Google A2A — Deep Dive

Generated: 2025-09-11 07:06 UTC

Google A2A (Agent-to-Agent) Protocol — Deep Dive

Generated: 2025-09-11 06:58 UTC

Introduction

Why this matters: Agent-to-Agent protocols (A2A) define how autonomous agents discover, negotiate, and delegate tasks among themselves. In multi-agent systems, A2A enables specialization, scaling, and complex workflows where different agents collaborate to complete user goals. For enterprise MCP setups, A2A ties into orchestration, trust, and interoperability across internal/external agents.

Suggested Interview Answer (short): A2A protocols allow agents to locate peers, negotiate task contracts, and route subtasks to specialized agents — enabling composed, multi-agent workflows with governance.

Core Concepts

Agent Discovery & Registry

Explanation: A central registry or service mesh that tracks available agents, capabilities, and endpoints. Agents register capabilities (e.g., "billing-agent", "compliance-checker").

ASCII Diagram:

```
[Registry] <--- register --- [Agent A]
   ^
   | discover
   v
[Agent B] <--- register --- [Agent C]
```

Negotiation & Contracts

Agents negotiate terms: input/output formats, SLAs, auth tokens, and cost (if applicable). Contracts might be lightweight JSON schemas validated at runtime.

Message Routing & Protocols

Agents communicate via standardized messages (request, propose, accept, complete). Transport can be HTTP/webhooks, message buses, or gRPC streams. Security and authentication are essential.

How to Hook It Up (Integration with MCP & Clients)

Agent Registry (example)

```
# registry stores agent info and capabilities
registry.register({"name":"billing-agent","url":"https://billing.internal","capabilities":["charge","ref
agents = registry.discover(capability="charge")
```

Delegation Flow (pseudo)

```
# Agent A receives user request, decides to delegate
candidate = registry.discover("compliance-check")
resp = http.post(candidate["url"]+"/a2a/task", json={"task":"check_policy","payload":{...}})
if resp.status_code==200:
    # handle result
```

```
        pass
```

Reusing repo scaffolding: You can implement a simple `agent_registry` module in `server/` and add MCP tools/resources that perform discovery and delegation. The MCP client can mediate trusted calls on behalf of agents.

Gaps & Challenges

- **Trust & Security:** How to ensure an agent can be trusted? Use mutual TLS, signed assertions, and short-lived tokens.
- **Interoperability:** Agents from different teams or vendors may use different message schemas — standardization or adapters are needed.
- **Failure modes:** Network partitions, agent crashes, and partial failures require timeouts, retries, and fallback strategies.

Enterprise Considerations

- **Governance:** Audit trails for delegated tasks, policy engines to block unsafe delegations, and approval workflows for sensitive tasks.
- **Observability:** Track task handoffs, end-to-end traces, latencies, and success rates across agents.
- **Operationalization:** Define SLAs for agent response, capacity planning, and scaling policies for agent instances.

Code Examples (A2A patterns)

Simple registry (in-memory)

```
class AgentRegistry:
    def __init__(self):
        self.agents = []
    def register(self, info): self.agents.append(info)
    def discover(self, capability): return [a for a in self.agents if capability in a.get("capabilities"
```

Secure delegation (concept)

```
# mutual TLS or token exchange before delegating
token = auth_service.issue_short_lived_token(agent_id="A", scope="delegate")
resp = requests.post(target_url, json=payload, headers={"Authorization": f"Bearer {token}"}, timeout=5)
```

Interview Angles & Suggested Answers

Q: How would you ensure trust between agents?
A: Use mutual TLS, signed capabilities assertions, short-lived tokens, and a centralized policy engine to vet agents before they register.

Q: How do you handle heterogeneous agents with different schemas?
A: Use adapters/mediators, schema registries, or require agents to publish capability schemas; validate messages at runtime and translate as needed.

Appendix & Further Reading