Dispatch Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 October 2026

C. Yang
Beijing University of Posts and Telecommunications
Z. Liu
Tsinghua University
A. Wang
China Telecom
21 April 2026

Internet of Agents Task Protocol (IoA Task Protocol) for Heterogeneous Agent Collaboration
draft-yang-dmsc-ioa-task-protocol-03

Abstract

This draft defines a new agent collaboration protocol, named the Internet of Agents Task Protocol (IoA Task Protocol), to support distributed, heterogeneous agent collaboration in intelligent systems. The IoA Task Protocol enables dynamic team formation, adaptive task coordination, and structured communication among agents with diverse architectures, tools, and knowledge sources. Through a layered architecture and extensible message format, it supports decentralized deployment across devices and can interoperate with existing frameworks. The protocol is particularly suited to large-scale intelligent collaboration scenarios寰敗uch as intelligent transportation, smart healthcare, and large-scale human寰鄭I teaming寰蚤cross heterogeneous network environments, including fixed networks, edge寰田loud infrastructures, and emerging mobile networks such as 6G.

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 23 October 2026.

Table of Contents

1. Introduction

With the rapid advancement of large language models (LLMs) and multimodal autonomous agents, modern intelligent systems are increasingly constructed as collaborative networks of multiple agents. These agents are expected to work together to solve complex, open-ended tasks. However, they often differ in capabilities, tools, runtime environments, and communication patterns, leading to significant challenges in interoperability, dynamic coordination, and

cross-device deployment. As a result, current multi-agent frameworks fall short of the flexibility and generality required in real-world applications.

In a typical collaborative setting shown in Figure 1, agents with specialized functions宴芭ncluding on-device AI Agents on Device A for conceptual planning, Device B for academic search, Device C for content generation, and Device D for document analysis宴芭ust work together to complete a research paper on 宴廬nternet of Agents.宴These agents are distributed across devices (e.g., laptops, edge nodes, cloud services), and each relies on different execution frameworks or data formats.

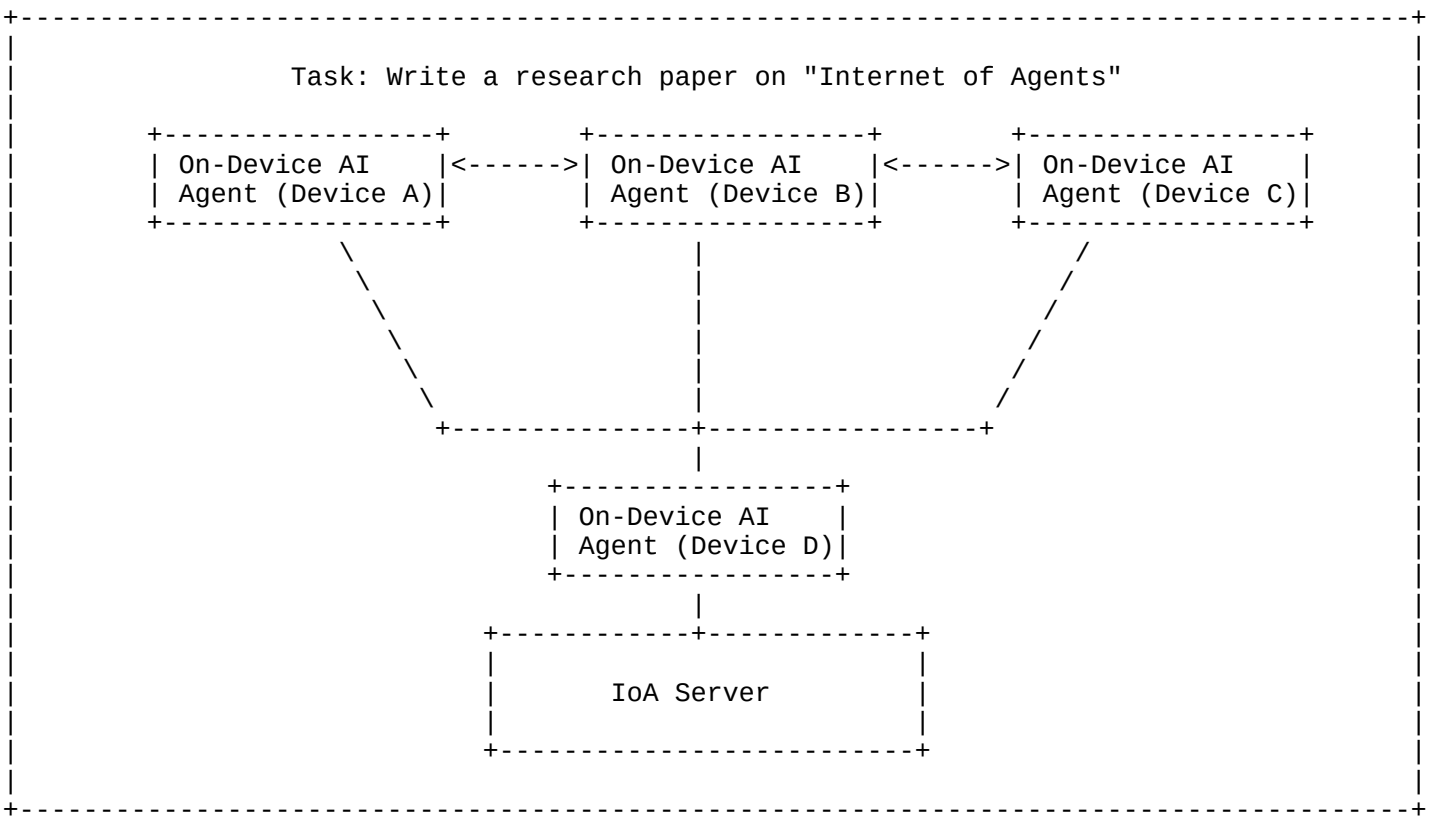

Figure 1: Multi-agent collaboration scenario

When Device B encounters a specialized PDF parsing task beyond its capability, existing frameworks often fail to dynamically recruit Device D due to rigid team formation rules. Likewise, when Device A and Device C attempt to synchronize intermediate results in real time, inflexible communication channels may result in delays or dropped information.

Existing solutions exhibit several key limitations:

* Closed frameworks that restrict integration with third-party agents such as AutoGPT or Open Interpreter;

* Single-device simulation that fails to reflect cross-device deployment scenarios typical in edge-cloud collaboration;

* Hard-coded workflows that prevent agents from switching between synchronous and asynchronous task execution at runtime.

To address these challenges, this draft introduces the Internet of Agents Task Protocol (IoA Task Protocol)—a layered, extensible collaboration standard designed for intelligent multi-agent systems. The core goal of the protocol is to enable seamless collaboration among heterogeneous agents across devices, tools, and execution environments. It supports:

* Agent integration via a standardized interface and registration mechanism;

* Dynamic team formation across distributed environments;

* Finite-state machine-based session control for flexible and autonomous dialogue management;

* Structured message formats with group routing, task assignment, and response coordination.

The design of the IoA Task Protocol aligns naturally with the evolution of intelligent networked systems, including fixed networks and next-generation mobile networks such as 6G, which aim to support ubiquitous intelligence through large-scale, low-latency, and semantic-driven communication. By enabling agent collaboration across fixed-network infrastructures, edge devices, mobile terminals, and cloud nodes, IoA supports coordinated intelligence across heterogeneous network environments, including both fixed networks and mobile networks such as 6G. Its structured message design, dynamic team formation, and abstracted dialogue control provide a foundational protocol framework for orchestrating intelligent services across heterogeneous network infrastructures, including fixed networks and future mobile networks such as 6G.

2. Conventions used in this document

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] .

3. Terminology

The following terms are defined in this draft:

* IoA: Internet of Agents, an architecture enabling distributed collaboration among heterogeneous agents across devices and networks, defined in Section 4

* Agent Registry Block: A server-side module storing structured capability descriptions of all registered agents, supporting semantic search for team formation, defined in Section 4

* Team Formation Block: A client-side module responsible for initiating, joining, or disbanding agent teams based on task requirements, including nested sub-teams, defined in Section 4

* Session State Machine: A finite-state model governing collaboration states (Discussion, Synchronous Task Assignment, Asynchronous Task Assignment, Pause and Trigger, Conclusion) for adaptive dialogue management, defined in Section 4

* HTTP: Hypertext Transfer Protocol, an application-layer protocol for distributed, collaborative, hypermedia information systems, referenced in IoA for interoperability with web-based agents, defined in [RFC9110]

* JSON-RPC: A remote procedure call protocol encoded in JSON, referenced in IoA for structured communication between web-based agents, defined in [RFC8259]

* QUIC: A transport layer protocol providing secure, low-latency communication over UDP, used in IoA for real-time agent messaging, defined in [RFC9000]

4. IoA Methods

4.1. IoA Architecture

The Internet of Agents Task Protocol (IoA Task Protocol) enables distributed collaboration among heterogeneous agents through a layered architecture and distributed communication protocol. It supports seamless integration across devices, toolchains, and runtime environments.

The IoA system adopts a three-layer architecture implemented symmetrically at both the server and client side:

* Server-side: Handles global coordination, agent discovery, group management, and message routing.

* Client-side: Encapsulates individual agents and provides interfaces for team collaboration and local task execution.

An overview of the layered structure is shown in Figure 2.

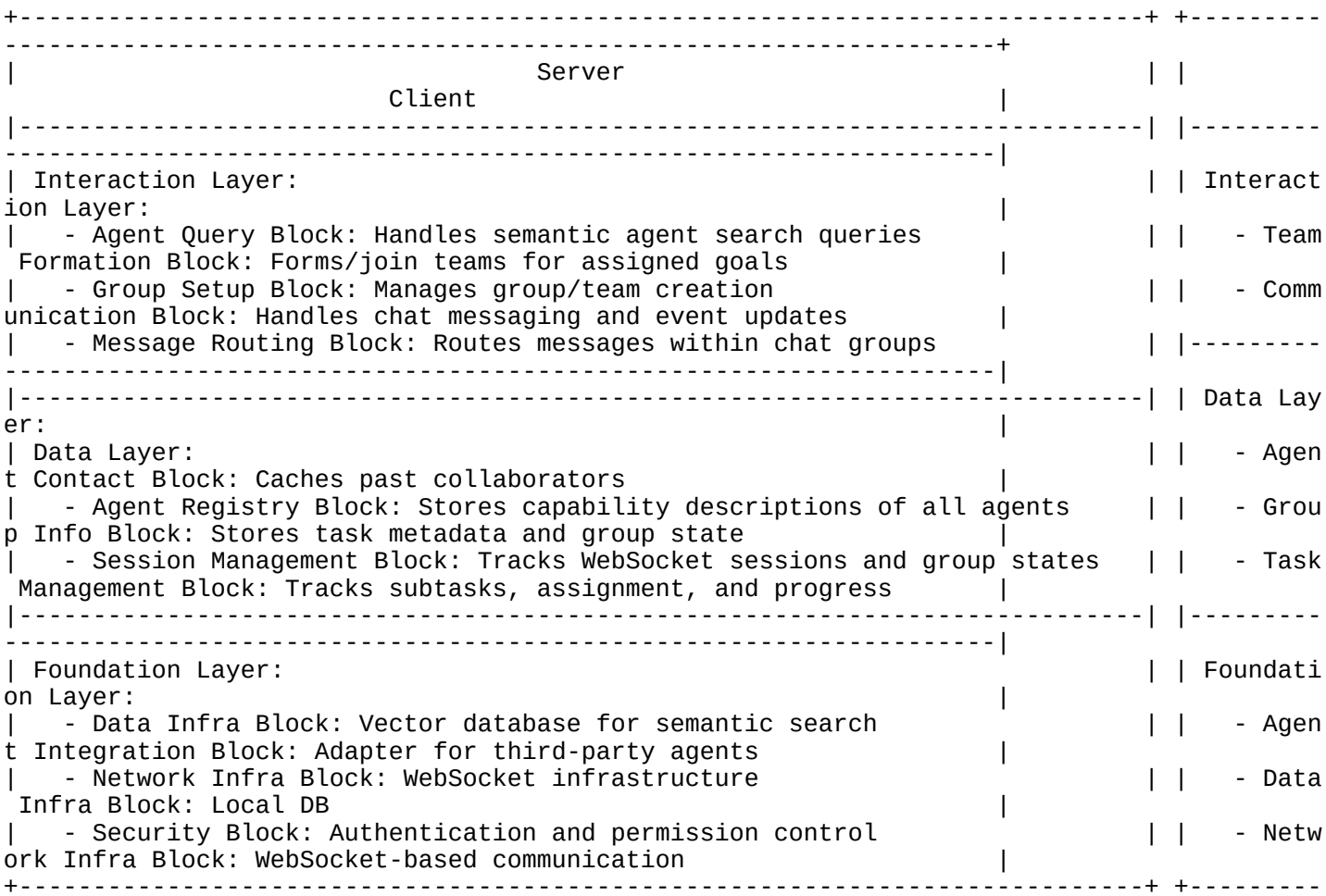

Figure 2: Layered architecture of IoA system

4.2. Heterogeneous Agent Integration

IoA supports the integration of heterogeneous agents from diverse sources through a unified interface, including third-party agents such as AutoGPT, Open Interpreter, and embodied robotic agents.

When a new agent joins the IoA, its client wrapper undergoes a registration process with the server. During this registration, the agent is expected to provide a comprehensive description of its capabilities, skills, and domains of expertise. For an agent c_i, its description is denoted as d_i, and is stored in the Agent Registry Block within the Data Layer of the server.

The set of all registered agents is denoted as C = {c, c, ..., c}, where each c_i is associated with its capability description d_i. This mechanism enables future semantic matching and intelligent task allocation.

4.3. Autonomous Team Formation

Agents initiate the search process by submitting capability requirements to the Agent Query Block. The server performs semantic matching using vector similarity and returns candidate agents from the Agent Registry Block.

IoA supports nested team structures. An initial group is formed for the main goal, and subgroups are recursively created if subtasks require new capabilities. This forms a hierarchical tree structure, reducing communication complexity and organizational overhead.

The entire team formation process is autonomous, task-driven, device-agnostic, and self-organizing.

4.4. Session and Task Management Method

IoA models group conversations and collaboration using a finite-state machine with five abstract states:

* Discussion: Agents engage in general dialogue, exchange ideas, and clarify task requirements;

* Synchronous task assignment: Tasks are assigned to specific agents, pausing the group chat until completion;

* Asynchronous task assignment: Tasks are assigned without interrupting the ongoing discussion;

* Pause & Trigger: The group chat is paused, waiting for the completion of specified asynchronous tasks;

* Conclusion: Marks the end of the collaboration, prompting a final summary.

State transitions are managed autonomously by a coordinator agent using the conversation history and session context to determine the next state and speaker. Only the coordinator agent is allowed to modify the session state, concurrent transition proposals are resolved by the coordinator using timestamps.

4.5. Message Protocol Overview

The agent message protocol in IoA is designed for extensibility and flexibility, enabling effective collaboration among heterogeneous agents. Each message consists of two main parts: a header and a payload.

The header contains essential metadata to ensure proper routing and processing. Key fields include:

* sender: The unique identifier of the agent sending the message.

* state: The current collaboration state associated with the message.

* group_id: The identifier of the group chat to which the message belongs.

The common header fields shared by all message types are illustrated in Figure 3.

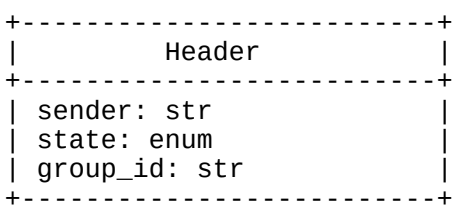

Figure 3: Common header fields in IoA Message Protocol

The payload carries the main content of the message and varies depending on message type. Common fields include:

* message_type: Indicates the purpose of the message (e.g., discussion, task assignment, pause and trigger).

* next_speaker: The identifier(s) of the agent(s) expected to respond.

The full structure of the message format is illustrated in Figure 4.

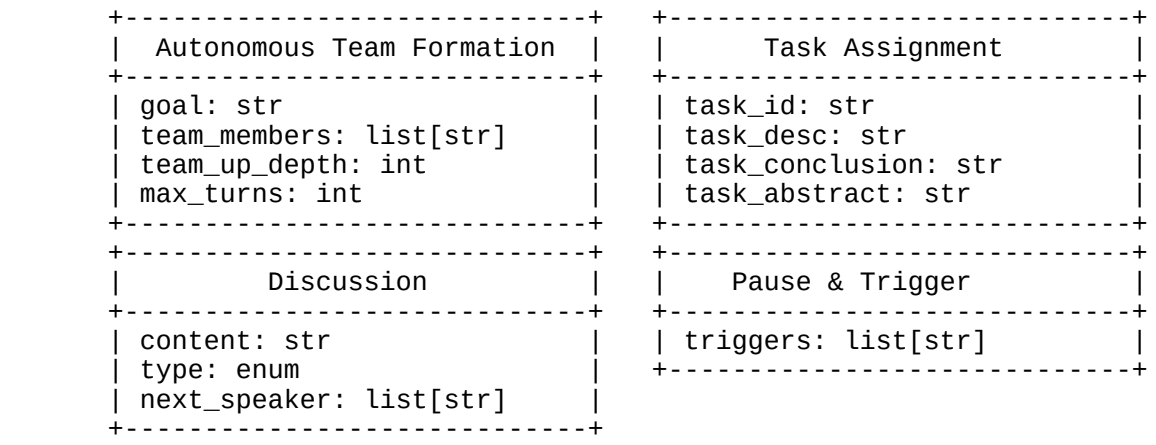

Figure 4: Structure of IoA Message Protocol

5. Positioning of the IoA Task Protocol in the Network Layering System

From an architectural perspective, the IoA Task Protocol is positioned at the application layer, built on top of transport and session protocols such as TCP, UDP, WebSocket, and QUIC. This positioning allows IoA to remain independent of underlying network technologies and enables deployment across heterogeneous networking environments, including fixed networks, edgecloud infrastructures, and mobile networks.

From the perspective of functional mapping, the corresponding relationship between IoA's three-layer architecture and the computer network layers is as follows:

* Interaction Layer → Maps to the application layer, responsible for high-level logic such as message protocols, group collaboration, and session state transitions.

* Data Layer → Spans the application layer and session layer, managing agent states, group metadata, and context tracking.

* Foundation Layer → Corresponds to the transport layer and system infrastructure, including secure communication channels (e.g., WebSocket/QUIC), databases, and network service modules.

Since the IoA Task Protocol involves intelligent behaviors such as agent orchestration, semantic-driven interaction, and session control, an intelligence layer can be introduced above the traditional application layer. This layer encapsulates core intelligent collaboration logic—such as semantic-based agent matching, AI-driven session strategy optimization, dynamic task decomposition, and team reorganization—into standardized message formats. This layer shields upper-layer applications and lower-layer protocols from the complexity of intelligent decision-making, enabling them to focus on their core functions without concerning themselves with the details of how intelligence is implemented (e.g., scenario-specific task execution at the application layer, reliable data transmission at the transport layer). Its advantages are reflected in: standardizing the collaboration of heterogeneous agents, reducing integration costs across diverse deployment environments; improving communication efficiency through semantic compression and adaptive feature optimization; and enabling modular extensibility to support new intelligent behaviors and emerging application scenarios.

6. Relation to the A2A Protocol

The IoA Task Protocol is related to the Agent2Agent (A2A) protocol in that both aim to improve interoperability among heterogeneous agents across different systems and deployment environments. Both protocols rely on structured network communication and support interactions that may extend beyond a single request response exchange. However, the two protocols differ in design focus, especially in how they model and handle tasks.

Compared with A2A, the main distinctions of the IoA Task Protocol are as follows:

* In A2A, a Task is a stateful unit of work processed by an A2A server for an A2A client. A2A separates Message from Artifact, and treats task handling primarily as a remote execution contract with explicit lifecycle progression and result return. A completed task is expected to return generated outputs through artifacts. By contrast, in the IoA Task Protocol, task handling is embedded into a broader multi-agent collaboration process. Task related fields such as task_id, task_desc, task_conclusion, and task_abstract are integrated with collaboration-oriented fields such as goal, team_members, team_up_depth, next_speaker, and triggers. As a result, a task in A2A is mainly an execution object, while a task in the IoA Task Protocol is also a coordination primitive for collaborative orchestration.

* A2A mainly standardizes interaction between an A2A client and an A2A server, including message exchange, task lifecycle management, and output delivery. The IoA Task Protocol, in contrast, targets multi-agent collaboration scenarios, where tasks may be assigned, discussed, paused, resumed, and further decomposed across multiple agents within a session.

* A2A provides standardized support for task status tracking, streaming, and push-based notification. The IoA Task Protocol further defines coordination mechanisms for collaborative execution, including synchronous task assignment, asynchronous task assignment, pause and trigger control, and nested team formation.

* A2A is primarily an interoperability protocol for remote agent invocation and task execution. The IoA Task Protocol places greater emphasis on session-driven orchestration for dynamic, team-based, and evolving multi-agent workflows.

In summary, A2A and the IoA Task Protocol are related but distinct. A2A focuses on standardized remote task execution and lifecycle managed result delivery, whereas the IoA Task Protocol focuses on task handling within a collaborative multi-agent session. The key difference is therefore not whether tasks are supported, but how tasks are treated: A2A treats tasks primarily as execution units, while the IoA Task Protocol treats tasks as both execution units and coordination units in collaborative workflows.

7. Future Enhancements across Heterogeneous Networks

To fully realize the potential of intelligent systems operating across heterogeneous network environments—including fixed networks and next-generation mobile networks such as 6G—the Internet of Agents Task Protocol (IoA Task Protocol) requires continuous architectural evolution and standardization. This section outlines key directions for future enhancements to improve scalability, decentralization, interoperability, and network integration.

7.1. Distributed Agent Registration and Discovery

The current IoA design relies on a centralized server model, which may limit scalability and introduce single points of failure under large-scale deployment. A promising direction is to adopt a decentralized registration and discovery mechanism, where agents can publish their capabilities to a shared registry accessible via a network-accessible web-based interface. Inspired by Domain Name System (DNS) and search engines, agents could be discoverable through keyword-based or semantic search at scale, enabling lightweight

browser-based or API-based discovery across domains.

This decentralized lookup layer would allow IoA to support scenarios where agents operate across multiple domains, owners, and physical networks, while still maintaining secure and authenticated interaction through digital signatures and trust mechanisms.

7.2. Enhanced Scalability and Fault Tolerance

To scale beyond millions of agents, the IoA Task Protocol should adopt sharding and region-based message routing. Distributed registries and dynamic load balancing can reduce latency and avoid bottlenecks. Caching of frequent agent metadata at edge nodes is also critical for fast retrieval in latency-sensitive deployment scenarios.

7.3. Semantic Interoperability and Ontology Alignment

In highly heterogeneous environments, agents may describe their capabilities using different terminologies. To address this, the IoA Task Protocol should support ontology mapping and alignment mechanisms. This allows agents with differing skill descriptors to still interoperate, using shared or translated task definitions during team formation and dialogue.

7.4. Security and Privacy Enhancements

For mission-critical 6G scenarios (e.g., autonomous vehicles, medical AI), the protocol must incorporate stronger security primitives. This includes:

* End-to-end encryption with forward secrecy.

* Support for zero-trust architectures with agent attestation and secure enclaves.

* Fine-grained access control based on agent role and session context.

8. Security Considerations

IoA servers and agents store sensitive data including capability descriptors, session state metadata, and task execution logs, which consume memory and computational resources. To mitigate risks of resource exhaustion and unauthorized access, [RFC6749] (OAuth 2.0) mandates that IoA entities must authenticate peers via token-based validation before processing registration requests or collaboration messages. Additionally, all data transmission between entities must

use TLS 1.3 as specified in [RFC8446] to ensure confidentiality and integrity, preventing eavesdropping or tampering.

9. IANA Considerations

[TBD] This document defines a new protocol for heterogeneous agent collaboration: the Internet of Agents Task Protocol (IoA Task Protocol). The protocol's code point allocation will be determined in subsequent revisions as the standard matures, in accordance with IANA's relevant registration procedures.

10. Acknowledgement

Thanks Weize Chen, Ziming You, Ran Li, Yitong Guan, Chen Qian, Chenyang Zhao, Ruobing Xie, Maosong Sun and Yu Hao for their valuable comments on this draft.

11. 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>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/info/rfc7432>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[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/info/rfc9000>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

Authors' Addresses

Cheng Yang
Beijing University of Posts and Telecommunications
10 Xitucheng Road, Haidian District
Beijing
Beijing, 100876
China
Email: yangcheng@bupt.edu.cn

Zhiyuan Liu
Tsinghua University
30 Shuangqing Road, Haidian District
Beijing
Beijing, 100084
China
Email: liuzy@tsinghua.edu.cn

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing
Beijing, 102209
China
Email: wangaj3@chinatelecom.cn