Nick Clark · Published July 2, 2026 · PDF

How to Build a Mesh Network That Keeps Working When Disconnected

If you are building for edge fleets, field sensors, tactical links, or interplanetary relays, you already know the hard part is not connectivity, it is what happens when connectivity is gone. This guide walks through an architectural approach where the data object carries its own routing, policy, and consensus rules, so nodes can act on it during long disconnections without a central server. The approach described here is disclosed in United States Patent Application 19/366,760; it is an architecture you build, not a shipping library you install. The home inventive step is the Memory-Native Protocol inventive step.

What You Are Building

You are building a network that keeps making forward progress while it is partitioned. Not a network that reconnects gracefully after an outage, but one where a node holding a message during a multi-hour or multi-day disconnection can still route it, apply access policy to it, and even participate in a decision about it, all before the partition heals.

This is the problem faced by anyone whose links are intermittent by design: sensor fleets that surface for minutes a day, disaster-response radios with no backhaul, ships and aircraft that leave coverage for hours, and deep-space relays where a round trip is

measured in minutes or hours. The common failure is that the coordination logic (who is allowed to do what, whose vote counts, where a message should go next) lives on a server the node cannot currently reach.

The architectural move described here, disclosed in United States Patent Application 19/366,760, is to stop treating messages as inert packets and instead make the message itself the carrier of its own governing context. Each unit of transmission is an "agent": a signed data object with an embedded memory field that holds its own lineage, access history, and policy references. Because the rules travel with the data, a disconnected node has everything it needs to act correctly without calling home.

Why the Obvious Approaches Fall Short

The conventional stack is built for stateless, address-based delivery. TCP/IP, DNS, and REST APIs move bytes between endpoints and delegate session continuity, trust evaluation, and policy enforcement to layers above the packet. This is a deliberate and successful design for the connected internet. The structural consequence, though, is that the coordination state lives outside the message: in a session table, an auth server, a routing controller, or a database. When a node is cut off from those, it can forward blindly but it cannot decide anything.

The standard delay-tolerant answer, store-and-forward (as in DTN bundle protocols), solves the transport half of the problem: a node holds a bundle until a contact opens, then passes it on. That is genuinely useful and this architecture is designed to run on top of it. But classic store-and-forward moves opaque payloads. The node carrying your data during the partition still cannot tell whether it is allowed to read it, whether a proposed change to it is authorized, or which neighbor is the trustworthy next hop, because that judgment normally requires reaching an external authority.

Distributed ledgers remove the central server but replace it with a different dependency: globally synchronized state. Reaching agreement typically means talking to a quorum of validators over a shared log. In a partitioned network that quorum may be on the far side of the gap, so progress stalls exactly when you need it. The gap that remains, across all three approaches, is that decisions depend on something the disconnected node cannot touch.

The Architecture

The disclosed approach closes that gap by relocating the decision-making inputs into the data object. The specification calls this a memory-native protocol substrate. Four ideas carry the weight.

1. The agent carries its own governing context. Per the spec, each agent comprises a unique identifier, a payload, a memory field, a transport header, and a cryptographic signature. The memory field is the key structure: an append-only record holding a signed lineage of prior mutations, an access log of node interactions, and policy references that point to (or embed) the governance rules for that agent. The transport header carries propagation constraints such as time-to-live, trust radius, semantic class, latency sensitivity, and quorum priority. Because each agent carries its own execution context, trust parameters, and routing constraints internally, a node can evaluate it without persistent sessions or source-address routing. The spec states directly that embedded policy is what "enables secure operation in disconnected or intermittently connected networks, such as IoT or interplanetary systems."

2. Every node verifies before it acts, using only the agent. 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 rejected and the rejection is logged locally. Each individual trace entry in the memory field is also separately signed by the node that produced it and chained by

cryptographic hash, which preserves both tamper-evidence and chronological order across trust zones. This is what makes it safe to accept an agent that has spent days in transit through nodes you have never spoken to: its integrity is self-verifying.

3. Routing is trust-scoped and local, not table-based. The dynamic routing protocol (DRP) scores candidate next hops from signals it reads out of the agent's own memory field (access-log outcomes, prior traces, embedded policy) combined with local trust inference, rather than from a global routing table. In the spec's worked example, a node builds a local trust graph from its own access records, scores each neighbor, excludes any below a policy-defined trust threshold or too costly against the remaining TTL, selects the best next hop, and appends the chosen path to the agent's memory trace. A disconnected node can compute all of this offline because every input is either agent-resident or locally held.

4. Consensus is dynamically scoped from embedded policy, with no shared ledger. The adaptive consensus protocol (ACP) lets nodes evaluate a mutation proposal carried in an agent's memory field without centralized coordination or globally synchronized state. There are no fixed validator sets and no persistent governance registry. Instead, quorum eligibility is scoped by the policy references embedded in the agent: each node evaluates its own eligibility, voting weight, and policy alignment autonomously from the agent alone. Crucially for disconnection, the spec says votes "may be accumulated locally or distributed via DRP to other eligible quorum participants." Each vote is itself an agent. So a decision can be assembled incrementally as partitions open and close, and the accumulated approval or rejection trace is appended to the originating agent's memory for downstream auditability. The spec also notes ACP runs in a stateless mode where all eligibility and weighting derive solely from the agent's memory field at runtime.

Underlying all of this is store-carry-forward at the transport layer. The spec states the substrate runs unmodified over TCP/IP, HTTP, WebSockets, WebRTC, mesh relay, or delay-tolerant networking, and that nodes may cache unresolved agents, reroute them

via delay-tolerant protocols, or propagate them along broadcast overlays. Because an agent carries all context needed for execution, it can "propagate and be validated even after long delays."

How to Approach the Build

You are implementing this yourself. The steps below sequence the work and stay faithful to the disclosed design. The sketches are illustrative only.

Step 1: Define the agent envelope. Everything hinges on this. Fix a canonical, deterministic serialization (the same bytes must be reproducible on any node for signatures to verify) with these fields:

```
Agent {
  uid             // globally unique, cryptographic root of lineage
  payload         // your application data
  memory_field {
    lineage[]      // signed, hash-chained mutation history
    access_log[]   // node interactions: read/write/exec + trust metadata
    policy_refs[]  // embedded stubs or references to policy agents
    traces[]       // routing outcomes, health, decisions
  }
  transport_header // TTL, trust_radius, semantic_class, latency, quorum_pri
  signature        // over canonical(uid, payload, memory_field, transport_h
}
```

Get the canonicalization right first. If two nodes serialize the same agent differently, every signature check downstream fails.

Step 2: Implement the verify-parse-decide receive path. Per the method claim, each node on receipt: verifies the signature; parses the transport header and memory field; determines routing eligibility and mutation scope by evaluating the access log and

policy references; executes the relevant protocol layers; appends a signed trace; and only then forwards. Make signature failure a hard stop with a local log entry, never a silent drop.

Step 3: Represent policy as agents, resolved locally. Encode governance (who may mutate, quorum thresholds, role definitions) as policy agents that are either embedded in the proposing agent or referenced and resolvable from a node-local or cached table. The spec is explicit that policy agents "do not typically mutate themselves" and act as versioned authorities. Caching them is what lets a partitioned node enforce policy with no lookup.

Step 4: Build DRP as a local scoring function. Given an agent and the node's local trust graph, score each neighbor from access-log history and any cached health signals, drop candidates below the policy trust threshold or beyond the remaining TTL, pick the best, and append the decision to the trace. No global table. Design it to return a "hold and carry" outcome when no eligible next hop is currently reachable, which is the disconnected case.

Step 5: Build ACP for incremental, offline-tolerant quorum. When an agent's memory field carries a mutation proposal, have the node check its own eligibility from the embedded policy, cast a vote as a new agent weighted by its trust and domain scope, and accumulate votes locally or forward them by DRP. Aggregate against the quorum logic encoded in the agent (for example the spec's illustration of "3 out of 5 votes with cumulative weight at or above 2.0"), then append approval or a rejection/quarantine flag. Because votes are agents, they survive partitions and merge when contact resumes.

Step 6: Add the optional feedback and indexing layers. A network health monitoring system can emit signed health agents (congestion, latency variance, entropy, cache pressure) that neighbors use to adjust routing preferences, quorum

thresholds, or reclassification, again with no central controller. A dynamic indexing protocol can locally restructure how agents are grouped based on entropy. The spec presents both as optional; add them when you need self-regulation, not on day one.

Step 7: Choose a mode per node class. Run constrained devices in stateless mode (all decisions from the agent alone) and capable nodes in memory-aware mode (a persistent trust graph for better routing and quorum forecasting). The spec describes evolutionary deployment: a node can start as a stateless router and adopt more layers over time without changing its identity or coordination logic.

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; you implement the agent envelope, the receive path, DRP, ACP, and policy resolution yourself, and the correctness of your canonical serialization and key management is on you.

The method is disclosed in a patent filing. It is not presented here as a benchmarked or production-proven product, and this guide states no performance figures, throughput numbers, or delivery guarantees beyond what the specification describes qualitatively. The one quantitative timing the specification offers, its definition of "near real-time" as roughly 250 milliseconds, describes local processing latency, not any promise about partitioned end-to-end delivery.

The approach also does not repeal physics or partition tradeoffs. If a quorum of eligible voters is unreachable for the entire life of an agent, the decision waits; embedding the rules removes the dependency on a central server, but it cannot manufacture participants that never come into contact. Delivery still depends on a contact eventually opening. And it is not a fit where you actually want strong global serialization or a

single canonical ledger of record, because the design deliberately avoids globally synchronized state. Treat it as a fit for intermittently connected, trust-divergent, decentralized environments, which is exactly the domain the specification targets.

Disclosure Scope

The architecture described in this guide, a memory-native protocol substrate in which each transmitted agent carries verifiable lineage, embedded policy, and trust-scoped routing so that routing, mutation, and consensus can proceed without centralized coordination, global consensus, or persistent session state, is disclosed in United States Patent Application 19/366,760. This guide is educational. It describes an architectural approach a developer can implement, and it is not a shipping software product, a warranty, a benchmark, or an offer of software. Every mechanism above is drawn from that filing; where the specification is silent, this guide makes no claim.

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)