Self Project 2025-06-03

MCP 2.0

A multi-agent platform for orchestrating tools, memory, context, knowledge, and distributed workflows over gRPC and Protocol Buffers.

No cover image available.

Project Snapshot

DATE
2025-06-03

CATEGORY
Self Project

SHORT DESCRIPTION
A multi-agent platform for orchestrating tools, memory, context, knowledge, and distributed workflows over gRPC and Protocol Buffers.

TAGS / TECH STACK
MCP · AI Infrastructure · Agentic AI · Workflow Orchestration

Project Links

GitHub Repository

Full Project Content

MCP 2.0 is a multi-agent infrastructure platform that connects tools, memory, context, knowledge, and workflow orchestration through typed distributed services.

GitHub Repository: anubhaparashar/MCP2.0

Why This Project Matters

Multi-agent systems become difficult to govern when tool access, shared context, service discovery, authorization, and workflow state are handled through separate ad hoc integrations.

MCP 2.0 explores a structured control plane for those concerns. It uses gRPC and Protocol Buffers for explicit service contracts, then layers discovery, middleware, event exchange, and capability-aware access around the protocol boundary.

System Scope

The repository implements the foundations of a distributed agent platform:

- a registry service for service registration and lookup;
- a context and tool server for structured agent interactions;
- an event-bus service for asynchronous coordination;
- middleware for cross-cutting protocol behavior;
- authorization concepts for limiting agent and service capabilities;
- a client example that exercises the service contracts;
- PostgreSQL initialization for durable platform state; and
- Protocol Buffer definitions that keep requests and responses typed across services.

Architecture

```
Agents and Clients
       |
       v
Capability and policy checks
       |
       v
MCP 2.0 orchestration layer
  |-- Discovery and registry
  |-- Context and tool services
  |-- Event-driven coordination
  |-- Middleware and observability
       |
       v
Knowledge stores, enterprise tools, APIs, and other agents
```

The control plane separates protocol contracts from service implementation. An agent discovers a service, presents the appropriate capability context, and exchanges typed

messages without coupling every participant to custom integration code.

Protocol and Service Design

Protocol Buffers define the shared message schema, while gRPC provides transport and service interfaces. This combination supports explicit request and response types, service discovery, streaming patterns, and language-independent clients.

The platform design groups responsibilities into four layers:

Layer	Responsibility
Protocol	Typed messages and RPC service contracts
Control plane	Discovery, registration, routing, and delegation
Governance	Authentication, capability checks, middleware, and audit boundaries
Execution	Tool calls, context exchange, events, and distributed workflow steps

---

Repository Structure

```
MCP2.0/
|-- auth.py
|-- client_example.py
|-- context_tool_server.py
|-- event_bus_server.py
|-- init_db.sql
|-- middleware.py
|-- protos/
|   `-- mcp2.proto
|-- registry_server.py
`-- requirements.txt
```

The implementation is a platform prototype rather than a claim of production readiness. A production deployment would still require hardened identity, secrets management, policy administration, telemetry, failure recovery, load testing, and isolation for untrusted tool execution.

Engineering Value

MCP 2.0 demonstrates how multi-agent coordination can be treated as distributed-systems infrastructure instead of a collection of loosely connected prompts. Typed

contracts make service boundaries inspectable, while the registry and event layers give agents a consistent way to discover capabilities and coordinate structured work across services.

The project is especially relevant to agent platforms that need reusable tool access, shared context, durable service knowledge, and governed delegation without embedding every workflow inside one monolithic runtime.

Source: src/content/posts/mcp-2-0.md