Network Management Research Group
Internet-Draft
Intended status: Informational
Expires: 19 October 2026

C. Feng
Ruijie Networks

17 April 2026

Agentic Intent Network (AIN): A Routing-Based Architecture for AI Agent Coordination at Scale

draft-feng-nmrg-ain-architecture-00

Abstract

The rapid proliferation of autonomous AI agents across enterprise and Internet-scale deployments creates a structural challenge that existing agent frameworks cannot address: how to enable any agent to discover and invoke any other agent’s capabilities without pre-established bilateral integration, across organizational boundaries, at Internet scale. This document presents the Agentic Intent Network (AIN) as an architecture-level model for open, heterogeneous, dynamically evolving multi-agent coordination. It defines problem drivers, architectural and underlay requirements, architectural components, design invariants, scope boundaries, and a research agenda for the NMRG. Engineering protocol details are intentionally out of scope.

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 19 October 2026.

Copyright Notice

Table of Contents

1. Introduction

The deployment of autonomous AI agents is transitioning from isolated experimental systems to large-scale production environments. Individual enterprises deploy tens to hundreds of specialized agents; Internet-scale platforms may eventually host millions. Each agent encapsulates bounded capabilities; collaboration among agents is an engineering necessity for accomplishing complex tasks, not an optional feature.

The architecture problem is not local orchestration. It is global, open coordination: how can any agent discover and invoke any other agent’s capabilities without pre-established bilateral integration, and across heterogeneous frameworks and organizational boundaries? At scale, the pairwise integration cost grows quadratically, quickly overwhelming any system’s operational budget. At organizational boundaries, the trust and deployment assumptions embedded in today’s frameworks break down: an agent operated by one enterprise cannot discover or invoke an agent operated by another without custom, manually maintained integration agreements.

The name "Agentic Intent Network" reflects the three structural commitments of the architecture. "Agentic" denotes that the participating entities are autonomous agents -- systems capable of independent reasoning and action -- rather than passive endpoints or simple services. "Intent" denotes that coordination is expressed as structured, capability-oriented requests, rather than as direct procedure calls or point-to-point messages; an intent captures what the originating agent wants accomplished, abstracted from which specific handler will accomplish it. "Network" denotes that the coordination substrate is organized as a routed network -- with a data plane, a control plane, addressing, and forwarding -- rather than as a registry, a broker, or an orchestration engine. Together, the name captures the core architectural claim: routing-based infrastructure is the appropriate model for open, scalable, heterogeneous agent coordination.

AIN proposes a routing-based answer, applying the structural logic of packet networking to inter-agent coordination. This document is an architecture description in the IRTF style: it specifies what AIN is, why it is needed, and what research questions it opens. Engineering protocol mechanisms and algorithmic details are intentionally handled in companion design-consideration documents.

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.

Intent:
: A structured representation of a task request, expressed as a capability class identifier and associated constraints.

Intent Datagram:
: The self-contained coordination unit carrying an intent, analogous to an IP datagram.

Intent Class:
: A hierarchical identifier used as the routing key for an Intent Datagram.

Intent Class OID (IC-OID):
: A globally unique, structured identifier for an Intent Class, suitable for machine processing and prefix aggregation.

Handler Instance Identifier (HID):
: A globally unique identifier for a specific Intent Handler instance.

Intent Router:
: A deterministic forwarding component that routes Intent Datagrams based on capability routing tables.

Intent Dispatcher:
: An application entity that decomposes compound intents into sub-intents and emits routable datagrams.

Intent Handler:
: An application entity that executes atomic intents.

Capability Advertisement (CAP):
: A control-plane message by which an Intent Handler announces capabilities.

Agent Domain:
: A collection of registered entities within a management and policy boundary, analogous to an IP AS.

Semantic Substrate:
: Shared naming, capability-description, and matching semantics consumed by AIN control-plane functions.

Capability Routing Table (CRT):
: A mapping from IC-OID prefixes to ranked candidate handler sets.

3. Problem Statement

3.1. The Multi-Agent Coordination Gap

Current AI ecosystems are rich in model capabilities and local agent frameworks, but poor in open coordination primitives. Systems can orchestrate agents that are already inside the same framework or trust domain, yet lack a common architecture for Internet-scale, cross-domain interaction among independently operated agents.

Consider a concrete scenario: an enterprise deploys a customer-service agent (Framework A) that needs to invoke a compliance-checking agent (Framework B, operated by a different business unit) and a logistics-status agent (a third-party API wrapper). Today, each of these connections requires a separate, manually negotiated integration: agreed-upon API schemas, bilateral trust configuration, and per-link lifecycle management. None of the three agents can discover the others dynamically; each new agent added to the ecosystem requires O(existing agents) new integration agreements. This is not an implementation deficiency -- it is a structural absence of shared coordination infrastructure.

This creates a coordination gap: autonomous agents can reason and act locally, but cannot reliably discover, select, and invoke remote capabilities in a neutral, scalable, framework-independent way.

3.2. The O(N^2) Bilateral Integration Problem

Without shared routing infrastructure, N agents require up to N(N-1)/2 bilateral integrations to interoperate. Each integration entails interface mapping, trust negotiation, lifecycle management, and operational monitoring.

Engineering mitigations -- common API conventions, shared SDKs, service registries -- reduce implementation friction but do not change the underlying structure: each pair of agents that must interoperate still requires explicit, maintained coordination state. As agent populations grow and cross organizational boundaries, the per-pair cost cannot be engineered away; it must be eliminated architecturally.

The Internet solved an identical structural problem for host interconnection. Before shared routing infrastructure, connecting N hosts required bilateral reachability agreements. IP eliminated this by introducing a shared forwarding substrate: any host can reach any other host without pre-arrangement, because routing state is maintained by the network rather than by endpoint pairs. AIN applies the same logic to agent coordination: per-agent integration overhead is reduced from O(N) to O(1) by introducing shared capability routing infrastructure.

3.3. Limitations of Existing Frameworks

Existing frameworks and protocols (e.g., AutoGen [WU2023], LangGraph, A2A [A2A2025], MCP [MCP2024]) provide useful local mechanisms but do not, by themselves, satisfy the full set of open architectural needs.

AutoGen and LangGraph are intra-framework orchestration systems. They excel at coordinating agents that are co-deployed within the same runtime and trust boundary, but provide no mechanism for an agent in one framework to dynamically discover or invoke an agent in another. Interoperability across framework boundaries requires custom bridging code, reproducing the bilateral integration problem at the framework level.

MCP [MCP2024] addresses the connection between LLM-based agents and external tools or data sources. Its scope is LLM-to-resource integration, not agent-to-agent routing. MCP does not define how an agent discovers which other agents exist, selects among candidates, or routes a request across multiple coordination hops.

A2A [A2A2025] defines an interaction protocol for pairs of agents. It is analogous to HTTP: it standardizes the exchange format for a single interaction, but it does not address discovery, capability routing, or multi-hop forwarding. Two agents using A2A must still know each other’s endpoints in advance; the protocol does not provide the directory or routing substrate needed to find them.

They are typically optimized for one or more of:

* single-framework ecosystems,
* pre-coordinated trust and deployment assumptions,
* bounded organizational scope, and
* static or manually curated integration surfaces.

These assumptions limit their ability to serve as a shared, Internet-scale coordination substrate for heterogeneous agents. AIN is not a replacement for these frameworks; it is the missing coordination layer above and between them.

3.4. Why This Is a Network Management Research Problem

First, the AIN control plane introduces new convergence challenges that have no direct precedent in IP routing. Capability advertisements carry semantic identifiers (IC-OIDs) rather than topological addresses. Convergence correctness -- the guarantee that every Intent Router's CRT eventually reflects the true handler population without loops or black holes -- must be established over a semantic identifier space whose structure differs fundamentally from IP prefixes. Defining convergence conditions, bounding convergence time, and proving loop-freedom under dynamic handler populations are open research problems of the kind NMRG is positioned to address.

Second, the AIN coordination substrate is itself a networked system that requires operational management. As agent populations scale, operators need telemetry on capability routing behavior, anomaly detection for malformed or malicious CAPs, policy controls on which agents may advertise which capabilities, and mechanisms for graceful handler failure and capability withdrawal. These are directly analogous to the network management functions that NMRG has studied for IP infrastructure, now applied to a new class of coordination plane.

Third, practical AIN deployment is coupled with the underlying network and system environment in ways that require joint analysis. CAP propagation generates control-plane traffic whose volume and burstiness depend on agent population dynamics. Intent Datagram forwarding latency is bounded by both AIN routing behavior and underlay transport characteristics. Resource allocation across Intent Routers and Handlers must account for both coordination overhead and execution workload. These coupling effects are deployment-engineering questions that sit squarely within the scope of [NMRG-AI-DEPLOY] and related NMRG work.

4. Requirements

4.1. Design Principles

The requirements in this section are not assembled as a feature wishlist. They are derived from three architectural principles that have proven effective in building open, scalable networked systems, applied here to the agent coordination domain.

Principle 1: End-to-End Argument. Saltzer, Reed, and Clark [SRC1984] established that functions requiring application-specific knowledge should be implemented at the endpoints, not in the network substrate. Applied to AIN: task execution, semantic reasoning, and result interpretation belong at Handlers and Dispatchers. The routing fabric must remain thin, deterministic, and execution-agnostic. This principle drives R_local and the design invariants of Networking-Execution Separation and Payload Opacity (Section 5.6).

Principle 2: Narrow Waist Design. The Internet's scalability rests on a minimal common interface -- the IP datagram -- that decouples heterogeneous link technologies below from heterogeneous applications above. AIN adopts the same strategy: the Intent Datagram and IC-OID form a narrow waist enabling any agent framework to participate without requiring coordination fabric redesign. This principle drives R_open and R_hetero.

Principle 3: Decentralized Scalability. Centralized coordination introduces bottlenecks, single points of failure, and policy coupling that limit scale. Distributed routing protocols, from OSPF to BGP, demonstrate that local forwarding decisions based on distributed state can achieve global reachability without central oracles. AIN applies this to capability routing. This principle drives R_local and the bounded-convergence constraint in R_dynamic.

A fourth consideration -- Policy/Mechanism Separation -- informs the overall design: routing mechanisms are specified to be generic and stable; capability-matching semantics and selection policies are explicitly layered above the forwarding core, allowing them to evolve independently without requiring changes to the routing fabric.

4.2. Architectural Requirements

R_open (Open Participation):
: Any agent, regardless of framework, language, or deployment environment, MUST be able to register capabilities and invoke others without bilateral pre-registration. Derivation: Follows from the Narrow Waist principle. Requiring bilateral pre-registration reproduces the O(N^2) integration cost documented in Section 3.2. A shared registration and routing substrate eliminates this cost structurally, as IP eliminated the need for pairwise host reachability agreements.

R_local (Local Decision-Making):
: Each forwarding decision MUST be made locally using datagram-visible and locally maintained state; no centralized per-request routing oracle is assumed. Derivation: Follows from the End-to-End Argument and Decentralized Scalability. A centralized routing

oracle would become a bottleneck, a single point of failure, and a policy chokepoint -- reproducing architecturally the same problems that distributed routing protocols were designed to eliminate.

R_hetero (Heterogeneous Capability Matching):
The architecture MUST support heterogeneous handler types (LLM-based, deterministic, legacy-wrapped) without requiring routing-fabric redesign. Derivation: Follows from the Narrow Waist principle and Policy/Mechanism Separation. The routing fabric must remain agnostic to execution substrate. Capability-matching semantics reside in the Semantic Substrate (Section 5.4), not in forwarding components, allowing handler technology to evolve without architectural disruption.

R_dynamic (Dynamic Capability Discovery):
The architecture MUST support dynamic capability additions, updates, and withdrawals with bounded convergence behavior. Derivation: Follows from Decentralized Scalability and the operational reality of large-scale deployments. Agent populations are not static: handlers are deployed, updated, and retired continuously. The routing fabric must accommodate this dynamism while providing convergence guarantees sufficient for operational stability -- a direct analogue of liveness and loop-freedom requirements in IP routing protocols.

4.3. Underlay Requirements

The architectural requirements above define what the AIN coordination layer must provide. They depend, in turn, on two properties of the underlay transport fabric. These are stated as requirements on the underlay, not as design choices of the AIN architecture itself, consistent with the End-to-End Argument: AIN does not prescribe how reachability is achieved, only that it must be available.

R_u1 (Handler Reachability):
Any Intent Router MUST be able to deliver an Intent Datagram to any registered Intent Handler.

R_u2 (Callback Reachability):
Any Intent Handler MUST be able to deliver callback responses to the Originator endpoint indicated by datagram metadata.

This document intentionally does not mandate specific implementation patterns for satisfying R_u1 and R_u2. Overlay networks, service meshes, and conventional IP transport are all candidate mechanisms.

5. AIN Architecture

5.1. Architecture Overview

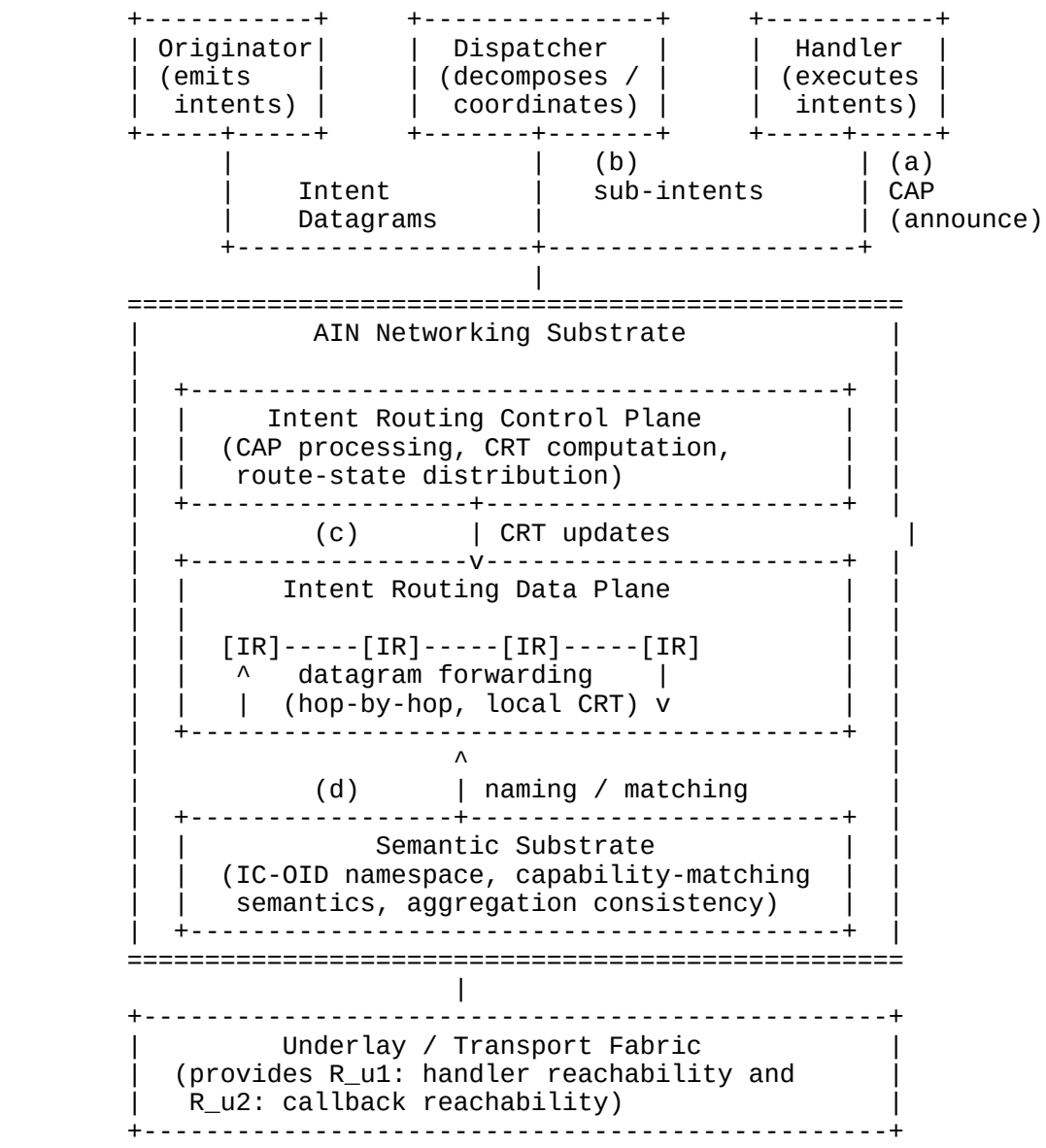
AIN is organized into three architectural parts:

1. (A) Networking: underlay/transport fabric, Intent Routing data plane, Intent Routing control plane, and Semantic Substrate.

2. (B) Application Entities: Originator, Dispatcher, and Handler roles.

3. (C) Runtime Dependencies: execution runtime and compute infrastructure used by application entities.

Figure 1 illustrates the layered structure and the principal interactions among components.



IR = Intent Router
CAP = Capability Advertisement
CRT = Capability Routing Table

Figure 1: AIN Architecture -- Layers and Principal Entities

The figure is read in two dimensions: vertical layering represents the separation of concerns, and the lettered interactions (a)--(d) represent the two principal information flows through the architecture.

Layering. Application Entities (top) sit above the AIN Networking Substrate and consume its services; they are not part of the forwarding core. This boundary enforces the Networking-Execution Separation invariant (Section 5.6): an Intent Router forwards datagrams but never executes the tasks they carry. Within the Networking Substrate, the Semantic Substrate occupies the base position because it provides the naming and matching semantics that the control plane depends on; it is not a forwarding layer. The Underlay/Transport Fabric is architecturally external: AIN states what reachability properties it requires (R_u1, R_u2) but does not prescribe how they are satisfied.

Control flow (interactions (a) and (c)). When a Handler registers or updates its capabilities, it emits a Capability Advertisement (CAP) into the networking substrate (a). The Intent Routing Control Plane processes incoming CAPs, applies matching semantics from the Semantic Substrate, and computes or updates entries in the Capability Routing Table (CRT) at each Intent Router (c). This is the AIN analogue of a routing protocol: CAP propagation drives distributed route-state convergence, so that every Intent Router eventually holds a CRT reflecting the current handler population.

Data flow (interactions (b) and forwarding in the data plane). An Originator emits an Intent Datagram carrying an IC-OID in its routing header. A Dispatcher may decompose a compound task into sub-intents and emit multiple datagrams (b). Each Intent Router performs a local CRT lookup on the IC-OID, selects a next hop from the ranked candidate set, and forwards the datagram -- without consulting any central oracle and without inspecting the payload (Payload Opacity, Section 5.6). Forwarding continues hop-by-hop until the datagram reaches a Handler that accepts and executes the intent. The Handler returns results via a callback path to the Originator endpoint identified in the datagram metadata.

Semantic Substrate (interaction (d)). The Semantic Substrate is consumed by the control plane, not the data plane. It provides the shared vocabulary that makes IC-OID prefix aggregation meaningful and capability matching consistent across heterogeneous handler types. Its role is analogous to the addressing and naming conventions that make IP prefix aggregation tractable: without shared semantics, CAP processing would require per-handler bespoke logic and could not scale.

Component	Responsibility	IP Analogy
Underlay/ Transport Fabric	End-to-end reachability under R_u1 and R_u2	Physical + Link
Intent Routing Data Plane	Stateless forwarding of intent datagrams	IP data plane
Intent Routing Control Plane	Capability advertisement and route-state maintenance	IP control plane
Semantic Substrate	Naming/matching semantics	Addressing
Application Entities	Intent origination, dispatch, and execution	End hosts

Table 1: AIN Architecture Overview (Conceptual)

5.2. Intent Datagram

The Intent Datagram is the fundamental coordination unit. It is conceptually defined as a triple:

D = <H, P, M>

where H is a routing header (intent class and routing-relevant fields), P is payload (opaque to routing), and M is metadata (e.g., correlation and callback information).

5.3. Intent Routing

Intent Routing separates:

* Data plane: deterministic per-datagram forwarding based on local route state.

* Control plane: capability advertisement processing and route-state computation/distribution.

This separation mirrors core Internet architectural practice and is central to AIN scalability and explainability.

5.4. Semantic Substrate

The Semantic Substrate defines shared capability-description and identifier semantics used by control-plane functions. It is a substrate for naming, matching, and aggregation consistency; it is not treated as an independent packet-forwarding layer.

5.5. Application Entities

AIN distinguishes three application-level roles:

* Originator: emits intents.

* Dispatcher: decomposes/coordinates compound tasks.

* Handler: executes atomic capabilities.

These roles are above the networking substrate and consume AIN services; they are not part of the forwarding core. A single physical deployment unit MAY simultaneously fulfill multiple roles (e.g., acting as both Dispatcher and Handler for different intents). Role assignment is decoupled from entity implementation, allowing flexible deployment topologies without requiring changes to the routing fabric or the Semantic Substrate.

5.6. Design Invariants

AIN relies on four architectural invariants:

1. Networking-Execution Separation: forwarding components do not perform task execution.

2. Hop-by-Hop Locality: forwarding is local and incremental.

3. Payload Opacity: forwarding is based on routing-relevant fields, not payload semantics.

4. Datagram Self-Containment: the datagram carries sufficient routing context for forwarding decisions.

5.7. Structural Correspondence with IP Networking

AIN intentionally mirrors key Internet architectural primitives:

* Intent Datagram ~ IP datagram

* Intent Router ~ IP router

* CRT ~ forwarding table

* Agent Domain ~ AS

* CAP propagation/control ~ routing control behavior

This correspondence is structural, not literal protocol reuse.

5.8. Relationship to Intent-Based Networking

[RFC9315] defines IBN intents as declarative goals for network behavior management. AIN intents are routable requests for AI-agent capability invocation. The two are complementary: IBN concerns network operation goals; AIN concerns inter-agent coordination routing.

5.9. Scope Boundaries

AIN is designed for open, dynamic, cross-domain agent coordination at scale. It is not appropriate for all agent interaction scenarios. In particular, AIN is not the intended solution for:

* Hard real-time tasks, where coordination latency introduced by capability routing is architecturally unacceptable.

* Fully static and deterministic workflows, where all handler assignments are known and fixed at design time, making dynamic capability discovery unnecessary.

* Closed single-framework deployments, where all participating agents share the same runtime and trust boundary and no cross-domain coordination is required.

Explicitly defining these boundaries is consistent with the End-to-End Argument: functions that do not benefit from routing-based indirection should not be routed. Recognizing these non-goals increases architectural clarity and helps implementers identify when AIN provides structural value.

6. Relationship to Existing NMRG Work

6.1. Relationship to draft-irtf-nmrg-ai-challenges

AIN adds a coordination-layer perspective to AI/network-management challenges, including semantic-route convergence, capability authenticity, and explainable multi-agent routing behavior.

6.2. Relationship to draft-irtf-nmrg-ai-deploy

AIN has deployment implications for capability advertisement traffic, convergence dynamics, and coupling between agent-coordination control behavior and network/system deployment choices.

6.3. AIN as a Third Research Dimension

AIN frames a third dimension alongside:

* AI for network management, and
* network/system support for AI services.

The third dimension is networked infrastructure for AI-agent coordination itself.

7. Open Research Problems

7.1. Phase 1: Foundations

* Capability description language and matching semantics.
* Intra-domain capability-routing protocol design.
* Minimal interoperable Intent Datagram schema.
* Baseline identity and trust for capability claims.
* Formal convergence and loop-freedom over semantic identifiers.

7.2. Phase 2: Scaling

* Inter-domain capability-routing policy and aggregation.
* Multi-metric optimization under heterogeneous constraints.
* Stability analysis for feedback-driven route scoring.
* Cross-tier latency and resource budget allocation.

7.3. Phase 3: Ecosystem

* Governance of global IC-OID taxonomy roots and delegation.
* Security and adversarial robustness at ecosystem scale, including capability-claim fraud, routing-state poisoning, and intent privacy threats.

* Economic settlement and incentive alignment across domains.

8. Security Considerations

AIN introduces security concerns at two architectural levels.

Control-plane threats affect the integrity and availability of capability routing state. Capability-claim spoofing (a malicious entity advertising capabilities it does not possess) and routing-state poisoning (injection of malformed or adversarial CAPs to corrupt CRTs) are the primary risks. Semantic namespace abuse -- the registration of IC-OIDs intended to shadow or hijack legitimate capability classes -- represents a further control-plane attack surface.

Data-plane threats affect the confidentiality and availability of intent traffic. Privacy leakage of intent contents or originator identity and denial-of-service against routing or handler interfaces are the primary concerns.

Architecture-level mitigation directions include authenticated capability advertisements, protected control-plane exchanges, deployment of modern transport security, and explicit governance for identifier namespaces. Detailed threat modeling is identified as a Phase 3 research problem (Section 7.3).

9. IANA Considerations

This document requests no IANA actions.

10. References

10.1. 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/info/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/info/rfc8174>.

[RFC9315] Clemm, A., Ciavaglia, L., Granville, L., Ed., and J. Tantsura, "Intent-Based Networking - Concepts and Definitions", RFC 9315, DOI 10.17487/RFC9315, October 2022, <https://www.rfc-editor.org/info/rfc9315>.

10.2. Informative References

[A2A2025] Google DeepMind, "Agent2Agent (A2A) Protocol", 2025, <https://github.com/google-deepmind/agent2agent>.

[MCP2024] Anthropic, "Model Context Protocol", 2024, <https://modelcontextprotocol.io/>.

[NMRG-AI-CHALLENGES] IRTF NMRG, "Challenges for Network Automation with AI/ML Techniques", Work in Progress Internet-Draft, draft-irtf-nmrg-ai-challenges, 2024.

[NMRG-AI-DEPLOY] IRTF NMRG, "Network and System Considerations for Deploying AI Services", Work in Progress Internet-Draft, draft-irtf-nmrg-ai-deploy, 2024.

[RFC7575] Farrell, S. and H. Tschofenig, "Architectural Considerations in Smart Object Networking", RFC 7575, DOI 10.17487/RFC7575, July 2015, <https://www.rfc-editor.org/info/rfc7575>.

[SRC1984] Saltzer, J., Reed, D., and D. Clark, "End-to-End Arguments in System Design", ACM Transactions on Computer Systems Vol. 2, No. 4, pp. 277-288, November 1984, <https://doi.org/10.1145/357401.357402>.

[WU2023] Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., White, R., Burger, D., Lara, V., Foyer, A., and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation", arXiv 2308.08155, 2023.

Author’s Address

Chong Feng
Ruijie Networks
Email: fengchongllly@gmail.com