Nick Clark · Published July 2, 2026 · PDF

How to Build a Delay-Tolerant Network for Space Communications

If you are designing communications for lunar relays, deep-space probes, or any link where round-trip latency runs minutes to hours and connectivity comes and goes, session-based networking breaks down. This guide walks through an architectural approach in which each message carries its own trust, policy, and history so nodes can route, validate, and forward it during long blackouts with no live coordinator. The approach is disclosed in United States Patent Application 19/366,760; it is an architecture you build yourself, not a shipping library. Its home inventive step is the Memory-Native Protocol inventive step.

What You Are Building

You are building a network that keeps working when the link does not. In space communications the defining constraints are long, variable propagation delay and frequent disconnection: a probe behind a planet, a rover waiting on an orbiter pass, a lunar relay that is only in view for part of the day. A request and its response may be separated by minutes or hours, and no two endpoints are reliably online at the same moment. Anything that assumes a live session, a synchronous handshake, or a reachable central controller stalls the instant the link drops.

The searcher's real question is: how do I move data, and enforce who is allowed to change it, across links that are asynchronous and often broken, without a coordinator that everyone can reach? This guide describes an architecture where the unit that crosses the network is not a stateless packet but a self-contained, cryptographically signed object that carries its own routing constraints, policy, and history. Nodes store it, carry it, and forward it opportunistically, and every node can decide what to do with it using only what the object carries. That approach is the substance of United States Patent Application 19/366,760, and it is what the rest of this guide teaches you to build.

Why the Obvious Approaches Fall Short

The conventional internet stack (TCP/IP, DNS, REST) is built for stateless packet transmission with external layers handling session continuity, trust, and policy. This design is excellent for well-connected terrestrial networks. Its assumptions simply are not available across a space link: TCP wants a round-trip handshake and timely acknowledgments, DNS wants a reachable resolver, and REST wants a live server on the other end. When the round trip is an hour and the peer is over the horizon, these are not present.

Delay-tolerant networking as a field already answers part of this. The standard move is store-and-forward: a node holds a bundle of data until a next hop becomes available, then passes it along, hop by hop, tolerating long gaps. This solves custody and delivery. What it does not inherently solve is governance during the disconnection. If a message needs to be routed according to trust, checked against an access policy, or approved by some quorum before a change is committed, and the authority that would make those decisions is unreachable for an hour, a store-and-forward layer that only moves opaque payloads has nowhere to get the answer. You either block until the coordinator is reachable (defeating the point) or forward blindly (defeating the governance).

The structural gap is this: in a disconnected network, the decision context has to travel with the data, because the thing that would normally supply that context is not online. Address-based routing, centralized trust assignment, and globally synchronized consensus all assume reachability that a space link cannot promise.

The Architecture

The disclosed approach closes that gap by making the message itself the carrier of its execution context. Every field below traces to the filed specification.

The agent is the unit of transmission. Instead of a stateless packet, the primary object is what the disclosure calls an agent: a cryptographically self-contained operand with five parts. A unique identifier (UID) anchors its identity and lineage. A payload carries the actual semantic data. A memory field holds a signed lineage record, access logs, and references to the policies that govern it. A transport header encodes delivery constraints, including time-to-live (TTL), trust radius, semantic class, latency sensitivity, and quorum priority. A cryptographic 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. This is what lets the object be trusted after an arbitrarily long delay: authenticity is verified from the object, not from a live session.

The memory field carries governance, so disconnection does not block decisions. The memory field is append-only and hash-chained, with each entry signed by the node that contributed it. It records mutation lineage (the sequence of structural changes and the policies under which they were made), an access log of prior node interactions, and policy references that point to policy agents encoding permission rules and quorum thresholds. Because the policy the agent must satisfy is referenced or embedded directly in the object, a node can evaluate authority locally, using only the

agent's embedded memory, with no external session verification or off-chain lookup. The specification calls out precisely this property as enabling secure operation in disconnected or intermittently connected networks, including interplanetary systems.

A composable protocol stack interprets the agent locally. Each node runs a horizontally composable stack whose behavior is driven by the agent's metadata rather than by node-local session state. The disclosure names four layers. A semantic memory layer extracts lineage, policy references, trust scores, and tags from the memory field. A dynamic routing protocol (DRP) chooses next hops from trust scope, access history, and policy constraints rather than static addresses, and can suppress unreliable paths. A dynamic indexing protocol (DIP) optionally restructures how agents are organized in response to entropy and semantic drift. An adaptive consensus protocol (ACP) evaluates any mutation proposal the agent carries. Each layer appends a trace to the memory field as it acts, so downstream nodes can validate or replay what happened, even across asynchronous or intermittently connected systems.

Consensus is scoped to the agent, not the globe. This is the part that matters most for space. ACP lets distributed nodes evaluate a mutation proposal without centralized coordination or globally synchronized state. Quorum eligibility is scoped dynamically by the policy references embedded in the agent's own memory field: each node evaluates its own eligibility, voting weight, and policy alignment autonomously from what the agent carries. A vote is itself an agent, weighted by the voter's trust score and domain scope. The quorum logic (for example, a threshold of a minimum number of votes and a cumulative trust weight) is encoded in the memory field. Votes can be accumulated locally or propagated to other eligible participants via DRP. Because there is no fixed validator set and no persistent governance registry, a quorum can form among whichever eligible nodes are reachable within a communication window, and the accumulated approval trace travels on with the agent.

Store-carry-forward falls out of the model. The disclosure states that the memory-bearing nature of agents lets the system function in asynchronous, delay-tolerant conditions: an agent carries all the context needed for execution (policy, mutation proposal, quorum metadata, routing constraints), so it can propagate and be validated even after long delays. Nodes may cache unresolved agents, reroute them via delay-tolerant protocols, or propagate them along broadcast overlays. The substrate is explicitly described as sitting above the transport layer and running unchanged over TCP/IP, HTTP, WebSockets, WebRTC, mesh relay, or delay-tolerant networking. TTL and trust radius in the transport header bound how far and how long an agent should keep being carried.

Stateless nodes can still participate. For resource-constrained or transient endpoints, nodes can run in stateless mode: with no persistent memory between evaluations, they rely exclusively on the embedded agent data for trust evaluation, quorum participation, and policy enforcement. The disclosure names IoT devices, ephemeral containers, and anonymized relays as examples. This is what lets a minimal relay with intermittent uptime take part without a full stack.

How to Approach the Build

The following steps are the order a developer would implement the architecture. The sketches are illustrative pseudocode, faithful to the disclosure, not a package you can install.

1. Define the agent object and its serialization. Fix a canonical serialization for UID, payload, memory field, and transport header, then sign that serialization. The canonical form is load-bearing: verification depends on both ends serializing identically.

```
Agent = { uid, payload, memory_field, transport_header, signature }
transport_header = { ttl, trust_radius, semantic_class, latency_sensitivity,
memory_field = { lineage[], access_log[], policy_refs[], traces[] }   // app
signature = sign(private_key, canonical(uid, payload, memory_field, transpor
```

2. Implement receive-and-verify first. Before any routing logic, a node re-serializes the received agent and validates the signature against the sender's public key. Reject and log on failure. Everything downstream assumes a verified agent.

3. Model policy as referenced or embedded policy agents. A policy agent encodes who may mutate, what quorum is required, and role permissions. Resolve it from a local or cached copy so a node can evaluate authority with no network call. Design for the offline case as the default, not the exception.

4. Build the routing layer to score from memory, not addresses. On receipt, parse the transport header and memory field, read the access log for the behavior of neighboring nodes, optionally fold in local health signals, assign trust scores to candidates, and drop candidates below a policy-defined threshold or over their TTL cost. Append the chosen path to the memory trace before forwarding.

```
on_receive(agent, node):
    if not verify_signature(agent): reject_and_log(agent); return
    ctx = parse(agent.transport_header, agent.memory_field)
    if expired(ctx.ttl) or out_of_scope(ctx.trust_radius): drop_or_quarantin
    if agent.has_mutation_proposal(): run_consensus(agent, node)   // ACP
    next = score_candidates(ctx.access_log, local_health, policy_threshold)
    append_trace(agent, node, decision)
    forward_or_carry(agent, next)   // hold if no next hop is currently reac
```

5. Add the consensus path for anything that mutates shared state. When an agent carries a mutation proposal, validate its embedded policy reference, check this node's own eligibility, cast a trust-weighted vote as a new agent, and accumulate votes against the quorum logic encoded in the memory field. Approvals and rejections are appended to the agent's trace so the outcome is auditable later. Scope quorum to whoever is reachable and eligible in the current window rather than waiting for a global set.

6. Bound propagation with TTL and trust radius, and implement store-carry-forward explicitly. If no eligible next hop is reachable, hold the agent and retry on the next contact window. Use TTL and trust radius to decide when to stop carrying and when to quarantine.

7. Layer indexing and health feedback last, and only where nodes can afford them. DIP (entropy-driven restructuring) and NHMS (health agents that adjust routing and quorum thresholds) are optional. Core nodes can run the full stack; edge relays can run routing plus verification only. The disclosure describes exactly this mixed, evolutionary deployment where nodes begin as stateless routers and adopt more layers as capacity allows.

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 format, the signature scheme, the stack layers, and the storage yourself. The disclosure describes a design; it is not a benchmarked or productized system, and it states no throughput, latency, or delivery guarantees for a space link, so you should not read one into it. The one timing figure the specification gives, roughly 250 milliseconds for its notion of near real-time, is a definition of a term, not a performance claim about interplanetary operation.

The approach also does not remove physics or replace lower layers. It sits above the transport, so you still need a real delay-tolerant transport or link layer underneath to actually carry bits across the gap, along with the radios, coding, and scheduling that live below it. Consensus here is scoped to reachable, eligible nodes; it is deliberately not global agreement, so if your problem genuinely requires a single globally synchronized ledger state, this model is not that. Key management, clock handling across long delays, and payload semantics are yours to design. And in a well-connected terrestrial setting with a reachable coordinator, a conventional stack may be simpler; the payoff here is specifically disconnection tolerance and travelling governance.

Disclosure Scope

The architecture described in this guide is disclosed in United States Patent Application 19/366,760. This guide is educational: it explains an approach a developer can implement, grounded in that filing. It is not a warranty, a specification of a shipping product, or an offer of software, and it does not grant any license. Any real technologies named for context are described only to situate the approach and remain the property and standards of their respective owners.

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)