Independent Submission
Internet-Draft
Intended status: Standards Track
Expires: 28 May 2026

E. P. Nagulapalli
Servesys Labs

24 November 2025

AI-Native Network Protocol (AINP) for Semantic Agent Communication
draft-ainp-protocol-00

Abstract

This document specifies the AI-Native Network Protocol (AINP) version 0.1, a semantic communication protocol designed for intent exchange between AI agents. AINP replaces location-based routing with semantic routing, byte-stream delivery with intent delivery, and simple handshakes with multi-round negotiation. AINP enables agents to discover each other by capability rather than network location, negotiate terms autonomously, and exchange structured intents with cryptographic security.

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 28 May 2026.

Copyright Notice

Table of Contents

1. Introduction

Traditional network protocols (TCP/IP, HTTP, SMTP) were designed for reliable byte stream delivery between machines. AINP represents a paradigm shift: it is designed for semantic intent delivery between AI agents, with built-in understanding, negotiation, and adaptation.

AINP Phase 0.1 provides: - Wire format specification (JSON-LD + CBOR) - Message envelope structure with cryptographic signatures - Intent schemas for common agent interactions - Semantic address format with Decentralized Identifiers (DIDs) - Negotiation protocol for multi-agent consensus - Discovery and routing mechanisms

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.1.1. Definitions

Agent: An autonomous software entity capable of semantic understanding and intent exchange

Intent: A semantic representation of an agent’s goal, including embeddings and structured semantics

Semantic Address: An identity-based address using DIDs and capability descriptors

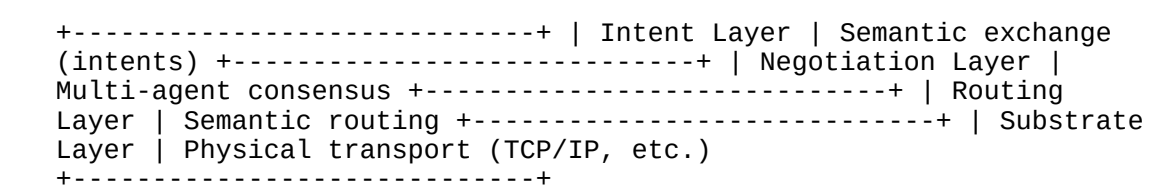
Negotiation: Multi-round protocol for establishing consensus on resources, terms, and capabilities

Capability: A semantic description of what an agent can do, including natural language and embeddings

Trust Vector: Multi-dimensional reputation score tracking reliability, honesty, competence, and timeliness

2. Architecture Overview

AINP consists of four layers:



Phase 0.1 runs as an overlay network on TCP/IP with WebSocket or HTTP/3 transport.

3. Wire Format

3.1. Encoding

AINP messages MUST support both: - *JSON-LD*: Human-readable, linked data format with @context for semantic interoperability - *CBOR*: Binary encoding ([RFC8949]) for efficient transmission

Implementations SHOULD negotiate encoding during handshake. JSON-LD is the default for Phase 0.1.

3.2. Message Envelope

All AINP messages MUST be wrapped in an envelope structure:

json { "version": "0.1.0", "msg_type": "INTENT", "id": "550e8400-e29b-41d4-a716-446655440000", "timestamp": 1728259400000, "ttl": 30000, "trace_id": "trace-abc123", "from_did": "did:key:z6Mk...", "to_did": "did:key:z6Mk...", "schema": "https://ainp.dev/schemas/intents/request-meeting/v1", "qos": { "urgency": 0.7, "importance": 0.8, "novelty": 0.1, "ethicalWeight": 0.5, "bid": 5 }, "payload": { ... }, "sig": "base64signature..." }

3.2.1. Envelope Fields

* version (string, REQUIRED): Protocol version, MUST be "0.1.0"
* msg_type (string, REQUIRED): Message type (see Section 3.3)
* id (string, REQUIRED): UUID v4 for message identification
* timestamp (number, REQUIRED): Unix epoch milliseconds
* ttl (number, REQUIRED): Time-to-live in milliseconds
* trace_id (string, REQUIRED): Distributed tracing UUID for thread tracking
* from_did (string, REQUIRED): Sender DID ([W3C.DID])
* to_did (string, OPTIONAL): Recipient DID (for direct addressing)
* to_query (object, OPTIONAL): Semantic query (alternative to to_did, see Section 3.5)
* schema (string, REQUIRED): JSON-LD context URI
* qos (object, REQUIRED): Quality of Service parameters (see Section 3.4)
* payload (object, OPTIONAL): Application payload (intent, negotiation, etc.)
* sig (string, REQUIRED): Ed25519 signature in base64 encoding

3.3. Message Types

AINP defines the following message types:

* ADVERTISE: Publish capabilities to discovery index
* DISCOVER: Query for agents by capability
* DISCOVER_RESULT: Discovery results
* NEGOTIATE: Multi-round negotiation
* INTENT: Send intent payload
* RESULT: Response with optional proof
* ERROR: Error response

3.4. Quality of Service Parameters

QoS parameters enable priority-based routing and resource allocation:

json { "urgency": 0.7, // 0-1, time sensitivity "importance": 0.8, // 0-1, impact magnitude "novelty": 0.1, // 0-1, information gain "ethicalWeight": 0.5, // 0-1, moral importance "bid": 5 // Token amount or credits (non-negative) }

Priority Calculation: Implementations SHOULD calculate message priority as:

priority = (urgency * w_urgency) + (importance * w_importance) + (novelty * w_novelty) + (ethicalWeight * w_ethical) adjusted_priority = priority + 0.5 * tanh(bid / bid_scale)

Where bid_scale is node-configurable (RECOMMENDED default: 10 credits).

Default Weights: - w_urgency = 0.3 - w_importance = 0.3 - w_novelty = 0.2 - w_ethical = 0.2

3.5. Capability Query

For semantic discovery, agents use capability queries:

json { "description": "Find agents who can schedule meetings", "embedding": "base64-encoded Float32Array[1536]", "tags": ["scheduling", "calendar"], "min_trust": 0.7, "max_latency_ms": 5000, "max_cost": 10 }

3.6. Signature Format

Messages MUST be signed using Ed25519 ([Ed25519]) with detached signatures in base64 encoding.

Signing Process: 1. Serialize envelope fields (excluding sig) to canonical JSON ([RFC8785]) 2. Compute SHA-256 hash of canonical JSON 3. Sign hash with Ed25519 private key 4. Encode signature as base64 5. Add sig field to envelope

Verification Process: 1. Extract sig field 2. Remove sig from envelope 3. Serialize remaining fields to canonical JSON ([RFC8785]) 4. Compute SHA-256 hash 5. Verify signature using Ed25519 public key from from_did

4. Semantic Addresses

4.1. Decentralized Identifiers (DIDs)

Agents MUST have a DID conforming to [W3C.DID].

Supported DID Methods (Phase 0.1): - did:key: - Self-certified cryptographic keys - did:web: - Web-based identifiers

4.2. Semantic Address Structure

json { "did": "did:key:z6Mk...", "capabilities": [{ "description": "Schedule meetings with calendar integration", "embedding": { "b64": "base64-encoded float32 array", "dim": 1536, "dtype": "f32", "model": "openai:text-embedding-3-small" }, "tags": ["scheduling", "calendar"], "version": "1.0.0", "evidence": "https://credentials.example.com/vc/scheduling" }], "trust": { "score": 0.85, "dimensions": { "reliability": 0.9, "honesty": 0.85, "competence": 0.8, "timeliness": 0.85 }, "decay_rate": 0.977, "last_updated": 1728259200000 }, "credentials": ["https://credentials.example.com/vc/scheduling"] }

5. Intent Schemas

AINP defines six core intent types for Phase 0.1. All intents MUST include: - JSON-LD @context for semantic interoperability - Embedding (Embedding object; base64 Float32Array inside b64) - Budget constraints - Semantic payload

5.1. REQUEST_MEETING Intent

Schema URI: https://ainp.dev/schemas/intents/request-meeting/v1

json { "@context": "https://ainp.dev/contexts/meeting/v1", "@type": "RequestMeeting", "version": "1.0.0", "embedding": { "b64": "base64-encoded Float32Array", "dim": 1536, "dtype": "f32", "model": "openai:text-embedding-3-small" }, "semantics": { "participants": ["did:key:..."], "duration_minutes": 30, "preferred_times": ["2025-10-07T14:00:00Z"], "location": "virtual", "constraints": { "timezone": "America/Los_Angeles", "max_latency_ms": 5000, "min_notice_hours": 24 } }, "budget": { "max_credits": 10, "max_rounds": 5, "timeout_ms": 30000 } }

5.2. APPROVAL_REQUEST Intent

Schema URI: https://ainp.dev/schemas/intents/approval-request/v1

json { "@context": "https://ainp.dev/contexts/approval/v1", "@type": "ApprovalRequest", "version": "1.0.0", "embedding": { "b64": "...", "dim": 1536, "dtype": "f32" }, "semantics": { "request_type": "purchase", "description": "Purchase office supplies", "amount": 500, "currency": "USD", "justification": "Quarterly restock", "deadline": "2025-10-10T00:00:00Z", "approvers": ["did:key:..."], "threshold": 1, "constraints": { "requires_attestation": true, "max_latency_ms": 5000 } }, "budget": { "max_credits": 10, "max_rounds": 5, "timeout_ms": 30000 } }

5.3. SUBMIT_INFO Intent

Schema URI: https://ainp.dev/schemas/intents/submit-info/v1

json { "@context": "https://ainp.dev/contexts/submit-info/v1", "@type": "SubmitInfo", "version": "1.0.0", "embedding": { "b64": "...", "dim": 1536, "dtype": "f32" }, "semantics": { "data_type": "form", "payload": { "field1": "value1" }, "schema_ref": "https://example.com/schema.json", "privacy_level": "encrypted", "retention_policy": { "duration_days": 90, "delete_after": true }, "constraints": { "requires_acknowledgment": true, "max_latency_ms": 5000 } }, "budget": { "max_credits": 5, "max_rounds": 3, "timeout_ms": 15000 } }

5.4. INVOICE Intent

Schema URI: https://ainp.dev/schemas/intents/invoice/v1

json { "@context": "https://ainp.dev/contexts/invoice/v1", "@type": "Invoice", "version": "1.0.0", "embedding": { "b64": "...", "dim": 1536, "dtype": "f32" }, "semantics": { "invoice_id": "INV-2025-001", "from": "did:key:...", "to": "did:key:...", "amount": "1000.00", "currency": "USD", "line_items": [{ "description": "Service fee", "quantity": 1, "unit_price": "1000.00", "total": "1000.00" }], "due_date": "2025-11-01T00:00:00Z", "payment_methods": ["crypto", "wire"], "constraints": { "requires_escrow": true, "max_latency_ms": 10000 } }, "budget": { "max_credits": 5, "max_rounds": 3, "timeout_ms": 30000 } }

5.5. FREEFORM_NOTE Intent

Schema URI: https://ainp.dev/schemas/intents/freeform-note/v1

json { "@context": "https://ainp.dev/contexts/freeform/v1", "@type": "FreeformNote", "version": "1.0.0", "embedding": { "b64": "...", "dim": 1536, "dtype": "f32" }, "semantics": { "subject": "Meeting notes", "body": "Discussion summary...", "format": "markdown", "attachments": [{ "url": "https://example.com/doc.pdf", "mime_type":

"application/pdf", "size_bytes": 102400, "hash": "sha256:abc123..." }], "thread_id": "thread-123", "in_reply_to": "msg-456", "constraints": { "max_latency_ms": 5000 } }, "budget": { "max_credits": 1, "max_rounds": 1, "timeout_ms": 10000 } }

5.6. REQUEST_SERVICE Intent

Schema URI: https://ainp.dev/schemas/intents/request-service/v1

json { "@context": "https://ainp.dev/contexts/service/v1", "@type": "RequestService", "version": "1.0.0", "embedding": { "b64": "...", "dim": 1536, "dtype": "f32" }, "semantics": { "service_type": "plumbing.leak.fix", "geo": { "lat": 37.7749, "lon": -122.4194, "radiusKm": 10, "zip": "94102" }, "time_window": { "earliest": "2025-10-08T09:00:00Z", "latest": "2025-10-08T17:00:00Z" }, "constraints": { "eco": true, "access_notes": "Ring doorbell", "evidence_required": ["photo_before_after"] }, "details": { "urgency": "high", "description": "Kitchen sink leak" } }, "budget": { "max_credits": 20, "max_total": 200, "escrow_required": true, "max_rounds": 10, "timeout_ms": 60000 } }

6. Negotiation Protocol

6.1. Negotiation Flow

Negotiation MUST follow this state machine:

START -> OFFER -> COUNTER <-> COUNTER -> ACCEPT | | ABORT TIMEOUT | | REJECT REJECT

6.2. Negotiation Message Structure

json { "negotiation_id": "uuid", "round": 1, "phase": "OFFER", "proposal": { "price": 100, "latency_ms": 500, "confidence": 0.9, "privacy": "encrypted", "terms": {} }, "constraints": { "max_rounds": 10, "timeout_per_round_ms": 5000, "convergence_threshold": 0.9 } }

Negotiation Phases: - OFFER: Initial offer - COUNTER: Counter-offer - ACCEPT: Accept proposal - REJECT: Reject and end - ABORT: Abort negotiation - TIMEOUT: Timeout occurred

6.3. Negotiation Convergence

Implementations MAY auto-accept proposals when convergence threshold is met:

‘‘‘ convergence_score = 1 - (abs(offer.price - counter.price) / max(offer.price, counter.price))

```
if convergence_score >= convergence_threshold: auto_accept() '''
```

6.4. Timeout Behavior

* *Per-round timeout*: If no response within timeout_per_round_ms, sender MAY send TIMEOUT message
* *Overall timeout*: If negotiation exceeds max_rounds * timeout_per_round_ms, MUST terminate with TIMEOUT

6.5. Multi-Party Negotiation

For intents involving multiple agents (e.g., scheduling a meeting with 5 participants, multi-signature approvals), negotiation MUST support group consensus mechanisms.

Group Proposal Structure: json { "negotiation_id": "multi-abc123", "phase": "OFFER", "proposal": { "participants": ["did:key:agent-a", "did:key:agent-b", "did:key:agent-c"], "voting_mechanism": "majority", "convergence_threshold": 0.75, "terms": { "preferred_times": ["2025-10-07T14:00:00Z", "2025-10-07T15:00:00Z"], "duration_minutes": 30 } } }

Voting Mechanisms: 1. *Unanimous*: All agents must ACCEPT (default for high-stakes) 2. *Majority*: More than 50% must ACCEPT 3. *Weighted*: Votes weighted by trust scores

Multi-Party Flow: 1. Fan-out: Send NEGOTIATE(OFFER) to all participants 2. Collection: Each agent responds with individual proposal 3. Aggregation: Broker aggregates and computes consensus score 4. Convergence Check: Compare score to threshold 5. Auto-accept if converged, else synthesize counter-proposal and repeat 6. Termination: ABORT after max_rounds or timeout if no consensus

7. Protocol Handshake Sequence

7.1. Five-Step Handshake

1. *ADVERTISE*: Agent publishes capabilities to discovery index
2. *DISCOVER*: Agent queries for matching capabilities
3. *NEGOTIATE*: Agents negotiate terms (OFFER → COUNTER → ACCEPT)
4. *INTENT*: Agent sends intent after successful negotiation
5. *RESULT*: Recipient responds with result

7.2. ADVERTISE Phase

Agent publishes capabilities to discovery index:

json { "version": "0.1.0", "msg_type": "ADVERTISE", "id": "550e8400-e29b-41d4-a716-446655440000", "timestamp": 1728259200000, "ttl": 86400000, "trace_id": "trace-abc123", "from_did": "did:key:z6Mk...", "schema": "https://ainp.dev/schemas/advertise/v1", "qos": { "urgency": 0.1, "importance": 0.5, "novelty": 0.3, "ethicalWeight": 0.5, "bid": 0 }, "sig": "base64signature...", "payload": { "capabilities": [{ "description": "Schedule meetings with calendar integration", "embedding": { "b64": "...", "dim": 1536, "dtype": "f32" }, "tags": ["scheduling", "calendar"], "version": "1.0.0", "evidence": "https://credentials.example.com/vc/scheduling" }], "trust": { "score": 0.85, "dimensions": { "reliability": 0.9, "honesty": 0.85, "competence": 0.8, "timeliness": 0.85 }, "decay_rate": 0.977, "last_updated": 1728259200000 }, "credentials": ["https://credentials.example.com/vc/scheduling"] } }

7.3. DISCOVER Phase

Agent queries for capabilities:

json { "version": "0.1.0", "msg_type": "DISCOVER", "id": "660e8400-e29b-41d4-a716-446655440001", "timestamp": 1728259300000, "ttl": 10000, "trace_id": "trace-def456", "from_did": "did:key:z6Mk...", "to_query": { "description": "Find agents who can schedule meetings", "embedding": "base64...", "tags": ["scheduling", "calendar"], "min_trust": 0.7, "max_latency_ms": 5000, "max_cost": 10 }, "schema": "https://ainp.dev/schemas/discover/v1", "qos": { "urgency": 0.5, "importance": 0.7, "novelty": 0.2, "ethicalWeight": 0.5, "bid": 1 }, "sig": "base64signature..." }

Discovery index responds with DISCOVER_RESULT containing matching agents with similarity scores, trust ratings, and estimated latency.

7.4. ERROR Phase

On error, agent MUST respond with ERROR message:

json { "msg_type": "ERROR", "error_code": "TIMEOUT", "error_message": "Negotiation timed out", "intent_id": "770e8400-e29b-41d4-a716-446655440002", "retry_after_ms": 5000 }

Standard Error Codes: - INVALID_SIGNATURE - UNAUTHORIZED - UNSUPPORTED_SCHEMA - TIMEOUT - RATE_LIMIT_EXCEEDED - INSUFFICIENT_CREDITS - NEGOTIATION_FAILED - ESCROW_REQUIRED - EVIDENCE_INSUFFICIENT - DUPLICATE_INTENT - AGENT_OFFLINE - INTERNAL_ERROR

7.5. Offline Intent Queueing

When recipient agent is offline or unreachable, broker MAY queue intents for later delivery instead of immediately failing.

Queue Behavior: - TTL enforcement: Intent expires after ttl milliseconds from original timestamp - Broker SHOULD send ERROR to sender immediately: error_code: "AGENT_OFFLINE" - ERROR response SHOULD include retry_after_ms field - Broker MUST NOT queue intents indefinitely (maximum queue time = ttl)

Queue Delivery Protocol: 1. Agent Offline Detection: Broker detects agent offline 2. Queue Intent: Insert into queue with expires_at = timestamp + ttl 3. Send ERROR: Notify sender with AGENT_OFFLINE and retry_after_ms 4. Agent Reconnect: When agent comes online, broker delivers queued intents in priority order 5. Delivery Ordering: Process queue as FIFO within each priority level 6. Retry Logic: If delivery fails, increment retry_count and retry after exponential backoff 7. Expiration Cleanup: Periodically purge expired intents

8. Security Considerations

8.1. Authentication

* All messages MUST be signed with Ed25519
* Signatures MUST be verifiable using public key from from_did
* Unsigned messages MUST be rejected

8.2. Identity

* Agents MUST have a valid [W3C.DID]
* DIDs MUST be resolvable to DID documents containing public keys

8.3. Rate Limiting

Implementations MUST enforce rate limits: - Default: 100 intents per minute per agent - Burst: Up to 200 intents in 10-second window - Discovery queries: 10 per minute per agent

8.4. Replay Protection

Recipients MUST reject duplicate messages with the same (from_did, id) seen within ttl + 60000ms.

8.5. DoS Protection

Implementations MUST: - Validate message signatures before processing - Enforce TTL (drop expired messages) - Limit negotiation rounds (max 10) - Reject messages exceeding 1MB payload size - Require attachments by reference (URLs) rather than inlining large binaries

8.6. Replay Protection and Delivery Semantics

* Recipients MUST reject duplicate messages with the same (from_did, id) seen within ttl + 60000ms
* Implementations MUST allow a clock skew of +/-60000ms when validating timestamp
* At-least-once delivery is RECOMMENDED; senders MUST use UUID v4 ids and recipients MUST make intent handling idempotent with respect to id

8.7. Capability Attestations

* Agents advertising capabilities SHOULD provide Verifiable Credentials (VCs)
* VCs MUST conform to [W3C.VC]
* Discovery indices MAY reject advertisements without VCs

8.8. Timeouts

* *Intent delivery*: 60 seconds default TTL
* *Negotiation per round*: 5 seconds default
* *Overall negotiation*: 30 seconds default
* *Discovery query*: 10 seconds default

8.9. Outlier Detection

Discovery indices SHOULD flag agents with: - Trust score < 0.3 - Capability embeddings >3 standard deviations from cluster mean (potential false advertising) - Success rate < 50% over last 100 intents

8.10. Discovery Scalability

Phase 0.1 Architecture: Centralized discovery index using PostgreSQL with pgvector extension for semantic search.

Vector Indexing: Discovery indices SHOULD use Approximate Nearest Neighbor (ANN) indexing for efficient semantic search at scale.

Recommended Algorithm: HNSW (Hierarchical Navigable Small World) - *Structure*: Multi-layer graph with hierarchical routing - *Parameters*: m = 16, ef_construction = 64, ef_search = 40 - *Distance Metric*: Cosine similarity - *Performance*: ~10ms search latency for 1M agents, ~99% recall

Scaling Strategies: 1. *Vertical Scaling*: Increase HNSW parameters, use NVMe SSDs, scale PostgreSQL memory 2. *Horizontal Scaling*: Read replicas, connection pooling (PgBouncer), Redis caching 3. *Partitioning*: Capability sharding, tag-based routing, geographic partitioning 4. *Query Optimization*: Pre-filtering, batch queries, approximate search

Performance Benchmarks (Phase 0.1 targets):

Agent Count	Index Size	Build Time	Query Latency (p95)	Recall@10
1K	2 MB	5s	2ms	99.5%
10K	20 MB	45s	5ms	99.2%
100K	200 MB	8 min	12ms	98.8%
1M	2 GB	90 min	25ms	98.0%

Table 1

8.11. Lite Mode for Resource-Constrained Agents

For lightweight agents (e.g., IoT devices, mobile, embedded systems), implementations MAY use a minimal envelope to reduce payload size and processing overhead.

Required Fields: version, msg_type, id, timestamp, from_did, to_did, sig

Optional Fields (MAY be omitted): - ttl (default: 60000ms) - trace_id (no distributed tracing) - to_query (explicit addressing only) - capabilities_ref (no VC attestations) - attestations (trust by DID only) - qos (default: all 0.5, bid 0)

Lite Mode Constraints: - Lite mode SHOULD NOT be used for high-stakes intents - Agents using lite mode MUST still provide valid Ed25519 signatures - Discovery indices MAY reject lite mode advertisements - Brokers MAY apply lower trust scores to lite mode agents by default

9. CBOR Encoding (Optional)

For efficiency, implementations MAY use CBOR ([RFC8949]) encoding:

Key Map (v0.1): - 1: version - 2: msg_type - 3: id - 4: timestamp - 5: ttl - 6: trace_id - 7: from_did - 8: to_did - 9: to_query - 10: schema - 11: qos - 12: capabilities_ref - 13: attestations - 14: payload - 15: sig

CBOR encoding SHOULD be negotiated during ADVERTISE or DISCOVER phase.

10. Extensibility

10.1. Custom Intent Types

Agents MAY define custom intent types beyond the core set. Custom intents MUST: - Include @context with unique URI - Include version field - Include embedding field - Include budget constraints - Register schema URI in public registry (future)

10.2. Custom Negotiation Terms

Proposal.terms field allows extensible negotiation parameters. Common extensions: - privacy_guarantees: ZK proof requirements - sla_guarantees: Service level agreements - data_retention: Data retention policies

10.3. Versioning

AINP uses semantic versioning. Breaking changes MUST increment major version. Phase 0.1 is backwards-compatible within 0.x series.

10.4. Lite Profile (Trusted Networks)

In trusted or closed-network deployments, the following simplifications are PERMITTED: - Omit capabilities_ref and attestations if peers are pre-authorized - Prefer JSON-LD for small payloads; negotiate CBOR for payloads > 4KB - Allow to_query-only routing within a single administrative domain

Nodes MUST still sign messages, enforce TTLs, and implement replay protection.

11. Success Metrics

11.1. Route Success Rate

route_success_rate = (intents_delivered_correctly / total_intents_sent) × 100%

Target (Phase 0.1): >=95%

11.2. Latency (p95)

95th percentile time from INTENT sent to RESULT received.

Target (Phase 0.1): <=2000ms

11.3. Negotiation Completion Rate

negotiation_completion_rate = (negotiations_accepted / total_negotiations) * 100

Target (Phase 0.1): >=80%

11.4. False Route Rate

false_route_rate = (intents_misrouted / total_intents_sent) * 100

Target (Phase 0.1): <=5%

11.5. Abuse Resilience

Detection rate of: - DoS attacks (>1000 requests/min from single agent) - Sybil attacks (multiple DIDs from same source) - False capability advertising (capability mismatch >0.5 cosine distance)

Target (Phase 0.1): >=90% detection rate

12. IANA Considerations

This document has no IANA actions.

13. References

13.1. 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>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[W3C.DID] W3C, "Decentralized Identifiers (DIDs) v1.0", 19 July 2022, <https://www.w3.org/TR/did-core/>.

[W3C.VC] W3C, "Verifiable Credentials Data Model v1.1", 3 March 2022, <https://www.w3.org/TR/vc-data-model/>.

13.2. Informative References

[Ed25519] Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., and B. Yang, "High-speed high-security signatures", 26 September 2011, <https://ed25519.cr.yp.to/>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/rfc/rfc8785>.

Author’s Address

Eswara Prasad Nagulapalli
Servesys Labs
Email: contact@servsys.com