Nick Clark · Published April 25, 2026 · PDF

How to Build Agent-to-Agent Messaging Without a Central Broker

You are building a network of agents that must exchange state directly, but every design keeps funneling through a broker, a session store, or a shared ledger that becomes the single point of failure. This guide shows an architectural approach where the message carries its own routing, policy, and trust context so nodes can decide locally, with no central coordinator. The approach described here is disclosed in United States Patent Application 19/366,760 (not a shipping library); its home inventive step is the Memory-Native Protocol inventive step.

What You Are Building

You have a set of autonomous agents, distributed across nodes, that need to exchange state and coordinate changes to shared structure. The obvious infrastructure answer is a broker: a message queue, a pub/sub hub, or a coordination service that every message flows through. That broker gives you ordering, delivery, and a place to enforce access rules, but it also becomes the thing that must be up, must be trusted, and must scale with your whole network.

This guide describes an architecture for doing without it. The goal is direct agent-to-agent exchange where each node can decide, on its own, whether to accept, route, mutate, or reject a message, using only what the message carries and what the node

locally knows. No central broker, no shared session store, no global ledger.

This is aimed at developers building decentralized AI meshes, federated knowledge networks, IoT fleets, or any system where intermittent connectivity, cross-organizational trust boundaries, or the absence of a central authority are structural facts rather than temporary inconveniences. The approach is disclosed in United States Patent Application 19/366,760. It is an architecture you implement yourself, not a package you install.

Why the Obvious Approaches Fall Short

Conventional network stacks (TCP/IP, DNS, REST, content distribution networks) are built for stateless packet transmission. They treat the data as transient and push session continuity, trust evaluation, and policy enforcement into separate layers outside the message. That separation is exactly what forces you toward a central coordinator: because the packet carries no durable state and no rules of its own, some other component has to remember who is talking to whom, what they are allowed to do, and whether a proposed change is legitimate.

The common workarounds each move the bottleneck without removing it. A message broker centralizes delivery and ordering. A shared session store centralizes state. A distributed ledger removes the single server but replaces it with global consensus, meaning every participant must agree on one ordered history before anything is final. That is a strong guarantee, but it is heavy: it assumes broad connectivity, a fixed or slowly changing validator set, and a willingness to synchronize globally. In asynchronous, disconnected, or federated environments (edge fleets, cross-organization networks, high-latency links) those assumptions do not hold cleanly.

The structural gap is this: the message itself is dumb, so something else has to be smart, and that something else becomes central. The approach below closes the gap by making the message the smart, self-contained unit.

The Architecture

The core inversion disclosed in the filing is to make the primary unit of the protocol a memory-bearing agent rather than a stateless packet. Governance logic is embedded directly in the data object, so each layer of the stack consults the message before acting, and the data itself initiates and constrains its own journey.

The agent structure. Each agent is a cryptographically self-contained operand with five parts: a unique identifier (UID), a payload (arbitrary semantic data, such as executable logic or structured knowledge), a transport header, a memory field, and a digital signature. The signature is computed over a canonical serialization of the UID, payload, memory field, and transport header, signed with the originating node's private key. On receipt, a node re-serializes and validates against the sender's public key; if validation fails, the agent is discarded and the rejection is logged locally. This is what lets a message be trusted without a broker vouching for it.

The memory field is the load-bearing idea. It is an append-only record containing verifiable lineage (the sequential history of structural changes the agent has undergone, hash-chained and individually signed by each contributing node), access logs (read, write, and execution events with timestamps and trust metadata), and policy references (pointers to policy agents that encode the governance rules). Because each agent carries its own history and its own rules, a node can evaluate authority locally using only the agent's embedded memory, with no external session verification or off-chain lookup. That property is what makes disconnected and intermittently connected operation possible.

Policy travels with the message. A policy reference resolves to a policy agent, which specifies which entities may mutate the agent, what quorum must be satisfied, and which behaviors are permitted. Per the disclosure, policy agents are themselves memory-bearing agents that encode governance logic; they are typically not self-

mutating and act as versioned authorities that other agents are evaluated against. They can be embedded directly as canonical identifiers or resolved by alias against a zone-local table scoped to the agent's declared trust domain.

A horizontally composable protocol stack. Rather than a vertically integrated, centrally orchestrated stack, the disclosed stack is a set of modular layers that each read the agent and act on it. The filing describes four:

- A **semantic memory layer** parses the memory field first, extracting lineage, policy references, trust scores, and tags so every layer above operates as memory-informed logic.
- A **dynamic routing protocol (DRP)** selects the next hop by trust, not by address. On arrival it parses the transport header (time-to-live, trust radius, semantic class) and the memory field, builds a local trust graph from access-log history, optionally folds in health signals, assigns trust scores to candidate neighbors, and forwards to the best eligible node. The worked example in the filing scores candidates against a policy-defined threshold and excludes nodes that fall below it or that exceed the agent's remaining TTL budget.
- A **dynamic indexing protocol (DIP)** is optional. It restructures semantic groupings (split, merge, reclassify) when entropy thresholds computed from agent-resident data are crossed. The filing is explicit that these are soft index anchors, not persistent containers, and that DIP can operate purely on UID-native lineage without any alias system.
- An **adaptive consensus protocol (ACP)** handles mutation proposals. This is the part that replaces global consensus. Instead of a fixed validator set, ACP dynamically scopes quorum eligibility from the policy references embedded in the agent's own memory field. Each node evaluates its own eligibility, voting weight, and policy alignment autonomously. Votes are themselves agents (each with its own UID and trace), weighted by the submitting node's trust score and domain scope, and

aggregated against the quorum logic encoded in the proposing agent's memory (for example, a stated threshold of 3 of 5 votes with cumulative weight at least 2.0). On resolution, an approval or rejection trace is appended to the agent's memory.

Every layer leaves a trace. After a layer executes, it appends a signed trace entry to the memory field. Downstream nodes use those traces to validate or replay outcomes, which is what preserves auditability and trust continuity across asynchronous hops without a central log.

Health as an in-band signal. The filing describes a network health monitoring system (NHMS) whose observations (congestion, latency variance, semantic entropy, cache pressure) are emitted as ordinary agents called health agents. They route through the same DRP and let nodes locally adjust routing preferences, raise quorum thresholds, or trigger reindexing, again without a central monitor.

Taken together: the message carries lineage, policy, and trust context; nodes decide locally; quorum is scoped per-agent instead of globally. That is the broker-free, ledger-free coordination model.

How to Approach the Build

The following is an ordered path to implement the architecture yourself. The interface sketches are illustrative and faithful to the disclosure; they are not a library to drop in.

1. **Define the agent envelope first.** Everything depends on it. Fix a canonical serialization for the five fields, because your signature depends on byte-for-byte agreement between sender and receiver.

```
// Illustrative only (shape, not an implementation)
Agent {
  uid:             UID
  payload:         bytes           // semantic data
  transportHeader { ttl, trustRadius, semanticClass, latencySensitivity, c
  memoryField {
    lineage:       [signed structural-change records, hash-chained]
    accessLog:     [ {node, op, timestamp, trustMeta} ]
    policyRefs:    [canonical id or alias]
    traces:        [signed trace entries]   // append-only
  }
  signature:       sig over canonical(uid, payload, transportHeader, memory
}
```

2. **Implement signature verify-then-parse as the node entry point.** A node must verify the signature before it trusts any field. Reject and locally log on failure. This is your substitute for a broker acting as gatekeeper.
3. **Build the semantic memory layer as the first read.** Parse the memory field and surface lineage, policy references, and any trust scores. Every later decision reads from here.
4. **Implement the DRP as local, trust-scored forwarding.** Maintain a per-node trust graph derived from prior memory-field evaluations. On each agent, score neighbors from access-log history (success rates, policy-violation frequency, responsiveness), apply the policy-defined minimum trust threshold, respect TTL and trust radius, forward to the best eligible node, and append the chosen path to the trace. Drop or flag agents that exceed TTL or violate scope.
5. **Model policy agents as first-class, referenceable objects.** Decide up front whether policies are embedded canonically or resolved by alias against a zone-local table. Evaluate them with locally cached rules so a disconnected node can still enforce access, mutation, and routing permissions.

6. **Add ACP only where mutations happen.** When a memory field carries a mutation proposal, trigger consensus. Have each node check its own eligibility against the referenced policy agent, cast votes as agents weighted by trust and domain scope, and aggregate against the quorum logic embedded in the proposing agent. Append an approval or rejection/quarantine trace on resolution. Nodes that only route messages do not need this layer.
7. **Treat health as agents, not as a dashboard.** If you need adaptivity, emit signed health agents and let receiving nodes adjust DRP preferences, quorum thresholds, or DIP behavior under their local policy.
8. **Add DIP last, and only if you need adaptive structure.** It is explicitly optional; stateless deployments can omit it.
9. **Choose your deployment mode per node.** The disclosure supports stateless mode (all decisions from the agent's own data) and memory-aware mode (a persistent local memory graph for richer trust modeling). Edge nodes can run just DRP plus a simplified memory layer; core nodes can run the full stack. Nodes of different capability interoperate.

Design tradeoffs to weigh as you go: per-message overhead grows because each agent carries its lineage and policy context; you are trading a central bottleneck for larger, self-describing messages. Trust-scoped routing means eventual, path-dependent delivery rather than the total ordering a broker or ledger can offer. And quorum scoped to a single agent's policy is finer-grained than global consensus but gives you no global agreement across unrelated agents, which is usually the point.

What This Does Not Give You

This is an architecture, not a drop-in library. There is no package to install and nothing here "just works" out of the box; the interface sketches above are illustrative, and you implement the substrate yourself. The method is disclosed in a patent filing; it has not

been presented here as a benchmarked or production-proven product, and this guide states no performance figures beyond what the filing itself defines.

It is also not a fit for every problem. If your application genuinely requires a single total order across all events (some financial settlement models, for instance), per-agent scoped quorum does not provide that, and a ledger or ordered broker may be the right tool. If your network is small, well-connected, and centrally administered, a broker is simpler and the per-message overhead of carrying lineage and policy will not pay for itself. The approach earns its keep specifically where central coordination is impractical: asynchronous, disconnected, federated, or trust-divergent environments. The correctness of the whole model rests on sound key management and canonical serialization; get those wrong and signature validation, which is the trust anchor here, gives you nothing.

Disclosure Scope

The architecture described in this guide is disclosed in United States Patent Application 19/366,760, "Cognition-Compatible Network Substrate and Memory-Native Protocol Stack." Its home inventive step is the Memory-Native Protocol inventive step: making the message a cryptographically signed, memory-bearing agent that carries its own verifiable lineage, embedded policy, and trust-scoped routing so that nodes coordinate locally without a central broker or shared ledger. This guide is educational. It explains an approach a developer can build; it is not a warranty, a specification, or an offer of software, and nothing here should be read as a claim that a benchmarked or productized implementation is being distributed.

Memory-Native Protocol (/memory-native-protocol) All 40 steps → (/inventive-steps)

Authority intrinsic to the object. Routing by semantic properties.

U.S. 19/366,760 (/patents/19-366760)

PRIMARY TECHNICAL DISCLOSURE

- Memory-Native Networking: A Cognition-Compatible Protocol Substrate (/articles/memory-native-networking-a-cognition-compatible-protocol-substrate)

SECONDARY TECHNICAL

- Protocol-Native Carriers: Agents as the Fundamental Unit of Transmission (/articles/memory-native-protocol/protocol-native-carrier)
- Dynamic Routing Protocol: Memory-Aware Path Selection for Semantic Agents (/articles/memory-native-protocol/dynamic-routing)
- Trust-Weighted Route Scoring: Dynamic Path Selection Through Policy-Defined Trust Thresholds (/articles/memory-native-protocol/trust-weighted-routing)
- Network Health Monitoring System: Signed Health Agents as Distributed Operational Telemetry (/articles/memory-native-protocol/network-health-monitoring)
- Health Agents as Semantic Objects: Operational Metrics That Route Like Any Other Agent (/articles/memory-native-protocol/health-agents)
- Dynamic Indexing Protocol: Entropy-Driven Restructuring of Semantic Flows (/articles/memory-native-protocol/dynamic-indexing)
- Soft-Index Anchors: Ephemeral Index Points Inferred From Agent Lineage (/articles/memory-native-protocol/soft-index-anchors)
- Adaptive Consensus Protocol: Memory-Native Quorum Without Fixed Validator Sets (/articles/memory-native-protocol/adaptive-consensus)
- Trust-Weighted Voting in ACP: Domain-Scoped Votes Accumulated Against Agent Memory (/articles/memory-native-protocol/acp-trust-voting)
- Dynamic Alias Resolution: Zone-Local Semantic Aliases Resolved Through Transport Headers (/articles/memory-native-protocol/alias-resolution)
- Horizontally Composable Protocol Stack: Independent Layers Operating in Parallel (/articles/memory-native-protocol/composable-stack)
- Transport-Layer Agnosticism: One Protocol Stack Above Any Carrier (/articles/memory-native-protocol/transport-agnosticism)
- Federated Semantic Zone Deployment: Heterogeneous Nodes Coordinating Across Trust Boundaries (/articles/memory-native-protocol/federated-zones)
- Health-Triggered Quorum Adjustment: Dynamic Thresholds From Network Stability Signals (/articles/memory-native-protocol/health-triggered-quorum)

- The Agent Is the Wire Format: A Self-Contained Unit on the Network (/articles/memory-native-protocol/governed-mesh-wire-format)
- Hop-History Relay and In-Band Chain of Custody (/articles/memory-native-protocol/hop-history-relay)
- Mobile Store-and-Forward Without Cellular Backhaul (/articles/memory-native-protocol/mobile-store-and-forward)

APPLICATIONS · GENERAL

- A Memory-Native Coordination Fabric for Multi-Agent AI Orchestration (/articles/memory-native-protocol/multi-agent-orchestration-fabric)
- Edge Routing Without a Central Control Plane: Compliance-Grade Routing Authority at the Edge (/articles/memory-native-protocol/edge-routing)
- Broker-Free IoT Device Mesh Governance at Scale (/articles/memory-native-protocol/iot-mesh)
- V2V Communication Without Roadside Infrastructure: Memory-Native Trust for Autonomous Vehicles (/articles/memory-native-protocol/autonomous-vehicle-networking)
- Military Mesh Networks Without Central Routing Authority (/articles/memory-native-protocol/military-mesh-networks)
- Decentralized Smart City Infrastructure Without a Central Control Platform (/articles/memory-native-protocol/smart-city-infrastructure)
- Delay-Tolerant Satellite Routing Governance for LEO Constellations (/articles/memory-native-protocol/satellite-communication)
- Industrial IoT Protocols Without Broker-Centralized Authority: A Memory-Native Substrate for Credentialed OT Telemetry (/articles/memory-native-protocol/industrial-iot-protocols)
- Healthcare Device Mesh Networking for Fault-Tolerant Clinical Data (/articles/memory-native-protocol/healthcare-device-mesh)
- Contested-Mesh Radio for Defense and Public Safety (/articles/memory-native-protocol/contested-mesh-radio)
- Expeditionary Mesh for GNSS-Denied Operations (/articles/memory-native-protocol/expeditionary-mesh)
- Maritime, Agricultural, and Mining IoT Mesh Without Cellular Backhaul (/articles/memory-native-protocol/maritime-iot-mesh)
- Why Mesh Networks Stall in Contested, Multi-Vendor Deployments: Node-Resident Governance and the Carried-Authority Fix (/articles/memory-native-protocol/carried-authority-ceiling)
- How to Contain a Compromised Node in a Distributed Network Without Trusting It (/articles/memory-native-protocol/malicious-host-contained)
- Delay-Tolerant and Interplanetary Autonomy: Carrying Authority When There Is No Link Home (/articles/memory-native-protocol/disconnected-and-interplanetary)

APPLICATIONS · SPECIFIC

- Starlink Alternative for Governed Mesh Routing: Why Satellite Handover Authority Stays Terrestrial (/articles/memory-native-protocol/starlink)
- Zigbee vs Governed IoT Messaging: Why the Mesh Routes Frames but Carries No Authority (/articles/memory-native-protocol/zigbee)
- Does Matter Let Governance Travel With the Message? (/articles/memory-native-protocol/matter)
- Helium Alternative for Governed IoT Transport: Decentralized Coverage Plus Message-Borne Governance (/articles/memory-native-protocol/helium)
- LoRaWAN Alternative for Governed IoT: Memory-Native Messages vs Passive Payloads (/articles/memory-native-protocol/lorawan)
- Beyond the Tailscale Coordination Server: Governed Mesh Networking Where Authority Travels With the Packet (/articles/memory-native-protocol/tailscale)
- QUIC vs Content-Scoped Authority: A Memory-Native Protocol Layer Above QUIC (/articles/memory-native-protocol/quic-protocol)
- MQTT vs Memory-Native Protocol: Where IoT Messaging Authority Should Live (/articles/memory-native-protocol/mqtt)
- CoAP Brought REST to Constrained Devices. The Protocol Carries No Governance Semantics. (/articles/memory-native-protocol/coap)
- gRPC Alternative for Governed Agent Execution: Where the Memory-Native Protocol Fits (/articles/memory-native-protocol/grpc)
- ZeroMQ vs Memory-Native Protocol: Brokerless Sockets Without Carried Authority (/articles/memory-native-protocol/zeromq)
- WireGuard vs Memory-Native Protocol: Governed Payloads Above the Tunnel (/articles/memory-native-protocol/wireguard)
- Nebula vs a memory-native protocol: does the mesh still depend on a central certificate authority? (/articles/memory-native-protocol/nebula-mesh)
- Calico Enforces Network Policy at the Kernel. A Governed Alternative Carries Policy in the Packet. (/articles/memory-native-protocol/calico)
- Cilium vs Memory-Native Protocol: Where Governance Lives in the Stack (/articles/memory-native-protocol/cilium)
- Weave Net Alternative for Governed Agent Execution: Where the Memory-Native Protocol Fits (/articles/memory-native-protocol/weave-net)
- Persistent Systems Wave Relay vs Protocol-Native Authority Semantics (/articles/memory-native-protocol/persistent-systems)
- Does Silvus StreamCaster Provide a Payload Governance Layer? (/articles/memory-native-protocol/silvus-streamcaster)

- Rajant Kinetic Mesh and Payload-Level Governance: A Memory-Native Layer Above the Link (/articles/memory-native-protocol/rajant-kinetic-mesh)
- TrellisWare TSM vs Governed Observation Admissibility: Routing Is Not Authority Resolution (/articles/memory-native-protocol/trellisware-tsm)
- Autotalks Craton2 vs Governed V2X: The Authority Layer Above the Chipset (/articles/memory-native-protocol/autotalks-craton2)
- Qualcomm 9150 C-V2X vs Memory-Native Behavioral Authority (/articles/memory-native-protocol/qualcomm-9150)
- Does NXP RoadLink Govern What a V2X Message Is Authorized to Do? (/articles/memory-native-protocol/nxp-roadlink)
- Chroma Vector Database vs a Governed Memory-Native Substrate (/articles/memory-native-protocol/chroma-vector-db)
- Milvus Alternative: Governed Agent Memory Beyond the Vector Database (/articles/memory-native-protocol/milvus-vector-db)
- Pinecone Alternative for Governed Agent Memory (/articles/memory-native-protocol/pinecone-vector-db)
- Qdrant Alternative: Governed, Portable AI Memory Beyond the Vector Database (/articles/memory-native-protocol/qdrant-vector-db)
- Weaviate Alternative for Governed Vector Memory: The Memory-Native Protocol (/articles/memory-native-protocol/weaviate-vector-db)
- Anduril Lattice Mesh vs Carried Governance: A Memory-Native Protocol Comparison (/articles/memory-native-protocol/anduril-lattice-mesh)
- Shield AI Hivemind vs Governed Team Coordination: The Authority Layer Above Onboard Autonomy (/articles/memory-native-protocol/shield-ai-hivemind)
- Bundle Protocol v7 / DTN (NASA ION) vs a memory-native protocol: where do trust, policy, and consensus live? (/articles/memory-native-protocol/bundle-protocol-dtn)
- IOTA (Tangle) alternative: agent-carried trust without a shared ledger (/articles/memory-native-protocol/iota-tangle)
- Model Context Protocol (MCP) vs a memory-native protocol: where trust, lineage, and policy live (/articles/memory-native-protocol/model-context-protocol)

Memory-Native Protocol overview → (/memory-native-protocol)