AGENTIC NETWORKING · A MAYA FOUNDER WHITEPAPER

maya.

Identity, Visibility, and Control for the Agent Era

Rebuilding the network's core abstractions around the agent, not the IP.

By **Daljeet Singh**, Founder · Mayagentic Inc. · Running on AWS today. Private beta.

The network went blind exactly when it mattered most

For thirty years, the network identified things by where they were. An IP address, a port, a 5-tuple. That was a fine proxy for who and what, back when one machine ran one workload and the map of the network was the map of the org.

That assumption is now broken in the most consequential way it has ever been broken.

Agents, autonomous software that reasons, calls tools, and talks to other agents, are arriving in fleets. They are numerous, ephemeral, and chatty. Dozens of them share a single host. They spin up and die in seconds. They share IP addresses. They talk to each other (agent-to-agent, "A2A") and to tools and services over protocols like MCP, almost always over TLS the network was never designed to see into.

So at the precise moment the things on your network became autonomous and numerous, the network lost the ability to see them. It sees IPs and ports, the one identity dimension agents have made meaningless. It cannot tell you which agent is talking, to whom, doing what, or whether it should be allowed to.

This is not a monitoring gap. It is a structural blindness. And every security, compliance, and operational control you have built on top of the network inherits it.

Figure 1. On the pre-agent host, one IP maps cleanly to one workload, so a 5-tuple is a fair proxy for identity. On the agent-era host, many ephemeral agents share a single address, and the same field can no longer say which agent did anything.

The reframe: agent identity is the handle

For thirty years every network primitive keyed on the same thing: where a packet came from and where it was going. Load balancing keyed on the connection. Quality of service keyed on port. Firewalls keyed on the 5-tuple. Discovery keyed on the DNS name. All of them answered "which box, which port, which flow," because when one host ran one workload, *where* was a fair proxy for *who*.

Agents break that proxy. The unit that matters is no longer the host or the flow. It is the agent: a thing with a capability, a cost, a level of trust, a policy about what it may touch, often carrying live state mid-task. Where it runs is an accident of scheduling. So every primitive that keys on *where* is now keyed on the wrong thing.

Agent identity is the handle the network should have keyed on, and until now it did not exist in a form the network could use. Maya derives it on the wire, per agent, without changing the agent. Once you hold that handle, re-keying the rest of the stack onto it is the work.

We did the first primitive already: permission. Detect when a declared agent reaches something it was never authorized to reach, and enforce it from the wire. That is the agentic firewall, and it is the proof the handle carries. We took a per-agent identity, keyed a real enforcement decision on it, and it holds on live traffic. **The firewall is not the product. It is the evidence.** Everything that follows is the same move applied to the next primitive.

Agentic Networking is our name for that program: rebuilding the network's core abstractions, identity, permission, forwarding, discovery, and provenance, around the agent, not the IP. This paper is about why that has to happen, and how we are doing it.

Why now: the attack surface arrives before the controls

Every shift in computing creates an attack surface before it creates the controls to manage it. Agents are no exception, and the surface is unusually exposed because agents *act*. Consider what an agent fleet actually exposes:

- **Lateral movement, agent-to-agent.** A compromised or hijacked agent does not sit still. It talks to the other agents it has access to. East-west A2A traffic is the new lateral-movement plane, and it is almost entirely unobserved today.
- **Identity confusion and impersonation.** When many agents share a host and the network identifies by IP, two agents are indistinguishable on the wire. One can be mistaken for another, or impersonate another, and your controls cannot tell the difference.
- **Exfiltration and rogue behavior.** An agent given a tool and a goal can reach outward in ways no one explicitly authorized. Without per-agent attribution, "which agent made this call" has no answer.

The uncomfortable part: your current stack is blind to all of it, because it identifies by the wrong thing. A firewall that reasons in IPs and ports cannot enforce "agent A may talk to agent B but not agent C." It does not know A, B, or C exist.

Why the incumbents cannot simply catch up

It is tempting to assume the existing network-security vendors will add an "agent feature" and the problem goes away. They will not, and the reason is architectural, not a matter of effort. Traditional firewalls and the network-security stack are pre-agent by construction:

- **They identify by IP, port, and 5-tuple**, the exact dimension agents have invalidated. Bolting agent-awareness onto a 5-tuple engine is rebuilding the engine.
- **They were not born in the cloud.** Most are network appliances in VM clothing, a hardware mindset wrapped in a virtual machine, not a cloud-native control plane. They scale, deploy, and price like appliances.
- **They are closed.** Proprietary by design, optimized for lock-in rather than for the open, standards-based, multi-cloud world agents actually live in.
- **Their operational model is heavy.** Instrument everything, terminate TLS, manage keys, configure per workload. That model does not survive contact with fleets of ephemeral agents.

The same limit applies to the cloud-native identity tools, not just legacy firewalls. Cilium and the service meshes key on identity the orchestrator hands them: a hash of pod labels, assigned at pod creation, scoped to the cluster. That is not agent identity and it is not derived from the wire. One pod running three agents is

one identity to them and three to Maya. They see the pod the platform told them about. They do not see the agent. Closing that gap is not a feature they ship next quarter. It is a different place to stand.

We are not trying to build a better firewall. We are operating at a layer the firewall never reached: agent identity. That is not a feature you add to a packet-and-port engine. It is a different plane.

What Maya is: the network stack whose unit is the agent

Maya rebuilds the network's foundational services around the agent. The cleanest way to understand it is by analogy to the network primitives you already know. Maya is, in effect, a new network stack whose unit is the agent rather than the host.

Classic primitive	Maya's agent-era equivalent	What it does	Status
IP address (identity, addressing)	`AID` — Agent Identity	A per-agent identity attributed on the wire. The stable unit of visibility and enforcement.	`Shipping`
DHCP (onboarding)	`Stitch` — host-side onboarding	Brings agents onto the fabric as they appear on a host, with no code change to the agent.	`Shipping`
Firewall, ACLs (permission)	Declared authorization, enforced on deviation	Each agent's authorized reach, declared up front and enforced from the wire when the agent deviates.	`Shipping`
DNS (discovery)	MCP-aware reachability	Governs which tools and services a declared agent may reach, at the identity layer.	`Direction`
Load balancing (distribution)	`AILB` — Agentic Load Balancing	Distributing work across agents by agentic properties, not round-robin.	`Direction`
Flow logs, audit (provenance)	Agent provenance chain	Attests the chain of agent identities a request traversed, across hops.	`Direction`
Policy config (control)	Declared intent compiler	Say what you want in plain terms; the fabric compiles it to enforceable agent policy.	`Direction`

The pieces that deliver this today:

- **Warp**, the enforcement data plane. An eBPF-based appliance that sees, attributes, and governs agent traffic in the kernel data path.
- **Weaver**, the control and classification plane. Builds the live picture of which agent is which and which is talking to which.
- **Loom**, a single pane of management and enforcement for every agent you bring onto the fabric.

- **Stitch**, a lightweight host-side component that brings agents onto the fabric. It touches the host to attribute and steer. It does not carry your payload.

Maya is the fabric for agent-to-agent communication. One identity model, one place to see it all, one place to enforce.

How it works, and the principles that make it different

Three design choices separate Maya from "a firewall that learned the word agent."

1. Identity is attributed on the wire. No SDK, no sidecar, no code change.

Maya attributes each agent's traffic from signals already present on the host and in the flow. You declare an agent's identity with a single environment variable. Nothing is injected into the agent. An agent appears on a host and Maya attributes its traffic automatically. This is the foundation of zero-instrumentation: you plug in the fabric and the agents light up, identified and observable, without anyone wiring them up one by one. Attribution computed at the edge also means no central registry has to agree, in real time, on who an agent is, which is what lets the model scale.

2. No decryption, no termination. Governance at the identity layer.

Maya never terminates TLS, never decrypts your traffic, and never holds your keys. It governs which agent is talking to which agent, tool, or service, across MCP, A2A, and any other agent protocol. It reasons about the envelope: identity, destination, volume, timing. It does not open the letter.

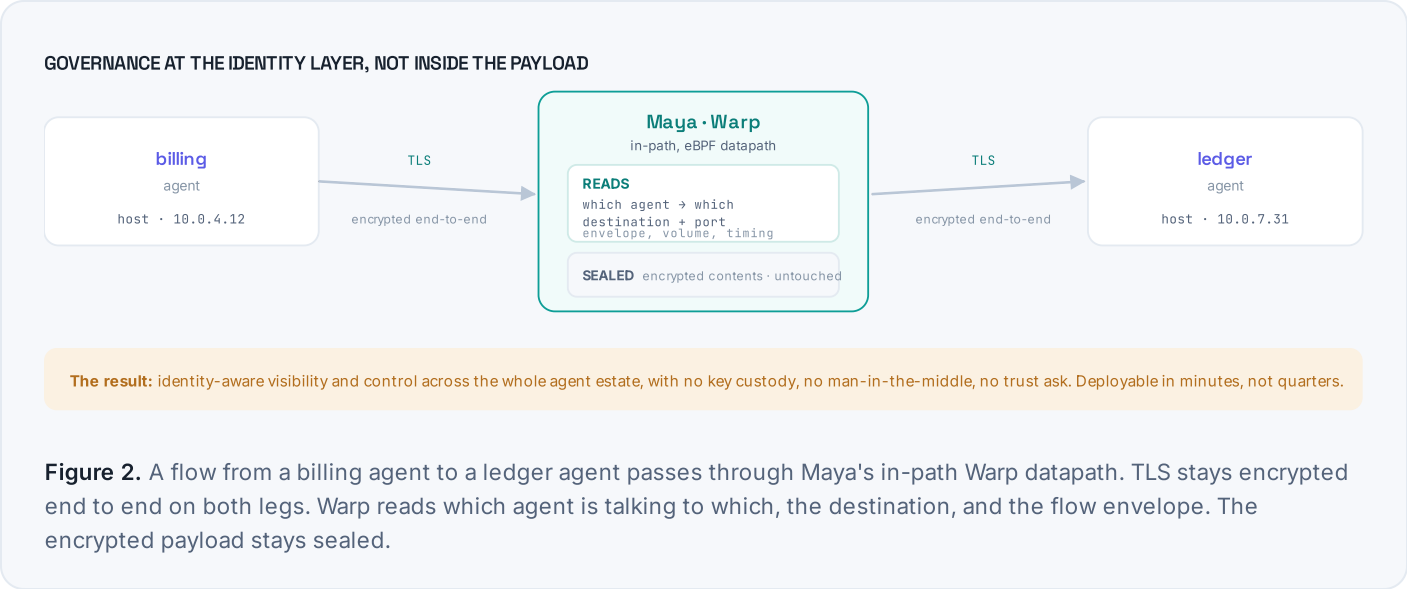

Figure 2. A flow from a billing agent to a ledger agent passes through Maya's in-path Warp datapath. TLS stays encrypted end to end on both legs. Warp reads which agent is talking to which, the destination, and the flow envelope. The encrypted payload stays sealed.

This is deliberate, and it is a feature, not a limitation. Decryption means key custody, a man-in-the-middle posture, and a heavy trust ask of every customer. Maya asks for none of that. You get identity-aware visibility and control over your entire agent estate without handing anyone the keys to read it. That is what makes it deployable in minutes instead of quarters.

3. It complements the identity standards you already have.

Maya is designed to complement SPIFFE, not compete with it. SPIFFE issues and attests workload identity at the application layer. It says, cryptographically, who a workload is. Maya is the network-enforcement plane for that identity. It sees identity expressed on the wire and governs what it may reach. SPIFFE issues; Maya enforces. Together they close the loop from "who you are" to "what you may do on the network." The wire-level enforcement is a standalone plane, and it works with or without SPIFFE present.

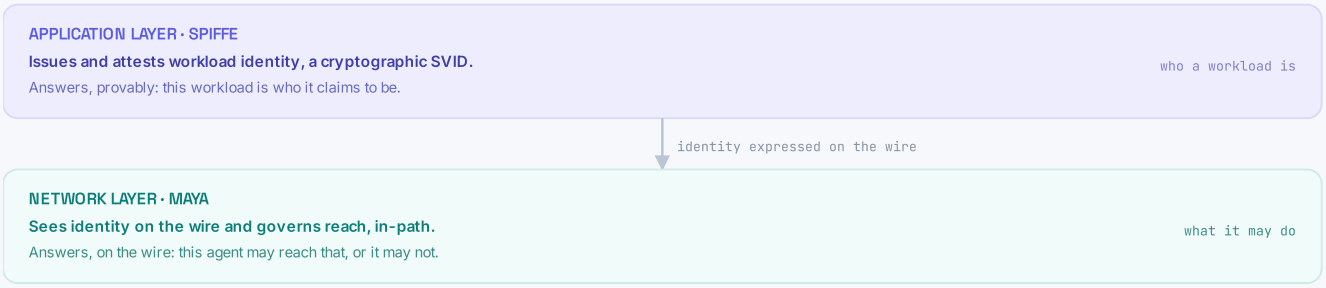

Figure 3. SPIFFE issues and attests a cryptographic workload identity at the application layer. That identity, expressed on the wire, hands off to the network layer, where Maya sees it and governs what it may reach. Together they run from who you are to what you may do on the network.

4. It enriches the observability you already bought.

Maya emits standard OpenTelemetry (OTLP) out of the box, into your existing Datadog, Grafana, or whatever you already run. And because Maya's per-agent identity shares a common key (the agent's cgroup) with host-based agent telemetry, your platform can correlate Maya's network-identity view with your existing application and runtime telemetry. Maya is not one more console to watch. It is a high-value signal that makes the observability investment you have already made smarter.

What is shipping, and where we are headed

We would rather be believed than impressive, so here is the honest tense.

Today

Per-agent identity attributed on the wire, with no code change to the agent. Cross-host agent-to-agent visibility and per-agent enforcement within your environment, on AWS, in your own VPC. Payload-blind by construction. Native OTLP export. In private beta, onboarding a small number of design partners.

By design, not by accident

Same-host agent-to-agent traffic is left unmediated on purpose. Co-located agents that never cross a wire do not need a wire-level control in the way cross-host traffic does. Maya governs the edges that leave a host.

Designed and sequenced

The inter-Warp fabric that carries one identity model across regions, providers, and geographies. The forwarding path already treats a peer in another region as just another next-hop, so extending the fabric is control-plane work on a data plane that is already built, not a new data plane. The agent-era primitives marked *Direction* above, reachability governance, agentic load-balancing, provenance, and plain-language policy, build on the same identity foundation.

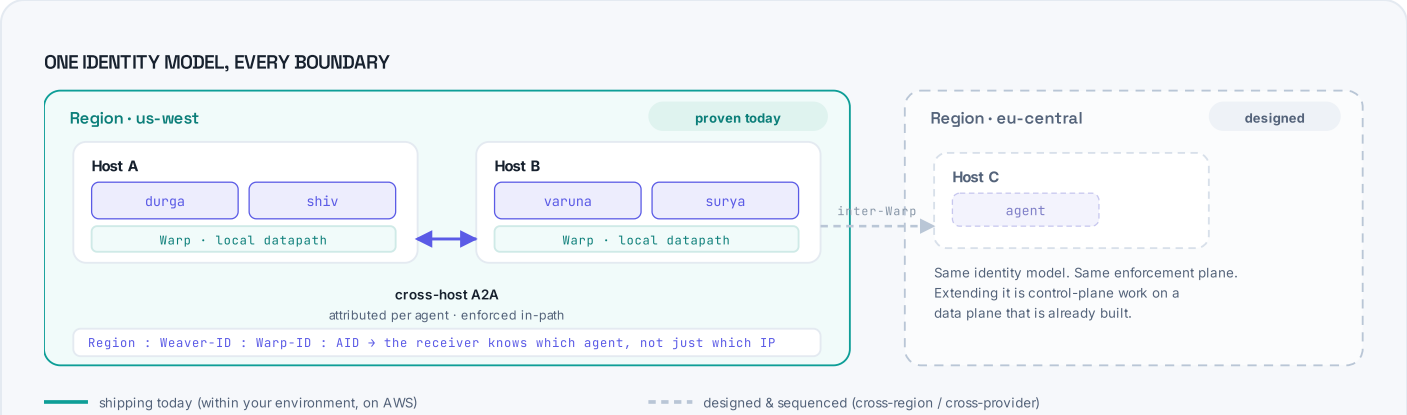

Figure 4. Within us-west, two hosts each run agents behind a local Warp datapath, and cross-host A2A traffic is attributed per agent and enforced in-path. Identity rides in-band, so the receiver knows which agent, not just which IP. A second eu-central region, drawn dashed, shares the same identity model and enforcement plane, connected by a designed inter-Warp link.

We are clear about which is which in technical conversations, and we intend to earn the rest.

Where this goes: the agent network at scale

What we have described is the foundation: one wire-derived handle, and permission already re-keyed onto it. Here is the trajectory the handle makes possible. Two of these are policy on the handle. Two are new layers that make agentic networking a stack rather than a control.

A logical identity built for millions

Maya's logical agent identity carries full scope, `Maya-AIAS : Region : Weaver-ID : Warp-ID : AID`, so an agent is uniquely addressable across a customer's entire domain, spanning regions, providers, and geographies. Because the underlying physical attribution is computed at the edge, identity shards without coordination: every enforcement point can agree on who an agent is without a global consensus bottleneck. That is the property that lets the fabric scale toward millions of agents in a single Maya domain, not as a marketing number, but as an architectural consequence of how identity is computed. It is also the property the layers below stand on: a globally unique, stable identity is what makes it safe to route to an agent or attest a chain of them.

Forwarding: distribution and prioritization on agentic properties

Once the network knows agent identity, it can decide where an agent's traffic goes and how much it gets, keyed on the agent rather than the connection. Distribute work by capability, by trust and reputation, by cost, by latency class, by which model backs an agent, by affinity to an ongoing A2A conversation. Reachability governance and agentic load-balancing are the same forwarding primitive with different actions. Balancing and steering agents by what makes them *agents* is a capability that only exists once the network is agent-aware. It is where agentic networking stops being defensive and starts being a platform.

Provenance: the identity chain across hops

A request crosses agent A to agent B to agent C. Because agent identity rides with the traffic, the fabric can attest the chain of identities a request traversed: this database call originated from agent A, three hops back, through B. Nobody has this today, because nobody had durable per-agent identity on the wire to chain. It is also a direct compliance artifact: proof that a data access traces to an authorized originator across an agent mesh, which is exactly what a regulated buyer under DORA or SOX has to show. Provenance is a direction, built on the same identity foundation that ships today.

Figure 5. Because agent identity rides with the traffic across hops, the fabric can attest the chain of agent identities a request traversed. That chain is a direct compliance artifact: proof that a data access traces to an authorized originator across the agent mesh.

One fabric, every boundary

Cross-host today; cross-region, cross-provider, cross-geo next. The same identity model and the same enforcement plane, everywhere agents talk. On AWS today, with the cloud-native architecture to follow agents wherever they run.

The world we are building toward

The internet spent two decades becoming identity-aware for people: single sign-on, zero-trust, per-user policy. We did it because access without identity is chaos, and at human scale, chaos is a breach.

Agents are arriving at a scale that dwarfs the human internet, and they act on their own. They need the same thing humans got, identity-aware, observable, enforceable communication, except the volume and the autonomy mean it has to live in the network itself, not bolted on top.

That is the world we are building: one where agent-to-agent communication is governed by default. Where you can deploy a fleet of a thousand agents, or a million, and see every agent you bring onto the fabric, know who each is talking to, and enforce what they may do, across every host, region, and cloud, without changing a single line of their code.

The network has to become agent-aware. We think it should be open, cloud-native, zero-instrumentation, and built for scale from the first line. That is Maya.

Building this era with us

Plug it in and watch your agents come alive, identified, mapped, and governable, in minutes, with nothing to instrument. We are actively onboarding design partners. If you are running agents, or about to be, and the blindness described in this paper sounds familiar, we want to talk.

Running on AWS today. Private beta. · Mayagentic Inc. · mayagentic.com

This document describes Maya's architecture and direction. Some capabilities are available today; others describe where the platform is headed. We are clear about which is which in technical conversations, and we intend to earn the rest.