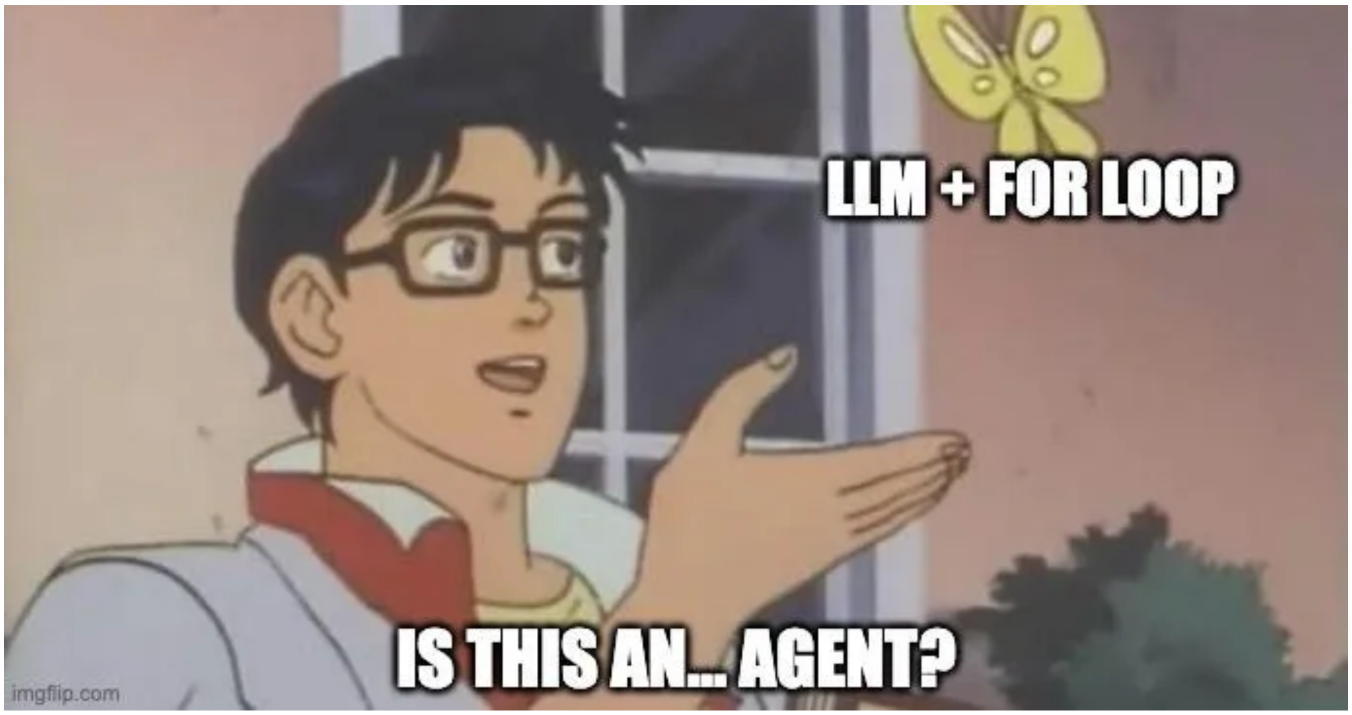
Game of Agents — Episode 1: Let there be Agents

5 min read · Draft

Abhishek Singh



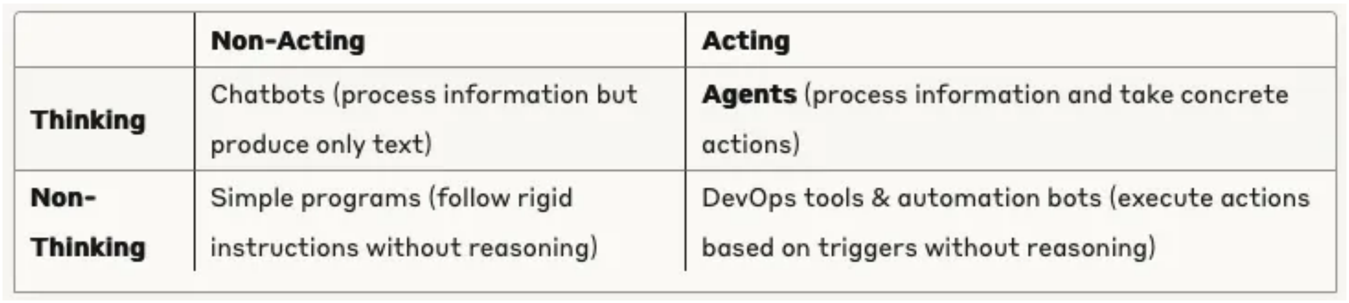
Agency lies in the eye of the beholder. Image Credits — Nathan Lambert, The AI Agent spectrum

Prehistory: Concepts and Capabilities

Long before ChatGPT captivated the masses, the seeds of AI agents were quietly taking root. While traditional programming has always involved functions “taking actions” in response to inputs, these systems lacked the scope of decision-making capabilities.

As large language models became mainstream as chatbots, a small group of researchers and developers recognized something profound: LLMs weren't just text generators — they could serve as connective tissue between disparate systems. This insight catalyzed projects like LangChain, which popularized the concept of chaining LLM outputs to trigger sequences of operations. Systems researchers began referring to these integrated workflows as "Compound Systems," where AI models became embedded components within a larger system.

While both "thinking" and "acting" is not unique in Computer Science, their unique combination of capabilities that agents bring is that they can "think" (process information and reason) and "act" (execute functions or commands that modify their environment).



	Non-Acting	Acting
Thinking	Chatbots (process information but produce only text)	**Agents** (process information and take concrete actions)
Non-Thinking	Simple programs (follow rigid instructions without reasoning)	DevOps tools & automation bots (execute actions based on triggers without reasoning)

History: Standards and Adoption

The emergence of reliable function calling — where LLMs could not only generate text but also invoke specific operations in external systems — marked a crucial turning point. As developers recognized this potential, a wave of SDKs emerged, each attempting to standardize agentic workflows. However, most adopted a "walled garden" approach, keeping their entire technology stack within one library trying to provide end-to-end infrastructure, APIs, all under one package. Notable examples included LangChain, LlamaIndex, Semantic Kernel, AutoGPT, BabyAGI, and numerous Web3 startups.

Anthropic decided to play a different game by introducing the Model Context

Protocol (MCP) — a lightweight specification rather than yet another SDK. The protocol defined how Claude (and other models) would communicate with external tools through a standardized interface, emphasizing simplicity over comprehensive features.

Within six months, the AI development landscape transformed. Tens of thousands of tools, servers, and agents adopted MCP, extending across digital platforms and even beginning to interface with the physical world. This explosive growth stemmed from two critical factors:

1) Perfect Timing: MCP arrived when excitement around agents was peaking, and language models had just reached the reliability threshold necessary for dependable function calling.

2) Consumer Distribution: The Claude desktop application provided immediate access to this ecosystem, allowing users to connect their AI assistant to previously inaccessible tools, from sophisticated 3D modeling software like Blender to everyday services like Spotify and CRMs.

When people ask what innovation MCP actually represents, the answer is counterintuitive: its strength lies in what it deliberately omitted. By providing only the essential communication interface without enterprise-grade security features or complex authentication mechanisms, MCP achieved widespread adoption among developers and enthusiasts. This same minimalism, however, has limited its penetration in enterprise deployment, where robust security and compliance features are non-negotiable requirements.

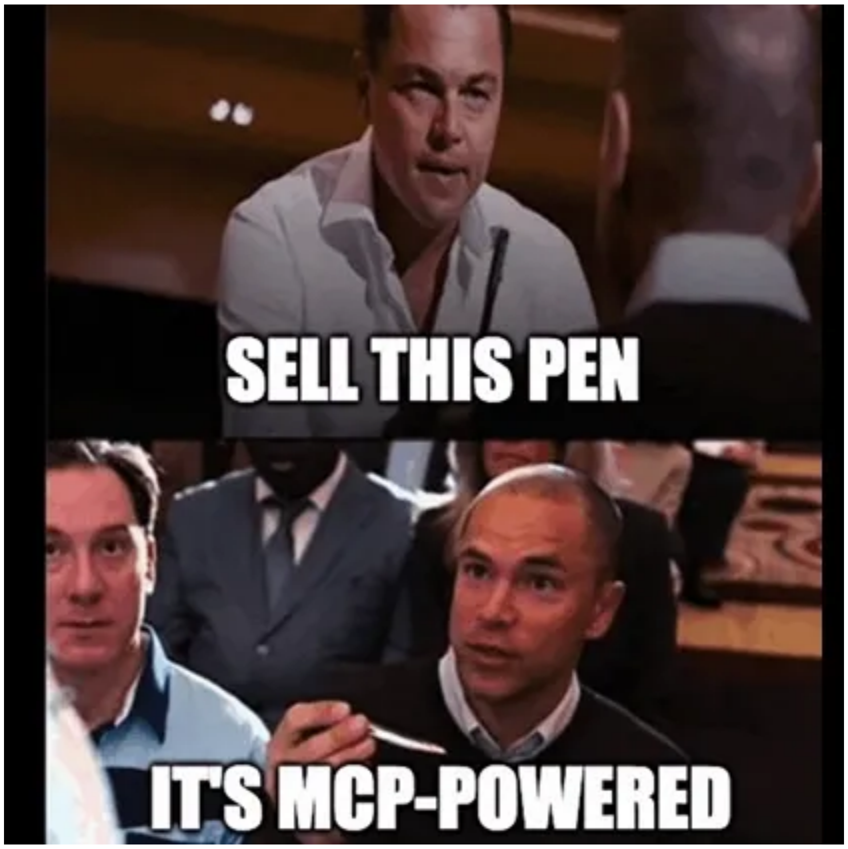


Image Credits - https://dev.to/lovestaco/model-context-protocol-the-secret-sauce-behind-smart-ai-tools-5mc

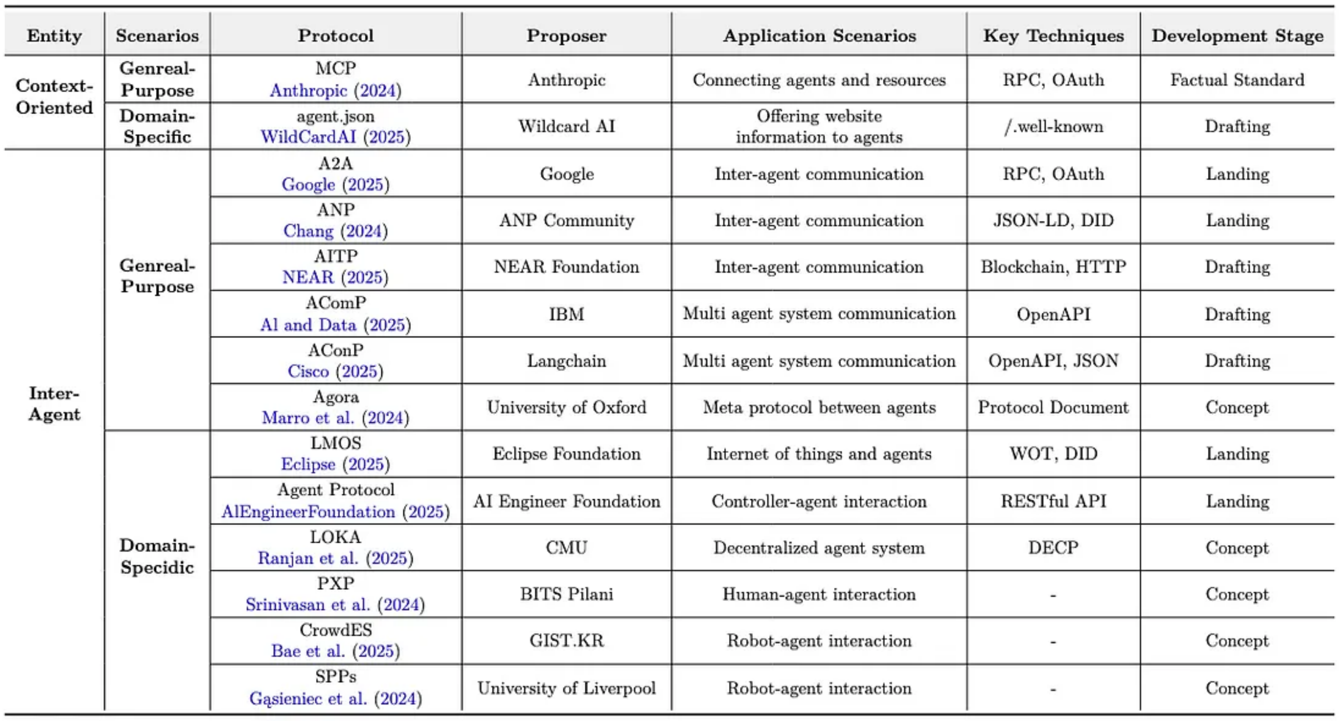
Present: Protocol Wars

This is where Google's A2A comes in. In principle, it is positioned as complementary to MCP and comes with additional enterprise features for reliability, security, and collaboration. Beneath this harmonious public narrative, however, lies a more competitive reality. Despite claims of solving different problems, MCP and A2A share remarkably similar technical foundations — both build on JSON-RPC, utilize HTTP and Server-Sent Events (SSE) as transport layers, and follow client-server architecture patterns. This technical overlap reveals the first hint that they're occupying the same strategic territory.

History teaches us that protocol wars rarely end with peaceful coexistence. From VHS vs. Betamax to Bluetooth vs. previous wireless standards, one implementation typically emerges dominant. The winners control not just a specification document, but the entire direction of technology development, partnership ecosystems, and ultimately, market leverage. This raises a fundamental question about our AI future: will we see an open, interconnected web of agents communicating freely across platforms, or will we instead witness the rise of closed ecosystems where a single entity controls the rules of engagement?

The Protocol Wars aren't just A2A vs. MCP — other players are entering the battlefield with their own approaches. Initiatives like ANP seek to establish more decentralized alternatives, while AGNTCY by Cisco aims for a protocol driven by industrial collaboration. Each claims a unique angle while fundamentally competing for the same prize: becoming the standard bearer for how intelligent agents communicate in the AI economy.

For developers and enterprises, this competition creates difficult decisions about which standard to invest in. Early choices could lead to either a strategic advantage or technical debt as these protocols evolve and market winners emerge.



Entity	Scenarios	Protocol	Proposer	Application Scenarios	Key Techniques	Development Stage
Context-Oriented	Genreal-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
		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

A well curated list of Agent Protocols. Taken from Yang, Yingxuan, et al. "A Survey of AI Agent Protocols."

Future: Consensus and Coordination

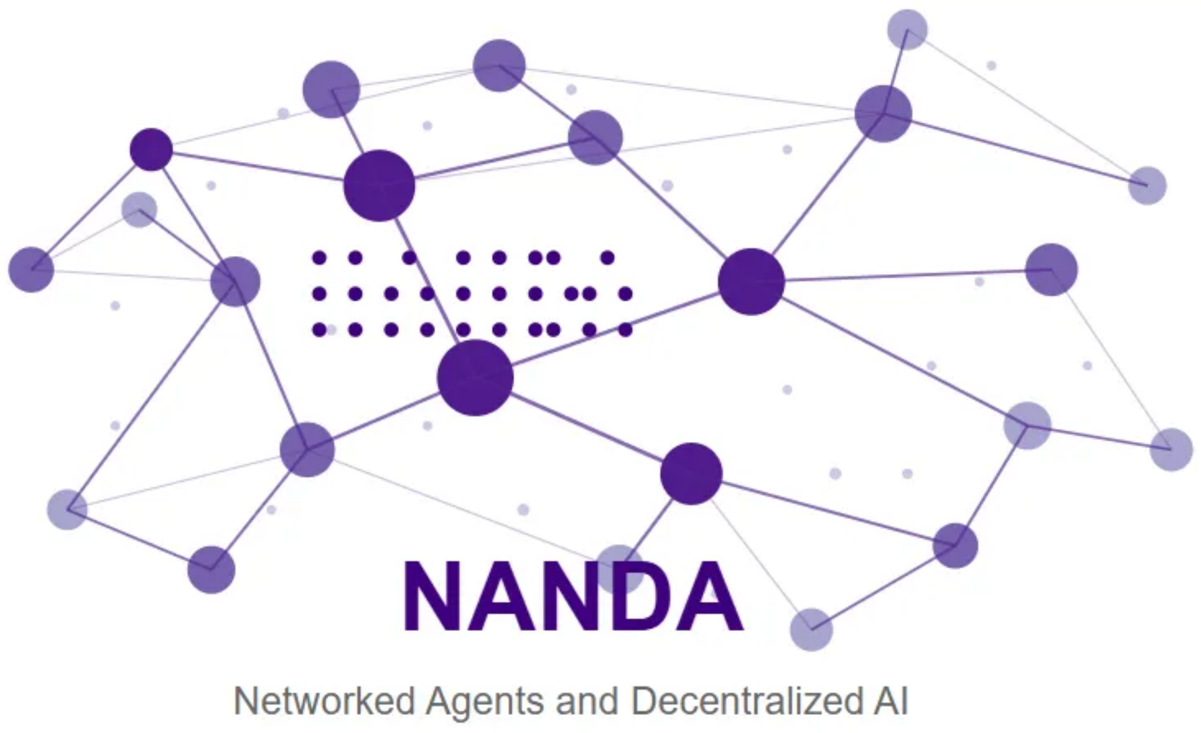


Image Credits — Project NANDA, MIT Media Lab

While agent protocols like MCP and A2A have made significant progress in solving the communication challenge between AI systems, they don't fully address the broader coordination problem. Communication is necessary but insufficient for a true web of agents to emerge. This is where NANDA (Networked Agents and Decentralized AI) from MIT Media Lab enters the picture. Unlike single-purpose protocols, NANDA builds upon existing standards like MCP to create a comprehensive architecture for distributed agent intelligence, functioning more like the World Wide Web than any single protocol.

Prof. Ramesh Raskar and his team at MIT have been developing building blocks of NANDA for over a decade, designing it to provide critical infrastructure for agent discovery, verification, and safe interaction at scale. The project introduces mechanisms for agents to find each other across networks, systems for querying distributed knowledge, and verification protocols for trustworthy interactions — elements missing from current communication-focused standards.

Reaching consensus on agent protocols remains crucial; fortunately, they share enough technical similarities that whatever standard the industry settles on should provide a sufficient foundation. The more challenging frontier lies in building coordination layers that establish trust, developer tools, incentive mechanisms, and user-friendly applications. The most pressing question now becomes: assuming AI agents are proliferating across networks, how do we find them? This is what we will discuss next: building a registry for the agent ecosystem as DNS is to the web.

Edit profile

Written by Abhishek Singh

0 followers · 2 following