

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: 12 April 2026

Z. Liang, Ed.  
E. Cui  
China Telecom Research Institute  
Y. Cheng  
University of Science and Technology Beijing  
9 October 2025

AgentDNS: A Root Domain Naming System for LLM Agents  
draft-liang-agentdns-00

## Abstract

This document describes AgentDNS, a root domain naming and service discovery system designed for Large Language Model (LLM) agents. Inspired by the traditional Domain Name System (DNS), AgentDNS enables agents to autonomously discover, resolve, and securely invoke third-party agent and tool services across different vendors. AgentDNS introduces a unified namespace, semantic service discovery, protocol-aware interoperability, and unified authentication and billing. The system aims to address interoperability, service discovery, and trust management challenges in large-scale agent collaboration ecosystems.

## 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 12 April 2026.

## Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Related Work . . . . .	5
3. AgentDNS Architecture . . . . .	7
3.1. AgentDNS System Overview . . . . .	7
3.2. Service Naming . . . . .	10
3.3. Service Discovery . . . . .	10
3.4. Service Resolution . . . . .	11
3.5. Unified Authentication and Billing . . . . .	12
4. AgentDNS Case Study . . . . .	13
5. Future Opportunities . . . . .	15
6. Conclusion . . . . .	16
7. IANA Considerations . . . . .	16
8. Security Considerations . . . . .	16
9. References . . . . .	16
9.1. Normative References . . . . .	16
9.2. Informative References . . . . .	17
Authors' Addresses . . . . .	21

## 1. Introduction

The rapid evolution of LLM agents across industries has introduced new challenges in enabling seamless multi-agent collaboration. Existing protocols such as the Model Context Protocol (MCP) [MCP-Spec] and Agent-to-Agent (A2A) protocol [A2A-Spec] have improved agent-tool interoperability and communication. However, these efforts lack a standardized root naming and discovery infrastructure to support autonomous cross-vendor interactions. As a result, collaborations between agents still demands significant manual effort, preventing the realization of true autonomous cooperation. The specific challenges are as follows:

- \* **The Service Discovery Challenge:** LLM agents typically generate an action plan, where each action may require calling external tools or agent services [RFC6763]. However, currently, services from different vendors are not standardized in naming or management, which forces developers to manually maintain service information for each tool or agent. This lack of standardization

makes it impossible for LLM agents to autonomously discover other agents or tool services, hindering seamless integration and collaboration between agents across different platforms.

- \* **\*The Interoperability Challenge:** Currently, different vendors' agents or tool services support various interoperability or communication protocols, with typical examples including MCP, A2A, and ANP (Agent Network Protocol Project 2025) [ANP-Doc]. We anticipate that more interoperability protocols will emerge in the future. However, LLM agents are unable to autonomously recognize and adapt to these differences, meaning they still require manual configuration and management. This lack of flexibility in handling diverse protocols limits seamless agent-to-agent and agent-to-tool communication across platforms.
- \* **\*The Authentication and Billing Challenge:** Cross-vendor collaboration is further complicated by security and authentication challenges. Each service provider typically requires proprietary API keys, necessitating manual configuration of multiple authentication credentials for agents. This adds significant overhead and disrupts seamless integration. In addition, billing systems are fragmented across vendors, requiring manual intervention for payment setup. As a result, agents are unable to autonomously discover and invoke new third-party paid agents or tool services without manual configuration.

To address these challenges, this document introduces AgentDNS, the first root domain naming and resolution system designed for LLM agents. Inspired by the Internet's DNS [RFC1035], AgentDNS introduces a unified namespace (\*agentdns://\*), natural language-based service discovery, and unified authentication and billing. As shown in Figure 1, AgentDNS is compatible with protocols such as MCP and A2A, allowing them to coexist seamlessly.

<preamble>: Illustrates the interaction between AgentDNS, A2A protocol, MCP, LLMs, and enterprise applications across different agent frameworks.

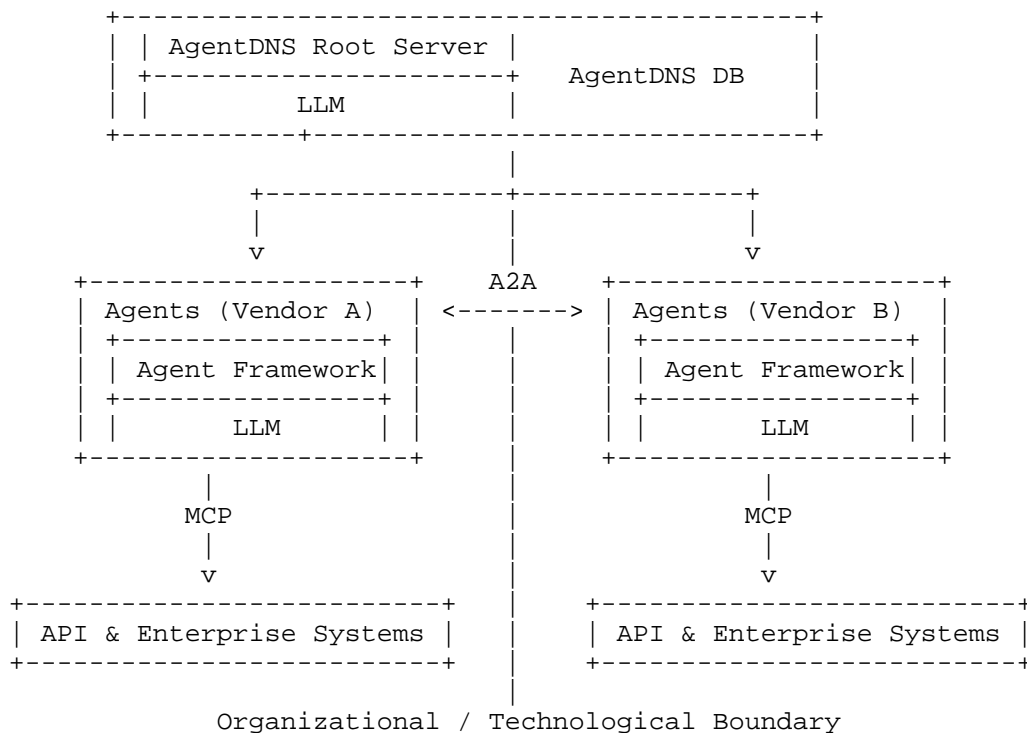


Figure 1: AgentDNS system and its relationship with A2A and MCP Protocols

With AgentDNS, agents can autonomously discover services, authenticate, and interoperate seamlessly across organizational boundaries. AgentDNS redefines the multi-agent collaboration ecosystem through four key functions:

- \* **\*Unified Namespace with Semantic Information:** AgentDNS introduces a semantically rich naming scheme (e.g., agentdns://org/category/name) for agents and tool services, decoupling service identifier name from physical addresses such as URLs. This also enables the semantic embedding of agent capabilities into their identifier name, facilitating more efficient classification and retrieval of agent and tool services.

- \* **\*Natural Language-Driven Service Discovery:**\* Agents can interact with the AgentDNS root service using natural language queries to discover third-party agents or tool services. They can obtain the corresponding service identifier names and related metadata, including physical addresses, capabilities, and communication protocol, etc. Agents can also dynamically request the AgentDNS root service to resolve an identifier name and retrieve the latest metadata as needed.
- \* **\*Protocol-Aware Interoperability:**\* AgentDNS enables agents to dynamically discover the supported interoperability or communication protocols of third-party agents and tool services by resolving their identifier names into metadata. This metadata includes not only network addresses and capabilities, but also the specific protocols (e.g., MCP, A2A, ANP) each service supports. Based on this information, agents can autonomously select and adapt to the appropriate protocol for communication, eliminating the need for manual configuration.
- \* **\*Unified Authentication and Billing:**\* AgentDNS replaces fragmented API keys with a single-sign-on mechanism. Agents authenticate once with the AgentDNS root server to obtain time-bound access tokens, valid across all registered services. For billing, AgentDNS serves as a unified billing platform: users pre-fund accounts, usage costs are tracked and deducted in real-time, and payments are automatically settled across vendors. This enables transparent billing and autonomous access to paid services by agents.

## 2. Related Work

### \*LLM Agents\*

LLM agents have rapidly emerged as a pivotal research frontier in artificial intelligence, driven by their transformative potential to bridge human-AI collaboration and autonomous problem-solving. In the industrial, several LLM agents have been launched, such as Deep Research [Deep-Research], Manus [Manus], and Cursor (Anysphere 2025) [Cursor], etc. Unlike traditional AI systems constrained by predefined rules, LLM agents leverage the general-purpose reasoning, contextual understanding, and multi-task capabilities of LLMs to dynamically adapt to complex environments. LLM agents have demonstrated broad application prospects across various fields. The future of LLM agents is expected to trend towards multi-agent collaboration. Researchers are increasingly interested in how to design efficient communication protocols and coordination mechanisms [hou2025model] [marro2024scalable] that enable seamless cooperation among agents. This collaborative approach is seen as a key direction for advancing the capabilities

and applications of LLM agents in the coming years.

**\*Agent Interaction Protocols\***

**\*Model Context Protocol.\*** The Model Context Protocol (MCP) (Hou et al. 2025) is an open standard developed by Anthropic, designed to facilitate seamless interactions between LLM models and external tools, data sources, and services. Inspired by the concept of a universal adapter, MCP aims to simplify the integration process, much like how a USB-C port allows various devices to connect effortlessly. MCP operates on a client-server architecture. The AI application (such as a chatbot or an integrated development environment) acts as the host and runs an MCP client, while each external integration runs as an MCP server. The server exposes capabilities such as functions, data resources, or predefined prompts, and the client connects to it to utilize these capabilities. This design allows AI models to interact with external systems without directly accessing APIs, thereby enhancing security and reducing the complexity of custom integrations.

**\*Agent-to-Agent Protocol.\*** The Agent-to-Agent (A2A) protocol (Google 2025) is introduced by Google, aimed at enabling seamless communication and collaboration between LLM agents, regardless of their underlying frameworks or vendors. A2A was developed in collaboration with over 50 technology partners, including major companies like Atlassian, Salesforce, SAP, and MongoDB. This protocol uses HTTP-based APIs and JSON data format, ensuring compatibility and ease of integration with existing enterprise IT systems. It supports various communication patterns, including request-response, event-based communication, and streaming data exchange. A2A complements protocols like MCP, which focuses on providing tools and context for agents. A2A focuses on agent-to-agent communication, allowing them to work together more effectively.

**\*Domain Naming System\***

The Domain Name System (DNS) [danzig1992analysis] [cheshire2013rfc6763] serves as the critical naming and discovery infrastructure for the human internet, translating memorable domain names (e.g., example. com) into physical addresses (IP addresses) through its hierarchical, decentralized architecture. While DNS effectively decouples human-readable names from machine-level addressing, its design proves inadequate for the emerging agent Internet where LLM agents require autonomous service discovery and interoperability. Traditional DNS lacks three critical capabilities essential for agent ecosystems: service discovery through natural language, querying service metadata beyond physical addresses (including capabilities, protocols,

etc.), and unified authentication and billing. These limitations necessitate AgentDNS—a purpose-built system that preserves DNS’s core benefits of naming and resolution while introducing agent-specific innovations.

### 3. AgentDNS Architecture

#### 3.1. AgentDNS System Overview

AgentDNS is a root service system for agent service naming, discovery, and resolution, enabling seamless service discovery, cross-vendor interoperability, unified authentication and billing. As shown in Figure 2, the AgentDNS root server is the central hub of the entire system, which connects agent users (e.g., Agent A) with service providers (e.g., Agent Service B, Tool Service D). The AgentDNS root server comprises components including service registration, service proxy pool[RFC5625], service management, service search, service resolution, billing, authentication, etc. The core components are as follows:

- \* **Service Registration Component:** Agent or tool service vendors register their services through this component. The process begins with organization registration, where developers first create an organization account. Under the organization’s domain, they then setup a service category and name to generate a globally unique service identifier name (e.g., agentdns://org/category/name). Concurrently, developers must submit service metadata to bind with the identifier, including network addresses (e.g., endpoints, URLs), supported interoperability protocols (e.g., MCP, A2A), detailed service capabilities, etc. This metadata is securely stored in the AgentDNS database. During subsequent service discovery or resolution phases, the system performs semantic matching by analyzing the identifier’s category and the metadata. This ensures precise retrieval of services aligned with an agent’s operational requirements [RFC7720].
- \* **Service Proxy Pool:** After a vendor registers a service, AgentDNS creates a corresponding service proxy within the Service Proxy Pool. This proxy acts as an intermediary, forwarding service invocation requests from user agents to the vendor’s actual service endpoint. In other words, the user agent sends a service request to the AgentDNS root server, which then routes the request to the appropriate vendor for execution. This design allows user agents to authenticate only once with AgentDNS, eliminating the need for separate registration and authentication with each individual vendor.

- \* **Service Search Component:** User agents can send natural language queries to the AgentDNS root server to discover relevant third-party agents or tool services. This component interprets the query and performs intelligent retrieval using a combination of keyword matching and retrieval-augmented generation (RAG) [gao2023retrieval] techniques. Based on the search results, it returns a list of top-k candidate services that match the agent's intent. Each result includes the service identifier name, physical endpoint, supported communication protocols, capability descriptions, and pricing information. The user agent can then evaluate these candidates and choose the most appropriate one for execution. Once selected, the agent can directly invoke the service by using the appropriate protocol and access the physical endpoint with an authentication token issued by AgentDNS.
- \* **Service Resolution Component:** User agents can cache service identifier names and, during subsequent invocations, dynamically request the AgentDNS root server to resolve these identifiers and get the latest metadata as needed.
- \* **Service Management Component:** This component handles the lifecycle of these service proxies, including their creation, updates, and deletion, ensuring that the proxy infrastructure remains up-to-date and synchronized with the underlying services.
- \* **Service Billing Component:** This component is responsible for tracking and processing service invocation costs. Users only need to settle payments directly with AgentDNS, which then handles the backend settlement with individual vendors. This design significantly simplifies the billing process for users by eliminating the need for managing multiple vendor-specific payment systems, enabling a streamlined and unified billing experience.
- \* **Authentication Component:** This component handles identity verification and access control for both user agents and service providers. Instead of requiring agents to authenticate separately with each vendor, AgentDNS offers a unified authentication mechanism. User agents authenticate once with the AgentDNS root server and receive a time-bound access token. This token can be used to securely access any registered third-party service without additional logins. By centralizing authentication, this component ensures secure, efficient, and scalable access across a heterogeneous agent ecosystem, while also reducing the operational burden on both users and service vendors.

<preamble>:

This figure illustrates the architecture of the AgentDNS system, its AP I interface,

and interactions with agents and services.

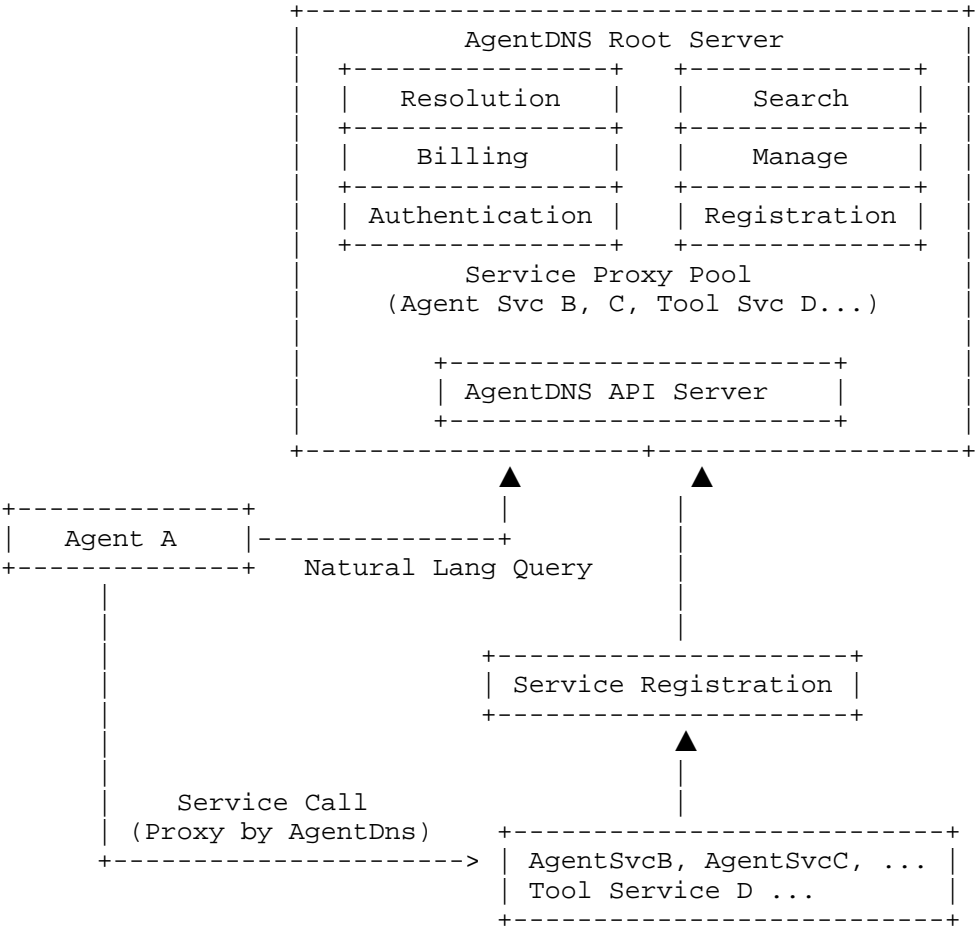


Figure 2: AgentDNS System Architecture

Together, these components form the backbone of AgentDNS, providing a unified framework that supports natural language-driven discovery, protocol-aware interoperability, trustless authentication, and unified billing-paving the way for truly autonomous multi-agent ecosystems. Next, we will provide a detailed introduction to AgentDNS’s service naming, service discovery, service resolution, and unified authentication and billing mechanisms.

### 3.2. Service Naming

The AgentDNS service naming system provides a structured and globally unique service identifier name for each registered agent or tool service. The identifier name follows the format as shown in Figure 3. The organization represents the name of the registering entity, such as a company, university, or research lab. Each organization must go through a registration and verification process to ensure uniqueness and authenticity. The category denotes the functional domain or classification of the agent service. This can be chosen manually by the developer or automatically generated by AgentDNS, and it supports hierarchical structures—allowing for multi-level categories using slashes (e.g., academic/nlp/summarization). Finally, the name is the unique identifier for the specific agent within the organization and category. This name must be explicitly defined by the developer. Together, this structured naming convention ensures precise identification, facilitates organized discovery, and supports scalable service management within the AgentDNS ecosystem.

`<preamble>:`  
 This figure shows the structure of an AgentDNS service name, including  
 prefix,  
 organization, category, and agent name.

Service Identifier Format:

```
agentdns://{organization}/{category}/{name}
  |           |           |           |
  Prefix    Organization  Category   Name
```

Example:

```
agentdns://example/academic/paperagent
```

Figure 3: AgentDNS Service Naming

### 3.3. Service Discovery

The service discovery process is illustrated in Figure 4. In step 1, Agent A initiates a natural language query to the AgentDNS root server, describing the desired service. In the example, Agent A is looking for an intelligent Agent capable of analyzing academic papers. In step 2, upon receiving the request, AgentDNS searches through its registry of available services to identify those with the required capabilities. It returns a list of service identifiers along with corresponding metadata, such as the proxy's physical address, supported protocols, pricing information, and more. This discovery process employs a hybrid retrieval mechanism that combines

keyword matching and RAG. Specifically, we construct a knowledge base using the capability descriptions of registered services. During service discovery, hybrid retrieval is performed over these capability descriptions to identify candidates that best match the user agent’s intent. In step 3, after receiving the service information, Agent A uses the appropriate protocol and an authentication token issued by AgentDNS to directly access the physical proxy address and initiate a service call. Finally, in step 4, the AgentDNS proxy forwards the request to the actual service endpoint hosted by the vendor, ensuring seamless interaction between Agent A and the service provider.

<preamble>:Illustrates how Agent A discovers and interacts with an academic paper analysis agent through the AgentDNS Root Server and proxy layers.

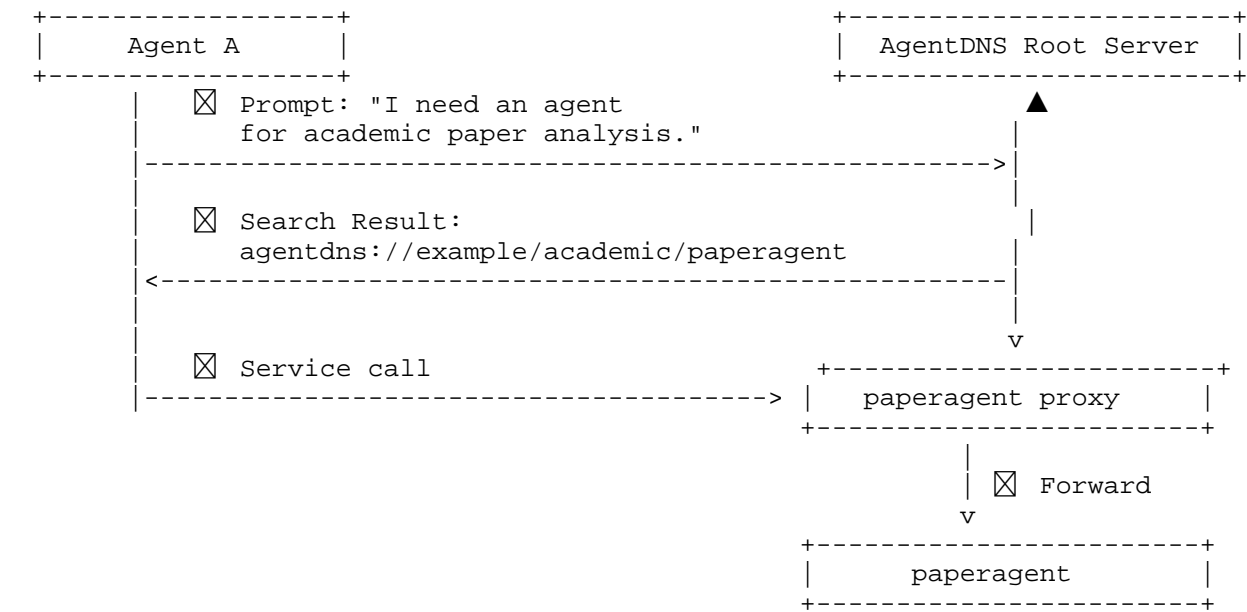


Figure 4: AgentDNS Service Discovery Workflow

3.4. Service Resolution

As previously mentioned, user agents can cache service identifier names and request the AgentDNS root server for updated metadata when needed. This functionality helps reduce the frequency of accessing AgentDNS, improving response times and lowering operational costs. The service resolution process is illustrated in Figure 5. In step 1, agent vendors update the metadata associated with their agent services. In step 2, Agent A sends a resolution request to the

AgentDNS root server, providing the cached service identifier name to retrieve the latest information. In step 3, AgentDNS locates the most recent metadata based on the identifier and returns it to Agent A, ensuring that the service invocation uses up-to-date information.

<preamble>:  
Root Server, Illustrates how an agent (paperagent) updates its metadata to the AgentDNS  
eve updated and how another agent (Agent A) resolves the service via AgentDNS to retri  
metadata.

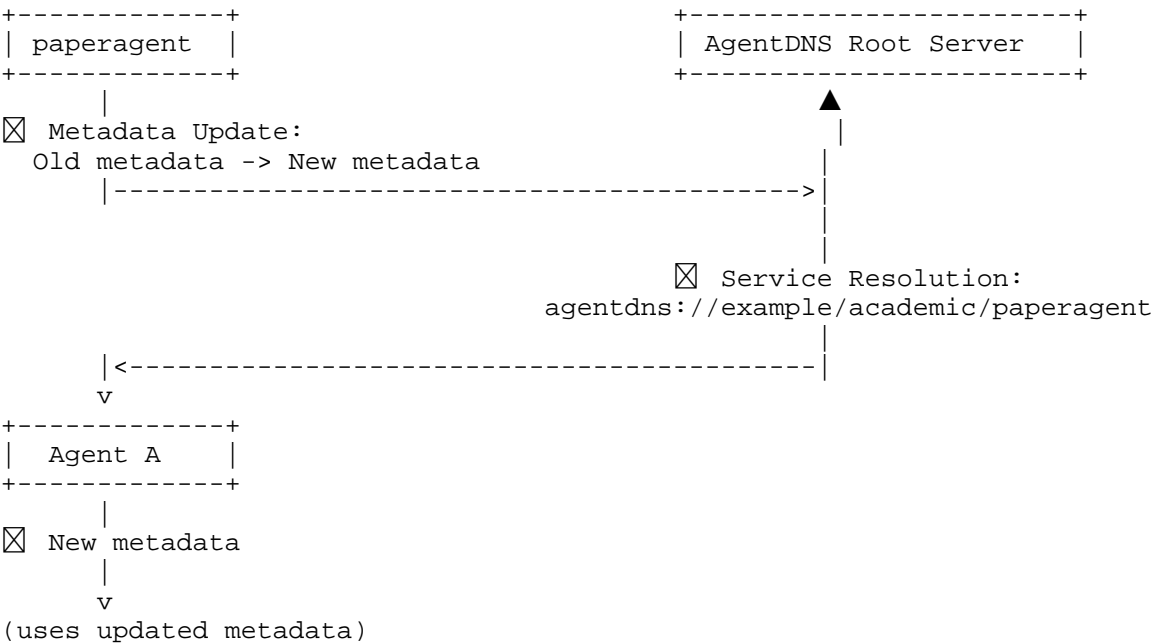


Figure 5: AgentDNS service resolution

3.5. Unified Authentication and Billing

AgentDNS introduces a unified authentication and billing mechanism by acting as a proxy layer between user agents and third-party services. As shown in Figure 6, when a user agent (e.g., Agent A) authenticates once with the AgentDNS root server using its own access key (Key A), it gains the ability to seamlessly invoke multiple external agent or tool services without needing to manage individual credentials for each provider. Internally, the AgentDNS root server maintains a service proxy pool that forwards user requests to the corresponding third-party services. For each thirdparty service, the proxy uses the appropriate authentication key (e.g., Key B, C, or D), which

corresponds to the access control requirements of the service provider. This abstraction decouples the user agent from vendor-specific authentication logic. Moreover, billing is centralized: user agents are charged by AgentDNS based on their usage, while AgentDNS handles settlements with the respective thirdparty services. This model simplifies cross-vendor interoperability, enforces secure access, and enables consistent billing across a heterogeneous service ecosystem.

<preamble>:Agent A resolves services via the AgentDNS Root Server, including billing, authentication, and service redirection using keys.

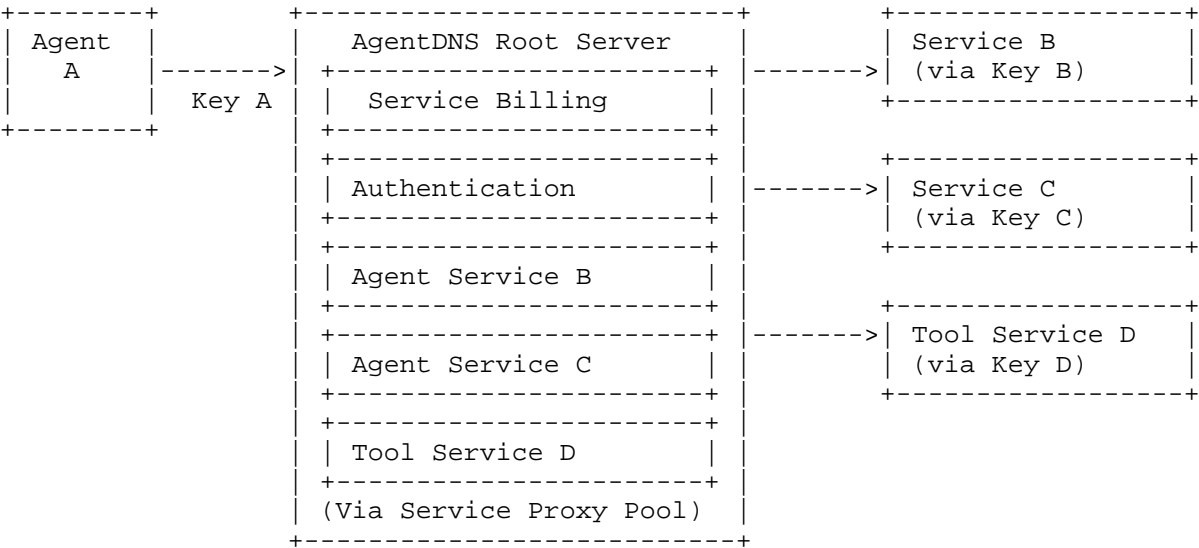


Figure 6: AgentDNS Unified Authentication and Billing

4. AgentDNS Case Study

In this section, we present a case study illustrating the interaction between an agent and the AgentDNS root server. The case demonstrates the complete agent workflow—from generating an action plan [huang2024understanding], to querying the AgentDNS root server for service discovery, and finally to executing the planned actions.

The full process is illustrated in Figure 7. After receiving a user request—such as “Help me research agent communication protocols and write a survey report”—the agent first invokes a LLM to generate an action plan. As shown in Figure 7, the generated plan in this case is structured in JSON format and consists of multiple steps. Each step includes a description of its purpose, whether it requires a

service, and a natural language description of the desired service functionality. These services correspond to third-party agent or tool services. For example, Step 1 requires a search service to retrieve relevant keywords, while Step 3 calls for a standards retrieval service to query documents from organizations like IEEE (IEEE Standards Association 2025) [ieee2025] or ITU-T (International Telecommunication Union 2025) [itut2025].

```
<preamble>:Illustrates how an LLM-generated action plan is executed through AgentDNS-based
```

service discovery and agent invocation.

[illegible]

Uses AgentDNS Root Server

Figure 7: AgentDNS Case Study

After generating the action plan, the agent submits a natural language query to the AgentDNS root server to discover suitable third-party services. For instance, in Step 1, the agent sends the tool function description directly to AgentDNS, which uses intelligent retrieval methods to identify matching services. Suppose AgentDNS returns a service named `agentdns://example/search/searchagent`; it also provides metadata such as the physical endpoint, supported protocols, service cost, capabilities, and available APIs.

The agent uses this information to invoke the selected third-party service.

Following service discovery, the agent enters the action execution phase. During this stage, it executes the steps of the action plan in sequence. When a step requires a service, the agent uses the

corresponding protocol to access the thirdparty service obtained from AgentDNS and passes the result to the next step. For steps that do not involve external services, the agent inputs the step purpose description, along with previous outputs and prompt instructions, into the LLM for generation. This process continues until all steps in the action plan are completed.

This case study presents a simplified example, while in practice, the structure and format of an action plan can be adapted to suit different needs. Importantly, the third-party service descriptions within the action plan are expressed in natural language, which means they are not tightly coupled with specific service identifiers, tool names, or endpoint URLs. AgentDNS plays a critical role in decoupling the foundational agent model from vendor-specific details such as service names, tool identifiers, and physical addresses, enabling more flexible and scalable agent architectures.

## 5. Future Opportunities

While AgentDNS addresses fundamental challenges in service discovery, interoperability, and billing in the agent ecosystem, numerous directions remain open for future exploration. These include decentralized and federated architecture, AgentDNS-compatible agent planning LLMs, privacy-preserving [RFC8932] and trusted discovery, as well as AgentDNS service discovery optimization. First, while the current design of AgentDNS adopts a centralized architecture, future iterations may benefit from decentralized or federated [huang2024aggregate] architecture, such as blockchain [karaarslan2018blockchain]. This would improve robustness, reduce the risk of single points of failure, and enhance trust in cross-organizational collaborations. Second, training and fine-tuning agent planning LLMs [wang2023describe] [hu2024agentgen] specifically compatible with AgentDNS is also an important direction. This can involve constructing agent planning datasets and fine-tuning LLMs to enhance their compatibility with AgentDNS. Alternatively, reinforcement learning techniques [wen2024reinforcing] [jin2025search] [qi2024webrl] [peiyuan2024agile] can be used to train agents to autonomously explore and optimize action sequences, dynamically selecting and combining various services registered in AgentDNS to maximize task success rates and efficiency. Third, security and privacy will remain central in crossvendor agent collaboration. Future directions may involve privacy-preserving search and resolution, using technologies such as homomorphic encryption [buban2025encrypted], differential privacy, and secure multi-party computation. AgentDNS could also integrate trust and reputation systems to allow agents to evaluate service quality and security risks before invocation. Finally, the optimization of AgentDNS service discovery and retrieval remains a critical area for

improving system performance and user experience.

## 6. Conclusion

The rapid advancement of LLM agents has exposed critical gaps in cross-vendor service discovery, interoperability, and authentication, hindering the vision of autonomous multiagent collaboration. This document introduces AgentDNS, a unified root domain naming system designed to bridge these gaps by providing a semantically rich namespace, natural language-driven service discovery, protocol-aware interoperability, and trustless authentication and billing. By decoupling agent identifiers from physical addresses and embedding dynamic metadata resolution, AgentDNS enables agents to autonomously discover, resolve, and securely invoke services across organizational and technological boundaries. Our architecture and case studies demonstrate its potential to streamline multi-agent workflows, reduce manual overhead, and foster an open ecosystem for agent collaboration. Future works include decentralized and federated architecture, AgentDNS-compatible agent planning LLMs, privacy-preserving and trusted discovery, as well as AgentDNS service discovery optimization, etc.

## 7. IANA Considerations

This memo includes no request to IANA.

## 8. Security Considerations

This document should not affect the security of the Internet.

## 9. References

### 9.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<https://www.rfc-editor.org/info/rfc5625>>.

- [RFC7720] Blanchet, M. and L. Liman, "DNS Root Name Service Protocol and Deployment Requirements", BCP 40, RFC 7720, DOI 10.17487/RFC7720, December 2015, <<https://www.rfc-editor.org/info/rfc7720>>.
- [RFC8932] Dickinson, S., Overeinder, B., van Rijswijk-Deij, R., and A. Mankin, "Recommendations for DNS Privacy Service Operators", BCP 232, RFC 8932, DOI 10.17487/RFC8932, October 2020, <<https://www.rfc-editor.org/info/rfc8932>>.

## 9.2. Informative References

- [MCP-Spec] Model Context Protocol Specification Authors, "Model Context Protocol (MCP) Specification", March 2025, <<https://modelcontextprotocol.io/specification/2025-03-26>>.
- [A2A-Spec] Agent2Agent Protocol Specification Authors, "Agent2Agent (A2A) Protocol Specification", 2025, <<https://github.com/google/A2A>>.
- [ANP-Doc] Agent Network Protocol Authors, "Agent Network Protocol (ANP) | Complete Guide to Agent Network Protocol", 2025, <<https://agentnetworkprotocol.com/en/docs/>>.
- [Deep-Research] OpenAI, "Introducing Deep Learning Research", 2025, <<https://openai.com/index/introducing-deep-research/>>.
- [Manus] Monica, "Manus: A New Kind of AI", 2025, <<https://manus.im/>>.
- [Cursor] Cursor, "Cursor: A New Kind of AI Editor", 2025, <<https://cursor.com/en/agents>>.
- [hou2025model] Hou, Xinyi., Zhao, Yanjie., Wang, Shenao., and Haoyu. Wang, "Model context protocol (mcp): Landscape, security threats, and future research directions", arXiv preprint arXiv:2503.23278 2025, 2025.
- [ding2024large] Ding, Han., Li, Yinheng., Wang, Junhao., and Hang. Chen, "Large language model agent in financial trading: A survey", arXiv preprint arXiv:2408.06361 2024, 2024.

[chu2025llm]

Chu, Zhendong., Wang, Shen., Xie, Jian., Zhu, Tinghui., Yan, Yibo., Ye, Jinheng., Zhong, Aoxiao., Hu, Xuming., Liang, Jing., Yu, Philip S., and others, "Llm agents for education: Advances and applications", arXiv preprint arXiv:2503.11733 2025, 2025.

[alzubi2025open]

Alzubi, Salaheddin., Brooks, Creston., Chiniya, Purva., Contente, Edoardo., von Gerlach, Chiara., Irwin, Lucas., Jiang, Yihan., Kaz, Arda., Nguyen, Windsor., Oh, Sewoong., and others, "Open deep search: Democratizing search with open-source reasoning agents", arXiv preprint arXiv:2503.20201 2025, 2025.

[chen2024mindsearch]

Chen, Zehui., Liu, Kuikun., Wang, Qiuchen., Liu, Jiangning., Zhang, Wenwei., Chen, Kai., and Feng. Zhao, "Mindsearch: Mimicking human minds elicits deep ai searcher", arXiv preprint arXiv:2407.20183 2024, 2024.

[luo2025large]

Luo, Junyu., Zhang, Weizhi., Yuan, Ye., Zhao, Yusheng., Yang, Junwei., Gu, Yiyang., Wu, Bohan., Chen, Binqi., Qiao, Ziyue., Long, Qingqing., and others, "Large Language Model Agent: A Survey on Methodology, Applications and Challenges", arXiv preprint arXiv:2503.21460 2025, 2025.

[googlea2a]

Google, "An open protocol enabling communication and interoperability between opaque agentic applications.", 2025.

[marketsandmarkets\_ai\_agents]

{MarketsandMarkets}, "AI Agents Market", 2025.

[guo2024large]

Guo, Taicheng., Chen, Xiuying., Wang, Yaqi., Chang, Ruidi., Pei, Shichao., Chawla, Nitesh V., Wiest, Olaf., and Xiangliang. Zhang, "Large language model based multi-agents: A survey of progress and challenges", arXiv preprint arXiv:2402.01680 2024, 2024.

[han2024llm]

Han, Shanshan., Zhang, Qifan., Yao, Yuhang., Jin, Weizhao., Xu, Zhaozhuo., and Chaoyang. He, "LLM multi-agent systems: Challenges and open problems", arXiv preprint arXiv:2402.03578 2024, 2024.

[yang2025survey]

Yang, Yingxuan., Chai, Huacan., Song, Yuanyi., Qi, Siyuan., Wen, Muning., Li, Ning., Liao, Junwei., Hu, Haoyi., Lin, Jianghao., Chang, Gaowei., and others, "A Survey of AI Agent Protocols", arXiv preprint arXiv:2504.16736 2025, 2025.

[yan2025beyond]

Yan, Bingyu., Zhang, Xiaoming., Zhang, Litian., Zhang, Lian., Zhou, Ziyi., Miao, Dezhuang., and Chaozhuo. Li, "Beyond self-talk: A communication-centric survey of llm-based multi-agent systems", arXiv preprint arXiv:2502.14321 2025, 2025.

[cheshire2013rfc6763]

Cheshire, Stuart. and Marc. Krochmal, "RFC 6763: DNS-Based Service Discovery", 2013.

[danzig1992analysis]

Danzig, Peter B., Obraczka, Katia., and Anant. Kumar, "An analysis of wide-area name server traffic: A study of the internet domain name system", Conference proceedings on Communications architectures \& protocols 1992, 1992.

[openai2025deepresearch]

{OpenAI}, "Introducing Deep Research", 2025.

[manus2025]

{Monica}, "Leave it to Manus", 2025.

[cursor2025]

}, {Anysphere., "Cursor: The AI Code Editor", 2025.

[li2024improving]

Li, Yunxuan., Du, Yibing., Zhang, Jiageng., Hou, Le., Grabowski, Peter., Li, Yeqing., and Eugene. Ie, "Improving multi-agent debate with sparse communication topology", arXiv preprint arXiv:2406.11776 2024, 2024.

[marro2024scalable]

Marro, Samuele., La Malfa, Emanuele., Wright, Jesse., Li, Guohao., Shadbolt, Nigel., Wooldridge, Michael., and Philip. Torr, "A scalable communication protocol for networks of large language models", arXiv preprint arXiv:2410.11905 2024, 2024.

[huang2024understanding]

Huang, Xu., Liu, Weiwen., Chen, Xiaolong., Wang, Xingmei., Wang, Hao., Lian, Defu., Wang, Yasheng., Tang, Ruiming., and Enhong. Chen, "Understanding the planning of LLM agents: A survey", arXiv preprint arXiv:2402.02716 2024, 2024.

[gao2023retrieval]

Gao, Yunfan., Xiong, Yun., Gao, Xinyu., Jia, Kangxiang., Pan, Jinliu., Bi, Yuxi., Dai, Yixin., Sun, Jiawei., Wang, Haofen., and Haofen. Wang, "Retrieval-augmented generation for large language models: A survey", arXiv preprint arXiv:2312.10997 2023, 2023.

[ieee2025] Association, IEEE Standards., "IEEE Standards Association Official Website", 2025.

[itut2025] Union, International Telecommunication., "ITU-T: Telecommunication Standardization Sector", 2025.

[li2021b] Li, Zecheng., Gao, Shang., Peng, Zhe., Guo, Songtao., Yang, Yuanyuan., and Bin. Xiao, "B-DNS: A secure and efficient DNS based on the blockchain technology", IEEE Transactions on Network Science and Engineering 2021, 2021.

[karaarslan2018blockchain]

Karaarslan, Enis. and Eylul. Adiguzel, "Blockchain based DNS and PKI solutions", IEEE Communications Standards Magazine 2018, 2018.

[huang2024aggregate]

Huang, Chih-Kai. and Guillaume. Pierre, "Aggregate Monitoring for Geo-Distributed Kubernetes Cluster Federations", IEEE Transactions on Cloud Computing 2024, 2024.

[wang2023describe]

Wang, Zihao., Cai, Shaofei., Chen, Guanzhou., Liu, Anji., Ma, Xiaojian Shawn., and Yitao. Liang, "Describe, explain, plan and select: interactive planning with llms enables open-world multi-task agents", Advances in Neural Information Processing Systems 2023, 2023.

[hu2024agentgen]

Hu, Mengkang., Zhao, Pu., Xu, Can., Sun, Qingfeng., Lou, Jianguang., Lin, Qingwei., Luo, Ping., and Saravan. Rajmohan, "Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation", arXiv preprint arXiv:2408.00764 2024, 2024.

[wen2024reinforcing]

Wen, Muning., Wan, Ziyu., Wang, Jun., Zhang, Weinan., and Ying. Wen, "Reinforcing LLM Agents via Policy Optimization with Action Decomposition", The Thirty-eighth Annual Conference on Neural Information Processing Systems 2024, 2024.

[jin2025search]

Jin, Bowen., Zeng, Hansi., Yue, Zhenrui., Yoon, Jinsung., Arik, Sercan., Wang, Dong., Zamani, Hamed., and Jiawei. Han, "Search-rl: Training llms to reason and leverage search engines with reinforcement learning", arXiv preprint arXiv:2503.09516 2025, 2025.

[qi2024webrl]

Qi, Zehan., Liu, Xiao., Iong, Iat Long., Lai, Hanyu., Sun, Xueqiao., Zhao, Wenyi., Yang, Yu., Yang, Xinyue., Sun, Jiadai., Yao, Shuntian., and others, "WebRL: Training LLM Web Agents via Self-Evolving Online Curriculum Reinforcement Learning", arXiv preprint arXiv:2411.02337 2024, 2024.

[peiyuan2024agile]

Peiyuan, Feng., He, Yichen., Huang, Guanhua., Lin, Yuan., Zhang, Hanchong., Zhang, Yuchen., and Hang. Li, "AGILE: A Novel Reinforcement Learning Framework of LLM Agents", Advances in Neural Information Processing Systems 2024, 2024.

[buban2025encrypted]

Buban, James., Zhang, Hongyang., Angione, Claudio., Yang, Harry., Farhan, Ahmad., Sultanov, Seyfal., Du, Michael., Ma, Xuran., Wang, Zihao., Zhao, Yue., and others, "Encrypted Large Model Inference: The Equivariant Encryption Paradigm", arXiv preprint arXiv:2502.01013 2025, 2025.

#### Authors' Addresses

Zhiyuan Liang (editor)  
China Telecom Research Institute

Email: liangzy17@chinatelecom.cn

Enfang Cui  
China Telecom Research Institute  
Email: cuief@chinatelecom.cn

Yujun Cheng  
University of Science and Technology Beijing  
Email: yjcheng@tsinghua.edu.cn