Game of Agents — Episode 2: The Great Library of Alexandria 2.0

8 min read · Draft

Abhishek Singh

This is the Episode 2 of the series Game of Agents. For episode 1 visit here.

Imagine a world where accessing AI capabilities is as simple as expressing a desire: “Redesign my kitchen within the budget of $5k” or “Help me reproduce this scientific paper.” In this ideal future, an intelligent system would determine which specialized agents to engage, how they should collaborate, and what tools they need — all without requiring you to know the specifics. This frictionless experience represents the north star for agent discovery.

But how does the system know which agents exist and what they can do? We have two possible approaches. Either we need a comprehensive directory that catalogs available agents and their capabilities, or we need a sophisticated recommendation system that matches requirements to appropriate agents and tools.

However, even the most advanced recommendation system requires foundational knowledge of what exists to recommend from. A recommendation engine cannot suggest an agent it doesn’t know about, just as a librarian cannot recommend a book that isn’t in their catalog. The point is: regardless of how intelligent or intuitive we make the discovery experience, a registry of agents is necessary. One could argue that indexing webpages did not require a webpage registry. However, agents are a different ballgame entirely.

Why the Web of Agents is Different: History Never Repeats, But It Often Rhymes

The ongoing debates around agent registries are reminiscent of countless protocol wars witnessed throughout computing history: DNS vs. WINS, HTML vs. SGML, TCP/IP vs. OSI, JSON vs. XML. Yet the agent ecosystem truly does present novel challenges that make traditional discovery mechanisms insufficient. Let's explore why this isn't just another iteration of the same old story.

The Ephemeral Nature of Agents: Nothing Like We've Seen Before

The web was built on permanence — stable URLs and persistent content that search engines could methodically crawl reliably. Agents, however, are going to be more transient and dynamic. This ephemeral quality will vary based on the kind of agent:

- **On-demand service agents**: These might run continuously but activate their full capabilities only when needed. They're like a restaurant that exists in a physical location but only prepares food when ordered — lurking in the shadows, waiting for the right moment to spring into action.
- **Task-specific ephemeral agents**: Created to solve a single problem, then terminate once complete. Think of them as specialized contractors hired for a particular job — the computational equivalent of mayflies: brilliant, specialized, and gone before you know it.
- **Agent swarms**: Large groups of simple agents that form temporarily to tackle complex problems through collective intelligence, disbanding when the task is complete. Imagine the emergent intelligence of an ant colony, assembling and dissolving with breathtaking speed.

We also have personal agents that are not like traditional servers but can still respond to requests from a trusted group of friends or family agents. None of these fit neatly into the crawler-based paradigm that worked so well for the document-centric web. By the time a crawler finds them, many agents will have already transformed or disappeared entirely.

No Hyperlinks to Follow: The Missing Semantic Glue

The web's brilliance came from hyperlinks — Berners-Lee's elegant solution that

created a navigable graph structure powering everything from PageRank to the semantic web. But in the agent ecosystem, there are no equivalent hyperlinks. No breadcrumbs to follow.

An agent specializing in financial analysis doesn't inherently know about or link to agents that excel at data visualization, even if they would be perfect partners. The agent ecosystem is less like the web's elegant citation network and more like a bustling bazaar where merchants don't know of each other's existence — potential synergies lost in the noise.

We've seen this problem before. Pre-DNS internet users maintained the HOSTS.TXT file — a centralized registry of all computer names on the ARPANET. As the network grew, this approach became unsustainable. DNS emerged as a distributed solution, combining central coordination with distributed authority. Today's agent ecosystem faces a similar inflection point, but with challenges orders of magnitude more complex.

The Same LLM, Different Contexts: Minecraft with Infinite Blocks

Perhaps the most fascinating aspect of the agent ecosystem is that many agents are essentially the same underlying large language models, just operating with different contexts, tools, or information sources. Two agents might have identical underlying models but completely different capabilities based on their prompting strategies and tool access.

The agent ecosystem is less like a library of distinct books and more like a chemistry lab where a handful of elements combine in countless ways to produce different compounds. Traditional categorical taxonomies become hopelessly inadequate. Do we organize by underlying model? By capability? By domain? Each choice creates blind spots.

The Leanest Possible Registry: Lessons from HOSTS.TXT to DNS

When considering registry designs, we're reminded of the transition from HOSTS.TXT to DNS. The original HOSTS.TXT was beautifully simple — a mapping of names to IP addresses maintained at a single location (SRI-NIC). Engineers would email updates to be manually added. It worked perfectly… until it didn't.

At its essence, an agent registry needs just two things: identifiers and endpoints. Like DNS translating domain names to IP addresses, a minimalist registry simply answers the question "how do I reach agent X?" without concerning itself with what that agent does. This bare-bones approach provides the foundation for discovery without the complexity of capability descriptions, versioning, or authentication mechanisms.

But this extreme minimalism creates echoes of past architectural debates. A registry that doesn't describe capabilities pushes the burden of agent selection elsewhere in the system. Without a recommendation service on top of it, this registry defeats much of the purpose of discovery.

Protocol coupling creates another design choice. A registry could remain entirely agnostic to how agents communicate (like DNS is to HTTP), simply providing connection information and letting agents handle the details themselves. Alternatively, it could integrate deeply with protocols like MCP or A2A, facilitating not just discovery but seamless interaction.

History teaches us that the most enduring protocols strike a balance. DNS began simply but evolved to include service records (SRV), text records (TXT), and other extensions. Similarly, the most successful agent registry will likely start with a minimal viable implementation that solves immediate pain points, while architecting for future extensibility.

Who Owns the Library? Governance Models Through the Ages

The question of who controls the agent registry reminds us of countless governance battles throughout tech history. From the Internet Engineering Task Force (IETF) to the World Wide Web Consortium (W3C), from ICANN to the ISO. Each approach leaves distinct fingerprints on the technologies they oversee.

The App Store Model: Digital Feudalism Returns

The App Store Model positions major AI companies as both infrastructure providers and gatekeepers — an approach reminiscent of AOL's walled garden or Microsoft's 1990s platform dominance. Under this approach, companies like OpenAI, Anthropic, or Microsoft would maintain registries for their own agent ecosystems,

vetting submissions for quality and compliance while extracting value through listing fees or revenue shares.

This creates powerful economic engines and consistent user experiences, as seen in mobile app marketplaces that prioritize security and usability. However, these walled gardens inevitably create innovation bottlenecks — agents deemed competitive or controversial face rejection, pricing models become non-negotiable, and smaller developers struggle against platform-favored incumbents. The registry effectively becomes a competitive moat protecting the platform's interests, potentially restricting cross-platform agent functionality and creating vendor lock-in.

While this creates powerful economic engines and consistent user experiences, history has taught us the limitations of digital feudalism. Examples include AOL, MiniTel etc. These closed ecosystems initially flourished but ultimately collapsed under their own weight.

The VISA Model: Consortium Politics Redux

The VISA Model establishes the registry as critical infrastructure operated by a specialized consortium — similar to how VISA processes payments across different banks without being a bank itself. This consortium-operated agent registry would create standardized listings, verification processes, and discovery mechanisms usable across multiple AI platforms.

This model balances competing interests through governance structures where various stakeholders (AI companies, enterprise users, developers, regulatory bodies) collaboratively establish policies. The resulting registry would likely enforce stronger interoperability requirements and more balanced economic relationships than platform-owned alternatives, while still maintaining accountability and quality standards.

This approach mirrors successful industry consortia like the Wi-Fi Alliance or Bluetooth SIG. Historically, consortium approaches have excelled at creating stable, industry-wide standards with balanced economic relationships. However, they typically move at glacial speeds and often favor established players.

The Blockchain Model: Decentralization's Siren Song

The Blockchain Model removes central authorities entirely through distributed technologies — an approach that echoes earlier peer-to-peer systems from Napster to BitTorrent. This approach enables permissionless registration as any agent can list itself with no gatekeeping. While using cryptographic mechanisms for maintaining the integrity of the system.

Registry data lives across many nodes rather than in central databases, making the system resilient to censorship and control. This radically open model maximizes innovation freedom and prevents monopolistic rent-seeking, but creates challenging tradeoffs: quality assurance becomes difficult without central verification, security threats multiply without coordinated responses, and user experience often suffers from complexity.

Most critically, fully decentralized systems struggle with governance — how to update protocols, resolve disputes, or fund maintenance without clear decision-making structures.

Existing Attempts: The Great Registry Race Has Begun

The race to build the definitive agent registry is already underway, with different players taking distinctive approaches reflecting their institutional DNA. Let's examine the current contenders:

NANDA: The Academic-Corporate Consortium

NANDA operates as an academic-corporate consortium bridging institutional boundaries. It adds rigorous OSI layer like multi-tiered approach with strong governance structures. NANDA prioritizes trust and neutrality — functioning like a "Switzerland" of agent registries where institutional credibility outweighs commercial interests.

This approach excels for the vision of the Internet of Agents and mirrors how DNS was initially deployed. Its multi-institutional structure provides legitimacy, and its focus on neutral governance creates a trustworthy foundation for sensitive applications. However, academic projects may move more slowly than market-driven alternatives — a trade-off between thoughtful design and rapid iteration.

ANP: Full Decentralization

ANP embodies radical decentralization as "the HTTP of the agent internet." Using W3C Decentralized Identifiers and semantic web technologies (JSON-LD), ANP enables peer-to-peer agent discovery through both active (.well-known paths) and passive (registration) mechanisms.

This approach maximizes agent autonomy and permissionless innovation without central authorities. While offering the greatest freedom and privacy protection, ANP faces challenges with complexity overhead, quality assurance, and governance. Its technical sophistication enables rich agent relationships but requires more complex implementation.

MCP: Pragmatic Minimalism

MCP has had plans for a registry for a long time, but has not officially announced it yet. Its current registry GitHub repo takes a crowdsourced, minimalist approach with its community-maintained registry.

Originally focused on tool access, MCP's registry provides straightforward discovery without complex verification or semantic frameworks. Its philosophy: let the ecosystem evolve organically. While architecturally less sophisticated than alternatives, MCP benefits from widespread industry adoption.

However, unlike its protocols, the registry has not seen any adoption or usage by clients, including its own Claude Desktop — suggesting a disconnect between protocol success and registry implementation.

A2A: Flexible Enterprise Focus

A2A implements a flexible, multi-tiered discovery architecture using "Agent Cards" — JSON capability descriptors. Rather than mandating a single discovery pattern, A2A advocates for direct configuration, known endpoints, and both centralized and decentralized registries.

This hybrid approach balances enterprise security requirements with ecosystem openness, functioning well in both controlled corporate environments and open marketplaces. A2A emphasizes practical deployment options with strong security fundamentals, though at the cost of some added implementation complexity.

In its current form, the A2A registry has a catch-all framing, but most organizations would seek consensus on a single approach for this critical infrastructure.

Looking Forward: The Next Episode

As with all protocol wars, what happens next will be determined not only by technical merits but by adoption dynamics, business incentives, and path dependencies. The history of technology is littered with superior protocols that were lost to more widely implemented alternatives.

Edit profile

Written by Abhishek Singh

0 followers · 2 following

Recommended from Medium