AI Agent Communications in the Future Internet -- Paving A Path toward Agentic Web

Qiang Duan * and Zhihui Lu

Posted Date: 5 February 2026

doi: 10.20944/preprints202602.0306.v1

Keywords: agentic AI; multi-agent systems; agent communications; agentic web; the future internet

 doi:10.20944/preprints202602.0306.v1

Article

AI Agent Communications in the Future Internet – Paving A Path Toward Agentic Web

Qiang Duan [1,*] and Zhihui Lu [2]

[1] Information Sciences & Technology Department, Pennsylvania State University Abington College
[2] School of Computer Science and Artificial Intelligence, Fudan University
* Correspondence: qduan@psu.edu

Abstract

The rapid evolution of artificial intelligence technologies toward the agentic AI paradigm enables the emergence of Agentic Web in the future Internet. Agent communication plays a critical role in constructing the Agentic Web but faces unique challenges posed by the edge-network-cloud continuum in the future Internet. This paper provides a comprehensive overview of state-of-the-art agent communication protocols and technologies, evaluating their readiness to support the construction of the Agentic Web. We first survey representative communication protocols and analyze the key technologies they employ, assessing their effectiveness in addressing the challenges for agent communications in the future Internet. We then identify critical gaps between existing approaches and the requirements of the Agentic Web, propose a unified architectural framework grounded in virtualization and service-oriented principles, and outline key research directions needed to advance toward a fully realized Agentic Web.

Keywords: agentic AI; multi-agent systems; agent communications; agentic web; the future internet

1. Introduction

The recent evolution of artificial intelligence is moving toward an agentic AI paradigm, in which AI agents can autonomously perceive their environment, make decisions, take actions, and interact with one another [1,2]. Multi-agent interactions are expected to be a key attribute of the emerging agentic AI paradigm, and Multi-Agent Systems (MASs) comprising agents that interact, either cooperatively or competitively, form a foundation for realizing agentic AI [3]. Therefore, agent communications, the mechanisms that enable information exchange among autonomous AI agents, play a crucial role in constructing effective MASs [4].

A main obstacle to effective agent interactions in MASs is the lack of standardized communication protocols to ensure interoperability among autonomous AI agents, which could be developed by different vendors, implemented with heterogeneous technologies, configured for diverse roles, and deployed by numerous organizations. Agent communications have recently emerged as an active research area attracting extensive attention from both academia and industry, and exciting progress has been made in this field [5]. A variety of agent communication protocols have been proposed over the past two years, including the Agent2Agent (A2A) protocol, the Agent Communication Protocol (ACP), and Agent Network Protocols. In addition, various new developments in agent protocols are currently ongoing.

The recent rapid progress in agentic AI technologies has enabled the widespread deployment of MASs across various application scenarios, ranging from cloud-based data centers to edge computing-based Internet of Things (IoT) systems, spanning the entire Internet, from public backbone networks to wireless mobile networks at the edge [6]. Therefore, it is expected that numerous AI agents, comprising various MASs for diverse applications, will be deployed across highly distributed networking environments in the future Internet, forming an *Agentic Web* – a web of AI agents interconnected through the future Internet's infrastructure.

 doi:10.20944/preprints202602.0306.v1

The Agentic Web, built upon the future Internet, introduces unique challenges for agent communications. Driven by recent developments in networking technologies, especially network virtualization, software-defined networking, and network-as-a-service, networking systems are merging with edge/cloud computing to form a converged edge-network-cloud continuum [7]. Such a transition in networking transforms the Internet from a data transport system to an infrastructure for integrated communication and computation. Therefore, the construction of Agentic Web, which interconnects numerous diverse AI agents through the large-scale edge-network-continuum in the future Internet, is a critical, exciting, and challenging research problem that must be fully resolved to realize the notion of agentic AI.

The currently available agent communication protocols, although they have shown encouraging performance in various MAS scenarios, are often designed for particular networking environments rather than explicitly for the Agentic Web in the future Internet. A variety of papers have been published to review the developments in the technologies and protocols for agent communications; for example, the surveys presented in [5,8–10]. However, these works either focus on a particular protocol, e.g., A2A as in [8,9], overlook some projects related to agent communications, e.g., LMOS and AGNTCY missed in [10,11], or lack reflection of the latest progress in this field, e.g., the merger of A2A and ACP protocols. Also, none of these surveys particularly assesses agent communication protocol designs against the specific demands of Agentic Web in the future Internet.

In this paper, we aim to provide a comprehensive overview of the state of the art in agent communication technologies with respect to their readiness to address the unique challenges of constructing the Agentic Web in the future Internet. Specifically, in this paper, we discuss the key functionalities of agent communication for constructing an Agentic Web, identify the main challenges to agent communication for Agentic Web introduced by the edge-network-cloud continuum in the future Internet, present a survey of the representative protocols for agent communications, and assess key technologies leveraged in these protocols in terms of their effectiveness in addressing the challenges to agent communications in the future Internet. Based on the survey and assessment, we analyze the gap between current agent communication protocols and the Agentic Web objective, propose a unified architectural framework for Agentic Web grounded in the virtualization and service-oriented principles, and discuss key topics and potential directions for future research to realize this framework.

2. AI Agent Communications for Agentic Web

2.1. The Role of AI Agent Communications in Agentic Web

Agent communications enable interactions among autonomous AI agents to support the intricate dynamics in MAS. An agent communication protocol leverages the computational and communication resources in the Internet infrastructure to build an information-exchange platform that supports interactions among AI agents in a MAS. Therefore, agent communication plays a crucial role in realizing the Agentic Web and thus significantly impacts the performance of Agentic AI applications in the future Internet.

Figure 1 presents a layered architecture that illustrates the role of agent communications in the Agentic Web deployed on the future Internet [9]. In this architecture, the agent communication layer sits between the upper-layer multi-agent systems built for various Agentic AI applications and the underlying Internet infrastructure comprising an edge-network-cloud continuum. The core functionalities of the agent communication layer can be categorized into two groups that are respectively for i) management of the inter-agent communication systems and ii) transportation for information exchange between AI agents. It is worth noting that the functions in these two categories, though focusing on distinct aspects of agent communication systems, are closely related. For example, virtually all the management functions rely on the information transport capabilities to accomplish their goals, while the operations of information transport functions are governed by the management functions

 doi:10.20944/preprints202602.0306.v1

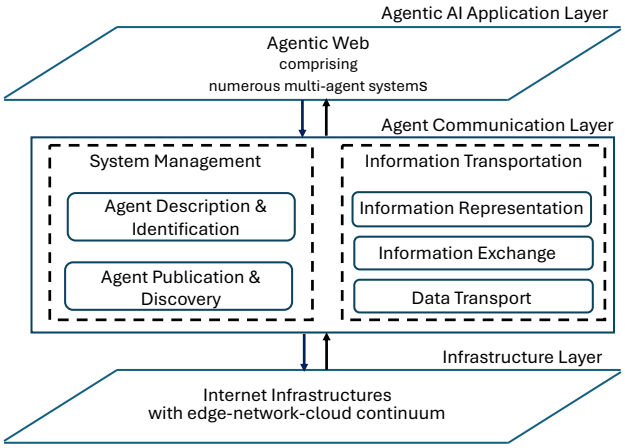

Figure 1. Agent communications for Agentic Web in the future Internet

2.2. Key Functionalities of Agent Communications for Agentic Web

2.2.1. System Management Functions

The system management functions govern the description, publication, discovery, identification, and authentication of the AI agents in the communication system. The agent description function articulates an agent's capabilities and services in a machine-readable, readily comprehensible format for other agents. Agent publication and discovery functions ensure each agent's description is accessible to other agents, allowing them to locate available agents and select the appropriate one(s) to communicate with to accomplish specific tasks. Agent identification and authentication functions assign each agent a unique identity and rigorously verify it. The system management aspect of an agent communication protocol is also expected to provide the support required by upper-layer MAS management, typically for agent-orchestration functionalities.

2.2.2. Information Transportation Functions

Transportation for information exchange refers to the actual transmission of information between AI agents. The functions in this category include information representation, information exchange, and data transport. Information representation dictates how exchanged information is presented in a standard format, with options including structured formats such as JSON and unstructured formats such as natural language, which can be understood by heterogeneous agents. The information exchange function controls message exchange patterns, such as synchronous versus asynchronous messaging, between AI agents. The data transport function ensures the delivery of inter-agent messages across a network, including sub-functions such as sending, forwarding, routing, and receiving.

2.3. Process of Agent Communication in Agentic Web

The overall structure of the agent communication system in a simple example MAS and its operation process are illustrated in Figure 2. The agent protocol operated by the system involves the following actors in the communications process. A *user*, which can be a human or a software application, invokes a request (intention) to the MAS to fulfill a specific need. A *client agent*, an AI agent in the MAS with which the user has direct access, translates the user's intent into requests for specific tasks that can be delegated to other agents. Then the client agent interacts with a *system controller*, which can be implemented as a central server or via a decentralized cooperation mechanism, to discover and select the appropriate *remote agents* that meet the task requirements. The agent communication protocol also specifies standards for agent descriptions and provides a mechanism for disseminating them. After agent discovery, the client agent begins operating communication sessions with the selected remote agents, including representing information in formats the remote agents can understand and managing message exchange patterns with them. These communication sessions utilize the computing and networking resources in the Internet infrastructure to ensure data transport between agents. Each

 doi:10.20944/preprints202602.0306.v1

remote agent is responsible for processing the delegated tasks and providing results to the client agent, which then renders the returned results into the final outcome and responds to the user.

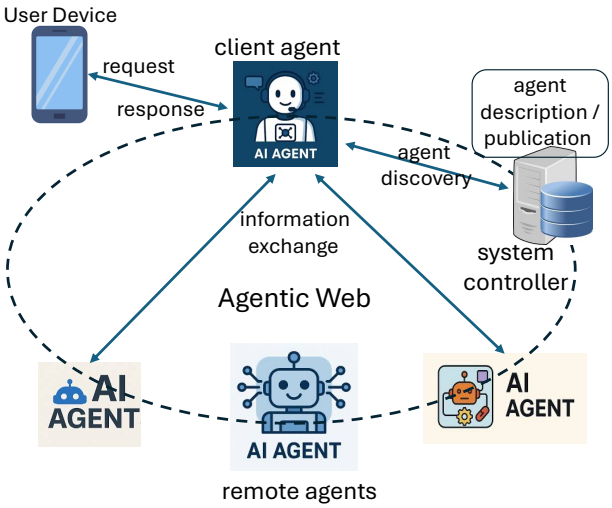

Figure 2. The agent communication process in Agentic Web.

2.4. Challenges to AI Agent Communications in Future Internet

Deploying MAS across an edge-network-cloud continuum to build an Agentic Web in the future Internet introduces unique challenges related to heterogeneity, scalability, dynamism, and efficiency that stress agent communication technologies and protocols.

2.4.1. The Challenge of System Heterogeneity

The Agentic Web in the future Internet is inherently a highly heterogeneous system. It comprises a wide range of AI agents with diverse skills, roles, and configurations. These agents are developed by different vendors, operated in different organizations, and tailored toward different application objectives. Furthermore, the AI agents in Agentic Web can be deployed across hosts with heterogeneous implementation technologies and various computing capacities, ranging from battery-powered edge devices to cloud servers in data centers, and connected to a variety of network environments, from wireless mobile networks to high-speed data center networks. Therefore, agent communication to achieve interoperability among diverse AI agents deployed across the Internet infrastructure, using heterogeneous networking and computing technologies, is particularly challenging.

2.4.2. The Challenge of Massive Scalability

The Agentic Web is expected to comprise a large number of agents, forming numerous MASs hosted on a vast array of edge/cloud devices distributed across large-scale Internet infrastructure, to serve a massive group of customers running various agentic AI applications. Information exchange among a large number of AI agents deployed across the Internet may generate a huge volume of communication traffic that overwhelms bandwidth-constrained networks, especially at the Internet edge. Managing and coordinating communications among numerous, geographically dispersed agents to accommodate potentially large volumes of information exchange presents significant scalability hurdles.

2.4.3. The Challenge of Environmental Dynamicity

Agentic Web in the future Internet will form a highly dynamic environment for inter-agent communication. The multi-agent interactions in the Agentic Web are expected to be dynamic, with individual agents regularly joining or leaving a MAS and cooperation patterns among agents evolving. The edge-network-cloud continuum in the Internet, which provides the infrastructure platform for the Agentic Web, is also highly dynamic. The availability of computational and communication

 doi:10.20944/preprints202602.0306.v1

resources in the infrastructure keeps varying; for example, edge devices may be switched on and off to save energy, while network bandwidth may fluctuate due to changes in transmission link states. Furthermore, the mobility of both mobile devices (as agent users) and edge devices (as agent hosts) introduces additional dynamism to agent communications.

2.4.4. The Challenge of Resource Constraints

The infrastructure for Agentic Web in the future Internet is built on the edge-network-cloud continuum, where computational and communication resources may be constrained, especially at the edge. Edge devices hosting mobile agents often have restricted processing power, storage space, and battery capacities. Network connections among edge devices often rely on wireless channels with limited bandwidth. On the other hand, the LLMs that underpin modern AI agents are particularly resource-intensive and often exceed the capacity of most edge devices. Multi-modal agentic AI applications that process multimedia information may also trigger high-volume data transmission among collaborative agents. Therefore, resource constraints in Internet infrastructure and resource-demanding agentic AI applications together pose a severe challenge to the efficiency of agent communication for Agentic Web in the future Internet.

2.4.5. Intertwined Challenges to Agent Communications in Future Internet

The aforementioned challenges to agent communications in the future Internet are deeply and subtly intertwined. For example, environmental dynamicity requires frequent control actions and updates (e.g., to rediscover the optimal agents and adjust communication patterns). However, in a large-scale system with limited resources in some parts, this constant churn can create a signaling storm that consumes the very network and computational resources that are already severely constrained. Furthermore, the system's heterogeneity makes it even harder to design uniform policies to manage this dynamic, resource-constrained environment. Therefore, any viable agent communication technology for Agentic Web in the future Internet must be designed to address this entire complex system of interacting constraints, rather than a single challenge in isolation.

3. Representative Agent Communication Protocols

3.0.1. Landscape of Agent Communication Protocols

Research on communication technologies for AI agents has recently gained momentum, leading to a wide variety of protocol designs. Although excited by the rapid progress, researchers who have just entered this emerging field might find the broad range of protocols developed by various organizations, each focusing on different aspects of agent communication systems, overwhelming or even confusing. In this section, we present a landscape of current agent protocol developments, with the hope of providing readers with a big-picture view of the latest status of this area, without being exhaustive about every single protocol.

As depicted in Figure 3, the current representative agent protocols can be categorized based on their main design objectives into three groups: i) protocols for controlling communications among autonomous AI agents, ii) context-oriented protocols, represented by the Model Context Protocol (MCP) [12], designed for communications between an AI agent and its context, typically for using tools and/or accessing data sources; and iii) user-oriented protocols, for example the Agent-User Interaction protocol (AG-UI) [13], focusing on interactions between an agent and front-end applications. Protocols in these three categories complement each other and are all needed in a MAS; on the other hand, the inter-agent communication protocols play the key role in realizing the Agentic Web due to their explicit focus on multi-agent interactions.

 doi:10.20944/preprints202602.0306.v1

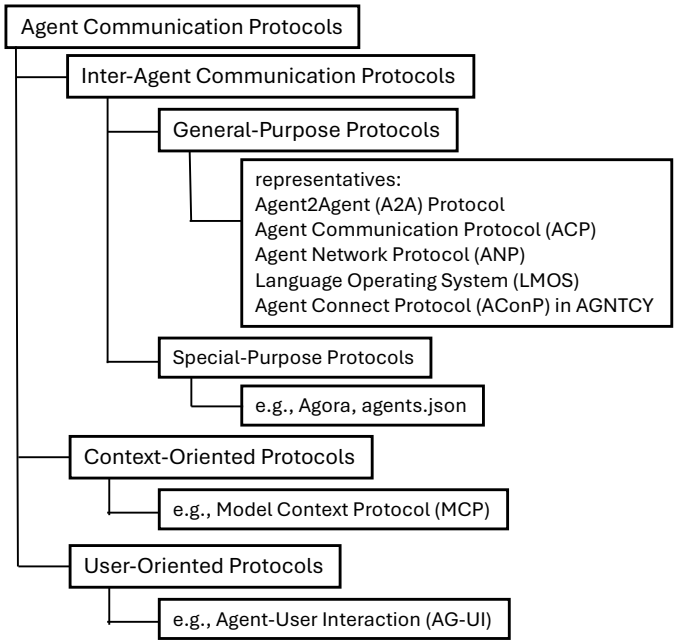

Figure 3. A Landscape of Representative Agent Communication Protocols.

The protocols for inter-agent communication can be further divided into general-purpose and special-purpose protocols. Each general-purpose protocol provides a full stack of functions for both system management and information transport in general MAS scenarios, while special-purpose protocols focus on specific aspects of inter-agent communication. Representative general-purpose agent protocols include the Agent2Agent Protocol (A2A) [14] initially developed by Google, Agent Communication Protocol (ACP) [15] initiated by IBM, Agent Network Protocol (ANP) [16] developed by the open-source community, Language Model Operating System (LMOS) [17] from the Eclipse Foundation, and Agent Connect Protocol (AConP) designed in the AGNTCY project [18]. Examples of general-purpose protocols include the Agora protocol [19] for negotiation of communication schemes between AI agents and agents.json [20] for agent description and discovery. The survey presented in this article focuses on comprehensive, general-purpose protocols for inter-agent communications.

3.1. Agent2Agent Protocol (A2A)

The Agent2Agent (A2A) protocol [14], initiated by Google and now an open standard managed by the Linux Foundation, is arguably the most widely adopted industry-backed protocol for inter-agent communications [9]. It is explicitly designed to allow agents from different vendors, built on diverse frameworks, to collaborate securely across enterprise environments. To achieve this objective, the protocol employs a client-server model based on established enterprise standards and follows a principle of "opaque execution" to enable agent collaboration without sharing their internal memory, prompts, or proprietary tool implementations [8].

In the A2A protocol architecture, client agents initiate requests and delegate tasks to one or more remote agents, which expose their access endpoints, accept tasks, process them, and return results. The Agent description in A2A uses the *Agent Card*, a structured JSON [21] file that typically contains metadata about an agent, including its name/identifier, a summary of its functions or "skills," and the endpoint URL for accessing the agent. The A2A protocol supports multiple schemes for agent publication/discovery. The open discovery method allows agents to make Agent Cards accessible via a standard "well-known" path *https://agent-domain/.well-known/agent.json*. The registry-based approach publishes Agent Cards on a central server, from which available clients can be discovered. The API-based discovery approach limits agents to offering customized API endpoints and delivering Agent Cards only to authenticated clients.

 doi:10.20944/preprints202602.0306.v1

The A2A protocol employs a structured approach for information representation, primarily utilizing JSON-RPC 2.0 as the standardized data exchange format for requests and responses. The fundamental unit of work is a *Task*, which has a unique ID and a defined lifecycle (e.g., submitted, working, completed, failed). The tangible output generated by the remote agent (e.g., a file, a block of text, structured JSON data) is referred to as an *Artifact*. The A2A protocol supports both synchronous and asynchronous message exchange between agents. In the synchronous mode, the client agent sends a request and waits for a direct response from the remote agent. A2A provides two asynchronous messaging schemes for long-running processes. Server-Sent Events (SSEs) [22] are used to stream updates from a remote agent to the client in real time. A client agent can also register webhooks to receive asynchronous push notifications whenever the task's status changes.

3.2. Agent Communication Protocol (ACP)

The Agent Communication Protocol (ACP) was an open standard introduced by IBM Research, associated with its BeeAI platform [15]. In the third quarter of 2025, ACP and its innovations have been incorporated into the A2A protocol, and both are now governed by the Linux Foundation. In this section, we briefly review the key features of the original ACP's design, as it addresses some of A2A's limitations and is particularly tailored for resource-constrained scenarios, such as edge computing systems.

The ACP protocol follows a different architectural philosophy than A2A. While A2A is based on JSON-RPC, ACP was designed as a REST-native protocol. It uses simple, well-defined REST endpoints and standard HTTP patterns (GET, PUT, POST, etc.) for all interactions [23]. The simplicity that comes from avoiding RPC and adopting RESTful interactions often makes ACP more lightweight, integration-friendly, and better aligned with modern edge-native development.

The ACP design achieves modality-agnosticity by using standard HTTP MIME types (e.g., application/json, image/png) to identify content, making the protocol inherently extensible to any data format (text, images, video) without requiring protocol changes. The protocol was designed primarily for asynchronous communications, ideal for long-running agent tasks, while still supporting synchronous requests.

Offline agent discovery is arguably ACP's most significant contribution. The *Agent Manifest* (ACP's equivalent of the Agent Card) was designed to support offline discovery, allowing agent metadata to be embedded directly in its distribution package (e.g., as labels on a container image). This feature enables agent discovery in scenarios where an agent may be inactive (and thus not serving its manifest) until needed, thereby better aligning with resource-constrained edge computing environments.

3.3. Agent Network Protocol (ANP)

Agent Network Protocol (ANP) is an open-source agent protocol developed to enable interoperability among agents across heterogeneous domains. ANP presents a decentralized vision for agent communication and adopts a peer-to-peer (P2P) architecture designed from the ground up for decentralization. The ANP architecture consists of three layers: the Identity and Encrypted Communication layer, the Meta-Protocol layer, and the Application Protocol layer [16].

The Identity and Encrypted Communication layer leverages W3C Decentralized Identifiers (DIDs) [24] to establish identity authentication and enable end-to-end encrypted communication across different platforms/domains. A DID provides a cryptographically verifiable, self-sovereign identity that is not controlled by any central authority [25].

Instead of being restricted to a fixed set of predefined schemes for inter-agent information exchanges, the Meta-Protocol layer in ANP allows agents to describe their requirements and constraints in natural language, which are then processed by LLMs for protocol negotiation, to enable agents to dynamically adjust communication parameters, define data formats, or even collaboratively generate the necessary protocol on the fly. On the other hand, to reduce communication costs, agents should avoid protocol negotiation whenever possible. Therefore, ANP designs a series of semantic Web

 doi:10.20944/preprints202602.0306.v1

standard-based schemes on the Application Protocol layer to make inter-agent communication more efficient and cost-effective.

A key function of the Application Protocol layer is agent description and discovery. This layer uses JSON-LD [26] to achieve semantic interoperability. By embedding linked data in a standard JSON format, JSON-LD allows agent descriptions to be both easily parseable by machines and semantically rich. ANP specifies a flexible, dual-mechanism approach for agent discovery. The active discovery method enables an agent to discover available agents within a known domain without a central registry by polling a standard URI: https://domain/.well-known/agent-descriptions. The passive discovery method creates a central or federated search service agent to which the available agents can submit the URLs of their agent descriptions.

3.4. Language Model Operating System (LMOS)

The Language Model Operating System (LMOS) is an open-source specification developed by the Eclipse Foundation for building and running enterprise-ready multi-agent systems [17]. Its core innovation is not the invention of a new agent protocol from scratch, but the strategic adaptation of the W3C Web of Things (WoT) standard [27] as the foundational architecture for inter-agent communication, thereby providing LMOS with mature, standardized solutions for agent description, discovery, and communication.

In LMOS, AI agents and tools are described using the WoT Thing Description (TD), a machine-readable, JSON-LD formatted document that semantically describes a "Thing" and its network-facing interfaces. The key information about an agent in TD includes Properties (data the agent exposes), Actions (functions the agent can perform), and Events (notifications the agent can emit).

The WoT-based agent discovery in LMOS supports three primary discovery methods. The centralized method allows agents to publish their TDs to a registry, from where any agent in the network can perform semantic queries to discover other agents. The local discovery method in LMOS enables agents to employ mDNS (Multicast DNS) [28] or DNS-SD (DNS Service Discovery) [29] to advertise their presence and discover peers on local area networks. LMOS also supports peer-to-peer metadata propagation, thereby eliminating reliance on a single registry and enabling decentralized agent discovery.

A key attribute of LMOS, inherited from the WoT architecture, is its protocol-agnostic nature. The TD for agent description separates the abstract inter-agent interaction from its concrete protocol implementation. This is achieved through forms specified in the TD, which provide protocol bindings that specify how agent communications use various transport protocols, such as HTTP, WebSockets [30], Message Queuing Telemetry Transport (MQTT) [31], and Constrained Application Protocol (CoAP) [32]. This allows a single LMOS agent to expose its capabilities simultaneously via HTTP (for a cloud orchestrator) and MQTT (for a local edge message bus), thereby ensuring maximal interoperability and flexibility.

3.5. Agent Connect Protocol (AConP) in the AGNTCY project

The Agent Connect Protocol (AConP) is the agent communication scheme developed in the AGNTCY project, now under the Linux Foundation's stewardship, with the explicit goal of building a modular, composable infrastructure stack for an open, decentralized Internet of Agents. The AGNTCY project provides a suite of components, including Open Agent Schema Framework (OASF), Agent Directory Service (ADS), Secure Low-latency Interactive Messaging (SLIM), and AConP, to address different aspects of the agent communication problem [33].

OASF is an extensible data model based on the Open Container Initiative (OCI) specification [34] that provides a standardized ontology for describing an agent's attributes, capabilities, and metadata. ADS is a distributed directory system designed to store and manage agent descriptions using OASF-defined schemas and to employ a distributed hash table (DHT) for scalable agent discovery. AConP is a REST-based API specified in OpenAPI [35] that defines a standard interface for agent invocation and configuration, with primary operations focused on the lifecycle management of "Runs" (executions)

 doi:10.20944/preprints202602.0306.v1

and "Threads" (conversational context). SLIM extends gRPC's standard request/reply and streaming patterns with native publish-subscribe (pub/sub) capabilities. SLIM also enables efficient transmission of binary payloads, thereby better supporting lightweight communications in resource-constrained environments.

The AGNTCY project presents a decentralized system for managing and verifying the identities of agents, tools, and multi-agent systems by leveraging W3C DIDs and Verifiable Credentials (VCs) [36]. In AGNTCY, a trusted "Issuer" (e.g., an organization such as Google) cryptographically signs an "Agent Badge," which is a W3C VC that attests to an agent's claims. This "Agent Badge" contains the agent's DID, its OASF schema definition, and its provable capabilities and provenance. When two agents need to start a communication session, one can present its VC, and the other can cryptographically verify the issuer's signature and trust the claims without contacting a central authority.

3.6. Mergence of A2A and ACP

The parallel development of multiple protocols, each targeting distinct agent communication environments, introduces a new interoperability challenge at the protocol level in the agentic AI ecosystem. For instance, an "edge agent" running ACP could not readily offload a computationally intensive task to a "Cloud agent" using A2A, necessitating complex translation layers or "bridge agents." On the other hand, all the protocols share fundamental goals: enabling agents to communicate across frameworks, organizations, and technology stacks [37].

The merger of A2A and ACP under the neutral governance of the Linux Foundation (LF AI&Data) to form a new A2A protocol is a recent community effort to address this challenge [38]. The merged A2A protocol has a hybrid architecture that integrates ACP's edge-native resilience with A2A's cloud-oriented design, and incorporates some features from other protocol developments (e.g., the AGNTCY project). In this section, we highlight the key enhancements introduced in the merged A2A protocol across three areas: agent description, agent discovery, and information transport.

The merged protocol embraces OASF, originally contributed by the AGNTCY project, and integrates the ACP's metadata schema into its Agent Card mechanism, which is inherited from A2A. The Agent Card is now a JSON-LD document with a schema that introduces critical fields for operational context, including agent resource constraints in addition to agent identity and capability information. This integration addresses a major deficiency in the legacy A2A and significantly enhances the protocol's capability to support interoperability across diverse agents and heterogeneous underlying infrastructures.

The new A2A protocol implements a dual-mode agent discovery system that reconciles the online requirements of the cloud with the offline realities of edge environments. For active, network-connected agents, the protocol conducts online discovery using the enhanced Agent Cards published at a registry or a well-known URI. The protocol also supports an offline agent discovery model. In this mode, the Agent Card metadata is serialized and embedded directly in the agent's distribution artifact, enabling client agents to discover available remote agents without waking their radios or processors.

The merged A2A protocol adopts a protocol negotiation strategy that supports multiple transport schemes, optimizing for the specific network environment [39]. The transport mechanisms supported by the new A2A include JSON-RPC 2.0, inherited from the legacy A2A, which is suitable for cloud-based enterprise environments, the REST-native messaging adopted in legacy ACP, which is more appropriate for edge/IoT applications, and the gRPC-based SLIM scheme originally developed in AGNTCY for low-latency and high-throughput message transport. The shift from a "one-size-fits-all" monolithic transport approach to a "meta-protocol" mechanism that adaptively utilizes an appropriate scheme based on communication requirements and deployment environments may greatly improve the protocol's performance in the highly dynamic future Internet.

4. Key Technologies for Agent Communications in Future Internet

In this section, we review the key technologies employed by representative inter-agent communication protocols to assess their readiness to the challenges of heterogeneity, scalability, dynamicity,

 doi:10.20944/preprints202602.0306.v1

and efficiency in building an Agentic Web in the future Internet. We present a taxonomy for and conduct analysis of the technologies for agent description, publication and discovery, identification and authentication, information representation, and information exchange, respectively, in the following subsections, and then give a summary of comparative analysis in Table 1 at the end of this section.

Table 1. Key technologies for agent communication and their capabilities of addressing challenges in the future Internet

Key Technologies for Agent Communications	**Capabilities of Addressing Internet Challenges**			
	Heterogeneity	**Scalability**	**Dynamicity**	**Efficiency**
Agent Description				
Syntactic descriptions	Low	Medium	Medium	High
Semantic descriptions	High	Medium	High	Low
Agent Publication & Discovery				
Centralized discovery	High	Low	Low	Low
Decentralized discovery	Medium	High	High	Medium
Agent Identification & Authentication				
CA-based authentication	High	Low	Low	Low
Peer-to-peer authentication	Medium	High	Medium	Medium
Information Representation				
Structured data formats	Medium	High	Medium	High
Unstructured natural languages	High	Medium	High	Low
Information Exchange				
Synchronous mode	Medium	Medium	Low	Medium
Asynchronous mode	High	Medium	High	High

4.1. Agent Description

Representative approaches employed by common agent communication protocols for agent description can be categorized as syntactic, semantic, and API schema-based.

4.1.1. Syntactic-based Methods

A syntactic description focuses on the form, structure, and grammar of metadata that describe AI agents' attributes and capabilities, providing a baseline agent description for interoperability. Syntactic description methods often use JSON files to construct Agent Cards or manifests. Another common approach to syntactic descriptions uses the OpenAI-based API schema [35] to describe AI agents' interfaces. Both schemes have been widely adopted in virtually all the representative agent communication protocols.

Heterogeneity: The lack of semantic expression in both JSON files and OpenAPI schema limits their abilities to handle ambiguity in heterogeneous agents' descriptions. Also, JSON files and API schemas often lack a standard for describing non-functional constraints (e.g., the host device's computing capacity), making it difficult to perform resource-aware agent discovery and selection across the heterogeneous edge-network-cloud continuum.

Scalability: Syntactic agent descriptions are typically lightweight and easy to index; therefore, they can better support high-volume queries than complex ontological descriptions. On the other hand, the constraint on syntactic information limits its ability to support scaling communications across diverse agents.

Dynamicity: OpenAPI is designed to describe static interfaces and does not inherently support dynamic updates. While the JSON file itself is static, its fields can be updated to adapt to changes in agent capabilities and deployment environments. However, real-time state changes (like battery drain) are better handled by a separate status protocol than by constantly updating a static manifest file.

Efficiency: JSON parsers are ubiquitous and highly optimized, minimizing overhead for both data processing and transmission. Code generators can compile OpenAPI specifications into efficient,

 doi:10.20944/preprints202602.0306.v1

native client libraries, eliminating the need for runtime schema introspection. Therefore, syntactic descriptions can be processed efficiently and also facilitate efficient agent discovery and selection.

4.1.2. Semantic Descriptions

A semantic description focuses on describing the meaning, intent, and context of the agent's capabilities using shared vocabularies and ontologies. JSON-LD with the @context field is a common format for presenting semantic agent descriptions, which has been employed in ACP, ANP, and LMOS. The OASF, which leverages a custom JSON-based record format inspired by the Open Cybersecurity Schema Framework (OCSF) [40], also provides semantic agent descriptions.

Heterogeneity: Semantic formats such as JSON-LD are explicitly designed to address the heterogeneity problem. By linking data to shared ontologies, they ensure that an agent from one vendor can understand the capabilities of an agent from another without ambiguity.

Scalability: While the semantic descriptions are text and scale well, the reasoning required to process them does not. As the number of agents grows, maintaining and synchronizing shared ontologies across the large-scale Internet becomes a significant management bottleneck.

Dynamicity: JSON-LD allows for flexible, decentralized extensibility. An agent can dynamically add new capability terms to its description without breaking existing parsers, which is vital for the evolution of multi-agent systems in the dynamic Internet environment.

Efficiency: Parsing and validating formal semantics is computationally expensive. JSON-LD, in particular, introduces high processing overhead for resolving linked data contexts, which can be prohibitive in resource-constrained edge computing environments.

4.2. Agent Publication and Discovery

Technologies for agent publication and discovery make agents' descriptions available to others. Typical publication and discovery mechanisms in common agent protocols can be classified as centralized or decentralized based on their control structures.

4.2.1. Centralized Methods

Centralized publication and discovery approaches require agents to actively register their descriptions with a central server or store them at a standardized web location, from which other agents can search and find agents that meet their requirements. The centralized approach has been leveraged in A2A, ACP, and LMOS.

Heterogeneity: A central registry can enforce a common description schema, such as an Agent Card or OASF, enabling heterogeneous agents to describe themselves and discover others in a standardized, interoperable way, thereby greatly enhancing the system's ability to address heterogeneity.

Scalability: Centralized agent publication and discovery methods may suffer from scalability issues due to potential performance bottlenecks and a single point of failure, especially in the large-scale Internet, where a vast number of highly distributed agents are deployed.

Dynamicity: Centralized methods require agents to explicitly register and update their descriptions, which becomes infeasible in highly dynamic networks where agents on mobile devices constantly change locations or lose connectivity. Furthermore, discovery fails entirely if the central server or location becomes unreachable, for example, due to a network or power outage.

Efficiency: With a centralized approach, every discovery action requires a round-trip communication session to a (potentially distant) central location, resulting in higher discovery latency and additional network overhead that reduces the efficiency of agent communication.

4.2.2. Decentralized Methods

Decentralized publication and discovery methods disseminate agent descriptions without a central server or web location, typically via peer-to-peer communication or local broadcast/multicast. This strategy has been adopted in ANP and AGNTCY, and the offline discovery in ACP and merged A2A is also considered a decentralized method.

 doi:10.20944/preprints202602.0306.v1

Heterogeneity: Decentralized publication/discovery mechanisms focus on disseminating agent descriptions without imposing a common schema; therefore, they typically require semantic descriptions, such as JSON-LD, to ensure interoperability across heterogeneous agents.

Scalability: Decentralized approaches distribute the discovery process across agents and eliminate performance bottlenecks and single points of failure, thereby greatly enhancing the scalability of agent communication systems.

Dynamicity: The peer-to-peer messaging or local multicast/broadcast used in decentralized agent discovery allows individual agents to update their descriptions as needed, making the system more adaptive to system dynamics. Offline discovery is especially well-suited for intermittently connected devices common in dynamic networks, such as IoTs.

Efficiency: On the one hand, decentralized discovery in local networks has very low latency, and offline discovery even generates zero network traffic. On the other hand, naive peer-to-peer messaging or network-wide broadcasts could cause overwhelming communication overheads in resource-constrained edge computing environments.

4.3. Agent Identification and Authentication

Representative technologies for the assignment and verification of agents' identities to establish trust in agent communication systems typically employ either central authority (CA)-based methods or cryptography-based peer-to-peer methods.

4.3.1. CA-based Authentication

Identification/authentication methods in this category rely on a central Identity Provider (IdP) to issue and validate credentials for establishing trust among autonomous agents. A common implementation of this strategy is the OAuth 2.0-based scheme [41] used in A2A and ACP.

Heterogeneity: OAuth 2.0 is a mature, ubiquitous standard supported by all cloud platforms and most enterprise information systems. Therefore, it facilitates interoperability across autonomous agents to face the heterogeneity challenge.

Scalability: This authentication model is architecturally bound to a central IdP for issuing and validating tokens, which becomes a performance bottleneck and a single point of failure, potentially degrading the scalability of agent communications.

Dynamicity: CA-based methods require each agent to contact a central IdP to obtain and refresh its time-bound access token; therefore, authentication may fail if the IdP becomes unreachable, which can affect performance in highly dynamic scenarios such as IoT with intermittent connections.

Efficiency: The regular communication between each agent and the IdP required for token acquisition and validation introduces not only latency in agent authentication but also additional bandwidth consumption, which may reduce efficiency for agent communications.

4.3.2. Peer-to-Peer Authentication

Decentralized peer-to-peer identification and authentication allow individual agents to control their own identities and verify others' identities without relying on any central authority. Both ANP and AGNTCY (AConP) embrace this strategy in their DID/VC-based agent identity/authentication schemes.

Heterogeneity: DID is a W3C standard widely adopted in edge and cloud computing, therefore facilitating interoperability across heterogeneous agents. However, the standard requires complex privacy key management and cryptographic operations, which may not be feasible in heterogeneous edge-network-cloud environments comprising a significant number of resource-constrained devices.

Scalability: Peer-to-peer authentication methods improve scalability in agent communications by allowing individual agents to create their own identifiers and verify credentials, thereby avoiding potential performance bottlenecks and single points of failure.

 doi:10.20944/preprints202602.0306.v1

Dynamicity: By allowing self-contained agent identities that can be verified by agents' digital signatures and resolved by peers through their public DID documents, peer-to-peer authentication methods are well-suited to the dynamic edge environments of the future Internet.

Efficiency: Once the public DID document is resolved, decentralized authentication is a local cryptographic operation that is more efficient than the network-bound OAuth validation required by CA-based methods. On the other hand, privacy key management and cryptographic processes may be computationally overwhelming for resource-constrained devices, especially at the network edge.

4.4. Information Representation

Information representations in agent communications define the formats for presenting the information exchanged between AI agents. Technologies for achieving this objective in representative inter-agent communication protocols leverage either structured data formats or unstructured natural language.

4.4.1. Structured Data Formats

The structured data formats used in agent communication protocols for information representation include JSON and its extension JSON-LD. JSON is a lightweight, text-based data format that contains no built-in semantics, while JSON-LD is designed to represent linked data across systems using URIs and to enrich JSON data with semantic meaning. Structured data can be transported between agents via either JSON-RPC (as in the A2A and ANP protocols) or RESTful JSON messaging (as in ACP, LMOS, and AConP).

Heterogeneity: Simple JSON is a syntactic standard that lacks the formal semantics needed to resolve ambiguity across agents' specific schemas, leading to poor interoperability among heterogeneous AI agents. JSON-LD carries both data and its formal, logical meaning, ensuring unambiguous interpretation and thus significantly enhancing its ability to address the heterogeneity challenge in agent communication. While the function-specific interface of JSON-RPC may hinder interoperability across heterogeneous agents, RESTful JSON uses the universal HTTP messages (e.g., GET, POST, PUT, DELETE) and standard URIs to provide uniformity, allowing various agents hosted on heterogeneous edge-cloud devices to interact.

Scalability: JSON and JSON-LD have complementary effects on the scalability of agent communications. JSON is well-suited for agent messaging at scale due to its low serialization overhead and fast parsing, but at the cost of poor interoperability. Although JSON-LD enables semantic interoperability that facilitates scalable discovery and capability matching across diverse agents, it also increases message size and computational overhead for context processing and reasoning. Therefore, using JSON for runtime message exchange and JSON-LD for control-plane functions, such as agent description and discovery, is an appropriate strategy to address the scalability challenge.

Dynamicity: The self-describing nature of JSON/JSON-LD enables dynamic, adaptive data queries and updates between agents and is therefore well-suited to highly dynamic edge computing systems, where nodes frequently join, leave, or evolve. Furthermore, LLMs are natively optimized for generating and parsing JSON, enabling an LLM-driven agent to dynamically create new message structures to adapt to emerging situations. The stateless RESTful JSON allows an agent to resume interrupted communications without restoring session state, thereby facilitating system resilience. On the other hand, RESTful JSON lacks native support for real-time "push" updates, requiring inefficient polling to handle dynamic events.

Efficiency: JSON provides low serialization overhead, fast parsing, and broad tooling support, making it efficient for information transport between AI agents. JSON-LD introduces additional processing and parsing overhead for interpreting its linked-data graph structure, making it unsuitable for resource-constrained scenarios such as IoTs. Therefore, choosing data formats that best fit the type of information exchanged and the deployment environment is critical to maintaining efficiency in agent communication.

 doi:10.20944/preprints202602.0306.v1

4.4.2. Unstructured Natural Languages

Since virtually all modern AI agents are built on LLMs with strong natural language processing capabilities, human languages provide a natural means of representing information in inter-agent communication to support multi-agent collaboration [42].

Heterogeneity: Natural language functions as a "universal solvent" for interoperability across heterogeneous AI agents. Unlike rigid protocols that require a prior schema integration, natural language allows agents to use LLMs to infer intent across disparate terminologies. This capability enables highly heterogeneous agents to collaborate spontaneously, effectively addressing the heterogeneity challenge.

Scalability: While natural language supports the open-ended collaboration conceptually, its scalability in practice is constrained by LLMs' limited context windows. As the number of agents increases, the conversation history grows superlinearly, forcing agents to compress or "forget" information, leading to a high "coordination tax" and information fragmentation [43]. Consequently, natural language-based information representations struggle to scale in highly distributed networks.

Dynamicity: Natural language offers exceptional adaptability in volatile network environments. It allows agents to dynamically discover peers and negotiate capabilities using high-level intents rather than static endpoints, making agent communication resilient to changes in both agents' capabilities and infrastructure resources. Furthermore, agents can utilize natural language (e.g., as in the Agora framework) to negotiate new structured data formats, thereby further enhancing the system's adaptability.

Efficiency: Natural language is inherently verbose, and processing it generates greater computational overhead and consumes more network bandwidth than structured formats. More critically, processing natural language using LLMs replaces deterministic, microsecond-scale parsing with probabilistic, millisecond-to-second-scale inference, creating an energy and latency bottleneck that makes it unsuitable for real-time control loops or resource-constrained edge devices.

4.5. Information Exchange

Information exchange technologies control the patterns for exchanging information between AI agents. The main approaches to information exchange in common agent communication protocols can be categorized as synchronous and asynchronous.

4.5.1. Synchronous Mode

Synchronous information exchange approaches follow a request/response pattern in which an agent sends a request message to a remote agent and waits for a response. Such a request/response process may be implemented using a variety of schemes, including HTTP REST calls, JSON-RPC, and gRPC.

Heterogeneity: On the one hand, request-response messaging schemes are typically well-defined standards that can be easily implemented in AI agents, thereby supporting interoperability between agents. On the other hand, synchronous information exchange is essentially a tightly coupled interaction, which poses challenges for communication among highly diverse AI agents.

Scalability: Although the stateless nature of the synchronous exchange pattern facilitates system scalability, it focuses on point-to-point connections between pairs of agents and may thus constrain scalability for communications across the massive number of AI agents deployed in the large-scale Internet.

Dynamicity: Synchronous information exchange approaches require a stable connection throughout the communication session, which may not be guaranteed in highly dynamic environments such as wireless mobile networks and edge computing systems. The tightly coupled interactions for synchronous information exchange are also ill-suited for agents hosted on mobile devices that may regularly move (thus change IP addresses) or go offline.

Efficiency: The efficiency of synchronous information exchange approaches varies with their specific implementation schemes. Among the typical schemes, gRPC is the most efficient and suitable

 doi:10.20944/preprints202602.0306.v1

for low-latency data transmission within a data center; HTTP REST calls have the highest overhead due to verbose (JSON or XML) payloads and heavy HTTP headers; and JSON-RPC falls between gRPC and HTTP REST calls in terms of efficiency.

4.5.2. Asynchronous Mode

Asynchronous information exchange approaches avoid tightly coupled interactions between agents, enabling more flexible communication patterns. Asynchronous information exchange can be implemented using a variety of schemes, including publish/subscribe, asynchronous streaming, and asynchronous queueing. The publish/subscribe scheme allows an agent (publisher) to send a message about a "topic" to a broker, which then delivers it to all agents that have subscribed to that topic. With asynchronous streaming, an agent (sender) opens a connection and pushes a continuous stream of messages to another agent (receiver). Using asynchronous queueing, a sender agent may push a message to a queue, from where a receiver agent can retrieve it later as needed.

Heterogeneity: The loose-coupling interactions enabled by asynchronous approaches allow information exchange between agents without being constrained by their implementations or hosting platforms, greatly facilitating communication across heterogeneous AI agents.

Scalability: Asynchronous approaches overcome the constraints of point-to-point sessions in the synchronous request-response pattern and are thus better suited to many-to-many information exchanges, thereby enhancing the scalability of agent communications. On the other hand, both the publish/subscribe broker and the asynchronous queue must be carefully designed to avoid creating a performance bottleneck that could degrade system scalability.

Dynamicity: Asynchronous information exchange approaches implemented with a publish/subscribe broker or an asynchronous queue are particularly well suited to highly dynamic communication environments with intermittent connectivity. With such approaches, a mobile edge agent can use a temporary network connection to publish data to a broker (or push it into a queue), which then delivers it to other agents (subscribers) when their connections are available. Agents can also dynamically update their subscriptions to various topics.

Efficiency: Asynchronous approaches are typically considered more efficient for information exchange than synchronous ones in most inter-agent communication scenarios. For example, MQTT, a common scheme for publish/subscribe operations, enables lightweight message delivery optimized for resource-constrained network environments such as IoTs and edge computing systems.

5. Unified Framework for AI Agent Communications in Agentic Web

In this section, we first analyze representative protocols for inter-agent communication and identify gaps between their capabilities and the demands of the Agentic Web. To address these gaps, we advocate a unified architectural framework for Agentic Web that embraces the virtualization and service-oriented paradigms for agent communication in the future Internet. We then discuss key topics and potential directions for future research to realize this Agentic Web framework.

5.1. Comparative Analysis of Agent Communications Protocols

5.1.1. A2A (Agent2Agent Protocol): A Cloud-Native Baseline

A2A is a representative agent communication protocol designed to support agentic AI applications in enterprise environments. Its technology bundle, leveraging HTTP, JSON-RPC, OAuth, and a centralized registry, is optimized for cloud-native data centers or enterprise networks, not the edge-network-cloud continuum across the Internet, and thus is not well prepared to address all the challenges introduced by the future Internet. In the Internet environment, the protocol may fail to scale due to its centralized registry bottleneck, suffer insufficient adaptability due to its online-only discovery and synchronous-first interaction schemes, and degrade communication efficiency due to HTTP/JSON-RPC overheads. While its opaque execution enables interoperability across diverse AI agents, the lack of infrastructure-related information in its Agent Card limits the protocol's ability to handle device-level heterogeneity.

 doi:10.20944/preprints202602.0306.v1

5.1.2. ACP (Agent Communication Protocol): An Edge-Native Contender

ACP is a protocol explicitly designed with local/edge autonomy as its focus, as reflected in its technology choices. It prioritizes efficiency with its lightweight, REST-native architecture and supports dynamicity through its async-first design with an offline agent discovery model. Its distributed architecture facilitates scalability; however, its "local-first" focus introduces scalability limitations (e.g., the absence of a scalable mechanism for agent discovery). A primary weakness of this protocol is its limited capacity to accommodate heterogeneity; it lacks built-in support for semantic agent descriptions, which risks brittle, ad hoc agent interactions and thus limits its performance in the future Internet, where a wide variety of AI agents are expected to be deployed.

5.1.3. ANP (Agent Network Protocol): A Decentralized Vision

ANP is an agent communication protocol designed following a decentralized vision. The key technologies leveraged by this protocol, for example, W3C DIDs for agent authentication and a P2P structure for agent discovery, are, in principle, well-suited to addressing scalability and dynamicity challenges. Its use of JSON-LD provides a strong solution for heterogeneity. However, the strengths of this protocol's forward-looking, decentralized architectural vision are offset by some designs ill-suited to the realistic edge-network-cloud continuum expected in the future Internet. The protocol may incur significant efficiency losses and moderate scalability limitations in resource-constrained edge computing environments, primarily due to its high computational overhead, for example, parsing JSON-LD and performing cryptographic operations for DIDs.

5.1.4. LMOS: A Hybrid Orchestration Model

The LMOS protocol aims to achieve a balance for a converged edge-network-cloud environment in the future Internet, primarily through an architectural framework that embraces pragmatic hybridity. It addresses scalability and dynamicity challenges by supporting both centralized and P2P mechanisms for agent discovery, and employing both enterprise OAuth and decentralized DIDs for agent identification/authentication. The protocol addresses heterogeneity through its transport abstraction layer, which bridges diverse network environments, combined with JSON-LD for semantic richness. The main weakness of this protocol, like ANP, lies in the potential efficiency costs introduced by some of its key functionalities, for example, parsing JSON-LD, on resource-limited edge/IoT devices.

5.1.5. AGNTCY: A Full-Stack Approach

Rather than a single protocol, AGNTCY offers a full-stack solution for agent communications, comprising four interrelated core components: OASF, ADS, SLIM, and AConP. The OASP allows agents to describe their operational context alongside functional skills, thereby enabling resource-awareness in agent description and discovery, which is required to address the heterogeneity challenge. The ADS implements a decentralized agent-discovery mechanism that is highly resilient to the large-scale edge-network-cloud continuum. The SLIM messaging scheme, built on gRPC and HTTP/2, offers a significant efficiency advantage over text-based JSON-RPC/RESTful JSON calls. On the other hand, the AGNTCY architecture design remains essentially cloud-centric and has not fully addressed opportunistic communication, delay-tolerance, or adaptive quality-of-service strategies needed in resource-constrained and dynamic edge computing systems.

5.1.6. Merged A2A Protocol: A Unified Standard

By synthesizing A2A's cloud-centric design with ACP's edge-native solution, the merged A2A protocol addresses the critical fragmentation between cloud and edge environments. By integrating A2A's opaque task-delegation model with ACP's embedded metadata and device-native capabilities, the protocol substantially improves interoperability through a dual-stack solution, which preserves JSON-RPC compatibility while introducing REST-aligned patterns suitable for constrained edge devices. The protocol enhances scalability by supporting both centralized and decentralized discovery with ACP's offline mechanism incorporated. The merged protocol adopts ACP's async-first design

 doi:10.20944/preprints202602.0306.v1

philosophy, making it resilient to intermittent connectivity in highly dynamic mobile edge environments. A hybrid authentication model supporting both OAuth and DIDs, ensuring trust management across both connected and disconnected operational contexts. Efficiency is also improved over the legacy A2A by adopting ACP's multi-part message structure, which allows handling of binary data without the overhead of pure JSON-RPC.

On the other hand, the highly diverse requirements of agentic AI applications necessitate deploying various multi-agent systems across heterogeneous infrastructures, making the "one-fit-all" strategy implemented through a single protocol for all agent communication scenarios less feasible in the future Agentic Web.

5.2. Limitations of the Current Agent Communication Protocols for Agentic Web

The survey and analysis presented in the previous sections indicate that, although encouraging progress has been made in the technologies and protocols for inter-agent communication, realization of the Agentic Web in the future internet still faces two main hurdles: the *Interoperability Crisis* and the *Infrastructure Gap*. All agent communication protocols are designed with interoperability among heterogeneous AI agents as their primary objective. However, the proliferation of agent protocol designs has led to the coexistence of competing protocols, thereby introducing interoperability issues on the communication level while addressing them at the agent level. The analysis of existing agent communication technologies has also shown that, although the current agent protocols are becoming more infrastructure-aware and providing various mechanisms with cloud-centric or edge-native designs, they overall still lack sufficient flexibility to adaptively choose the optimal mechanisms based on the current infrastructure status for meeting diverse agent communication demands, thus leaving the infrastructure gap unfilled.

Recent developments in agent communication technologies, exemplified by the AGNTCY framework and the merged A2A (combining the A2A and ACP protocols), indicate a trend toward a unified protocol for addressing interoperability issues at the protocol level, with enhanced resource awareness to address the infrastructure gap. However, the analysis reveals a significant disconnect: the technologies that are best for efficiency and dynamicity, which are critical for agent communication in edge computing environments, are often the worst for heterogeneity, which is a key requirement for enabling interoperability across heterogeneous agents. For example, agents in a cloud data center prefer high-throughput, low-latency gRPC (as pioneered by AGNTCY's SLIM), while agents on battery-powered mobile sensors tend to use lightweight, asynchronous transport schemes such as CoAP or MQTT. This implies that a single, monolithic protocol cannot satisfy all requirements of inter-agent communications in the vast, complex, and heterogeneous device-edge-network-cloud continuum of the future Internet.

We argue that the future of agent communication for the Agentic Web lies not in a single protocol but in a flexible, unified architectural framework that can accommodate the coexistence of hybrid protocols implemented with various technologies, and can adaptively leverage the appropriate protocol bundle to support the numerous MASs deployed upon heterogeneous infrastructures to meet the requirements of diverse agentic AI applications. The key requirements for realizing such a framework include the following two aspects: i) holisticity for cooperation across the layers of the architecture, including Agentic AI applications, multi-agent systems, the agent communication platform, and the underlying infrastructures; and ii) flexibility allowing coexisting hybrid agent communication protocols to be adaptively leveraged for meeting the diverse requirements of various MASs deployed upon heterogeneous infrastructures.

5.3. Service-Oriented Virtualization-based Framework for Agentic Web

Given the convergence of networking and edge/cloud computing in the future Internet, and inspired by the proven success of the virtualization paradigm and service-oriented architecture in both cloud/edge computing and future networking, we envision an Agentic Web architectural framework grounded in the principles of virtualization and service orientation.

 doi:10.20944/preprints202602.0306.v1

The key notion of virtualization lies in the decoupling of a function's capabilities from the (computational and communication) resources it consumes. Such decoupling allows virtualization to greatly facilitate a holistic architecture with cross-layer cooperation without being constrained by the implementation details of individual layers [44]. Virtualization also greatly enhances system flexibility by enabling diverse virtual functions to share a common infrastructure substrate, while each function uses a slice of the substrate isolated from the slices used by other functions [45].

Service orientation is an architectural principle that encapsulates system modules into self-contained, platform-independent *services* and enables loosely coupled interactions among heterogeneous modules through abstract service interfaces. Service-oriented architecture provides the flexibility and interoperability required for highly integrated, cross-platform, inter-domain communication environments and has thus been widely adopted and realized through the Everything-as-a-Service (XaaS) paradigm, not only in cloud/edge computing but also in the future Internet, e.g., via the Network-as-a-Service (NaaS) model [46].

A service-oriented virtualization-based architectural framework (SOVA) for Agentic Web is depicted in Figure 4. This architecture consists of the Virtualized Infrastructure layer at the bottom, the MAS Communication Platform layer in the middle, and the Agentic AI Application layer on the top.

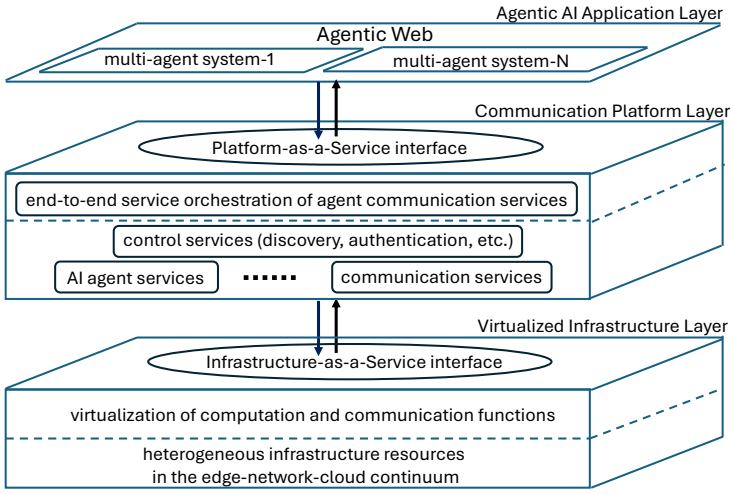

Figure 4. The SOVA architectural framework for Agentic Web

The Virtual Infrastructure layer comprises heterogeneous infrastructure resources that are encapsulated into virtual computation or communication functions and composed into various edge-network-cloud infrastructure services, which are then provisioned to the Platform layer via the Infrastructure-as-a-Service (IaaS) interface between the Infrastructure and Platform layers.

On the Platform layer, various AI agents and agent-communication functionalities are encapsulated as virtual services via the Agent-as-a-Service and Network-as-a-Service paradigms, each is hosted on its corresponding infrastructure service. Various multi-agent systems can be constructed through the composition of the needed service functions, including description, discovery, and orchestration of appropriate agent communication services, into end-to-end MAS services, which are then offered to the upper-layer agentic applications following the Platform-as-a-Service (PaaS) model through the interface between the Platform and Application layers.

On the top layer, various Agentic AI applications can be developed by leveraging the MAS services that the Platform layer has composed and provided to meet specific application requirements. These applications can then be delivered to end users as agentic AI services following the Application-as-a-Service (AaaS) model.

The virtualization principle, when embraced in this layered Agentic Web architecture, enables a set of virtual edge-network-cloud continuums, implemented using different agent communication protocols that leverage their corresponding slices of the infrastructure substrate to host various MASs.

 doi:10.20944/preprints202602.0306.v1

The service-oriented principle may be applied in two dimensions to the Agentic Web architecture: the vertical dimension across layers and the horizontal dimension within layers. In the vertical dimension, service-orientation allows loosely coupled interactions across inter-layer interfaces by abstracting layer implementations, thereby enabling holistic cross-layer cooperation in the architecture. In the horizontal dimension, MASs can be constructed by composing and orchestrating service functions that abstract agent communication capabilities, thus significantly enhancing the architecture's flexibility.

5.4. Possible Topics and Directions for Future Research

In this subsection, we identify some critical topics and potential directions for future research to realize the SOVA framework and enable the Agentic Web in the future Internet.

5.4.1. Flexible and Efficient Descriptions for Diverse Services

The XaaS paradigm adopted in the SOVA framework calls for further study of description approaches for the wide range of services across all layers of this framework, including network-compute infrastructure services, services that encapsulate agent control and information transport functions, and services provided by multi-agent systems. The expected service descriptions should strike a balance between rich semantics, which are required for interoperability and flexibility to address the challenges of heterogeneity and dynamicity, and simple representation, which is critical for efficiency and scalability in resource-constrained systems.

5.4.2. Capability-Based and Performance-Oriented Service Discovery and Selection

Service discovery and selection mechanisms are expected to play a crucial role in the SOVA framework and warrant further research. The discovery of the highly diverse services (including both agent and infrastructure services) in this framework and the selection of the appropriate set of services for building the required MASs should be both capability-based and performance-oriented. That is, the decision of service discovery and selection should be made based not only on what functionalities the available services provide, so that their composition can accomplish the tasks of the multi-agent system, but also based on the quality of service provisioning, so that the selected services can cooperate to meet the performance requirements of the MAS.

5.4.3. Inter-Domain and Cross-Layer Service Orchestration

Service orchestration is central to system management and control in the SOVA framework and thus presents a critical topic for future research. Two key aspects of service orchestration deserve thorough investigation. The first aspect is end-to-end cross-domain orchestration, which focuses on coordinating the diverse services across heterogeneous domains. The heterogeneity of domains may be caused by either their implementations (e.g., AI agents using different LLMs or utilizing heterogeneous infrastructures) or their administration (e.g., AI agents operated by different organizations). The other key aspect of service orchestration research is cross-layer cooperation, which requires the seamless integration of infrastructure-aware and performance-oriented service management to ensure that composite agent-communication services fully leverage underlying infrastructure resources while meeting end-to-end performance requirements. The promising service orchestration architecture in the SOVA framework is a hybrid approach that enables decentralized coordination across autonomous domains while allowing each domain to choose its own orchestration mechanism.

5.4.4. Balanced Layer-Decoupling with Cross-Layer Cooperation

Another interesting and important research topic is the technical strategy for balancing layer-decoupling and cross-layer cooperation in the SOVA framework. On the one hand, the virtualization principle of this framework requires decoupling higher-layer agent/communication service functions from their specific implementations in the underlying edge-network-cloud infrastructures, which is critical for enabling multiple virtual MASs to share the infrastructure substrate while utilizing diverse agent protocols tailored for different application requirements. On the other hand, infrastructure-

 doi:10.20944/preprints202602.0306.v1

aware, performance-oriented service orchestration requires cross-layer cooperation within the SOVA framework. Therefore, how to design cross-layer interfaces that balance these two competing demands remains an open issue for further study, and a key to solving it may lie in an optimal level of information abstraction that allows sufficient information to be exchanged between layers while maintaining the transparency of the underlying infrastructure to upper-layer functions.

6. Conclusions

In this paper, we survey the state-of-the-art agent communication protocols and technologies and assess their effectiveness in building the agentic Web when facing the challenges of heterogeneity, scalability, dynmicity, and efficiency introduced by the edge-network-cloud continuum in the future Internet. Our review demonstrates that although encouraging progress has been made in agent communication technologies, the current agent protocols are mainly designed with some specific deployment environments, either cloud-centric or edge-native systems, in mind as a focus, rather than having a holistic vision across the edge-network-cloud continuum in the future Internet. Therefore, no single available protocol is sufficiently ready to ensure the inter-agent communications required for the future Agentic Web. Our analysis also indicates that it is not feasible for any single, monolithic protocol design to satisfy all requirements for inter-agent communication across the vast, dynamic, and heterogeneous edge-network-cloud continuum of the Internet, supporting the diverse AI applications in the Agentic Web. Therefore, we argue that the future of agent communication for the Agentic Web lies not in a single protocol but in a flexible, unified architectural framework that can accommodate the coexistence of hybrid protocols implemented with various technologies, and can adaptively leverage the appropriate protocol bundle to support the numerous MASs deployed across heterogeneous infrastructures to meet the requirements of diverse agentic AI applications. In this direction, we advocate a service-oriented, virtualization-based framework for the Agentic Web and discuss key topics and potential directions for future research to realize the framework.

References

1. Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences* **2025**, *68*, 121101.
2. Sapkota, R.; Roumeliotis, K.I.; Karkee, M. AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenge. *arXiv preprint arXiv:2505.10468* **2025**.
3. Tran, K.T.; Dao, D.; Nguyen, M.D.; Pham, Q.V.; O'Sullivan, B.; Nguyen, H.D. Multi-agent collaboration mechanisms: A survey of LLMs. *arXiv preprint arXiv:2501.06322* **2025**.
4. Yan, B.; Zhou, Z.; Zhang, L.; Zhang, L.; Zhou, Z.; Miao, D.; Li, Z.; Li, C.; Zhang, X. Beyond self-talk: A communication-centric survey of LLM-based multi-agent systems. *arXiv preprint arXiv:2502.14321* **2025**.
5. Yang, Y.; Chai, H.; Song, Y.; Qi, S.; Wen, M.; Li, N.; Liao, J.; Hu, H.; Lin, J.; Chang, G.; et al. A Survey of AI Agent Protocols. *arXiv preprint arXiv:2504.16736* **2025**.
6. Zhang, R.; Liu, G.; Liu, Y.; Zhao, C.; Wang, J.; Xu, Y.; Niyato, D.; Kang, J.; Li, Y.; Mao, S.; et al. Toward Edge General Intelligence with Agentic AI and Agentification: Concepts, Technologies, and Future Directions. *IEEE Communications Surveys & Tutorials* **2026**, *28*, 4285–4318.
7. Duan, Q.; Wang, S.; Ansari, N. Convergence of networking and cloud/edge computing: Status, challenges, and opportunities. *IEEE Network* **2020**, *34*, 148–155.
8. Ray, P.P. A Review on Agent-to-Agent Protocol: Concept, State-of-the-art, Challenges and Future Directions. *Authorea Preprints* **2025**.
9. Duan, Q.; Lu, Z. Agent Communications toward Agentic AI at Edge – A Case Study of the Agent2Agent Protocol. In Proceedings of the 11th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom 2025), November 2025.
10. Ehtesham, A.; Singh, A.; Gupta, G.K.; Kumar, S. A Survey of Agent Interoperability Protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP). *arXiv preprint arXiv:2505.02279* **2025**.
11. Kong, D.; Lin, S.; Xu, Z.; Wang, Z.; Li, M.; Li, Y.; Zhang, Y.; Sha, Z.; Li, Y.; Lin, C.; et al. A Survey of LLM-Driven AI Agent Communication: Protocols, Security Risks, and Defense Countermeasures. *arXiv preprint arXiv:2506.19676* **2025**.

 doi:10.20944/preprints202602.0306.v1

12. Hou, X.; Zhao, Y.; Wang, S.; Wang, H. Model Context Protocol (MCP): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278* **2025**.
13. CopilotKit. Agent-User Interaction Protocol (AG-UI). https://docs.ag-ui.com/introduction. Accessed: 2026-1-31.
14. Google. Announcing the Agent2Agent Protocol (A2A). https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/. Accessed: 2026-1-31.
15. Besen, S.; Gutowska, A. What is Agent Communication Protocol (ACP)? https://www.ibm.com/think/topics/agent-communication-protocol. Accessed: 2026-1-31.
16. Chang, G.; Lin, E.; Yuan, C.; Cai, R.; Chen, B.; Xie, X.; Zhang, Y. Agent Network Protocol Technical White Paper. *arXiv preprint arXiv:2508.00007* **2025**.
17. Eclipse. Language Model Operating System (LMOS). https://eclipse.dev/lmos/. Accessed: 2026-1-31.
18. AGNTCY Origins. https://docs.agntcy.org/, 2025. Accessed: 2026-1-31.
19. Marro, S.; La Malfa, E.; Wright, J.; Li, G.; Shadbolt, N.; Wooldridge, M.; Torr, P. A Scalable Communication Protocol for Networks of Large Language Models. *arXiv preprint arXiv:2410.11905* **2024**.
20. WildcardAI. agents.json Specification. https://github.com/wild-card-ai/agents-json. Accessed: 2026-1-31.
21. IETF. RFC 8259: The JavaScript Object Notation (JSON) Data Interchange Format. https://datatracker.ietf.org/doc/html/rfc8259, 2017. Accessed: 2026-1-31.
22. WHATWG. HTML Living Standard: 9.2 Server-Sent Events. https://html.spec.whatwg.org/multipage/server-sent-events.html. Accessed: 2026-1-31.
23. Konopik, P.; Küng, J.; Rechberger, C.; Retschitzegger, W. REST: From architectural style to formal specifications for Web service engineering - A systematic literature review. *Journal of Systems and Software* **2020**, *166*, 110587.
24. W3C. Decentralized Identifiers (DIDs) Core Architecture, Data Model, and Representations, 2022.
25. Mazzocca, C.; Acar, A.; Uluagac, S.; Montanari, R.; Bellavista, P.; Conti, M. A survey on decentralized identifiers and verifiable credentials. *IEEE Communications Surveys & Tutorials* **2025**.
26. W3C. JSON-LD: A JSON-based Serialization for Linked Data version 1.1. https://www.w3.org/TR/json-ld11/, 2022. Accessed: 2026-1-31.
27. W3C. Web of Things (WoTs) Architecture. https://www.w3.org/TR/wot-architecture/Overview.html. Accessed: 2026-1-31.
28. IETF. RFC 6762: Multicast Domain Name System (DNS). https://datatracker.ietf.org/doc/html/rfc6762, 2013. Accessed: 2026-1-31.
29. IETF. RFC 6763: DNS-Based Service Discovery. https://datatracker.ietf.org/doc/html/rfc6763, 2013. Accessed: 2026-1-31.
30. IETF. RFC 6455: The WebSocket Protocol. https://datatracker.ietf.org/doc/html/rfc6455, 2011. Accessed: 2026-1-31.
31. OASIS. Message Queuing Telemetry Transport (MQTT) v5.0. https://www.oasis-open.org/standard/mqtt-v5-0-os/, 2019. Accessed: 2026-1-31.
32. IETF. RFC 7252: The Constrained Application Protocol (CoAP). https://datatracker.ietf.org/doc/html/rfc7252, 2014. Accessed: 2026-1-31.
33. Muscariello, L.; Pandey, V.; Polic, R. The AGNTCY agent directory service: Architecture and implementation. *arXiv preprint arXiv:2509.18787* **2025**.
34. OCI. Open Container Initiative Runtime Specification. https://specs.opencontainers.org/runtime-spec/?v=v1.0.2, 2025. Accessed: 2026-1-31.
35. OpenAPI. Specification (version 3.1.0). https://spec.openapis.org/oas/v3.1.0.html, 2021. Accessed: 2026-1-31.
36. W3C. Verifiable Credentials Data Model verson 2.0. https://www.w3.org/TR/vc-data-model-2.0/, 2025. Accessed: 2025-8-31.
37. Jankovics, V. AI Agent Protocols: ACP and A2A Unite. https://dotsquarelab.com/resources/acp-and-a2a-united. Accessed: 2026-1-31.
38. LFAI&Data. ACP Joins Forces with A2A. https://lfaidata.foundation/communityblog/2025/08/29/acp-joins-forces-with-a2a-under-the-linux-foundations-lf-ai-data/. Accessed: 2026-1-31.
39. CZ. Complete Guide: Agent2Agent (A2A) Protocol - The New Standard for AI Agent Collaboration. https://dev.to/czmilo/2025-complete-guide-agent2agent-a2a-protocol-the-new-standard-for-ai-agent-collaboration-1pph. Accessed: 2026-1-31.

 doi:10.20944/preprints202602.0306.v1

40. Agbabian, P. Understanding the Open Cybersecurity Schema Framework. Technical report, Linux Foundation, 2022.
41. IETF. RFC 6749: The OAuth 2.0 Authorization Framework. https://datatracker.ietf.org/doc/html/rfc6749, 2012. Accessed: 2026-1-31.
42. Huh, D.; Mohapatra, P. Grounding Natural Language for Multi-agent Decision-Making with Multi-agentic LLMs. *arXiv preprint arXiv:2508.07466* **2025**.
43. Qayyum, A.; Albaseer, A.; Qadir, J.; Al-Fuqaha, A.; Abdallah, M. LLM-Driven Multi-Agent Architectures for Intelligent Self-Organizing Networks. *IEEE Network* **2025**. https://doi.org/10.1109/MNET.2025.3605319.
44. Duan, Q.; Ansari, N.; Toy, M. Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks. *IEEE Network* **2016**, *30*, 10–16.
45. Zhang, S. An overview of network slicing for 5G. *IEEE Wireless Communications* **2019**, *26*, 111–117.
46. Duan, Q. Network-as-a-service in software-defined networks for end-to-end QoS provisioning. In Proceedings of the 23rd Wireless and Optical Communication Conference (WOCC), 2014, pp. 1–5.