AAIIMS

Instituto de Computação
UNIVERSIDADE ESTADUAL DE CAMPINAS

Divide, Conquer, and Standardize

A Recursive Architecture for Multi-Agent Systems (MAS)

Ronaldinho V. C. Olivera, Mateus E. R. da Silveira, Alejandro N. Arroyo, Allan M. de Souza, Júlio C. dos Reis

ICLR

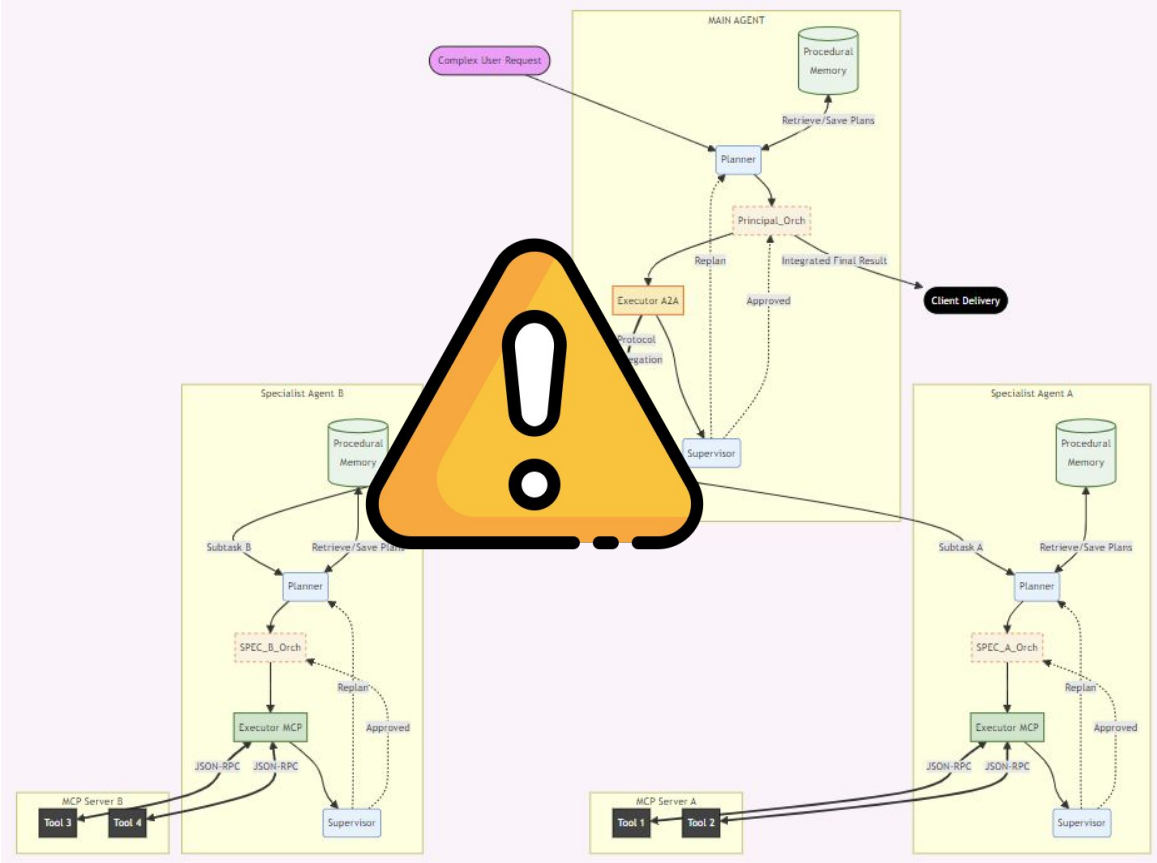
ICLR BLOGPOSTS 2026

The Fragmentation Challenge

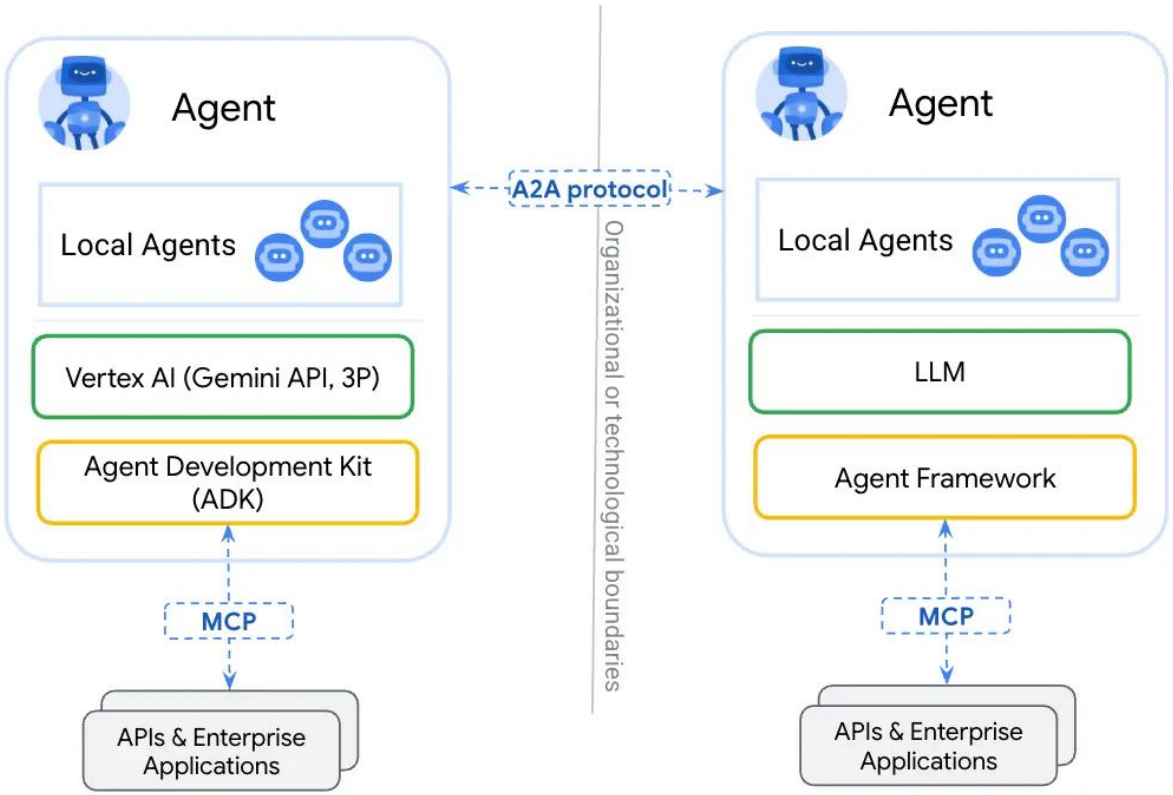
The scalability and robustness of current Multi-Agent Systems (MAS) are severely constrained by the heterogeneity of communication interfaces.

Enterprise systems often consist of agents built upon disparate technology stacks, resulting in siloed behaviors and inefficient collaboration.

This lack of interoperability forces developers to build ad-hoc integrations that are fragile, difficult to scale, and act as rigid communication barriers.



Convergence of Standards



Agent-to-Tool (MCP)

The Model Context Protocol (MCP) provides a universal, secure interface linking AI assistants with databases and external APIs. By abstracting the technical implementation via JSON-RPC, MCP ensures standardization in every environmental interaction.

Agent-to-Agent (A2A)

A2A acts as a social layer protocol enabling heterogeneous agents to discover each other, authenticate identities, and delegate goals. It centers on secure orchestration, allowing each agent to operate autonomously as a specialized black box.

Core Architecture Principles

Divide & Conquer

Resolves difficult problems by recursively decomposing main task requests into a logical execution graph of discrete, manageable subtasks.

Self-Similarity

Applies structural recursion. The global structure of the MAS is entirely isomorphic to the internal structure of a single specialized agent.

Procedural Memory

Empowers the system to evolve over time, transforming past execution traces into robust heuristics without expensive parameter retraining.

Basic FRACTAL Agent Anatomy

Planner Agent Utilizing LLM reasoning, it breaks down tasks into execution graphs, defining parallel and sequential dependencies guided by memory.

Orchestration Mechanism A deterministic execution engine. It strictly administers the control flow, managing queues and ensuring blocking dependencies are respected.

Executor Agent Acts as an intelligent MCP/A2A client, translating abstract subtasks into concrete executions agnostically to the infrastructure.

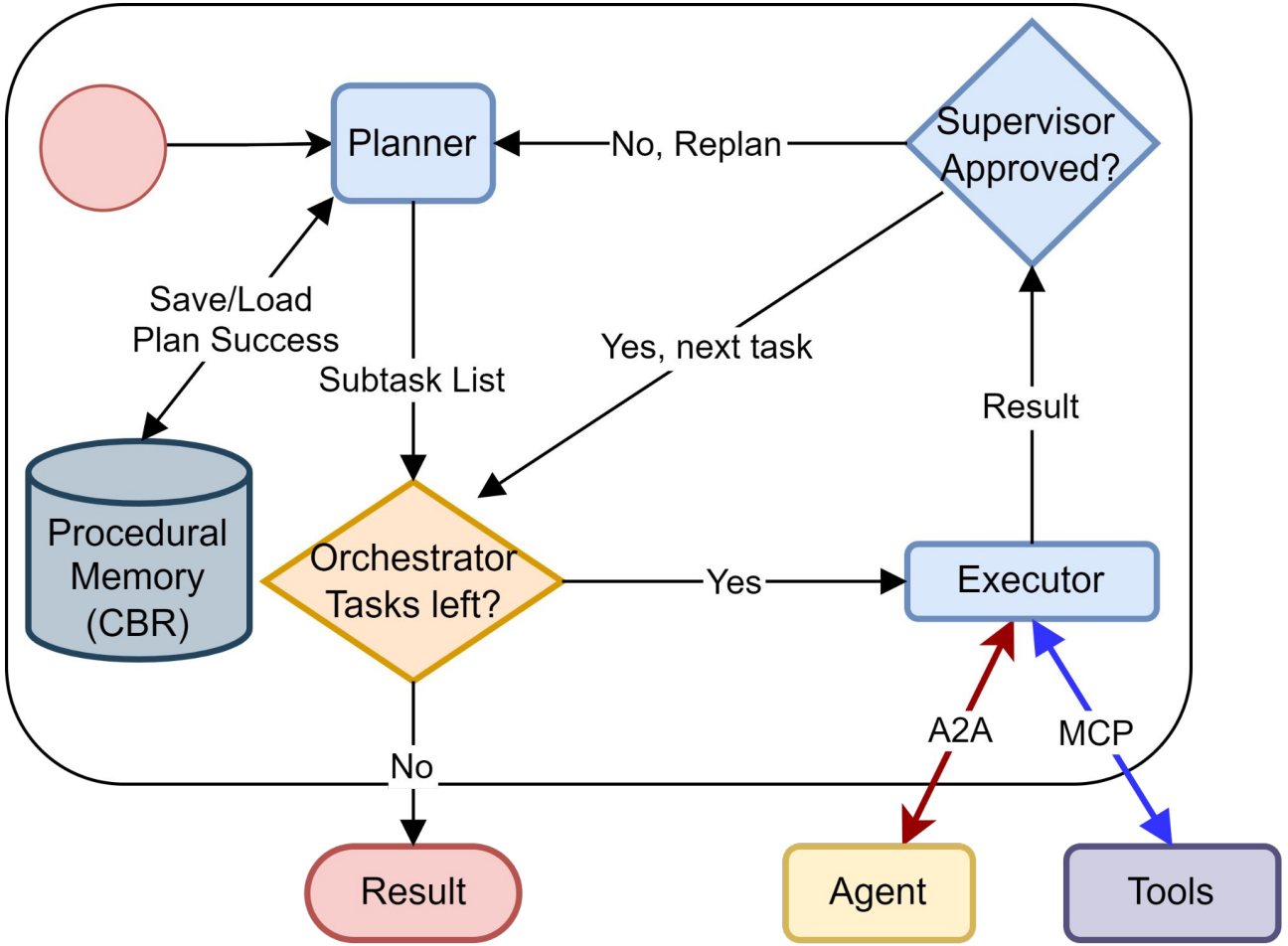
Supervisor Agent The quality control module. It strictly verifies output before task completion, triggering replanning loops to prevent cascading errors.

Recursive Architecture

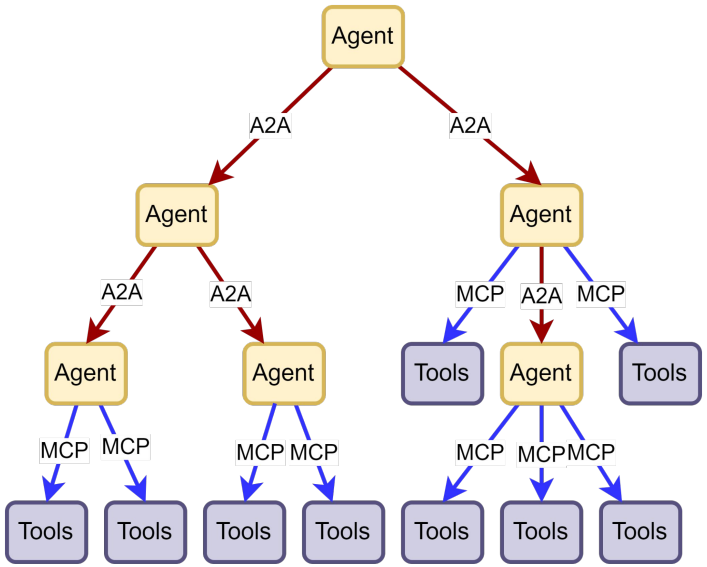
The core innovation of FRACTAL-MAS lies in its recursive scalability. It avoids centralized bottlenecks by applying the exact same topology at the System level.

At the top level, the MAS behaves as a single Fractal Agent. The **Main Agent** utilizes A2A to orchestrate Specialist Agents.

Internally, each **Specialist Agent** decomposes the delegated task and utilizes its own distinct tools via MCP. Internal states and memory remain isolated, preserving operational security.

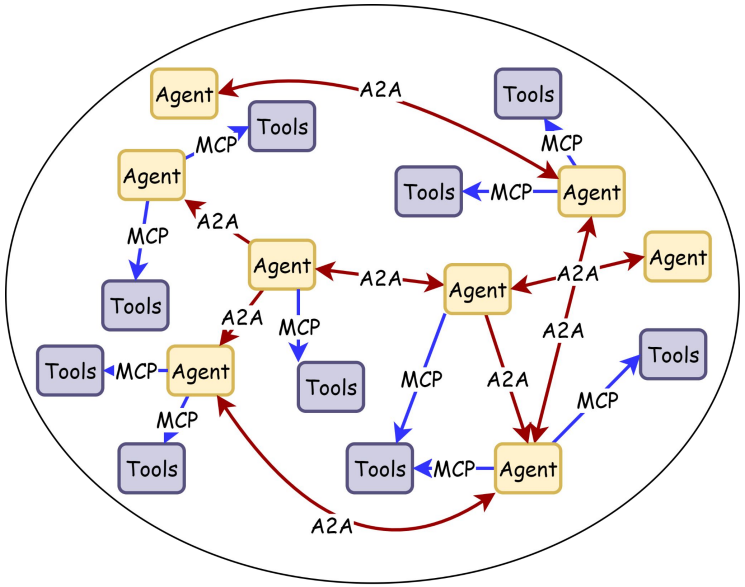


Agent-as-a-Tool Paradigm



Hierarchical Delegation

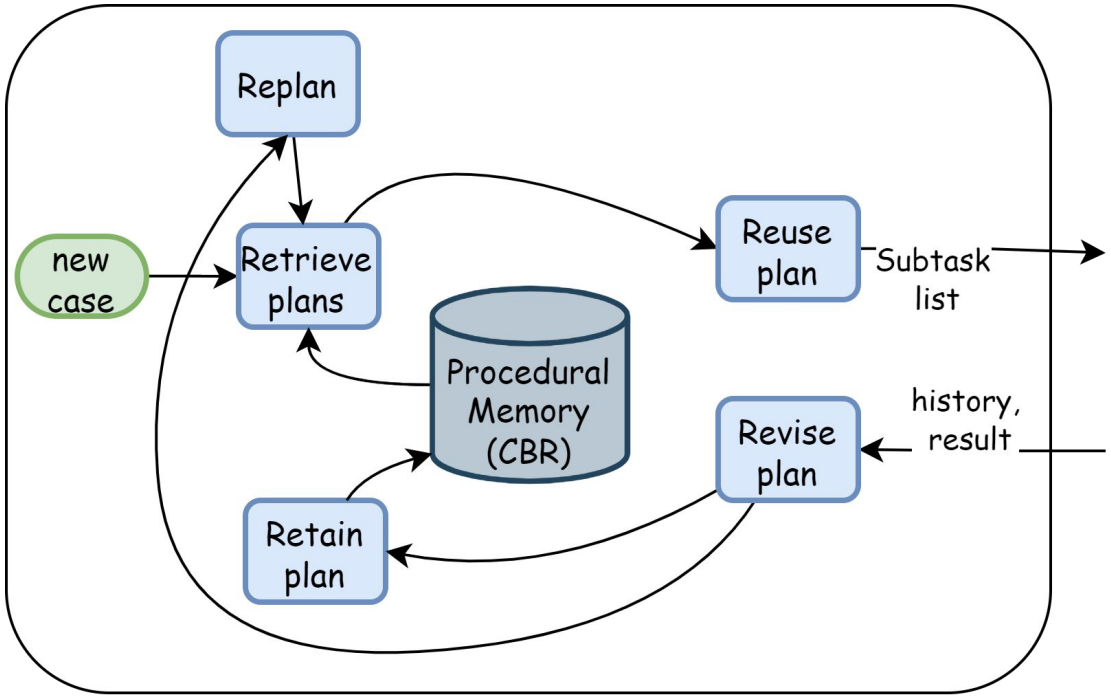
Main agents delegate complex tasks downwards to strict specialists, establishing a clean execution hierarchy.



Decentralized Topology

Agents function simultaneously as clients and servers, morphing rigid hierarchies into robust meshes.

Continuous Learning



Case-Based Reasoning (CBR)

In FRACTAL-MAS, intelligence resides in the systematic accumulation of operational experience via **Procedural Memory**.

Instead of starting from scratch, agents retrieve traces of past episodes (state, plan, reward). This guides the Planner with validated heuristics, drastically reducing hallucinations without the extreme computational overhead of fine-tuning models.

Comparative Analysis

Architectural Feature	Traditional Ad-Hoc MAS	FRACTAL-MAS
Integration Layer	Custom scripts and proprietary frameworks	Unified **A2A** and **MCP** protocols
Scaling Mechanism	Prone to bottlenecks & infinite loops	Isomorphic, recursive self-similarity
Error Handling	Cascading system-wide failures	Localized Supervisor replanning
Adaptability	Requires static updates or fine-tuning	Online learning via **Procedural Memory**