← Back to Insights

STRAVORIS

MCP vs A2A: The Protocol Stack Architects Need

Executive Summary

A consensus architecture for enterprise AI agent infrastructure is crystallizing around three complementary protocols: **MCP** (Model Context Protocol) for agent-to-tool connections, **A2A** (Agent-to-Agent) for multi-agent orchestration, and **WebMCP** for structured browser-based web access. As of March 2026, this three-layer stack has moved from theoretical proposal to industry-backed standard, governed under the Linux Foundation's Agentic AI Foundation (AAIF) — co-founded in December 2025 by Anthropic, OpenAI, Google, Microsoft, AWS, and Block.[4]

MCP is the most mature layer. Originally released by Anthropic in late 2024, it has crossed 97 million monthly SDK downloads (Python + TypeScript combined) and powers over 10,000 published MCP servers.[1][4] Every major AI platform — Claude, ChatGPT, Gemini, Copilot, Cursor, VS Code — now supports it natively. MCP solves a clearly defined problem: giving an AI model structured access to external data, APIs, and tools via a JSON-RPC 2.0 interface. It is production-ready today.

A2A is earlier in its lifecycle but accelerating. Launched by Google in April 2025, it now has over 100 enterprise supporters and has been adopted into the Linux Foundation alongside MCP.[5] A2A addresses a different problem entirely — how independently built agents discover each other, delegate tasks, and collaborate without sharing internal architectures. Major enterprises including Adobe, SAP, S&P Global, and Microsoft have announced A2A integrations, though production deployments remain limited compared to MCP.[7]

WebMCP is the newest entrant. Previewed in Chrome 146 (February 2026), this Google-Microsoft joint effort brings structured tool exposure to the browser layer, enabling AI agents to interact with websites through semantic APIs rather than DOM scraping. Early benchmarks show a 67% reduction in computational overhead and ~98% task accuracy compared to vision-based approaches.[9]

The critical insight for architects: **these protocols are not competitors — they occupy distinct layers of the same stack.** Organizations that treat them as alternatives risk either over-engineering (building custom orchestration when MCP suffices) or creating vendor lock-in (adopting proprietary agent frameworks that the three-layer open standard will replace). The window for making the right

architectural decisions is Q1–Q2 2026, before fragmented proprietary tooling becomes a migration problem.

However, significant security concerns remain unresolved. Research shows 88% of MCP servers require credentials, yet 53% rely on static API keys rather than modern OAuth delegation.[11] Tool poisoning attacks, prompt injection via tool descriptions, and audit logging gaps present real enterprise risk. The protocol stack is architecturally sound but operationally immature — organizations adopting it need to layer their own security controls on top.

Evidence Base & Methodology

Research Approach

This brief synthesizes findings from 16 sources collected on March 11, 2026. Research comprised 8 web searches across distinct angles (protocol comparisons, governance, adoption statistics, security criticism, case studies, WebMCP specification) and direct retrieval of 4 primary sources including the official MCP specification, Linux Foundation press release, and seed URLs from the idea file.

Source Composition

Source Type	Count	Examples
Official announcements / press releases	4	Linux Foundation, Anthropic, Google Cloud, OpenAI
Technical specifications / documentation	2	modelcontextprotocol.io, W3C WebMCP spec
Developer community / blog analysis	5	dev.to, DZone, InfoQ, DigitalOcean, VentureBeat
Security research / vulnerability reports	4	Astrix Security, Red Hat, Pillar Security, Practical DevSecOps
Analyst / market data	1	Gartner forecast (via secondary sources)

Date Range & Gaps

Evidence spans April 2025 (A2A launch) through March 2026. Notable gaps: (1) limited independent production deployment metrics for A2A — most evidence is vendor-announced partnerships rather than measured outcomes; (2) no peer-reviewed academic evaluation of the three-layer stack as a combined architecture; (3) WebMCP is in early preview with no production deployment data available.

The Three-Layer Protocol Stack

Architecture Overview

The emerging consensus architecture maps to three distinct layers, each solving a fundamentally different problem in agent infrastructure:[1]

Layer	Protocol	Purpose	Analogy	Maturity
Layer 1 — Web Access	WebMCP	Structured browser-to-website interaction for agents	The browser's DOM API	Early Preview (Chrome 146)
Layer 2 — Tool Integration	MCP	Agent-to-tool, API, and data source connections	USB / device drivers	Production (97M+ downloads)
Layer 3 — Agent Orchestration	A2A	Multi-agent discovery, delegation, and collaboration	Service mesh / API gateway	Early Adoption (v0.3)

This layered model mirrors patterns architects already understand from microservices: MCP is analogous to the interface between a service and its dependencies (databases, APIs, file systems), while A2A plays the role of service discovery and inter-service communication (comparable to gRPC + service mesh). WebMCP occupies a niche layer unique to agent architectures — the programmatic web access layer that replaces brittle screen scraping.

MCP: The Agent-to-Tool Layer

MCP uses a JSON-RPC 2.0 wire format with a stateful client-server model inspired by the Language Server Protocol (LSP).[2] The protocol defines three roles: **Hosts** (LLM applications initiating connections), **Clients** (connectors within the host), and **Servers** (services providing capabilities).

Servers expose four capability types:

- **Resources** — read-only data sources (files, database records, API responses)
- **Tools** — executable actions the AI model can invoke
- **Prompts** — reusable templates and workflows
- **Sampling** — reverse-direction LLM completions (server requests the client's model)

Clients can offer servers: **Sampling** (server-initiated LLM interactions), **Roots** (filesystem boundary inquiries), and **Elicitation** (requests for user input).[2]

Three transport mechanisms are supported: **stdio** (local processes), **SSE** (Server-Sent Events for remote connections), and **Streamable HTTP** (production API deployments).[1]

Ecosystem scale: As of March 2026, there are 5,800+ MCP servers in public registries, with built-in support across Claude Desktop, VS Code, Cursor, Windsurf, Zed, JetBrains IDEs, and ChatGPT. Official servers exist for GitHub, Slack, PostgreSQL, Google Drive, Stripe, AWS, Jira, Linear, and Notion.[1]

A2A: The Agent-to-Agent Layer

A2A uses a client-remote architecture over JSON-over-HTTP, deliberately designed to work with agents as opaque entities — unlike MCP, it does not require agents to share their internal architecture or tools.[1] [7]

Core A2A concepts:

- **Agent Cards** — JSON manifests served at `/.well-known/agent.json` describing an agent's capabilities, enabling automatic discovery
- **Tasks** — units of work with a defined state machine: submitted → working → input-required → completed / failed / canceled
- **Artifacts** — typed outputs/deliverables from completed tasks
- **Streaming** — real-time progress updates via Server-Sent Events

A2A's key differentiator from direct API integration:

Aspect	Direct APIs	A2A Protocol
Discovery	Manual configuration	Automatic via Agent Cards
Task tracking	Custom implementation	Built-in state machine
Async handling	Custom webhooks	Standardized push notifications
Agent substitution	Rewrite integrations	Change URL only
Error handling	Vendor-specific	Standardized failure states

In August 2025, IBM's Agent Communication Protocol (ACP) was merged into A2A, consolidating the two competing standards.[1] The protocol is now at version 0.3, which Google describes as bringing "a more stable interface to build against" and as "critical to accelerating enterprise adoption."[7]

WebMCP: The Browser Access Layer

WebMCP is a W3C Community Group standard jointly developed by Google and Microsoft. It enables browsers to expose structured tools to AI agents through the `navigator.modelContext` API.[9]

Two APIs are defined:

- **Declarative API** — handles standard actions defined directly in HTML forms
- **Imperative API** — supports complex, dynamic interactions requiring JavaScript

Despite sharing "MCP" in its name, WebMCP is architecturally distinct from Anthropic's MCP. It does not use JSON-RPC; it operates entirely client-side within the browser rather than as a back-end protocol.[9] The protocol is permission-first: the browser acts as a secure proxy requiring user confirmation before agents can execute sensitive tools.

Performance claims from early benchmarks: 67% reduction in computational overhead versus vision-based (screenshot) approaches, with task accuracy reaching approximately 98%.[9]

Governance and Industry Alignment

The Agentic AI Foundation (AAIF)

The Linux Foundation announced AAIF on December 9, 2025, anchored by three contributing projects: Anthropic's MCP, Block's goose (an open-source AI agent framework), and OpenAI's AGENTS.md (a markdown standard for AI coding agent guidance).[4]

Membership as of March 2026:

Tier	Count	Notable Members
Platinum	8	AWS, Anthropic, Block, Bloomberg, Cloudflare, Google, Microsoft, OpenAI
Gold	18	Cisco, Datadog, Docker, IBM, JetBrains, Okta, Oracle, Salesforce, SAP, Shopify, Snowflake, Twilio
Silver	22+	Elastic, Hugging Face, Uber, Zapier, Pydantic, Solo.io

The breadth of this membership — spanning cloud providers, enterprise software vendors, developer tooling companies, and security firms — is significant. It suggests the three-layer protocol stack has sufficient industry backing to become the de facto standard, reducing the risk of a standards fragmentation scenario.

Convergence Timeline

Date	Event	Significance
Late 2024	Anthropic releases MCP	First open protocol for agent-tool integration
April 2025	Google launches A2A	Agent-to-agent orchestration standard with 50+ launch partners
August 2025	IBM ACP merges into A2A	Consolidation eliminates competing standard
December 2025	AAIF formed under Linux Foundation	MCP, A2A, goose, AGENTS.md under neutral governance
February 2026	WebMCP previews in Chrome 146	Third layer of the stack emerges
February 2026	MCP hits 97M monthly SDK downloads	Confirms production-scale adoption
April 2026	MCP Dev Summit (scheduled)	First major community event, New York City

Security Landscape

MCP Security Concerns

Multiple security research reports published in 2025–2026 identify serious concerns with the MCP ecosystem, particularly around credential management and attack surface expansion.[11][12][13]

Credential management is the most urgent problem. An Astrix Security audit found that 88% of MCP servers require credentials, but 53% rely on static API keys or personal access tokens (PATs) that are long-lived and rarely rotated. Only 8.5% use OAuth.[11]

Known attack vectors:

- **Tool Poisoning** — Attackers embed hidden, malicious instructions in tool descriptions that MCP servers expose to AI models. Since tool descriptions go directly into the model's context, the AI may follow covert instructions (e.g., exfiltrating private data to an external server).[12]
- **Prompt Injection** — MCP expands the attack surface for prompt injection by giving AI models direct access to enterprise tools and data sources, enabling attackers to trigger automated actions beyond text generation.[13]
- **Session ID Exposure** — Earlier MCP implementations placed session IDs in URL query strings, leaking sensitive information through browser history, logs, and proxies.[12]
- **CVE-2025-6514** (CVSS 9.6) — A critical vulnerability in proxy configurations affecting MCP servers.[12]

Audit and compliance gaps: The MCP ecosystem lacks standardized audit logging, making forensic analysis of the chain from user query to tool action nearly impossible in many deployments.[12]

A2A and WebMCP Security Posture

A2A inherits standard HTTP security practices (TLS, OAuth, API keys) and treats agents as opaque services — which limits the blast radius of compromised agents but also means the protocol itself cannot inspect or validate agent behavior. WebMCP takes a permission-first approach with the browser acting as a security proxy, which is architecturally stronger but untested at scale.

Security Maturity Assessment

Protocol	Auth Model	Known CVEs	Audit Logging	Overall Risk
MCP	Mixed (53% static keys, 8.5% OAuth)	Yes (CVE-2025-6514, CVE-2025-68145)	No standard	High — requires layered controls
A2A	Standard HTTP (TLS + OAuth/API keys)	None published	Not specified	Medium — inherits HTTP maturity
WebMCP	Permission-first browser proxy	None (too new)	Browser-managed	Low (theoretical) — unproven at scale

Market Adoption and Production Evidence

MCP Adoption Metrics

MCP's adoption curve is the strongest signal in this research. Key data points:

- 97M+ monthly SDK downloads across Python and TypeScript[1]
- 10,000+ published MCP servers[4]
- 5,800+ servers in public registries[1] (the discrepancy with the 10,000 figure likely reflects the difference between public registries and total published servers including private/internal ones)
- Native support in every major AI platform: Claude, ChatGPT, Gemini, Copilot, Cursor, VS Code, JetBrains
- 1.1M+ public GitHub repositories importing an LLM SDK (+178% YoY)[4]

A2A Adoption Status

A2A adoption is earlier-stage and primarily measured through partnership announcements rather than usage metrics:

- 100+ enterprise supporters by February 2026[5]
- Key integrations: Adobe (distributed agent interoperability), SAP (Joule AI assistant), S&P Global (inter-agent data services), Microsoft (Azure AI Foundry + Copilot Studio)[7]
- System integrators Accenture, Deloitte, and PwC building A2A into enterprise AI practices[7]
- Comparus (IBM watsonx.ai) reported streamlined AI operations and faster task completion post-A2A integration[14]

Broader Market Context

Gartner forecasts that 40% of enterprise applications will feature task-specific AI agents by 2027, up from less than 5% in 2025.[5] A 2025 global AI survey found 29% of enterprises already running agentic AI in production, with 44% planning to join within a year.[7] These figures suggest the protocol stack is arriving at the right time — enterprise demand for agent infrastructure is real, not speculative.

Key Assumptions & Uncertainties

What the Evidence Does Not Resolve

1. **A2A production performance at scale.** While multiple enterprises have announced A2A integrations, there are no published benchmarks for latency, throughput, or reliability in production multi-agent systems. The protocol is at v0.3 — breaking changes remain possible.
2. **WebMCP adoption trajectory.** WebMCP is in early preview with zero production deployments. Whether website operators will implement WebMCP endpoints (adding work for them) at sufficient scale to make it useful for agents is an open question. The 67% overhead reduction and 98% accuracy figures come from controlled benchmarks, not production environments.
3. **Security convergence timeline.** The gap between MCP's adoption curve and its security maturity is the most significant risk factor. Whether AAIF governance will accelerate security standardization (mandatory OAuth, audit logging specs, tool validation) or whether the ecosystem will remain fragmented on security is unclear.
4. **Proprietary framework displacement.** Frameworks like LangChain, crewAI, and AutoGen have established user bases for multi-agent orchestration. Whether A2A will subsume these or coexist alongside them is not settled by the current evidence.
5. **Three-layer completeness.** The three-layer model (WebMCP / MCP / A2A) may not account for all necessary infrastructure. Authentication/authorization, observability, and rate limiting across agents may require additional standardization not currently addressed by any of the three protocols.

Where Expert Opinion Diverges

There is broad consensus that MCP is the right approach for agent-to-tool integration. There is less consensus on whether A2A's Agent Card discovery model will work in enterprise environments where agents are internal, not public — the `/.well-known/agent.json` convention assumes web-style discoverability that may not map to private networks. Some practitioners argue that existing service mesh tooling (Istio, Linkerd) could be adapted for agent orchestration more quickly than A2A can mature.

Strategic Implications

1. **Start with MCP now — it is the lowest-risk, highest-value layer.** MCP is production-ready with massive ecosystem support. Any organization building agent capabilities should standardize on MCP for tool integration immediately. The 10,000+ existing servers mean most common integrations (databases, APIs, SaaS tools) are already built. Building proprietary agent-to-tool connectors at this point creates unnecessary technical debt.

2. **Design for A2A, but defer production commitment to v1.0.** A2A's architecture (Agent Cards, task state machines, artifact handoff) is well-designed, but v0.3 is not production-stable. Architects should structure their agent systems to be A2A-compatible — independent, capability-described agents with clean task boundaries — without hard-coupling to the current protocol version. This maps directly to microservices best practices.

3. **Treat MCP server security as a day-one concern, not a backlog item.** The 53% static-key credential figure is alarming for enterprise deployments. Organizations should mandate OAuth for all MCP server connections, implement tool description validation (to mitigate poisoning attacks), and build audit logging around MCP interactions before going to production. Do not assume the protocol's security will improve — layer your own controls.

4. **Monitor WebMCP but do not build around it yet.** WebMCP's performance claims are promising, but it is pre-production and dependent on website operators implementing endpoints. For agent web access today, existing approaches (headless browsers, API integrations) remain more reliable. Revisit after Google I/O 2026 for clearer signals on adoption.

5. **Use AAIF governance as a vendor selection filter.** The breadth of AAIF membership (48+ organizations across all tiers) provides a reliable signal of which vendors are committed to open standards. When evaluating agent platforms, frameworks, or infrastructure vendors, check their AAIF membership or MCP/A2A compatibility as a proxy for standards alignment. Proprietary agent frameworks that ignore these standards will become integration liabilities.

6. **Map the three-layer stack to your existing architecture patterns.** For teams with microservices experience: MCP ≈ service clients/SDKs, A2A ≈ service mesh + discovery, WebMCP ≈ BFF (backend-for-frontend) pattern. This mental model reduces the learning curve and helps teams apply existing operational expertise (monitoring, circuit breaking, retry policies) to agent infrastructure.

References

1. "MCP vs A2A: The Complete Guide to AI Agent Protocols in 2026," *DEV Community (pockit_tools)*, 2026. dev.to. Accessed March 11, 2026.
2. "Specification — Model Context Protocol (2025-11-25)," *modelcontextprotocol.io*. modelcontextprotocol.io. Accessed March 11, 2026.
3. "MCP vs A2A: Protocols for Multi-Agent Collaboration 2026," *OneReach.ai*, 2026. onereach.ai. Accessed March 11, 2026.
4. "Linux Foundation Announces the Formation of the Agentic AI Foundation (AAIF)," *Linux Foundation*, December 2025. linuxfoundation.org. Accessed March 11, 2026.
5. "Announcing the Agent2Agent Protocol (A2A)," *Google Developers Blog*, April 2025. developers.googleblog.com. Accessed March 11, 2026.
6. "MCP joins the Agentic AI Foundation," *Model Context Protocol Blog*, December 2025. blog.modelcontextprotocol.io. Accessed March 11, 2026.
7. "Agent2Agent Protocol (A2A) Is Getting an Upgrade," *Google Cloud Blog*, 2025. cloud.google.com. Accessed March 11, 2026.
8. "MCP joins the Linux Foundation: What this means for developers," *GitHub Blog*, December 2025. github.blog. Accessed March 11, 2026.
9. "Chrome WebMCP: The Complete 2026 Guide to AI Agent Protocol," *DEV Community*, 2026. dev.to. Accessed March 11, 2026.
10. "Google AI Introduces the WebMCP," *MarkTechPost*, February 2026. marktechpost.com. Accessed March 11, 2026.
11. "State of MCP Server Security 2025: Research Report," *Astrix Security*, 2025. astrix.security. Accessed March 11, 2026.
12. "The Security Risks of Model Context Protocol (MCP)," *Pillar Security*, 2025. pillar.security. Accessed March 11, 2026.
13. "Plug, Play, and Prey: The Security Risks of the Model Context Protocol," *Microsoft Defender Cloud Blog*, 2025. techcommunity.microsoft.com. Accessed March 11, 2026.
14. "MCP vs A2A: Practical Enterprise Data Integration," *DZone*, 2026. dzone.com. Accessed March 11, 2026.
15. "Donating the Model Context Protocol and Establishing the Agentic AI Foundation," *Anthropic*, December 2025. anthropic.com. Accessed March 11, 2026.
16. "OpenAI Co-founds the Agentic AI Foundation under the Linux Foundation," *OpenAI*, December 2025. openai.com. Accessed March 11, 2026.

STRAVORIS

INNOVATE. INTEGRATE. ELEVATE.