Independent Submission
Internet-Draft
Intended status: Informational
Expires: 8 October 2026

C. Teodor
Vulture Labs

6 April 2026

Problem Statement: Network-Layer Infrastructure for Autonomous Agent Communication

draft-teodor-pilot-problem-statement-01

Abstract

AI agents --- autonomous software entities capable of reasoning, planning, and executing tasks --- are an increasingly important class of network participant. Current agent communication protocols operate exclusively at the application layer over HTTP, assuming the existence of stable endpoints, DNS names, and centralized infrastructure. No existing standard provides network-layer identity, addressing, or transport for agents. This document describes the problem space and identifies requirements for a network-layer infrastructure that would give agents first-class network citizenship, independent of the web infrastructure designed for human users.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 October 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

The internet's protocol stack was designed for human-operated devices with stable network attachments. IP addresses identify interfaces, DNS names identify services, and TLS certificates identify organizations. These assumptions break down for AI agents, which are transient software processes that may run behind NAT, migrate between hosts, and lack persistent network identity.

Recent standardization efforts for agent communication --- notably MCP [MCP] (agent-to-tool) and A2A [A2A] (agent-to-agent) --- have focused on application-layer protocols built on HTTP. These protocols define what agents say to each other but assume the underlying problem of how agents reach each other is already solved. For agents running in cloud environments with public endpoints, this assumption holds. For agents running on edge devices, behind corporate firewalls, on laptops, or in heterogeneous multi-cloud deployments, it does not.

The IETF has seen explosive activity in AI agent protocol standardization, with over thirty individual drafts filed in 2025-2026 covering agent identity ([I-D.prakash-aip], [I-D.ni-wimse-ai-agent-identity]), discovery ([I-D.nemethi-aid-agent-identity-discovery]), transport ([I-D.hood-independent-agtp], [I-D.li-atp]), routing ([I-D.du-catalist-routing-considerations]), frameworks ([I-D.rosenberg-aiproto-framework], [I-D.zyyhl-agent-networks-framework], [I-D.eckert-catalist-acip-framework]), and audit ([I-D.sharif-agent-audit-trail]). The CATALIST Birds of a Feather session at IETF 125 (March 2026, Shenzhen) was the first formal coordination effort, surveying the problem space ([I-D.yao-catalist-problem-space-analysis]) without yet chartering a working group.

Despite this volume, the vast majority of these drafts operate at the application layer over HTTP. Even the dedicated transport proposals (AGTP, ATP) define application-layer semantics carried over QUIC or TCP --- none provides an overlay network with virtual addressing, port-based multiplexing, and built-in NAT traversal at the network layer. The network-layer gap identified in the original version of this document remains unaddressed.

This document describes the problem of network-layer infrastructure for autonomous agent communication, identifies the gaps in existing protocols, and states requirements for a solution. It is modeled after [RFC7364], which performed a similar analysis for network virtualization overlays.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Agent: An autonomous software entity capable of reasoning, planning, and executing tasks without continuous human supervision. An agent may run as a process, container, or serverless function.

Overlay Network: A virtual network built on top of an existing network (the underlay). Overlay nodes communicate using encapsulated packets carried over the underlay.

Virtual Address: A network address assigned within the overlay address space, independent of the underlay IP address. A virtual address identifies an agent, not a network interface.

Registry: A service that assigns virtual addresses, maintains an address-to-locator mapping table, and provides bootstrap information for overlay participants.

Trust Handshake: A protocol exchange through which two agents establish a bilateral trust relationship with explicit mutual consent.

3. Problem Description

3.1. Agent Identity Is Coupled to Infrastructure

In current practice, agents are identified by URLs, DNS names, or API endpoints --- all of which are tied to the infrastructure hosting the agent, not to the agent itself. When an agent migrates to a different host, changes cloud provider, or restarts behind a different NAT binding, its identity changes. There is no stable identifier that follows an agent across these transitions.

This is analogous to the identity/locator conflation problem in IP networking, which motivated the Locator/ID Separation Protocol (LISP) [RFC9300]. In LISP, Endpoint Identifiers (EIDs) are separated from Routing Locators (RLOCs) so that an endpoint’s identity is independent of its network attachment point. Agents need the same separation: a permanent identity that is independent of the transient infrastructure hosting them.

The A2A protocol [A2A] identifies agents via "Agent Cards" served at well-known HTTPS URLs. The Agent URI scheme [I-D.narvaneni-agent-uri] proposes agent:// URIs but still requires DNS-resolvable endpoints. Both approaches require the agent to maintain a stable, publicly reachable web endpoint --- a requirement that excludes agents running on edge devices, behind NAT, or in ephemeral compute environments.

3.2. No Peer-to-Peer Communication Without Web Infrastructure

Both MCP [MCP] and A2A [A2A] require HTTP endpoints for communication. This means every agent must either have a publicly routable IP address or be fronted by a reverse proxy, load balancer, or API gateway. For two agents behind NAT to communicate, at least one must provision web infrastructure as an intermediary.

NAT traversal is a solved problem for specific domains: WebRTC handles it for browsers (at the cost of ICE/DTLS-SRTP/SDP negotiation complexity), and WireGuard [WIREGUARD] handles it for VPN tunnels. But no existing protocol provides NAT traversal specifically designed for agent-to-agent communication, with agent-native addressing and trust semantics.

An estimated 88% of real-world network environments involve some form of NAT. Agents running on laptops, IoT devices, edge servers, and mobile phones cannot participate in HTTP-based agent protocols without significant infrastructure provisioning.

3.3. No Agent-Native Trust Model

Existing trust models were designed for different participants:

* TLS: Trust is anchored in Certificate Authorities. Agents would need to obtain and manage X.509 certificates, adding operational complexity disproportionate to many agent interactions.

* SSH: Trust-on-first-use (TOFU) assumes a human operator who can verify a host key fingerprint. Autonomous agents have no human in the loop.

* OAuth/OIDC: Designed for user-to-service authorization, not peer-to-peer agent trust. Requires an authorization server as a trusted third party.

None of these models provide bilateral consent --- the property that both parties must explicitly agree before a communication relationship is established. For autonomous entities that may be operated by different organizations, bilateral consent is a natural trust primitive: neither agent should be reachable by the other until both have agreed.

3.4. No Lightweight Transport for Agent Streams

TCP and QUIC [RFC9000] are general-purpose transports optimized for web traffic patterns (request-response, large transfers, multiplexed streams). Agent communication patterns differ:

* Many agents exchange small, frequent messages (status updates, task delegations, sensor readings) where connection setup overhead dominates.

* Agents often maintain long-lived bidirectional streams for event-driven architectures, where TCP's head-of-line blocking is problematic.

* Agents may need port-based service multiplexing (echo on one port, task submission on another, events on a third) --- a concept that exists in TCP/UDP but has no equivalent in HTTP-based agent protocols.

While QUIC addresses head-of-line blocking through multiplexed streams, it does not provide agent addressing, discovery, or trust semantics. A transport designed for agents could provide these as built-in capabilities rather than requiring them to be layered on top.

3.5. Privacy Gaps in Agent Discovery

Current agent discovery mechanisms are designed for visibility:

* A2A Agent Cards are intended to be publicly discoverable at well-known URLs.

* DNS-SD and mDNS broadcast service availability to all listeners on a network segment.

* HTTP-based service registries typically allow any authenticated client to enumerate all registered services.

For agents, the default should be the opposite. An agent's existence and capabilities should not be disclosed to parties that have not been explicitly authorized. Mass enumeration of agent endpoints creates attack surface (reconnaissance for exploitation) and privacy risks (mapping an organization's agent infrastructure).

A privacy-by-default discovery model --- where agents are invisible until they explicitly opt in to specific peer relationships --- has no equivalent in current standards.

3.6. No Multi-Tenant Network Isolation

Current agent protocols assume flat, single-tenant deployments where all agents share the same namespace and trust domain. In practice, organizations deploy multiple agent teams serving different departments, projects, or customers. These teams need isolation:

* Agents in one project should not observe or interfere with agents in another.

* Administrative control (who can join a network, what ports are accessible, who can modify policy) should be scoped per network, not global.

* Compliance requirements (SOC 2, GDPR, HIPAA) demand audit trails recording who did what, when, and to which network --- at the infrastructure layer, not bolted on at the application layer.

No existing agent protocol or draft addresses multi-tenancy, role-based access control, or per-network policy enforcement. The closest analog is cloud VPC isolation, but VPCs operate at the IP layer and require cloud provider infrastructure. Agent networks need the same isolation primitives at the overlay layer, independent of the underlying cloud or network topology.

4. Requirements for a Solution

Based on the problems identified above, a network-layer infrastructure for agent communication should satisfy the following requirements:

4.1. Virtual Addressing

REQ-1:
Agents MUST receive stable virtual addresses that are independent of their underlying IP address, network attachment point, and hosting infrastructure.

REQ-2:
The addressing scheme MUST support hierarchical grouping (e.g., network or topic-based segmentation) to enable scoped communication boundaries.

4.2. NAT Traversal

REQ-3:
The system MUST provide automatic NAT traversal without requiring manual configuration of port forwarding, firewall rules, or relay proxies by the agent operator.

REQ-4:
NAT traversal MUST support direct peer-to-peer communication where possible, with transparent relay fallback when direct communication is not achievable.

4.3. Bilateral Trust Model

REQ-5:
Communication between agents MUST require explicit bilateral consent. Neither agent should be reachable by the other until both have agreed to establish a trust relationship.

REQ-6:
Trust relationships MUST be revocable. Revoking trust MUST immediately prevent further communication.

4.4. Lightweight Encrypted Transport

REQ-7:
The transport MUST provide reliable, ordered byte stream delivery (TCP-equivalent) and unreliable datagram delivery (UDP-equivalent) over the overlay.

REQ-8:
Encryption MUST be enabled by default for all data in transit, with no opt-in required from the agent developer.

REQ-9:
The transport MUST support port-based service multiplexing, allowing an agent to expose multiple services on different virtual ports.

4.5. Privacy-by-Default Discovery

REQ-10:

Agents MUST be private by default. An agent's virtual address, physical locator, and capabilities MUST NOT be disclosed to parties without an established trust relationship or shared group membership.

REQ-11:
It MUST be possible to establish trust with a private agent without first knowing its physical network location (i.e., via a trusted relay or rendezvous mechanism).

4.6. Multi-Tenant Isolation

REQ-12:
The system MUST support isolated network segments with independent membership control, role-based access (at minimum: owner, administrator, and member roles), and per-network policy enforcement. Operations within one network MUST NOT affect agents in another network.

4.7. Audit and Compliance

REQ-13:
The system MUST provide an audit trail recording security-relevant operations including node registration and deregistration, trust relationship changes, network membership modifications, role assignments, and policy updates. Audit records MUST include timestamps, actor identifiers, and old/new values for state mutations.

5. Existing Approaches and Gaps

5.1. MCP (Model Context Protocol)

MCP [MCP] standardizes the interface between AI models and external tools/resources. It uses JSON-RPC over HTTP with Server-Sent Events for streaming. MCP addresses agent-to-tool communication, not agent-to-agent communication, and provides no network-layer capabilities. It assumes agents can reach tool servers via HTTP.

5.2. A2A (Agent-to-Agent Protocol)

A2A [A2A] defines a protocol for agent interoperability: Agent Cards for discovery, task lifecycle management, and multimodal message exchange. A2A operates entirely over HTTP/HTTPS. It provides no NAT traversal, no overlay addressing, no built-in encryption beyond TLS, and no bilateral trust model. It assumes agents have reachable HTTP endpoints.

5.3. WebRTC

WebRTC provides peer-to-peer communication with NAT traversal via ICE, encryption via DTLS-SRTP, and data channels via SCTP. However, WebRTC was designed for browser-based audio/video communication. Its complexity (ICE candidate gathering, SDP offer/answer negotiation, DTLS-SRTP key exchange) is disproportionate for agent message exchange. WebRTC also lacks agent-specific concepts like virtual addressing, bilateral trust, and privacy-by-default discovery.

5.4. QUIC

QUIC [RFC9000] provides a modern transport with multiplexed streams, built-in encryption, and reduced connection setup latency. QUIC addresses transport-layer concerns but does not provide overlay addressing, agent identity, NAT traversal coordination, trust management, or discovery. It is a potential underlay transport for an agent overlay, not a complete solution.

5.5. libp2p

libp2p [LIBP2P] is a modular networking stack developed for decentralized applications, particularly in the blockchain ecosystem. It provides peer identity (via cryptographic keypairs), NAT traversal, and transport multiplexing. libp2p is the closest existing system to the requirements stated above. However, it uses unstructured peer IDs (not hierarchical addresses), is heavyweight (large dependency tree), is oriented toward content-addressed distributed systems rather than agent communication patterns, and lacks built-in bilateral trust or privacy-by- default semantics.

5.6. WireGuard

WireGuard [WIREGUARD] provides encrypted point-to-point tunnels with excellent performance. It uses Curve25519 for key exchange and ChaCha20- Poly1305 for encryption. WireGuard establishes tunnels between known peers with pre-shared public keys --- it does not provide dynamic discovery, agent addressing, or trust negotiation. It is a VPN, not an agent network.

5.7. LISP

The Locator/ID Separation Protocol [RFC9300] [RFC9301] separates endpoint identity from network location, providing a conceptual precedent for agent addressing. LISP’s EID-to-RLOC mapping system is architecturally similar to an agent registry that maps virtual addresses to physical locators. However, LISP operates at the IP layer for routing optimization, not at the application layer for

agent communication. It does not provide agent-specific trust models, privacy semantics, or built-in services.

5.8. AGTP (Agent Transfer Protocol)

AGTP [I-D.hood-independent-agtp] proposes a dedicated application-layer protocol for AI agent traffic with agent-native intent methods (QUERY, SUMMARIZE, DELEGATE, COLLABORATE). It correctly identifies that MCP and A2A are messaging-layer constructs that do not address the transport problem. However, AGTP operates over QUIC or TCP/TLS at the application layer --- it defines what agents say, not how they reach each other. It provides no overlay addressing, no virtual network primitives, no NAT traversal, and no multi-tenant isolation.

5.9. ATP (Agent Transfer Protocol)

ATP [I-D.li-atp] defines a two-tier architecture where agents connect to ATP servers, with DNS-based service discovery via SVCB records. It supports asynchronous messaging, synchronous request/response, and event-driven streaming. Like AGTP, ATP operates at the application layer and requires server infrastructure as an intermediary --- the same dependency on centralized endpoints that HTTP-based protocols impose. It does not address overlay networking, NAT traversal for serverless agents, or privacy-by-default semantics.

5.10. AIP (Agent Identity Protocol)

AIP [I-D.prakash-aip] defines verifiable, delegable identity for AI agents using Invocation-Bound Capability Tokens (IBCTs) with Ed25519 signatures and Biscuit-based delegation chains. AIP addresses the identity problem (REQ-1) with a cryptographically strong approach. However, it focuses exclusively on identity and authorization --- it provides no transport, addressing, or network-layer primitives. AIP's identity tokens could complement an overlay network that provides the missing transport substrate.

5.11. CATALIST Coordination

The CATALIST BoF at IETF 125 [I-D.yao-catalist-problem-space-analysis] surveyed the agent protocol landscape and began scoping what IETF should standardize. The problem space analysis identified categories including agent identity, discovery, communication, and governance. Notably, agent network routing [I-D.du-catalist-routing-considerations] defines forwarding based on Agent ID, Gateway ID, and Skill --- concepts that overlap with virtual addressing and service multiplexing. The ACIP framework [I-D.eckert-catalist-acip-framework] proposes agent-aware network infrastructure drawing from overlay/underlay designs. These

efforts validate the need for network-layer agent infrastructure but have not yet produced a concrete protocol specification.

6. Security Considerations

A network-layer infrastructure for agents introduces security considerations beyond those of traditional overlay networks:

Centralized Registry: A registry that assigns addresses and maintains locator mappings is a trusted third party. Compromise of the registry could allow address hijacking, locator spoofing, or metadata harvesting. The registry should support authentication, access control, and replication for high availability.

Overlay Header Metadata: Even with payload encryption, overlay packet headers may expose source and destination virtual addresses, port numbers, and packet sizes. Traffic analysis on the overlay is possible even when the underlay is encrypted.

Trust Model Assumptions: A bilateral trust model assumes that agents can make informed consent decisions. If an agent’s trust logic is compromised (e.g., by adversarial prompt injection), it may approve trust relationships it should reject. The trust model provides a mechanism, not a policy --- the security of trust decisions depends on the agent’s reasoning capability.

Key Management: Overlay encryption requires key exchange between peers. Anonymous key exchange (without identity binding) is vulnerable to man-in-the-middle attacks. Authenticated key exchange requires a mechanism to distribute and verify public keys, which depends on the registry’s integrity.

Multi-Tenant Control Plane: A multi-tenant registry introduces additional attack surface. Per-network role-based access control must be enforced consistently to prevent privilege escalation (e.g., a member modifying network policy). Admin token authentication for privileged operations must use constant-time comparison to prevent timing attacks. Audit trails must be tamper-evident and persist across service restarts to support forensic analysis (see also [I-D.sharif-agent-audit-trail]).

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

8.2. Informative References

[A2A] Google, "Agent-to-Agent Protocol", 2025, <https://google.github.io/A2A/>.

[I-D.du-catalist-routing-considerations] Du, Z., "Routing Considerations in Agentic Network", 2026, <https://datatracker.ietf.org/doc/draft-du-catalist-routing-considerations/>.

[I-D.eckert-catalist-acip-framework] Eckert, T., "Framework for Agent Communications Internet Protocol (ACIP)", 2026, <https://datatracker.ietf.org/doc/draft-eckert-catalist-acip-framework/>.

[I-D.hood-independent-agtp] Hood, C., "Agent Transfer Protocol (AGTP)", 2026, <https://datatracker.ietf.org/doc/draft-hood-independent-agtp/>.

[I-D.li-atp] Li, Y., "Agent Transfer Protocol (ATP)", 2026, <https://datatracker.ietf.org/doc/draft-li-atp/>.

[I-D.narvaneni-agent-uri] Narvaneni, S., "Agent URI Scheme", 2025, <https://datatracker.ietf.org/doc/draft-narvaneni-agent-uri/>.

[I-D.nemethi-aid-agent-identity-discovery] Nemethi, B., "Agent Identity and Discovery (AID)", 2026, <https://datatracker.ietf.org/doc/draft-nemethi-aid-agent-identity-discovery/>.

[I-D.ni-wimse-ai-agent-identity]
Ni, Y. and C. P. Liu, "WIMSE Applicability for AI Agents", 2026, <https://datatracker.ietf.org/doc/draft-ni-wimse-ai-agent-identity/>.

[I-D.prakash-aip]
Prakash, S., "Agent Identity Protocol (AIP)", 2026, <https://datatracker.ietf.org/doc/draft-prakash-aip/>.

[I-D.rosenberg-aiproto-framework]
Rosenberg, J., "A Framework for AI Protocols", 2025, <https://datatracker.ietf.org/doc/draft-rosenberg-aiproto-framework/>.

[I-D.sharif-agent-audit-trail]
Sharif, R., "Agent Audit Trail", 2026, <https://datatracker.ietf.org/doc/draft-sharif-agent-audit-trail/>.

[I-D.yao-catalist-problem-space-analysis]
Zhou, Y. and K. Yao, "Problem Space Analysis of AI Agent Protocols in IETF", 2026, <https://datatracker.ietf.org/doc/draft-yao-catalist-problem-space-analysis/>.

[I-D.zyyhl-agent-networks-framework]
Yao, Z., "A Framework for Agent Networks", 2025, <https://datatracker.ietf.org/doc/draft-zyyhl-agent-networks-framework/>.

[LIBP2P] Protocol Labs, "libp2p: A Modular Network Stack", 2023, <https://libp2p.io/>.

[MCP] Anthropic, "Model Context Protocol", 2024, <https://modelcontextprotocol.io/>.

[RFC7364] Narten, T., Ed., Gray, E., Ed., Black, D., Fang, L., Kreeger, L., and M. Napierala, "Problem Statement: Overlays for Network Virtualization", RFC 7364, DOI 10.17487/RFC7364, October 2014, <https://www.rfc-editor.org/rfc/rfc7364>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9300] Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, Ed., "The Locator/ID Separation Protocol (LISP)", RFC 9300, DOI 10.17487/RFC9300, October 2022, <https://www.rfc-editor.org/rfc/rfc9300>.

[RFC9301] Farinacci, D., Maino, F., Fuller, V., and A. Cabellos, Ed., "Locator/ID Separation Protocol (LISP) Control Plane", RFC 9301, DOI 10.17487/RFC9301, October 2022, <https://www.rfc-editor.org/rfc/rfc9301>.

[WIREGUARD] Donenfeld, J. A., "WireGuard: Next Generation Kernel Network Tunnel", 2017, <https://www.wireguard.com/papers/wireguard.pdf>.

Appendix A. Acknowledgments

The author thanks the participants of the IETF AI protocols discussions for their contributions to understanding the agent communication landscape.

Author’s Address

Calin Teodor
Vulture Labs
Email: teodor@vulturelabs.com