supplies it —▶ | ◀— Final Answer

Solid steps (①–④) run on every call • Highlighted step (⚡) keeps the task open when input is incomplete.

🔗 A2A JSON-RPC & Agent Card Specification: google.github.io/A2A/specification

16-Agent Polyglot Mesh: Google ADK + LangGraph + CrewAI

Dynamic discovery via Agent Registry + parallel A2A JSON-RPC across three agent frameworks.

Root Concierge Orchestrator (Google ADK)

Discovers specialists dynamically via Agent Registry & delegates over A2A.

Agent Registry + Shared Session & Memory Bank

Catalog of Agent Cards + shared state across all 16 containers.

inspiration_agent [Google ADK]

Destination & POI Ideation
(Parallel place + poi A2A calls)

planning_agent [Google ADK]

Flights, Seats, Hotels & Itinerary
(Parallel 5-specialist A2A fan-out)

booking_agent [LangGraph]

Reservations & Payment Checkout
(ReAct Graph over A2A JSON-RPC)

pre_trip_agent [CrewAI]

Visa, Medical, Weather & Packing
(Multi-Agent Crew + Grounding)

in_trip_agent [Google ADK]

Live Flight Monitor & Transit
(Parallel day_of + trip_monitor)

post_trip_agent [Google ADK]

Post-Trip Feedback & Learning
(Writes to Shared Memory Bank)

9 Leaf A2A Specialists: place • poi • flight_search • flight_seat • hotel_search • hotel_room • itinerary • day_of • trip_monitor

 Google ADK: [google.github.io/adk-docs](https://github.com/google/adk-docs)

 LangGraph: [langchain-ai.github.io/langgraph](https://github.com/langchain-ai/langgraph)

 CrewAI: docs.crewai.com

 GITHUB: <https://github.com/shahsohil123/GoogleADK/tree/main/travel-concierge>

Key Takeaways & Further Reading

Thank you! Connect on LinkedIn: [linkedin.com/in/sohil-vinod-shah](https://www.linkedin.com/in/sohil-vinod-shah)

1. MCP for Tools, A2A for Agents

Bounded single-shot actions belong behind MCP. Multi-turn, cross-team problem solvers belong behind A2A.

2. True Framework Interoperability

Google ADK, LangGraph, and CrewAI interoperate seamlessly via /.well-known/agent-card.json and JSON-RPC.

3. Controlled Flow + Parallel A2A

Keep routing deterministic in code, invoke A2A specialists in parallel, and share Session & Memory Bank state.

Specifications, Blogs & Documentation Links

 A2A Protocol Specification & GitHub: <https://google.github.io/A2A/>

 Google Blog — Announcing the Agent2Agent Protocol (A2A): <https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/>

 Google Agent Development Kit (ADK) Documentation: <https://google.github.io/adk-docs/>

 Model Context Protocol (MCP) Official Specification: <https://modelcontextprotocol.io/>

 Vertex AI Agent Engine (Runtime, Sessions & Memory Bank): <https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/overview>