Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Technical Taxonomy of LLM Agent Communication Protocols

Linus Sander

linus.sander@tum.de

Technical University of Munich

Habtom Kahsay Gidey

Technical University of Munich

Alexander Lenz

Technical University of Munich

Alois Knoll

Technical University of Munich

Research Article

Keywords: Agent Communication Protocols, Internet of Agents, Taxonomy Development, Multi-agent systems, LLM based Agents

Posted Date: September 29th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10919831/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Technical Taxonomy of LLM Agent Communication Protocols

Linus Sander, Habtom Kahsay Gidey, Alexander Lenz and Alois Knoll

Technische Universität München, Munich, Germany.

*Corresponding author(s). E-mail(s): linus.sander@tum.de; habtom.gidey@tum.de;
Contributing authors: alex.lenz@tum.de; knoll@tum.de;

Abstract

As large language models (LLMs) advance and multi-agent systems aim to overcome the limits of standalone agents, robust communication protocols are becoming essential infrastructure for distributed agent networks. Nonetheless, the fragmented protocol landscape presents a significant interoperability challenge. This study develops a technical taxonomy to classify and analyze LLM agent communication protocols. Following an established iterative method, we defined the taxonomy's purpose, meta-characteristic, and ending conditions, then performed five iterations, three empirical-to-conceptual and two conceptual-to-empirical, on nine actively maintained open-source protocols with demonstrable adoption. The taxonomy comprises five dimensions: counterparty, payload, interaction state, discovery mechanism, and schema flexibility. Classification reveals recurring architectural patterns: all sampled agent-to-agent protocols combine hybrid payloads with session-state persistence; most protocols support multiple predefined schemas, and two negotiate schemas at runtime, indicating a trend toward schema flexibility; decentralized discovery remains rare. Analysis suggests short-term convergence pressure toward protocols unifying agent-to-agent and agent-to-context (tool and data) communication. Long-term, however, no single protocol is likely to maximize versatility, efficiency, and portability simultaneously. The field will more likely evolve toward a federated, layered protocol stack. The framework guides protocol selection and highlights open research gaps such as privacy and policy enforcement.

Keywords: Agent Communication Protocols, Internet of Agents, Taxonomy Development, Multi-agent systems, LLM based Agents

1 Introduction

Recently, large language model (LLM) based agents have attracted significant attention in AI research (Han et al. 2024; Masterman et al. 2024; Xi et al. 2025). LLMs incorporate extensive knowledge (Yu et al. 2024) along with advanced reasoning and planning capabilities (Achiam et al. 2023; Bang et al. 2023; Valmeekam et al. 2023). Embedded as the agent's core and equipped with memory, sensors, and actuators, the LLM grants the agent the capacity to interact dynamically with its environment and to perform efficient multistep reasoning, thereby enabling it to solve complex real-world tasks (Liu et al. 2024; Sun et al. 2023; Hsiao et al. 2025).

Building on single LLM agents, a promising next frontier involves multiple cooperating agents. Such multi-agent systems (MAS) consist of specialized agents that communicate, collaborate, and debate with one another (Guo et al. 2024; Talebirad and Nadiri 2023). Rooted in the principles of collective intelligence (CI) (Malone and Bernstein 2015), this approach further enhances the system's problem-solving capabilities, yielding outcomes superior to those achievable by any individual agent (Guo et al. 2024; Chen et al. 2024; Hong et al. 2024). Here, one detail must not be overlooked: without a communication mechanism, no collective intelligence can arise. This crucial piece of infrastructure serves as the backbone that enables collective intelligence in MAS (Guo et al. 2024; Han et al. 2024; Yan et al. 2025).

To build and explore such systems, various frameworks exist that allow developers to construct MAS at varying levels of abstraction for diverse applications. Widely used examples include Microsoft's *AutoGen* (Microsoft 2023; Wu et al. 2024), *CrewAI* (CrewAI 2023), CAMEL (CAMEL 2023; Li et al. 2023), and *LangGraph* (LangChain 2024). This variety of frameworks, however, has introduced a new challenge: a common protocol becoming necessary to standardize agent communication. Such standardization is a fundamental step toward unlocking the full potential of distributed MAS, where heterogeneous agents can dynamically discover each other and collaborate seamlessly across diverse use cases (Cemri et al. 2025; Yan et al. 2025; Kong et al. 2025). Moreover, this standardization should extend to access to external systems, such as APIs or services, to enable seamless agent integration (Ehtesham et al. 2025). To overcome the antipattern of hard-coded communication pipelines, such a protocol must define the mechanics by which agents interact consistently, efficiently, and securely, with both other agents and external systems (Marro et al. 2024; Li et al. 2025; Kong et al. 2025; Yan et al. 2025).

As the field of LLM-based agents is still relatively new and rapidly evolving, no universally accepted standard has yet emerged. Numerous solutions overlap in capability and are not interoperable with one another. To date, explicit communication protocols have not been widely studied, yet they are becoming an essential prerequisite for efficient cooperation and scalable MAS (Yan et al. 2025; Li et al. 2025; Kong et al. 2025).

In this paper, we investigate the emerging landscape of communication protocols for LLM-based agents by developing a comprehensive taxonomy and analyzing nine concrete protocol implementations. Our taxonomy provides a coherent framework for understanding and classifying these protocols, as well as for tracking future developments.

Concretely, this paper makes four contributions. **(i)** We derive a five-dimensional technical taxonomy of LLM agent communication protocols, comprising *counterparty*, *payload*, *interaction state*, *discovery mechanism*, and *schema flexibility*, following Nickerson et al. (2013)'s iterative method over five documented iterations, and we additionally report the dimensions we considered and rejected, together with the reasons for rejection, so that the boundary of the taxonomy is traceable. **(ii)** We define and apply an explicit, reproducible sampling procedure with five conjunctive inclusion criteria over a multivocal search of academic and grey literature, yielding a sample of nine actively maintained, openly licensed protocols with working implementations. **(iii)** We classify all nine protocols along every dimension and report the rationale for each assignment, surfacing recurring architectural patterns, most notably that agent-to-agent protocols converge on hybrid payloads with session state, and that decentralized discovery remains rare. **(iv)** We interpret these patterns against Marro et al. (2024)'s communication trilemma and argue that no single protocol can dominate, proposing instead a candidate federated, layered protocol stack for the Internet of Agents and identifying privacy, policy, and compliance enforcement as an open gap common to all surveyed protocols.

Finally, we present our key findings and conclude with a brief outlook on future developments.

2 Background

This section lays the groundwork for developing the taxonomy. It first examines LLM-based agents and MAS, before turning to the foundations of communication protocols that enable standardized agent coordination and interaction.

2.1 LLM-Based Agents

Back in 1998, Russell (1998) argued that "AI is about the construction of intelligent agents", noting that such agents are crucial for coping with dynamic, partially observable, and stochastic environments. An agent is generally understood as an autonomous entity that perceives its environment and takes actions to achieve specified goals (Wooldridge and Jennings 1995; Mele 1995). While early AI research paid limited attention to agents, interest grew significantly from the mid-1980s onward (Jennings et al. 1998). Common defining characteristics of agents exhibiting genuinely intelligent behavior include autonomy, reactivity, proactivity, social ability, and strong reasoning capabilities (Nwana 1996; Russell 1998; Mele 1995).

2.1.1 LLM-Centric Control

With the rise of large language models (LLMs), LLM-based agents have emerged as a new paradigm (Xi et al. 2025). Trained on massive text corpora, models such as GPT-4 (2023) (Achiam et al. 2023), DeepSeek-V3 (2024) (Liu et al. 2024), or Claude Opus 4 (2025) (Anthropic 2025) have demonstrated impressive language understanding and generation capabilities. Beyond language processing, these models exhibit a range of emergent capabilities, including human-like reasoning, planning, decision-making (Bang et al. 2023; Liu et al. 2024; Achiam et al. 2023), self-reflection (Chen

et al. 2024; Madaan et al. 2023), zero-shot learning (Kojima et al. 2022), creativity (Li et al. 2025) and more (Zhao et al. 2023b).

Unlike narrow models, LLMs enable agents to often require little or no task-specific training data. They can perform informed actions out-of-the-box using their internal knowledge (Brown et al. 2020; Jin and Lu 2023; Ruan et al. 2023). In practice, an LLM agent can receive a high-level goal in text and autonomously plan and act to accomplish it, using the language model itself to guide multistep reasoning and planning (Yao et al. 2022; Besta et al. 2024; Yao et al. 2023). It is even argued that LLMs pave a promising path toward general AI agents (Xi et al. 2025) and that agents "further push the boundary of LLMs towards AGI" (Zhang et al. 2023a), with self-improvement, autonomy, and continual learning as key components.

A typical workflow for an autonomous LLM-based agent unfolds as follows. The agent first receives a *task* or high-level goal as text. It then begins to *think* by generating natural language, creating subtasks, and devising a plan. Next, it *acts* within its environment, executing the plan through appropriate tools. Finally, by *observing* the environment and reviewing outcomes, the agent gathers *feedback*, enabling it to re-plan, learn, and perform another iteration as needed (Yao et al. 2022; Wang et al. 2024).

2.1.2 Fundamental Components and Concepts

One of the core ideas behind LLM agents is to combine language-based reasoning with structured control, and equipping the agent with components such as long-term memory and specialized tools (Wang et al. 2024; Masterman et al. 2024; Castrillo et al. 2025). Figure 1 provides a high-level overview of an agent system's key components and their connections to the external world.

Reasoning and Planning

LLMs exhibit strong reasoning capabilities (Valmeekam et al. 2023), which form the foundation of successful problem-solving, decision-making, and planning (Evans et al. 1993; Xi et al. 2025). A wide range of techniques has been explored to enhance these capabilities further (Huang et al. 2024; Masterman et al. 2024), ranging from task decomposition and planning approaches to continuous self-reflection loops that support adaptation. Task decomposition, in particular, helps an agent produce more granular and reliable plans for each subtask (Wei et al. 2022; Yao et al. 2023; Besta et al. 2024). Self-reflection mechanisms enable the agent to detect errors, revise its strategy, build resilience through adaptive self-healing, and tackle complex, long-horizon tasks (Shinn et al. 2023; Yao et al. 2022; Gidey et al. 2023).

Knowledge and Memory

Thanks to the LLM acting as the central controller, the agent has access to a vast amount of encoded knowledge embedded within the foundation model's weights (Petroni et al. 2019; Yu et al. 2024). Beyond this, the agent also has access to in-context data, such as recent user conversations, records of tool invocations, and the agent's own thought history. This session-bounded data is commonly referred to as short-term memory or working memory (Zhang et al. 2024). Furthermore, the

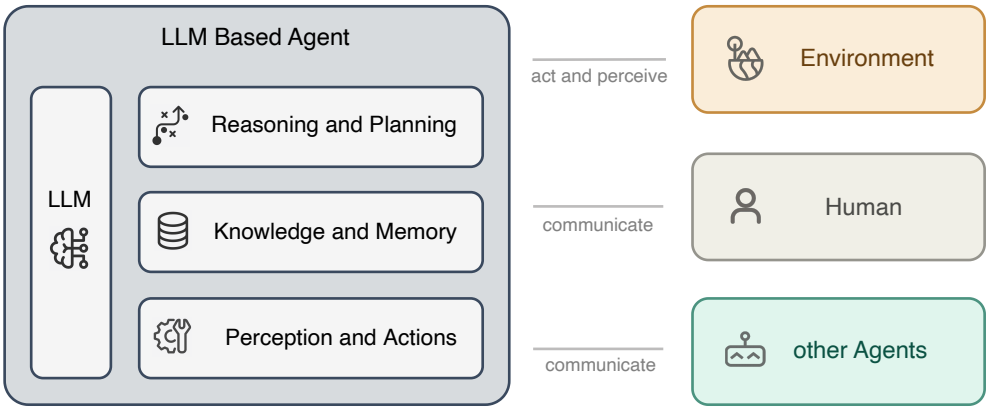


Fig. 1 Fundamental components of LLM agents

agent can be equipped with access to persistent external data sources, such as internal documents, external databases, and retrieval-augmented generation (RAG), sharing principles with document-based knowledge discovery (Hatalis et al. 2023; Lewis et al. 2020; Zeng et al. 2024; Gidey et al. 2022).

Perception and Action

To operate effectively in a specific environment, an agent must be able to perceive that environment, recognize its affordances, and act within it (Shen 2024; Gidey et al. 2025). Beyond the obvious input of textual information, multimodal perception can provide valuable additional capability. Visual input is particularly rich in information when the agent's operating sphere is a subset of the real world, necessitating resilient design patterns for visual processing (Guo et al. 2022; Peng et al. 2023; Pi et al. 2024; Gidey et al. 2026).

The interface to the environment is typically realized by providing tools the agent can use, marking a key difference between agentic systems and pure LLMs (Ruan et al. 2023; Shen 2024). By generating output in a specific format, typically a JSON snippet, the agent can invoke a tool with the required parameters and receive the resulting output upon execution (Shen 2024). Thanks to LLMs' powerful zero-shot and few-shot learning capabilities, new tools can be acquired simply by including their description in the prompt (Ruan et al. 2023; Yao et al. 2022). Additionally, models fine-tuned for tool use have demonstrated strong effectiveness (Shen 2024; Qin et al. 2024).

Applications

Equipped with these capabilities, LLM agents can operate far beyond simple conversational exchanges and traditional human-machine interactions.They can plan multistep processes, query knowledge sources, adapt over time, and more (Huang et al. 2024; Zhang et al. 2024). A variety of domains are now being tackled with specialized agents.

These include computer control, ranging from code generation and terminal troubleshooting to GUI automation (Bouzenia and Pradel 2025; Cao et al. 2024; Qian et al. 2023; Sager et al. 2025). Knowledge work and cognitive automation, including innovation support, scientific workflow assistance, and knowledge exploration, are also now well within reach (Li et al. 2025; Ghafarollahi and Buehler 2025; Lála et al. 2023; Katz et al. 2024; Gidey et al. 2023b). Across all of these areas, LLM agents are still at an early stage but are advancing rapidly (Wang et al. 2024).

Challenges

Despite their rapid advancement, LLM-based agents still face significant limitations that must be addressed for robust real-world deployment. Fundamentally, they are not immune to the inherent vulnerabilities of underlying foundation models, such as hallucination (Bang et al. 2023; Azamfirei et al. 2023), and finite context windows, which restrict the volume of historical data and environmental context they can process (Hsieh et al. 2024; Liu et al. 2024; Shi et al. 2023). Next, autonomously generated plans are not guaranteed to be feasible or efficient (Huang et al. 2024; Kim et al. 2023; Xie et al. 2025), and the reliable integration of multimodal information sources remains an ongoing challenge (Huang et al. 2024). Also, many LLM agent techniques heavily rely on pure prompting. Yet minor prompt variations can produce entirely different outputs (Chatterjee et al. 2024).

Given the capabilities of LLM based agents and the abstract ideas of CI, the next natural step is to explore the collaborative potential of multiple agents (Talebirad and Nadiri 2023; Tran et al. 2025). For instance, one could query a smaller LLM N times to produce a set of independent answers. By selecting the most common answer, this method can, interestingly, outperform a single call to a state-of-the-art model (Li et al. 2024). However, this captures only a narrow facet of CI, prompting a more fundamental question: what happens when we actively employ the communicative and social capacities of LLM agents to enable true cooperation? The following section addresses this question by a brief survey of the field of MAS.

2.2 Multi-agent systems

2.2.1 LLM Based Multi-agent systems

Multi-agent systems (MAS) consist of multiple agents that coordinate and communicate to accomplish a common goal. The idea is simple: independent, specialized agents jointly solve a task, thereby harnessing the benefits of collective intelligence (Guo et al. 2024; Talebirad and Nadiri 2023). Decomposing complex objectives into manageable subtasks and assigning them to specialized agents naturally simplifies problem-solving (Huang et al. 2024; Hong et al. 2024; Khot et al. 2022). Furthermore, diverse perspectives and varied interaction styles across agents, such as mutual feedback and debate, further enhance MAS problem-solving capabilities beyond what single-agent systems can achieve (Tillmann 2025; Guo et al. 2026; Fu et al. 2023). Such systems can tackle dynamic, complex tasks with strong performance and show emergent cognitive behavior (Duéñez-Guzmán et al. 2023; Guo et al. 2024).

2.2.2 Key Concepts

At a high level, an MAS is built from two essential elements: the agents themselves and the architecture that connects them. Each agent is characterized by a specific profile, including its role, abilities, and toolset, while the architecture serves as the core infrastructure that defines and enables collaboration among all agents (Li et al. 2024; Yan et al. 2025). Having outlined the core elements of individual LLM agents in Section 2.1, we now turn to the fundamental concepts of LLM-driven MAS.

The optimal topology of an agent ensemble remains a central research concern, bringing forth various architectural challenges in cognitive agent design (Gidey et al. 2023a). Different *architectures* exist, ranging from equi-level designs, in which all agents operate at the same hierarchical level, to hierarchical configurations, where one or several agents guide the group as a leader. Subsequently, the communication structure may be either fixed or dynamic, requiring rigorous methods to model such dynamic architectures.In fully dynamic systems, agent roles, relationships, and even the number of participating agents can evolve over time (Han et al. 2024; Masterman et al. 2024; Tillmann 2025).

Next, effective *coordination and decision-making* is vital. Agents must debate, consult one another, align their goals, and resolve misunderstandings. Because MAS often yield multiple possible solutions and parallel discussions, techniques are required to converge on a single solution. Typical strategies include majority voting, consensus-seeking methods, or introducing a judge agent to make the final decision. (Kaesberg et al. 2025; Huang et al. 2024).

When looking more closely at the individual entities, the agents, *profiling* becomes especially important in MAS. The specialization of each agent is critical for an effective system. Roles, characterizations, behavior descriptions, and skill sets must be carefully crafted, primarily through precise prompting and LLM model selection (Wang et al. 2024; Guo et al. 2026; Li et al. 2023).

To successfully enable CI within MAS, a *communication backbone* is crucial (Guo et al. 2024). Without communication, agents cannot cooperate, debate, nor coordinate their actions. Ensuring that interactions are efficient and robust is therefore essential for achieving a reliable and high-performing MAS (Han et al. 2024; Yan et al. 2025). Therefore, given that MAS are expected to grow in size and distribution (Wang et al. 2025; Li et al. 2024), standardized communication protocols are critical to achieve scalability and flexibility without sacrificing robustness (Ehtesham et al. 2025; Kong et al. 2025). They are key infrastructure elements that enable heterogeneous agents to connect reliably and dynamically. In essence, the communication infrastructure serves as the backbone enabling collective intelligence in MAS (Marro et al. 2024; Yan et al. 2025; Guo et al. 2024; Cemri et al. 2025).

2.3 Communication Protocols

In computer science, a *protocol* is a formal set of rules or conventions that governs how entities, such as programs, processes, or agents, communicate and interact. It specifies the format, sequence, and meaning of the messages exchanged, enabling heterogeneous systems to understand one another straight away (Pouzin and Zimmermann

1978; Tanenbaum and Wetherall 2011). As Cerf and Kahn (1974) note, computers must "share a common protocol (i.e., a set of agreed-upon conventions)" for data exchange to be meaningful.

In practice, a protocol specifies *what* data is communicated, *how* it is structured, and *when* it should be sent or acknowledged. By defining these dimensions precisely, protocols enable independent, potentially heterogeneous systems to communicate reliably. Often, they also include procedures for error handling, synchronization, and more, so that, even under adverse conditions, reliable communication is ensured (Pouzin and Zimmermann 1978; Tanenbaum and Wetherall 2011).

The internet itself is the most obvious example of a protocol's importance. Foundational standards such as the *Internet Protocol* (IP) and the *Transmission Control Protocol* (TCP), together with the *Hypertext Transfer Protocol* (HTTP), form today's backbone of global data exchange (Nath and Uddin 2015; Cerf and Kahn 1974).

Beyond this, MAS require application-level protocols clarifying how autonomous agents interact. Before the rise of LLMs, well-known examples included the protocols *Knowledge Query and Manipulation Language* (KQML) (Finin et al. 1994) and *FIPA Agent Communication Language* (ACL) (Labrou et al. 1999). KQML, developed in the early 1990s, was introduced as "a new language and protocol for exchanging information and knowledge" among software agents. Central to KQML is an extensible set of operations called *performatives*, such as *ask* and *tell*, which lays the groundwork for higher-level interaction patterns like negotiation (Finin et al. 1994; Labrou et al. 1999). Its successor, ACL, represented the first organized effort to standardize communication in the broader agent community. Building on speech-act theory, ACL lets agents express intentions, requests, information, and proposals while each message follows a formal structure with fields for sender, receiver, content, and more, enabling interoperability across heterogeneous agents (Labrou et al. 1999).

In LLM-based MAS, standardized communication protocols become essential for maintaining consistent, secure, and reliable interactions among heterogeneous agents (Yan et al. 2025; Ehtesham et al. 2025; Kong et al. 2025). As systems scale, these protocols become a prerequisite for robust multi-agent collaboration, enabling greater efficiency, adaptability, and overall performance with minimal human intervention (Li et al. 2025; Cemri et al. 2025).

3 Related Work

Several recent works have examined the emerging landscape of LLM agent communication protocols, highlighting both the rapid progress and the significant challenges that remain.

From a standardization standpoint, Li et al. (2025) argue that the current fragmentation of agent communication resembles the early "protocol wars" of networking and advocate for a unified framework. Next, Du et al. (2025) examine AI agent communication through the perspective of Internet architecture, distilling five key design principles, to guide the sustainable development of multi-agent ecosystems. Moreover, Kong et al. (2025) provide a comprehensive security-focused survey of agent communication, proposing a three-class taxonomy for categorizing agent communication, while

systematically analyzing their associated vulnerabilities and potential defense mechanisms. Complementing those perspectives, Ehtesham et al. (2025) offer a comparative analysis of four concrete interoperability protocols: MCP, ACP, A2A, and ANP, evaluating them across dimensions such as interaction modes, discovery mechanisms, and security models.

Further research exists on concrete protocols. Among the sample that is later examined, the Agora protocol (Agora Protocol 2025a) stands out for its scientific rigor and for emphasizing fundamental principles needed in an efficient, scalable, and robust design. In particular, Marro et al. (2024) identified a *trilemma* of three quality attributes, none of which can be fully optimized simultaneously. First, *versatility* requires the protocol to support multiple message types, from text only to structured data. Second, *efficiency* demands minimal computational and networking costs. Third, *portability* is of focus, reducing the effort required for any arbitrary agent to adopt and use the protocol (Marro et al. 2024).

Last but not least, Yang et al. (2025) also propose a taxonomy for LLM agent communication protocols. Their taxonomy comprises two dimensions: *object orientation*, which distinguishes *context-oriented* from *inter-agent* protocols according to the intended counterpart, and *application scenario*, which classifies protocols as either *general-purpose* or *domain-specific*. In addition to that, they collected a sample of 14 protocols and evaluated them against quality attributes such as efficiency, security, and scalability, identifying several emerging trends in agent development (Yang et al. 2025).

A distinct line of work proposes *layered architectures* for agent communication rather than individual protocols. Fleming et al. (2025) extend the OSI reference model with an agent communication layer, carrying message envelopes and speech-act performatives, and an agent semantic layer handling context discovery, disambiguation, and consensus. Xu et al. (2026) propose *Agent-OSI*, a six-layer stack separating connectivity and secure transport from messaging and routing, identity and authorization, settlement and metering, verifiable execution and provenance, and semantic orchestration. Addressing the trust dimension specifically, Zou et al. (2025) overlay decentralized identifiers, blockchain-anchored ledgers, and smart-contract policy enforcement onto existing agent-to-agent interactions, while Ranjan et al. (2025) propose a decentralized ecosystem built on verifiable credentials.

These proposals are architectural rather than empirical; each derives a layering from design principles, and none is obtained from a systematic classification of protocols in actual use. They also describe reference architectures rather than protocols with implementations, and therefore fall outside the sample delimited by IC1–IC5 in Section 4.2. Our contribution is complementary and proceeds in the opposite direction. We first classify nine maintained protocols along five dimensions and only then read a candidate layering off that classification (Section 6.1), so that every placement in Figure 4 rests on an evidenced characteristic rather than on a functional decomposition chosen a priori. The convergence of several independent proposals on a layered architecture is, in itself, further support for the federated-stack argument we develop in Section 6.1.

The works most closely related to this survey are those of Yang et al. (2025) and Ehtesham et al. (2025). However, the two-dimensional taxonomy proposed by Yang et al. (2025), lacks the granularity needed for a thorough characterisation of the field. Ehtesham et al. (2025) employ more detailed dimensions but focus on only four protocols and orient their analysis toward practical application rather than systematic, abstract classification.

4 Methods

4.1 Taxonomy Development Method

To develop the taxonomy itself, we followed the widely used method for taxonomy construction from Nickerson et al. (2013). This method was selected because it provides a structured, iterative, traceable, and reproducible procedure for constructing taxonomies, rather than relying on an ad hoc classification process (Oberländer et al. 2019). We first begin this section by outlining and clarifying the qualities that define a useful taxonomy, then discuss the iterative procedure that drives its development as applied in this study.

The term *taxonomy* can be used in several ways. In this study, the term *taxonomy* refers to a classification framework. Fundamentally, the goal is to provide a useful hierarchical classification system that organizes key concepts and their concrete values. It aims to capture the essence of a specific domain, thereby helping its users to understand, discuss, analyze, and observe complex concepts (Nickerson et al. 2013; Oberländer et al. 2019). In contrast to a *typology*, which is a conceptual classification approach, a taxonomy is empirical, derived from real-world data and observation (Bailey 1994). Following Nickerson et al. (2013), a taxonomy consists of several distinct dimensions, each of which contains two or more characteristics that should be mutually exclusive and collectively exhaustive (Nickerson et al. 2013; Oberländer et al. 2019).

Nickerson et al. (2013) define five qualitative attributes of a useful taxonomy. A taxonomy should be *concise*, enabling researchers to recall the entire taxonomy with ease; *robust*, containing enough dimensions and characteristics to differentiate objects in the domain meaningfully; *comprehensive*, able to classify every instance in the domain and contain all relevant object dimensions; *extendible*, allowing new characteristics and dimensions to be added without difficulty when necessary; and *explanatory*, offering dimensions and characteristics that provide a useful and explainable abstraction (Nickerson et al. 2013). These criteria served as subjective ending conditions for our taxonomy development process.

In this study, we rigorously follow the clear development process to ensure quality and reproducibility. The entire development process began with the definition of three foundational elements. First, the researcher should think of the overall *purpose* of the taxonomy: for what should it especially be used later and who are the main users. Next, a so-called *meta-characteristic* has to be set up, to provide a basis for all characteristics to be constructed later. A typical wording could be: 'Classify ... based on ...' and can encapsulate multiple facets. After establishing this anchor, the researcher selects objective *ending conditions* (e.g. no new dimensions emerge and

every characteristic is instantiated) and subjective ending conditions (namely, the five quality criteria outlined above) (Nickerson et al. 2013).

After these elements had been defined, we followed Nickerson et al.'s iterative development cycle. The development cycle goes over, right in the beginning, to a fundamental decision where each iteration requires a choice between two development paths. In the *conceptual-to-empirical* path, the researcher first identifies candidate dimensions from theory, applies them to a sample of concrete objects of the domain, and prunes or augments the constructed dimensions. On the alternative *empirical-to-conceptual* path, the researcher instead examines a sample of concrete objects, distills meaningful attributes and properties, and groups these into abstract dimensions that satisfy the basic quality criteria of sound taxonomy design. Each iteration ends with a review and evaluation of the resulting taxonomy against the predefined ending conditions. If these conditions are not satisfied, another iterative cycle begins, starting with the binary path choice (Nickerson et al. 2013).

Conceptually, this process mirrors the build-evaluate cycle in design science research (Hevner et al. 2004). Each iteration constructs a provisional taxonomy or schema and immediately evaluates it against empirical reality and the predefined quality criteria. This allows usefulness, purpose-fit, and traceability to be integrated into the taxonomy development process. The approach also encourages detailed documentation of the entire procedure, which supports both reproducibility and extensibility. In our study, this disciplined iterative procedure, bounded by explicit rules, was used to yield a high-quality taxonomy with unique dimensions and mutually exclusive, collectively exhaustive characteristics (Nickerson et al. 2013; Oberländer et al. 2019).

4.1.1 Use of large language models

Large language models were not used to generate the taxonomy, to select or classify the protocols, or to produce the analysis reported in this paper. All dimensions, classifications, and rationales were derived by the authors from the primary specifications. Language-model assistance was limited to language editing of author-written text. The authors take full responsibility for the content of the manuscript.

4.2 Communication Protocols Sample

This section outlines how we selected the nine protocols that ground our taxonomy development and offers concise summaries of each.

First, we define an *LLM agent protocol* as a specification whose stated purpose is to standardise the interaction between an LLM-based agent and an external counterpart, such as a tool, a data source, a service, or another agent **(IC1)**. The decisive criterion is design intent: more generic agent communication implementations and concepts, such as the *FIPA Agent Communication Language* (O'Brien and Nicol 1998) which implements speech-act theory, or the *Contract Net Protocol* (Smith 1980), an abstract task-sharing protocol for MAS, are excluded. This scope is further clarified in the *meta-characteristic* in Section 5.1.

The search strategy combined academic and grey literature, following the multivocal literature review guidelines of Garousi et al. (2019), which are appropriate for

domains in which practice outpaces peer-reviewed publication. Academic records were retrieved from IEEE Xplore, ACM Digital Library, ACL Anthology, Scopus, and arXiv using the query ‘("agent protocol" OR "agent communication" OR "agent orchestration" OR "agent interoperability") AND ("large language model" OR "LLM" OR "AI agent" OR "agentic AI" OR "multi-agent system")‘, restricted to publications from January 2023 onward. Since most protocols in this domain are released as repositories and specifications rather than as papers, the search was extended to GitHub (same keywords, results ordered by stars). All searches were conducted on 12 Mar 2026, and all repository metrics cited in this paper correspond to that date. In addition to scope, four further inclusion and exclusion criteria were applied, as summarized in Table 1:

Communication-layer primacy (IC2). A qualifying specification must define the interaction mechanism at the agent-communication layer, with at least one of the following explicitly addressed: message transport, session lifecycle, message structuring, or capability negotiation. Specifications that define only a method, vocabulary, or content schema carried over a separate agent-communication protocol do not meet this criterion. Building on generic web or RPC infrastructure such as HTTP is permitted; however, specifications that function purely as application or content layers on top of an existing agent-communication protocol are excluded. Also excluded are specifications limited to supporting functions such as naming, discovery, or registry services that delegate the actual interaction to a separate protocol. For example, NLWeb (nlweb.ai 2025) is excluded because it operates on top of MCP, whereas agents.json (Wildcard 2025) is included because it defines interaction directly over plain HTTP. Similarly, AgentDNS (Cui et al. 2025) and the Agent Name Service (Huang et al. 2025) are excluded because they specify a naming, discovery, and resolution layer and delegate actual invocation to a separate protocol such as MCP, A2A, or ANP.

Openness (IC3). Only protocols publicly accessible and released *in their entirety* under an open licence were retained. "Open licence" is operationalised here via the Open Definition: implementations must carry an OSI-approved licence, while specifications must carry either an OSI-approved or an open-content licence such as CC-BY. Moreover, openness is further assessed along two dimensions: *functional completeness* (the full protocol is openly available) and *vendor neutrality* (protocol can be independently implemented without dependence on any single vendor's runtime, token, or proprietary infrastructure). This requirement is motivated by the historical pattern that for major web standards it has been critical to be open source, in order to achieve widespread adoption and become a global standard (Gamalielsson et al. 2024).

Implementation (IC4). A usable implementation must exist, although it need not be production-ready: research and test prototypes qualify, as in the case of Agora (Marro et al. 2024). Purely conceptual proposals such as LOKA (Ranjan et al. 2025) were therefore excluded.

Maintenance (IC5). Public repositories also allow ongoing maintenance activity to be verified. A cutoff of 1 January 2025 was applied, corresponding to at least 12 months prior to the search date, such that any candidate lacking a substantive commit since that date was excluded. The *Agent Protocol* of Garg (2023), whose last non-README commit was dated 11 June 2024, is one such example. This specification

Table 1 Protocol inclusion criteria. All criteria are conjunctive: a candidate is included only if it satisfies IC1–IC5

ID	Criterion	Operationalisation	Excluded examples
IC1	Agent-interaction intent	Stated purpose is to standardise interaction between an LLM-based agent and an external counterpart (tool, data source, service, or agent)	FIPA-ACL (O'Brien and Nicol 1998), Contract Net Protocol (Smith 1980), Spatial Population Protocols (Gąsieniec et al. 2025)
IC2	Communication-layer primacy	Carries the agent interaction itself	llm.txt (llms-txt 2024), AGENTS.md (LF Projects 2025), NLWeb (nlweb.ai 2025), PXP Protocol (Mestha et al. 2025), AgentDNS (Cui et al. 2025), Agent Name Service (ANS) (Huang et al. 2025)
IC3	Openness	Publicly accessible and released under an open licence	Firecrawl (Firecrawl 2024), uAgents (Fetch.ai 2025), Skyfire (Skyfire 2025)
IC4	Implementation	At least one usable implementation exists, including research or test prototypes	LOKA (Ranjan et al. 2025), Agent Collaboration Protocols (ACPs) (Liu et al. 2025)
IC5	Maintenance	Repository has at least one substantive commit within the preceding 12 months	Agent Protocol (Garg 2023)

should not be confused with LangChain's identically named *Agent Protocol*, which is included in the sample and which we denote LAP throughout.

The selection process is summarised in Figure 2. The combined academic and repository search yielded 31 records, which reduced to 24 distinct candidate protocols after 7 duplicates were removed, chiefly cases in which a specification was retrieved both as a preprint and as its corresponding repository. Each of the 24 candidates was then assessed in full against the conjunctive criteria IC1–IC5. Because the criteria are conjunctive, a candidate was attributed to the first criterion it failed, evaluated in the fixed order IC1 to IC5, so that the exclusion counts partition the excluded set rather than overlapping. On this basis 3 candidates failed IC1 (agent-interaction intent), 6 failed IC2 (communication-layer primacy), 3 failed IC3 (openness), 2 failed IC4 (implementation), and 1 failed IC5 (maintenance), giving 15 exclusions in total. The excluded candidates are named in the final column of Table 1. The remaining nine protocols met all five criteria and are outlined in the following section. Tables 2 and 3

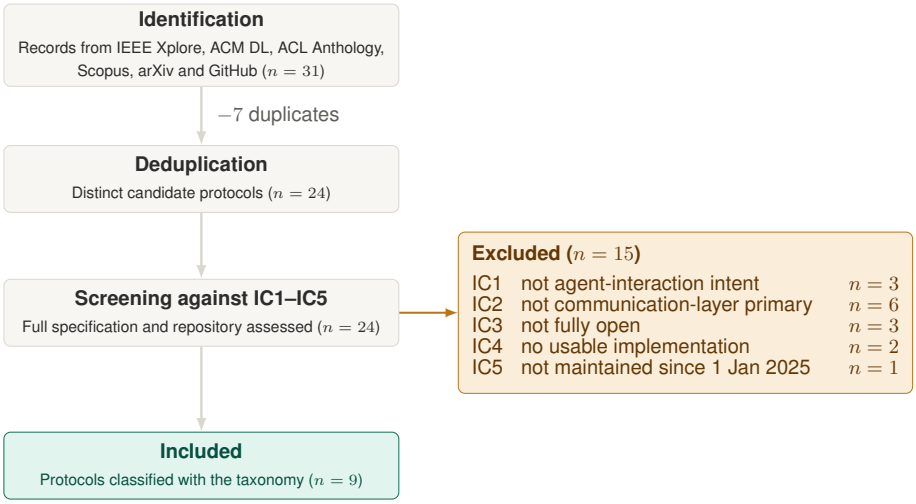


Fig. 2 Selection of the protocol sample, reported in the style of a PRISMA flow diagram and adapted for a multivocal literature review (Garousi et al. 2019). Criteria IC1–IC5 are defined in Table 1 and are conjunctive; each excluded candidate is attributed to the first criterion it failed, evaluated in the order IC1 to IC5, so the exclusion counts sum to the total without double counting

provide a quick overview, and further characteristics are discussed during taxonomy development.

4.2.1 Model Context Protocol (MCP)

Developed by Anthropic, the *Model Context Protocol* (MCP) provides a standardized way to augment any LLM with tools and contextual information by enabling existing applications, APIs, and raw data sources, in short almost any IT component, to present their context and capabilities to LLMs in a standardized manner (Anthropic 2024b,a).

At its core, MCP follows a straightforward client-server model. Host applications, such as Claude Desktop or IDE plug-ins, embed *MCP clients* that communicate with *MCP servers*, each of which exposes specific capabilities, including data retrieval, tool invocation, and prompt delivery for the LLM behind the MCP client (Anthropic 2024a).

4.2.2 Agent to Agent (A2A)

Google developed this protocol to address the rising number of siloed agent infrastructures. As agents are implemented and deployed across diverse frameworks and platforms, a standardized protocol is required to facilitate collaboration (Google 2025e). Its primary goal is to enable agent-to-agent connectivity, not to replace the already well-known MCP protocol (Google 2025b).

Designed to be secure and extensible by default, it implements a variety of concepts across multiple levels of communication, supporting both immediate and extended

tasks, including streaming updates, state synchronization, and artifact exchange (e.g. files and structured data) (Google 2025a,d).

4.2.3 LangChain Agent Protocol (LAP)

LangChain created a standardized API for deploying LLM agents in production. We refer to it throughout as LAP, to keep it distinct from the earlier and unrelated *Agent Protocol* of Garg (2023), which shares the name but is excluded from our sample under IC5. Specifically, they defined a RESTful API with a unified set of endpoints. The main endpoints are */runs* for executing agents, */threads* for managing multi-turn conversations, and */store* for long-term memory storage. Through these endpoints, any client can invoke, monitor, and persist agent workflows, irrespective of the agents' internal implementations (LangChain 2025b,a).

4.2.4 agents.json

This simple protocol defines a JSON specification through which existing websites and APIs can be discovered and interpreted by LLM based agents. Built on top of the *OpenAPI* standard, agents.json offers a structured, LLM-friendly, and stateless way to define and consume API workflows, letting agents execute multi-step tasks reliably with minimal boilerplate and without extensive prompt engineering (Wildcard 2025).

4.2.5 Agora

Oxford researchers introduced the *Agora Protocol* (Agora Protocol 2025a) to enable decentralized collaboration among diverse LLM agents, without central servers and pre-defined communication schemas. Its design tackles the *Agent Communication Trilemma*, balancing versatility (handling diverse message types), efficiency (minimizing token and API cost), and portability (working across agent platforms), in a single decentralized framework (Marro et al. 2024). For instance, it includes an on-the-fly protocol negotiation mechanism. Before exchanging actual data, agents share or even generate new communication schemas (Marro et al. 2024; Agora Protocol 2025b).

4.2.6 Agent Network Protocol (ANP)

Its vision is to become "the HTTP of the agentic web era" (ANP 2025). It is a peer-to-peer communication protocol designed to enable secure, decentralized collaboration among heterogeneous LLM-powered agents across the internet. The protocol is intentionally flexible and feature-rich out of the box, offering three layers: an *Identity Layer* for secure communication, covering authentication and end-to-end encryption; a *Meta Protocol Layer* that allows agents to negotiate communication styles dynamically, and an *Application Layer* that supports straightforward agent interaction through standardized descriptions and management (Chang 2024b; Chang et al. 2025).

4.2.7 LMOS

This protocol is part of the *Eclipse LMOS* ecosystem and enables agents and tools to easily publish, connect, and share their capabilities. As a special feature, it standardizes metadata and interaction patterns while remaining agnostic to the underlying transport layer, allowing seamless operation over HTTP, WebSocket, MQTT, and other protocols. Moreover, it emphasizes that it can readily support an internet-scale MAS with thousands of tools and agents, an *Internet of Agents* (Eclipse Foundation 2025a,b).

4.2.8 Agent Communication Protocol (ACP)

The ACP is a RESTful open standard, created by BeeAI and IBM under Linux Foundation governance, that enables seamless, structured communication, discovery, and coordination among heterogeneous LLM agents (ACP 2025; ACP 2025a).

ACP defines consistent HTTP/JSON interfaces that enable agent discovery, execution, and multimodal message exchange. It also implements so-called *ACP Servers*, where agents can register themselves for discovery and can likewise discover other publicly available agents (ACP 2025a).

4.2.9 agntcy

Its mission extends beyond defining a protocol for agent-to-agent communication, it seeks to establish an open-source infrastructure for the *Internet of Agents* (AGNTCY 2025c). For communication, it specifies the *Agent Connect Protocol*, an API extending *OpenAPI* that enables agents to connect, share state, stream results, and handle authentication and authorization seamlessly across frameworks and runtime environments. The potential standard also provides essential infrastructure elements, including a centralized discovery service, a fixed agent manifest format, and many more (AGNTCY 2025a,d,b).

5 Results: Taxonomy Design

We now proceed to construct the taxonomy by first defining its fundamental artifacts and summarizing the iterative development process. Rather than detailing each iteration, we then focus on examining both accepted and rejected dimensions in detail. Finally, we classify concrete protocols from our sample to illustrate the taxonomy's dimensions and their application.

5.1 Developing the Taxonomy

5.1.1 Scope

As discussed in Section 4, Nickerson et al. (2013)'s technique begins with the definition of three foundational artifacts that define the scope of the taxonomy.

The *purpose* of the taxonomy is to classify and distinguish LLM agent communication protocols based on the type of communication they facilitate. The taxonomy focuses on protocols that connect an LLM agent to another system, such as another

agent or a general information system. It operates primarily at the application level rather than the solution level, and therefore does not delve into concrete technical implementation details.

The *expected users* of the taxonomy are researchers and agent developers, as well as non-experts who simply want to access an agent through an API. Potential users may wish to obtain an overview of the forms of communication and guidance for protocol selection. In addition, a brief overview of how protocols differ overall, and where gaps and opportunities for further research and development exist, is of interest.

The *meta-characteristic* is defined as classifying the protocols according to the types of components involved and the characteristics of their communication.

5.1.2 Process and Quality

In total, we conducted five iterations: the first three followed the empirical-to-conceptual approach, and the final two followed the conceptual-to-empirical approach. Each of the first three iterations examined three of the nine collected protocols. The final two iterations complemented the analysis by consolidating accumulated knowledge and testing more abstract dimensions.

According to Nickerson et al. (2013), the iterative taxonomy development process ends when both objective and subjective conditions have been fulfilled. For the subjective conditions, we followed the principle that the fundamental quality conditions must be fulfilled: a taxonomy should be concise, robust, comprehensive, extensible, as well as explanatory. In addition, the general taxonomy definition that it should consist of mutually exclusive and collectively exhaustive dimension should hold. For the objective conditions, we selected a suitable subset of Nickerson et al. (2013)'s suggestions:

- no additional specific protocols need to be examined
- no new dimensions are added in the last iteration
- no dimensions or characteristics were merged or split in the last iteration
- every dimension and characteristic within a dimension is unique and not being repeated

Ultimately, the iterative approach, guided by clear ending conditions and the taxonomy's purpose and meta-characteristic, made development straightforward and appeared to strengthen its robustness and comprehensibility.

With respect to the basic quality criteria of a qualitative taxonomy, we focused on the following key properties. The developed taxonomy is concise, as it contains a clear and manageable number of dimensions and characteristics. It is also robust, in the sense that we examined exactly nine protocols in detail and discovered no new valid dimensions in the final iteration. Furthermore, comprehensiveness and explanatory quality are ensured, as we demonstrate that all protocols can be classified according to the taxonomy in a straightforward manner. Finally, the taxonomy satisfies the foundational principle of mutual exclusivity and collective exhaustiveness, which we systematically enforced during dimension construction.

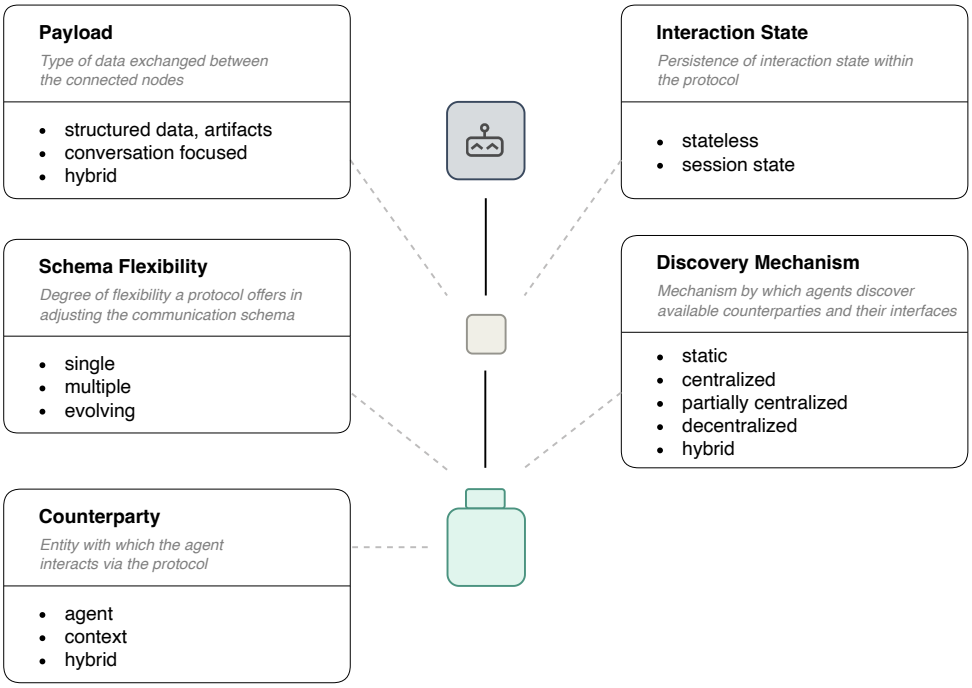


Fig. 3 Finalized Taxonomy: all dimensions and characteristics

5.2 Accepted Dimensions

Throughout the iterative process, various dimensions were examined based on discovered common and distinctive characteristics. All considered dimensions are now described in detail, separated into accepted and rejected ones.

For each dimension, we present its corresponding values and assess whether it satisfies mutual exclusivity and exhaustiveness. For accepted dimensions, we additionally cite foundational research supporting the dimension and its values.

The finalized taxonomy comprises five dimensions, as shown in Figure 3.

5.2.1 Counterparty

We begin with the most self-explanatory dimension: *counterparty*. It identifies the type of entity with which the agent interacts via the protocol. Its values are: *agent*, where the protocol connects one LLM agent to another; *context*, where non-agent entities such as tools, services, APIs, or data sources are connected; and *hybrid*, where both are supported.

The dimension is mutually exclusive and exhaustive, as *context* captures any non-agent counterparty by design. Although further subcategories within *context* could be defined, the dimension is intentionally kept concise.

This dimension aligns with Yang et al. (2025)'s protocol taxonomy. Their *object-orientation* dimension distinguishes *context-oriented* from *inter-agent* protocols,

corresponding to our *context* and *agent* values, respectively. We extend their scheme with a third value, *hybrid*, to accommodate protocols supporting both types.

5.2.2 Payload

Next, the *payload* dimension classifies the kind of data a protocol exchanges. A protocol falls into one of three categories: *structured data and artifacts*, where only structured data or artifacts, including meta-information, are exchanged; *conversation focused*, where the payload is message-centric and text is always part of the exchange; or *hybrid*, where both are supported.

To clarify this dimension further, we compare it with the three basic data types: structured, semi-structured, and unstructured (Eberendu et al. 2016). Rather than adopting that typology, our taxonomy classifies payloads according to the intended purpose a protocol is designed to support. A protocol is labeled *conversation focused* when text is always part of the payload and structured or semi-structured data appear only as optional extensions. Conversely, if the protocol can solely transmit structured data and artifacts drawn from any of Eberendu et al. (2016)'s three data types, it is assigned to the *structured data and artifacts* category.

5.2.3 Interaction State

This dimension captures the persistence of *interaction state* within a protocol, that is, whether the protocol implements a stateful unit of work between connected components. A protocol is either *stateless* or provides *session state* functionality.

Mutual exclusivity and exhaustiveness are guaranteed because the dimension is binary. A protocol that preserves context across messages is classified as *session state*, one that lacks such mechanisms is *stateless*.

Ling et al. (2004) identify four kinds of managed state in communication networks: stateless, soft state, session state, and hard state. Their definition of stateless, no information retained at the nodes, matches our equivalently named value directly. Session state, defined as information persisting only for the duration of a user session, corresponds to what we term *session statefulness* (Ling et al. 2004). The remaining categories, hard state, referring to long-term memory, and soft state, denoting temporarily cached information, fall outside the scope of this dimension. The dimension only concerns the duration of a single session and not what persists beyond it. This is further discussed in the context of persistent state in Section 5.3.3.

5.2.4 Discovery Mechanism

In dynamic, large-scale distributed systems, individual nodes are typically unaware of the precise identities or locations of other nodes. Therefore, efficient discovery mechanisms are essential for scalable and flexible agent networks (Ahmed and Boutaba 2010; Yang et al. 2025).

We identify two initial values: *static*, where the requesting agent must know the endpoint a priori, and *centralized*, where a registry maintains a list of accessible endpoints.

To ensure exhaustiveness, we drew on Ahmed and Boutaba (2010)'s survey of distributed search techniques in large-scale systems. They index two centralized content-sharing architectures, registry and index, which we combine into a single value, *centralized*, since we only are concerned with the discovery mechanism itself, not with how the registry is built. Their classification further motivates two additional values: *partially centralized*, where a limited set of supernodes assists with discovery, and *decentralized*, where no central authority exists and each peer maintains its own index, with discovery occurring via network broadcasting. Finally, we add a *hybrid* value for protocols that support multiple discovery mechanisms beyond static configuration.

Mutual exclusivity is guaranteed by arranging the values hierarchically. *Static* represents the baseline. If a protocol offers a *centralized*, *partially centralized*, or fully *decentralized* discovery mechanism, it is classified as such, even though static configuration remains universally available. A protocol supporting multiple non-static mechanisms is assigned to *hybrid*.

5.2.5 Schema Flexibility

Next, we introduce the *schema flexibility* dimension, which captures the degree of flexibility a protocol offers in adjusting communication schemas, message formats, or dialogue structure. A protocol may allow only a *single* schema, one basic interaction pattern defined before runtime. Alternatively, it may support *multiple* static schemas, established a priori, from which the requester can choose at runtime. Finally, if the interacting components can negotiate new interaction styles at runtime, the protocol is classified as *evolving*.

In this dimension, the term *schema* follows the standard computer-science definition: a formal structure representing an engineered artifact (Bernstein et al. 2011). In our context, it refers in particular to the structure of the exchanged content. For instance, a tool invoked by an agent may require a fixed schema with a predefined set of typed variables (Anthropic 2024a), whereas an LLM-based agent might accept only a single text input, subject to constraints such as a maximum length (LangChain 2025a).

It is important to note that this dimension does not characterize the structural nature of the payload, that role belongs to the *payload* dimension. Instead, it specifies whether exchanged information must conform to a fixed schema, whether agents may select from multiple predefined schemas, or whether the protocol supports runtime negotiation of schema structure.

Because a protocol can be described at more than one level, we state the unit of analysis explicitly. This dimension is assessed at the level of the *interaction schemas the specification itself admits and how a requester becomes bound to one*, not at the level of the wire envelope and not at the level of any single deployment. Three consequences follow. First, a protocol that fixes one message shape for every exchange, so that a requester has no schema to choose, is *single*, even when the content it carries is free-form: LAP is classified this way because its */runs* and */threads* endpoints impose one invariant request and response structure and the agent payload is opaque to the protocol (LangChain 2025a). Second, a protocol that defines a mechanism by which counterparties declare an open-ended set of typed schemas that a requester discovers

and selects among before invocation is *multiple*, even though the transport envelope is uniform: MCP is classified this way because each server advertises its tools with individual, machine-readable input schemas which the client retrieves and must satisfy per tool, so the set of admissible content schemas is determined by the counterparty rather than by the specification (Anthropic 2024a). The uniform JSON-RPC envelope is therefore not what is being classified. Third, *evolving* is reserved for protocols in which the schema itself is an object of negotiation at runtime, so that a structure which existed in neither party's prior repertoire can come into use during the exchange. The distinction between *multiple* and *evolving* is thus whether the admissible set is fixed before the interaction and merely selected from, or is extended by the interaction.

5.3 Rejected Dimensions

Having examined all accepted dimensions, we now turn to three rejected dimensions and explain the rationale for their exclusion from the taxonomy.

5.3.1 Initiative Flow

This dimension captures which party may initiate communication. Its proposed values are: *unidirectional*, where only the client can initiate; *context-bound bidirectional*, where either party may send messages within an active session but only one side can initiate it; and *open bidirectional*, where any party can initiate communication at any time.

The dimension is rejected on two grounds: most protocols fall under *context-bound bidirectional*, and any protocol could in principle be implemented symmetrically to support open bidirectional initiation. It therefore fails the test of mutual exclusivity and adds little analytical value.

5.3.2 Simplicity

The examined protocols differ notably in their simplicity and intended lightweightness. Some, such as *A2A* and *LMOS*, incorporate a broad range of functionalities (Google 2025d; Eclipse Foundation 2025c), whereas others, such as *Agora*, *agents.json*, and *ACP*, target specifically defined use cases and explicitly prioritize a lightweight design (Agora Protocol 2025b; Wildcard 2025; ACP 2025a).

Nevertheless, this dimension poses an immediate problem: there are no clear boundaries to ensure mutually exclusive values. The notion of simplicity is too vague to construct a meaningful qualitative dimension, and it is therefore rejected.

5.3.3 Persistent State

This dimension classifies protocols by their ability to store data about prior interactions beyond a single or multi-turn exchange. Unlike the *interaction state* dimension (Section 5.2.3), it concerns exclusively long-term statefulness. The stored content might include a textual summary of an earlier chat session or a negotiated sub-protocol preserved for future interactions. Although individual nodes could store such state at the implementation level, this dimension focuses on whether the protocol itself promotes any form of persistence.

Its proposed values are *none*, where no persistence features are offered; *metadata*, where the protocol supports storage of metadata such as a negotiated schema; *context*, where concrete interaction context such as session summaries is preserved; and *hybrid*, where both metadata and context are explicitly maintained.

In practice, most protocols already implement *metadata* persistence by exchanging component descriptions, such as the *Agent Discovery Card* in A2A (Google 2025c) or the *Agent Detail* endpoint in ACP (ACP 2025a). Certain protocols additionally offer explicit features for context persistence: for example, LAP exposes a *store* endpoint that manages a persistent key–value repository accessible across interaction *threads* (LangChain 2025a).

Nonetheless, the dimension is rejected. Nearly every protocol already exchanges minimal metadata for node identification, and the protocol itself does not determine what connected nodes ultimately store persistently. Moreover, providing persistent state mechanisms is not an essential objective of these protocols.

5.4 Classifying Concrete Protocols

We now revisit the protocols used during taxonomy construction and assign them to the appropriate characteristics within the accepted dimensions. In the interest of conciseness, we only discuss the most salient classifications, those that best clarify and validate the taxonomy. A complete overview is provided in Table 2 and Table 3, the former optimized for pattern detection and the latter for informative depth.

5.4.1 Counterparty

Various protocols are explicitly designed to facilitate agent-to-agent collaboration and thus address the counterparty type *agent*. Agora (Marro et al. 2024), ANP (Chang et al. 2025), LAP (LangChain 2025b), and agntcy (AGNTCY 2025a) are classified as such. Moreover, A2A and ACP further emphasize that agents should use these protocols specifically for inter-agent communication, in contrast to protocols such as MCP, which are intended for connecting to context (Google 2025b; ACP 2025b). An *MCP server* is defined as the counterparty addressed by the agent, which can expose resources (e.g. documents or data), tools, and prompts (Anthropic 2024a).

5.4.2 Payload

MCP and agents.json are the only protocols assigned to the *structured data and artifacts* payload type. MCP is designed for the exchange of structured requests and data responses between LLM-based agents and external tools, data sources, or services (Anthropic 2024a). Further, agents.json does not handle actual data exchange, its sole purpose is to provide agents with a structured description of an existing API (Wildcard 2025).

All remaining protocols are classified as *hybrid*. They typically define a core unit of exchange capable of encapsulating both simple strings and arbitrary elements. The exchange is thus not limited to conversational text, purely structured or contextual information can be transmitted as well.

Table 2 Classification matrix of the nine analyzed protocols, for cross-protocol pattern comparison

Dimension	Characteristics	MCP (Anthropic 2024b,a)	A2A (Google 2025a,b,e,c,d)	LAP (LangChain 2025b,a)	agents .json (Wildcard 2025)	Agora (Marro et al. 2024; Agora Protocol 2025a,b)	ANP (ANP 2025; Chang et al. 2025; Chang 2024b,a)	LMOS (Eclipse Foundation 2025a,c,b)	ACP (ACP 2025; ACP 2025b,a)	agntcy (AGNTCY 2025c,e,b,a)
Counterparty	Agent		x	x		x	x		x	x
	Context	x			x					
	Hybrid							x		
Interaction State	Stateless	x			x					
	Session state		x	x		x	x	x	x	x
Discovery Mechanism	Static	x		x	x	x				
	Centralized		x				x		x	x
	Partially centralized									
	Decentralized									
	Hybrid							x		
Payload	Struct. data, artifacts	x			x					
	Conversation focused									
	Hybrid		x	x		x	x	x	x	x
Schema Flexibility	Single			x	x					
	Multiple	x	x					x	x	x
	Evolving					x	x			

Table 3 Per-protocol classification and rationale for the nine analyzed protocols

Protocol	Counterparty	Interaction State	Discovery Mechanism	Payload	Schema Flexibility	Rationale
MCP (Anthropic 2024b,a)	Context	Stateless	Static	Structured data, artifacts	Multiple	Stateless core. Exposes tools; requires known endpoints.
A2A (Google 2025a,b,e,c,d)	Agent	Session state	Centralized	Hybrid	Multiple	Inter-agent tasks and streaming. Uses JSON card or registry.
LAP (LangChain 2025b,a)	Agent	Session state	Static	Hybrid	Single	REST API with fixed endpoints for multi-turn sessions.
agents.json (Wildcard 2025)	Context	Stateless	Static	Structured data, artifacts	Single	Stateless OpenAPI JSON spec for LLM API discovery.
Agora (Marro et al. 2024; Agora Protocol 2025a,b)	Agent	Session state	Static	Hybrid	Evolving	Decentralized. Plain-text schemas dynamically negotiated at runtime.
ANP (ANP 2025; Chang et al. 2025; Chang 2024b,a)	Agent	Session state	Centralized	Hybrid	Evolving	Stateful meta-layer dynamically negotiates structures and formats.
LMOS (Eclipse Foundation 2025a,c,b)	Hybrid	Session state	Hybrid	Hybrid	Multiple	Transport-agnostic. Standardizes metadata with hybrid discovery modes.
ACP (ACP 2025; ACP 2025b,a)	Agent	Session state	Centralized	Hybrid	Multiple	Multi-modal exchange. Agents register on centralized servers.
agntcy (AGNTCY 2025c,e,b,a)	Agent	Session state	Centralized	Hybrid	Multiple	OpenAPI extension for state/streams via central manifests.

5.4.3 Interaction State

All protocols designed for agent-to-agent communication support multi-turn interactions through stateful units of work persisting across message exchanges. These units are named differently across protocols, like threads (AGNTCY 2025e; LangChain 2025a), tasks (Google 2025e), or in the case of Agora, realized through a simple *multiround* flag that enables the message structure to maintain *session state* (Agora Protocol 2025b).

In contrast, the core *MCP* protocol and *agents.json* define no explicit mechanism for session statefulness and are therefore *stateless* by design (Anthropic 2024a; Wildcard 2025).

5.4.4 Discovery Mechanism

First, MCP illustrates an important distinction: although agents can discover a server's capabilities at runtime, the *discovery mechanism* dimension classifies MCP as *static*, since the requesting agent must still know the server's base address a priori (Anthropic 2024a).

A2A, by contrast, supports three discovery strategies, all relying on a *JSON Agent Card* as the agent's self-description. An agent may host its card at a standardized path on its own domain, share it through private channels, or publish and query cards via a centralized registry. Although static configuration is a primary discovery mechanism, the presence of a registry-based option places the protocol within the *centralized* discovery class (Google 2025d).

All other protocols fall into either the *static* or *centralized* category, with the exception of LMOS, which is classified as *hybrid* (Eclipse Foundation 2025b).

5.4.5 Schema Flexibility

Most protocols either specify a *single* concrete interface (LangChain 2025a; Wildcard 2025) or allow *multiple* distinct schemas to be defined in advance (Anthropic 2024a; Eclipse Foundation 2025c; AGNTCY 2025a).

More interesting are the two protocols implementing *evolving* schema functionality. In Agora, a core concept is the *Protocol Document*, a plain-text description of everything an agent must understand to follow a given protocol. These documents are actively constructed and negotiated at runtime. Initial interactions occur entirely in natural language, but once a more efficient protocol has been established, subsequent exchanges can adopt a more structured format, possibly without any message parameters (Marro et al. 2024). ANP similarly supports dynamic negotiation of payload formats and interaction structures, enabling agents to select or generate a schema at runtime (Chang 2024b).

6 Discussion

We have developed a qualitative taxonomy to efficiently classify and analyze communication protocols for LLM agents. This hierarchical classification system helps to structure and thereby clarify an increasingly complex, rapidly growing field. Future

protocols can be intuitively classified and compared at an abstract level, enabling researchers to identify overarching trends and anticipate potential innovations.

6.1 Insights and Implications

Let us consider some important findings and implications that emerged during taxonomy development, backed by the classifications of all nine protocols (see Table 2 and Table 3).

If a protocol implements agent-to-agent communication (7 of 9), a *session state* functionality is always available. This is expected, because contemporary LLM based agents rely on multi-turn interaction, which in turn requires state persistence. Moreover, none of the agent-to-agent protocols restrict themselves to *conversation-focused* payloads (7 of 9). Instead, all are consistently *hybrid*; while they can transmit textual messages, they also support purely structured payloads whenever appropriate. Furthermore, a clear trend towards schema flexibility emerges from the data. The majority of protocols permit multiple schema definitions (7 of 9), and two (2 of 9) additionally allow schemas to evolve during runtime. Schema negotiation is thereby regarded as a central component of agent-to-agent protocols, enabling LLM-based agents to fully leverage their capabilities. With respect to discovery mechanisms, only the LMOS protocol (Eclipse Foundation 2025b) truly incorporates a decentralized, peer-to-peer approach. Most protocols instead rely on either a centralized registry (4 of 9) or static configuration (4 of 9). Should the Internet of Agents emerge as a dominant paradigm, decentralized approaches should be explored more and may prove increasingly important.

Looking forward, a central question is whether the field will converge on a single, monolithic standard or adopt a more modular architecture. Initially, the functional overlap between emerging protocols might suggest a short-term convergence toward a unified standard capable of supporting both agent-to-agent and agent-to-context interactions. Notably, A2A and agntcy present themselves as extensions of MCP (Google 2025b; AGNTCY 2025d), though the primary organization behind MCP, Anthropic, has not officially endorsed either of them. While modifying an agent-to-agent protocol like A2A to incorporate MCP functionality would be relatively straightforward, drawing on historical precedents in computer networking, we argue that a monolithic "winner-takes-all" standard is unlikely to prevail. Instead, mirroring the layered OSI model, illustrated in Figure 4, our analysis suggests that the future of multi-agent system communication will evolve toward a federated, layered protocol stack. Layered architectures for agent communication have been proposed before on design-principle grounds, notably by Fleming et al. (2025) and Xu et al. (2026); what our classification adds is an empirical basis for the placement of concrete protocols within such a stack. Under this architecture, the Internet of Agents will likely utilize lightweight specifications, such as agents.json, for static capability discovery; defer to highly structured context-protocols, e.g., MCP, for secure tool execution; and reserve session-aware, schema-evolving protocols, e.g., ANP or Agora, for complex, multi-turn deliberations.

A notable gap across all investigated protocols is the widespread absence of privacy safeguards, compliance checks, and policy enforcement mechanisms. This shortcoming will become more severe as agents are increasingly deployed in safety-critical domains

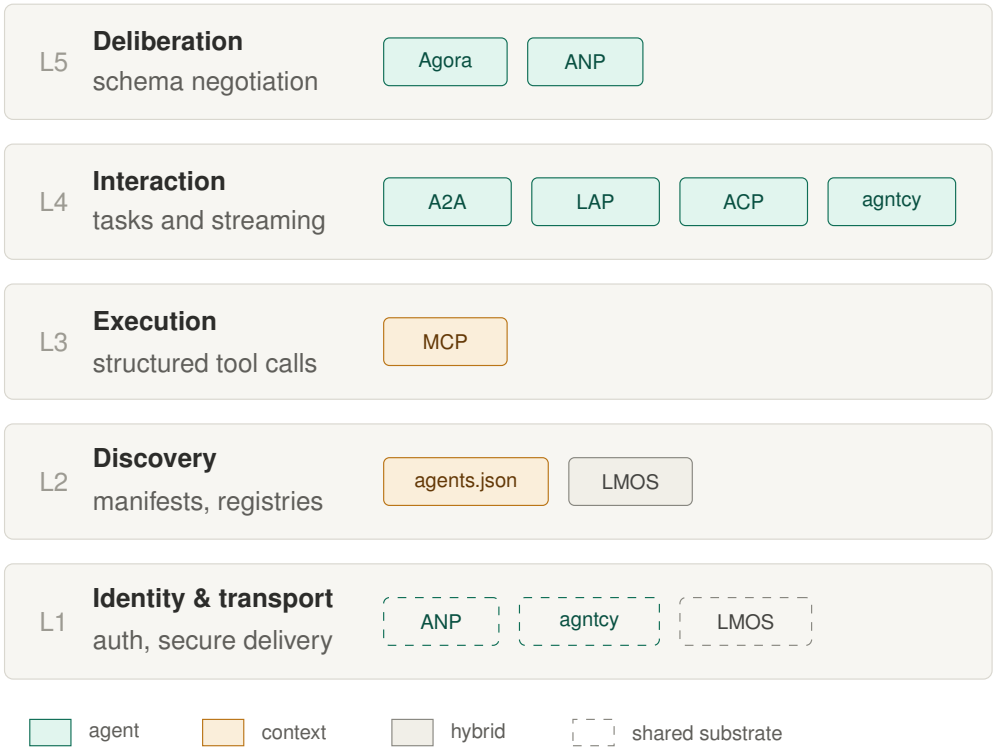


Fig. 4 A candidate layered protocol stack for the Internet of Agents. Layers are ordered by the communication trilemma: lower layers favor efficiency and portability (rigid schemas, stateless, structured payloads), upper layers favor versatility (session-aware, schema-evolving). Each protocol is placed based on its primary role in Table 3; chip color denotes the counterparty type, and dashed chips indicate protocols that additionally provide the identity/transport substrate. The layering is a proposed instantiation of the federated-stack argument in Section 6.1, not an observed fact, and it is one of several layered architectures proposed for the Internet of Agents (Fleming et al. 2025; Xu et al. 2026); the distinguishing feature here is that each placement follows from the classification rather than from an a priori functional decomposition.

such as healthcare or human resources, where mechanisms for protecting personal and security-sensitive data are essential (Guo et al. 2024).

6.2 Evaluating the Communication Trilemma

Applying Marro et al. (2024)'s Agent Communication Trilemma, introduced in Section 3, to our taxonomy reveals stark architectural trade-offs. The trilemma dictates that a protocol cannot simultaneously maximize versatility, efficiency, and portability.

This trade-off is highly visible at the extremes of our *Schema Flexibility* and *Payload* dimensions. Protocols designed primarily for context-interaction, such as MCP, maximize portability and efficiency by enforcing rigid schemas, stateless interactions,

and strictly structured data payloads. This strictness eliminates token-heavy negotiation, making them ideal for predictable, high-throughput agent-to-tool invocations. Conversely, protocols with *evolving* schemas, such as Agora and ANP, maximize versatility to facilitate dynamic, open-ended multi-agent debates. However, this adaptability introduces significant token overhead and latency during schema negotiation, severely reducing efficiency.

Interestingly, the taxonomy reveals that the majority of surveyed protocols, such as A2A, LMOS, and ACP, navigate the center of this trilemma. By adopting *hybrid* payload structures, *session state* persistence, and *multiple* pre-established schemas, they provide enough versatility to support complex interactions without incurring the extreme efficiency penalties of runtime protocol negotiation. Ultimately, mapping our taxonomy against the trilemma reinforces the conclusion from Section 6.1: the impossibility of a single, omnipotent protocol strongly supports the necessity of a layered protocol stack for the future Internet of Agents.

6.3 Limitations and Future Work

Three limitations bound the claims made above. First, the classifications reported in Tables 2 and 3 were produced by the authors reading each protocol's primary specification and repository, and were reconciled by discussion until consensus was reached. We did not conduct a formally independent coding exercise, and we therefore report no inter-rater agreement statistic. Judgements on dimensions where a specification is permissive rather than prescriptive, in particular *discovery mechanism* and *schema flexibility*, retain a degree of interpretive latitude, and a replication with independent coders would strengthen confidence in them. To support such replication we state the decision rule for every dimension explicitly in Section 5.2 and record the rationale for each individual assignment in Table 3, so that any disagreement can be located at a specific claim rather than at the classification as a whole. Second, the sample is a snapshot: all searches and repository metrics correspond to 12 Mar 2026, and a domain moving at this pace will have changed by publication, which is a further reason we treat the layered stack in Figure 4 as a proposed instantiation rather than an observed fact. Third, the taxonomy classifies what specifications state, not what deployments do; a protocol may be used in ways its specification does not anticipate, and conformance in practice is outside our scope.

Future research could enrich our taxonomy with further dimensions that capture the domain in greater detail. Promising extensions include more technical protocol dimensions, authentication and security mechanisms, and the means by which protocols embed policies and norms. Also, classifying new protocols as they emerge would both strengthen the taxonomy's validity and allow systematic tracking and analysis of ongoing developments within the domain. Moreover, the taxonomy provides a practical basis for evaluating which protocols are most suitable for specific use cases and application domains.

Finally, the governance gap identified in Section 6.1 raises a specific architectural question. Embedding policy, privacy, and compliance mechanisms into each communication standard would trade directly against the efficiency and portability poles of the

trilemma. An alternative, pursued by several recent proposals, is to locate these concerns in dedicated layers above transport (Xu et al. 2026; Fleming et al. 2025), with trust frameworks supplying the enforcement and audit machinery (Zou et al. 2025; Ranjan et al. 2025). Whether such a separation can be sustained once protocols are composed in practice is an open question, and one that our taxonomy is positioned to help address; the dimensions could serve as an admission rubric for protocol selection within a layered governance model, in which counterparty, payload, negotiated schema, and session state jointly determine which protocol an interaction is admitted to at a given layer.

7 Conclusion

In this paper, we have developed a comprehensive technical taxonomy of communication protocols for LLM-based agents. Motivated by the growing fragmentation of agent frameworks and protocol proposals, we examined how emerging protocols structure communication between LLM agents, other agents, tools, services, APIs, and external information systems. To develop the taxonomy, we followed a structured approach, defining the taxonomy's purpose, meta-characteristic, ending conditions, and iterative development process before applying the resulting framework to nine existing protocol implementations. By following a well-structured taxonomy development approach, we ensured methodological quality, producing a framework which allows efficient exploration and analysis of the diverse landscape of LLM agent communication protocols.

In addition, we identified several clear patterns by applying the taxonomy to nine concrete protocols. Most agent-to-agent protocols support multi-turn statefulness and the capacity to exchange both textual messages and structured data. Furthermore, schema negotiation is considered essential for efficient flexibility. Most importantly, our analysis suggests a possible short-term convergence pressure toward protocols that unify agent-to-agent and agent-to-context communication. In the long term, rather than converging on a single monolithic standard, the field will likely evolve toward a federated, layered protocol stack, similar to the OSI model in traditional computer networking. Such an architecture would seamlessly integrate specialized protocols, ranging from lightweight capability discovery specifications to highly structured tool execution and schema-evolving deliberation mechanisms, in order to support the full spectrum of agent communication scenarios efficiently. However, whether this leads to a single dominant standard or to a small set of interoperable protocol families remains an open question.

As LLMs continue to advance, LLM agents will grow more intelligent and capable of tackling real-world challenges. Likewise, LLM-powered multi-agent systems hold the potential to surpass the scalability constraints of single models. Consequently, the means by which agents communicate is a critical design element in building decentralized, flexible, and efficient agent networks, laying the groundwork for a future in which agents may operate partially or even fully autonomously in handling complex tasks.

Declarations

Funding

No funding was received for conducting this study.

Competing interests

The authors have no relevant financial or non-financial interests to disclose.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Data availability

No new data were generated. All material analysed in this study consists of publicly available protocol specifications and repositories, each cited in the reference list with the access date and, where applicable, the commit hash. The complete classification of all nine protocols, which constitutes the analytical output of this study, is reported in full in Tables 2 and 3.

Materials availability

Not applicable.

Code availability

Not applicable.

Author contribution

L. Sander and H. K. Gidey: Conceptualization, Methodology, Formal analysis, Writing – original draft. A. Lenz and A. Knoll: Supervision. All authors reviewed and approved the final manuscript.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., *et al.*: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023) https://doi.org/10.48550/arXiv.2303.08774

Ahmed, R., Boutaba, R.: A survey of distributed search techniques in large scale distributed systems. IEEE Communications Surveys & Tutorials **13**(2), 150–167 (2010) https://doi.org/10.1109/SURV.2011.040410.00097

ACP: Agent Communication Protocol (ACP). GitHub repository. Accessed 12 March 2026. (2025). https://github.com/i-am-bee

ACP: Core Concepts. Accessed 12 Mar 2026 (2025). https://agentcommunicationprotocol.dev/core-concepts

ACP: MCP and A2A - How does ACP compare to other AI protocols. Accessed 12 Mar 2026 (2025). https://agentcommunicationprotocol.dev/about/mcp-and-a2a

AGNTCY: Agent Connect Protocol. Accessed 12 Mar 2026 (2025). https://spec.acp.agntcy.org/#description/introduction

AGNTCY: Agent Discovery Service. Accessed 12 Mar 2026 (2025). https://docs.agntcy.org/dir/overview/

AGNTCY: AGNTCY. GitHub repository. Accessed 12 March 2026. (2025). https://github.com/agntcy

AGNTCY: AGNTCY Origins. Accessed 12 Mar 2026 (2025). https://docs.agntcy.org

AGNTCY: Threads Specification. Accessed 12 Mar 2026 (2025). https://spec.acp.agntcy.org/#tag/threads

Agora Protocol: Agora Protocol Python. GitHub repository. Commit: d4628c312d34d041bbdfe2b1bc8e7c0dab7f07d6. Accessed 12 March 2026. (2025). https://github.com/agora-protocol/python

Agora Protocol: Agora Standard Specification. Accessed 12 Mar 2026 (2025). https://agoraprotocol.org/docs/protocol/specification

Azamfirei, R., Kudchadkar, S.R., Fackler, J.: Large language models and the perils of their hallucinations. Critical Care **27**(1), 120 (2023) https://doi.org/10.1186/s13054-023-04393-x

ANP: AgentConnect. GitHub repository. Commit: c9270e23407eff7c0557281ef8178e0982afa7ec. Accessed 12 March 2026. (2025). https://github.com/agent-network-protocol/AgentConnect

Anthropic: MCP Core Architecture. Accessed 12 Mar 2026 (2024). https://modelcontextprotocol.io/docs/concepts/architecture

Anthropic: Model Context Protocol. GitHub repository. Accessed 12 March 2026. (2024). https://github.com/modelcontextprotocol

Anthropic: System Card: Claude Opus 4 & Claude Sonnet 4. Accessed 12 Mar 2026 (2025). https://www.anthropic.com/claude-4-system-card

Bailey, K.D.: Typologies and Taxonomies: An Introduction to Classification Techniques vol. 12. Sage, Thousand Oaks, CA (1994)

Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., *et al.*: Graph of thoughts: Solving elaborate problems with large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 17682–17690 (2024). https://doi.org/10.1609/aaai.v38i16.29720

Bang, Y., Cahyawijaya, S., Lee, N., Dai, W., Su, D., Wilie, B., Lovenia, H., Ji, Z., Yu, T., Chung, W., *et al.*: A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. In: Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-pacific Chapter of the Association for Computational Linguistics (volume 1: Long Papers), pp. 675–718 (2023). https://doi.org/10.18653/v1/2023.ijcnlp-main.45

Bernstein, P.A., Madhavan, J., Rahm, E.: Generic schema matching, ten years later. Proceedings of the VLDB Endowment **4**(11), 695–701 (2011) https://doi.org/10.14778/3402707.3402710

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.*: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020) https://doi.org/10.48550/arXiv.2005.14165

Bouzenia, I., Pradel, M.: You name it, I run it: An LLM agent to execute tests of arbitrary projects. Proceedings of the ACM on Software Engineering **2**(ISSTA), 1054–1076 (2025) https://doi.org/10.1145/3728922

CAMEL: CAMEL. GitHub repository. Commit: da86121346e79ba922a6f5635cf5213e576268dc. Accessed 12 March 2026. (2023). https://github.com/camel-ai/camel

Cui, E., Cheng, Y., She, R., Liu, D., Liang, Z., Guo, M., Li, T., Wei, Q., Xing, W., Zhong, Z.: AgentDNS: A Root Domain Naming System for LLM Agents (2025). https://doi.org/10.48550/arXiv.2505.22368 . https://arxiv.org/abs/2505.22368

Castrillo, V.d.L., Gidey, H.K., Lenz, A., Knoll, A.: Fundamentals of building autonomous LLM agents. arXiv preprint arXiv:2510.09244 (2025) https://doi.org/10.48550/arXiv.2510.09244

Chang, G.: ANP Agent Discovery Protocol Specification. Accessed 12 Mar 2026 (2024). https://agentnetworkprotocol.com/en/specs/08-anp-agent-discovery-protocol-specification/#anp-agent-discovery-protocol-specification-draft

Chang, G.: ANP Communication Meta-Protocol Specification. Accessed 12 Mar 2026

(2024). https://agent-network-protocol.com/specs/communication.html

Cerf, V., Kahn, R.: A protocol for packet network intercommunication. IEEE Transactions on communications **22**(5), 637–648 (1974) https://doi.org/10.1109/TCOM.1974.1092259

Chen, X., Lin, M., Schärli, N., Zhou, D.: Teaching large language models to self-debug. In: International Conference on Learning Representations, vol. 2024, pp. 8746–8825 (2024). https://doi.org/10.48550/arXiv.2304.05128

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) https://doi.org/10.48550/arXiv.2508.00007

Cemri, M., Pan, M.Z., Yang, S., Agrawal, L.A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., *et al.*: Why do multi-agent LLM systems fail? arXiv preprint arXiv:2503.13657 (2025) https://doi.org/10.48550/10.52202/085713-4082

Chatterjee, A., Renduchintala, H.K., Bhatia, S., Chakraborty, T.: POSIX: A prompt sensitivity index for large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024, pp. 14550–14565 (2024). https://doi.org/10.18653/v1/2024.findings-emnlp.852

CrewAI: CrewAI. GitHub repository. Commit: 2ab6c315448219bf7b4e68ece5d6dc8776004d9c. Accessed 12 March 2026. (2023). https://github.com/crewAIInc/crewAI

Chen, W., Su, Y., Zuo, J., Yang, C., Yuan, C., Chan, C.-M., Yu, H., Lu, Y., Hung, Y.-H., Qian, C., *et al.*: Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In: International Conference on Learning Representations, vol. 2024, pp. 20094–20136 (2024). https://doi.org/10.48550/arXiv.2308.10848

Cao, C., Wang, F., Lindley, L., Wang, Z.: Managing Linux servers with LLM-based AI agents: An empirical evaluation with GPT4. Machine Learning with Applications **17**, 100570 (2024) https://doi.org/10.1016/j.mlwa.2024.100570

Duéñez-Guzmán, E.A., Sadedin, S., Wang, J.X., McKee, K.R., Leibo, J.Z.: A social path to human-like artificial intelligence. Nature machine intelligence **5**(11), 1181–1188 (2023) https://doi.org/10.1038/s42256-023-00754-x

Du, C., Wang, C., Chao, Y., Xie, X., Cui, Y.: AI agent communication from internet architecture perspective: Challenges and opportunities. arXiv preprint arXiv:2509.02317 (2025) https://doi.org/10.48550/arXiv.2509.02317

Eberendu, A.C., *et al.*: Unstructured data: An overview of the data of Big Data. International Journal of Computer Trends and Technology **38**(1), 46–50 (2016)

Eclipse Foundation: Eclipse LMOS. GitHub repository. Accessed 12 March 2026. (2025). https://github.com/eclipse-lmos

Eclipse Foundation: LMOS Discovery. Accessed 12 Mar 2026 (2025). https://eclipse.dev/lmos/docs/lmos_protocol/discovery/

Eclipse Foundation: LMOS Introduction. Accessed 12 Mar 2026 (2025). https://eclipse.dev/lmos/docs/lmos_protocol/introduction/

Evans, J.S.B., Over, D.E., Manktelow, K.I.: Reasoning, decision making and rationality. Cognition **49**(1-2), 165–187 (1993) https://doi.org/10.1016/0010-0277(93)90039-X

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) https://doi.org/10.48550/arXiv.2505.02279

Fetch.ai: uAgents - Exchange protocol. Accessed 12 Mar 2026 (2025). https://uagents.fetch.ai/docs

Finin, T., Fritzson, R., McKay, D.P., McEntire, R., *et al.*: KQML-a language and protocol for knowledge and information exchange. In: 13th Int. Distributed Artificial Intelligence Workshop, pp. 93–103 (1994)

Firecrawl: Open Source vs Cloud. Accessed 12 Mar 2026 (2024). https://docs.firecrawl.dev/contributing/open-source-or-cloud

Fleming, C., Muscariello, L., Pandey, V., Kompella, R.: A layered protocol architecture for the internet of agents. arXiv preprint arXiv:2511.19699 (2025) https://doi.org/10.48550/arXiv.2511.19699

Fu, Y., Peng, H., Khot, T., Lapata, M.: Improving language model negotiation with self-play and in-context learning from AI feedback. arXiv preprint arXiv:2305.10142 (2023) https://doi.org/10.48550/arXiv.2305.10142

Garg, D.: Agent Protocol. GitHub repository. Commit: 5047199152dc6e1ba83320b1b59f94af24046dcf. Accessed 12 March 2026. (2023). https://github.com/Div99/agent-protocol

Ghafarollahi, A., Buehler, M.J.: SciAgents: Automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. Advanced Materials **37**(22), 2413523 (2025) https://doi.org/10.1002/adma.202413523

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N.V., Wiest, O., Zhang, X.: Large language model based multi-agents: A survey of progress and challenges. arXiv preprint arXiv:2402.01680 (2024) https://doi.org/10.48550/arXiv.2402.01680

Garousi, V., Felderer, M., Mäntylä, M.V.: Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and software technology **106**, 101–121 (2019) https://doi.org/10.1016/j.infsof.2018.09.006

Gidey, H.K., Hillmann, P., Karcher, A., Knoll, A.: Towards cognitive bots: Architectural research challenges. In: International Conference on Artificial General Intelligence, pp. 105–114 (2023). https://doi.org/10.1007/978-3-031-33469-6_11 . Springer Nature Switzerland Cham

Gidey, H.K., Hillmann, P., Karcher, A., Knoll, A.: User-like bots for cognitive automation: A survey. In: International Conference on Machine Learning, Optimization, and Data Science, pp. 388–402 (2023). https://doi.org/10.1007/978-3-031-53966-4_29 . Springer Nature Switzerland Cham

Guo, X., Huang, K., Liu, J., Fan, W., Vélez, N., Wu, Q., Wang, H., Griffiths, T.L., Wang, M.: Embodied LLM agents learn to cooperate in organized teams. IEEE Transactions on Computational Social Systems (2026) https://doi.org/10.1109/TCSS.2025.3637527

Gidey, H.K., Huber, N., Lenz, A., Knoll, A.: Affordance representation and recognition for autonomous agents. arXiv preprint arXiv:2510.24459 (2025) https://doi.org/10.48550/arXiv.2510.24459

Gąsieniec, L., Kuszner, Ł., Latif, E., Parasuraman, R., Spirakis, P., Stachowiak, G.: Anonymous Self-Stabilising Localisation via Spatial Population Protocols (2025). https://doi.org/10.4230/LIPIcs.ISAAC.2025.35 . https://arxiv.org/abs/2411.08434

Gidey, H.K., Kesseler, M., Stangl, P., Hillmann, P., Karcher, A.: Document-based knowledge discovery with microservices architecture. In: International Conference on Intelligent Systems and Pattern Recognition, pp. 146–161 (2022). https://doi.org/10.1007/978-3-031-08277-1_13 . Springer International Publishing Cham

Gamalielsson, J., Lundell, B., Brax, C., Persson, T., Mattsson, A., Gustavsson, T., Feist, J.: Open source software reference implementations for standards issued by different standards setting organisations: Availability, perceptions and practices. Journal of Standardisation **3**(2) (2024) https://doi.org/10.59490/jos.2024.7140

Gidey, H.K., Lenz, A., Knoll, A.: A pattern language for resilient visual agents. arXiv preprint arXiv:2604.28001 (2026) https://doi.org/10.48550/arXiv.2604.28001

Guo, J., Li, J., Li, D., Tiong, A.M.H., Li, B., Tao, D., Hoi, S.C.: From images to textual prompts: Zero-shot visual question with frozen large language models. arXiv preprint arXiv:2212.10846 (2022) https://doi.org/10.1109/CVPR52729.2023.01046

Gidey, H.K., Marmsoler, D., Ascher, D.: Modeling adaptive self-healing systems. arXiv

preprint arXiv:2304.12773 (2023) https://doi.org/10.48550/arXiv.2304.12773

Google: A2A. GitHub repository. Commit: 74ec26167cfb8e87d71dec9a2defabbbbfab55ac. Accessed 12 March 2026. (2025). https://github.com/a2aproject/A2A

Google: A2A and MCP: Complementary Protocols for Agentic Systems. Accessed 12 Mar 2026 (2025). https://a2aproject.github.io/A2A/latest/topics/a2a-and-mcp/

Google: Agent Discovery in A2A. Accessed 12 Mar 2026 (2025). https://a2aproject.github.io/A2A/latest/topics/agent-discovery/

Google: Agent2Agent (A2A) Protocol Specification. Accessed 12 Mar 2026 (2025). https://a2aproject.github.io/A2A/latest/specification/

Google: Key Concepts in A2A. Accessed 12 Mar 2026 (2025). https://a2aproject.github.io/A2A/latest/topics/key-concepts/

Hatalis, K., Christou, D., Myers, J., Jones, S., Lambert, K., Amos-Binks, A., Dannenhauer, Z., Dannenhauer, D.: Memory matters: The need to improve long-term memory in LLM-agents. In: Proceedings of the AAAI Symposium Series, vol. 2, pp. 277–280 (2023). https://doi.org/10.1609/aaaiss.v2i1.27688

Hsiao, V., Fine-Morris, M., Roberts, M., Smith, L.N., Hiatt, L.M.: A critical assessment of LLMs for solving multi-step problems: Preliminary results. In: AAAI 2025 Workshop LM4Plan (2025)

Huang, X., Liu, W., Chen, X., Wang, X., Wang, H., Lian, D., Wang, Y., Tang, R., Chen, E.: Understanding the planning of LLM agents: A survey. arXiv preprint arXiv:2402.02716 (2024) https://doi.org/10.48550/arXiv.2402.02716

Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly, 75–105 (2004) https://doi.org/10.2307/25148625

Huang, K., Narajala, V.S., Habler, I., Sheriff, A.: Agent Name Service (ANS): A Universal Directory for Secure AI Agent Discovery and Interoperability (2025). https://doi.org/10.1109/ICAIC67076.2026.11395757 . https://arxiv.org/abs/2505.10609

Hsieh, C.-P., Sun, S., Kriman, S., Acharya, S., Rekesh, D., Jia, F., Zhang, Y., Ginsburg, B.: RULER: What's the real context size of your long-context language models? arXiv preprint arXiv:2404.06654 (2024) https://doi.org/10.48550/arXiv.2404.06654

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Yau, S., Lin, Z., Zhou, L., *et al.*: MetaGPT: Meta programming for a multi-agent collaborative framework. In: International Conference on Learning Representations, vol. 2024, pp. 23247–23275 (2024). https://doi.org/10.48550/arXiv.2308.00352

Han, S., Zhang, Q., Yao, Y., Jin, W., Xu, Z.: LLM multi-agent systems: Challenges and open problems. arXiv preprint arXiv:2402.03578 (2024) https://doi.org/10.48550/arXiv.2402.03578

Jin, Z., Lu, W.: Tab-cot: Zero-shot tabular chain of thought. arXiv preprint arXiv:2305.17812 (2023) https://doi.org/10.48550/arXiv.2305.17812

Jennings, N.R., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Autonomous agents and multi-agent systems **1**(1), 7–38 (1998) https://doi.org/10.1023/A:1010090405266

Kim, G., Baldi, P., McAleer, S.: Language models can solve computer tasks. Advances in Neural Information Processing Systems **36**, 39648–39677 (2023) https://doi.org/10.52202/075280-1723

Kaesberg, L.B., Becker, J., Wahle, J.P., Ruas, T., Gipp, B.: Voting or consensus? Decision-making in multi-agent debate. arXiv preprint arXiv:2502.19130 (2025) https://doi.org/10.18653/v1/2025.findings-acl.606

Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. Advances in neural information processing systems **35**, 22199–22213 (2022) https://doi.org/10.52202/068431-1613

Katz, U., Levy, M., Goldberg, Y.: Knowledge Navigator: LLM-guided browsing framework for exploratory search in scientific literature. In: Findings of the Association for Computational Linguistics: EMNLP 2024, pp. 8838–8855 (2024). https://doi.org/10.18653/v1/2024.findings-emnlp.516

Kong, D., Lin, S., Xu, Z., Wang, Z., Li, M., Li, Y., Zhang, Y., Peng, H., Chen, X., Sha, Z., *et al.*: A survey of LLM-driven AI agent communication: Protocols, security risks, and defense countermeasures. arXiv preprint arXiv:2506.19676 (2025) https://doi.org/10.48550/arXiv.2506.19676

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., Sabharwal, A.: Decomposed prompting: A modular approach for solving complex tasks. arXiv preprint arXiv:2210.02406 (2022) https://doi.org/10.48550/arXiv.2210.02406

LangChain: LangGraph. GitHub repository. Commit: 139cad373b6cf712d0b6d456c698559b59073812. Accessed 12 March 2026. (2024). https://github.com/langchain-ai/langgraph

LangChain: Agent Protocol - OpenAPI Documentation. Accessed 12 Mar 2026 (2025). https://langchain-ai.github.io/agent-protocol/api.html

LangChain: LangChain Agent Protocol. GitHub repository. Commit: 5275a867c1e25df8f7a6c079660bfcdbafd04a27. Accessed 12 March 2026. (2025). https://github.com/langchain-ai/agent-protocol

LF Projects: AGENTS.md. GitHub repository. Commit: d1ac7f063d20e70015ed6732664049ae4ba9d74e. Accessed 12 March 2026. (2025). https://github.com/agentsmd/agents.md

Labrou, Y., Finin, T., Peng, Y.: Agent communication languages: The current landscape. IEEE Intelligent Systems and Their Applications **14**(2), 45–52 (1999) https://doi.org/10.1109/5254.757631

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., *et al.*: Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437 (2024) https://doi.org/10.48550/arXiv.2412.19437

Li, G., Hammoud, H., Itani, H., Khizbullin, D., Ghanem, B.: CAMEL: Communicative agents for" mind" exploration of large language model society. Advances in neural information processing systems **36**, 51991–52008 (2023) https://doi.org/10.5555/3666122.3668386

Ling, B.C., Kiciman, E., Fox, A.: Session state: Beyond soft state. In: NSDI, vol. 4, pp. 22–22 (2004). https://doi.org/10.5555/1251175.1251197

Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts. Transactions of the association for computational linguistics **12**, 157–173 (2024) https://doi.org/10.1162/tacl_a_00638

llms-txt: The /llms.txt file. Accessed 12 Mar 2026 (2024). https://llmstxt.org

Li, X., Liu, M., Yuen, C.: LLM agent communication protocol (LACP) requires urgent standardization: A telecom-inspired protocol is necessary. arXiv preprint arXiv:2510.13821 (2025) https://doi.org/10.48550/arXiv.2510.13821

Lála, J., O'Donoghue, O., Shtedritski, A., Cox, S., Rodriques, S.G., White, A.D.: PaperQA: Retrieval-augmented generative agent for scientific research. arXiv preprint arXiv:2312.07559 (2023) https://doi.org/10.48550/arXiv.2312.07559

Li, S., Padilla, S., Bras, P.L., Dong, J., Chantler, M.: A review of LLM-assisted ideation. arXiv preprint arXiv:2503.00946 (2025) https://doi.org/10.48550/arXiv.2503.00946

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., *et al.*: Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in neural information processing systems **33**, 9459–9474 (2020) https://doi.org/10.5555/3495724.3496517

Li, X., Wang, S., Zeng, S., Wu, Y., Yang, Y.: A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges. Vicinagearth **1**(1), 9 (2024) https://doi.org/10.1007/s44336-024-00009-2

Liu, J., Yu, K., Chen, K., Li, K., Qian, Y., Guo, X., Song, H., Li, Y.: ACPs: Agent Collaboration Protocols for the Internet of Agents (2025). https://doi.org/10.1109/IC-NIDC67200.2025.11390349 . https://arxiv.org/abs/2505.13523

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., *et al.*: Agentbench: Evaluating LLMs as agents. In: International Conference on Learning Representations, vol. 2024, pp. 52989–53046 (2024). https://doi.org/10.48550/arXiv.2308.03688

Li, J., Zhang, Q., Yu, Y., Fu, Q., Ye, D.: More agents is all you need. arXiv preprint arXiv:2402.05120 (2024) https://doi.org/10.48550/arXiv.2402.05120

Malone, T.W., Bernstein, M.: Handbook of Collective Intelligence. MIT press, Cambridge, MA (2015)

Mestha, H., Bania, K., Sathyanarayana, S.V., Liu, S., Srinivasan, A.: Multi-Turn Human-LLM Interaction Through the Lens of a Two-Way Intelligibility Protocol (2025). https://doi.org/10.48550/arXiv.2410.20600 . https://arxiv.org/abs/2410.20600

Masterman, T., Besen, S., Sawtell, M., Chao, A.: The landscape of emerging AI agent architectures for reasoning, planning, and tool calling: A survey. arXiv preprint arXiv:2404.11584 (2024) https://doi.org/10.48550/arXiv.2404.11584

Mele, A.R.: Autonomous Agents: From Self-control to Autonomy. Oxford University Press, New York, NY (1995)

Microsoft: AutoGen. GitHub repository. Commit: 7c536a8c95b9600f7c30e08473148839ecaf57c7. Accessed 12 March 2026. (2023). https://github.com/microsoft/autogen

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) https://doi.org/10.48550/arXiv.2410.11905

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., *et al.*: Self-refine: Iterative refinement with self-feedback. Advances in neural information processing systems **36**, 46534–46594 (2023) https://doi.org/10.52202/075280-2019

nlweb.ai: NLWeb. GitHub repository. Commit: 6c3398bfd183bc836a6ea24b78390db6d184cb3b. Accessed 12 March 2026. (2025). https://github.com/nlweb-ai/NLWeb

Nath, P.B., Uddin, M.M.: TCP-IP model in data communication and networking. American Journal of Engineering Research **4**(10), 102–107 (2015)

Nickerson, R.C., Varshney, U., Muntermann, J.: A method for taxonomy development and its application in information systems. European Journal of Information Systems **22**(3), 336–359 (2013) https://doi.org/10.1057/ejis.2012.26

Nwana, H.S.: Software agents: An overview. The knowledge engineering review **11**(3), 205–244 (1996) https://doi.org/10.1017/S026988890000789X

Oberländer, A.M., Lösser, B., Rau, D.: Taxonomy research in information systems: A systematic assessment. In: Proceedings of the 27th European Conference on Information Systems (ECIS), Stockholm and Uppsala, Sweden, pp. 1–17 (2019)

O'Brien, P.D., Nicol, R.C.: FIPA—towards a standard for software agents. BT Technology Journal **16**, 51–59 (1998) https://doi.org/10.1023/A:1009621729979

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A.H., Riedel, S.: Language models as knowledge bases? arXiv preprint arXiv:1909.01066 (2019) https://doi.org/10.18653/v1/D19-1250

Peng, Z., Wang, W., Dong, L., Hao, Y., Huang, S., Ma, S., Wei, F.: Kosmos-2: Grounding multimodal large language models to the world. arXiv preprint arXiv:2306.14824 (2023) https://doi.org/10.48550/arXiv.2306.14824

Pi, R., Yao, L., Gao, J., Zhang, J., Zhang, T.: PerceptionGPT: Effectively fusing visual perception into LLM. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 27124–27133 (2024). https://doi.org/10.1109/CVPR52733.2024.02561

Pouzin, L., Zimmermann, H.: A tutorial on protocols. Proceedings of the IEEE **66**(11), 1346–1370 (1978) https://doi.org/10.1109/PROC.1978.11145

Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., *et al.*: Chatdev: Communicative agents for software development. arXiv preprint arXiv:2307.07924 (2023) https://doi.org/10.18653/v1/2024.acl-long.810

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., *et al.*: ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In: International Conference on Learning Representations, vol. 2024, pp. 9695–9717 (2024). https://doi.org/10.48550/arXiv.2307.16789

Ruan, J., Chen, Y., Zhang, B., Xu, Z., Bao, T., Mao, H., Li, Z., Zeng, X., Zhao, R., *et al.*: TPTU: Task planning and tool usage of large language model-based ai agents. In: NeurIPS 2023 Foundation Models for Decision Making Workshop (2023). https://doi.org/10.48550/arXiv.2308.03427

Ranjan, R., Gupta, S., Singh, S.N.: Loka protocol: A decentralized framework for trustworthy and ethical AI agent ecosystems. arXiv preprint arXiv:2504.10915 (2025) https://doi.org/10.48550/arXiv.2504.10915

Russell, S.: Learning agents for uncertain environments. In: Proceedings of the Eleventh Annual Conference on Computational Learning Theory, pp. 101–103 (1998). https://doi.org/10.1145/279943.279964

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems **36**, 8634–8652 (2023) https://doi.org/10.52202/075280-0377

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E.H., Schärli, N., Zhou, D.: Large language models can be easily distracted by irrelevant context. In: International Conference on Machine Learning, pp. 31210–31227 (2023). https://doi.org/10.5555/3618408.3619699 . PMLR

Shen, Z.: LLM with tools: A survey. arXiv preprint arXiv:2409.18807 (2024) https://doi.org/10.48550/arXiv.2409.18807

Skyfire: Introduction to Skyfire. Accessed 12 Mar 2026 (2025). https://docs.skyfire.xyz/docs/introduction-to-skyfire

Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on computers **29**(12), 1104–1113 (1980) https://doi.org/10.1016/B978-0-934613-63-7.50039-5

Sager, P.J., Meyer, B., Yan, P., Wartburg-Kottler, R., Etaiwi, L., Enayati, A., Nobel, G., Abdulkadir, A., Grewe, B.F., Stadelmann, T.: AI agents for computer use: A review of instruction-based computer control, GUI automation, and operator assistants. arXiv preprint arXiv:2501.16150 (2025) https://doi.org/10.48550/arXiv.2501.16150

Sun, H., Zhuang, Y., Kong, L., Dai, B., Zhang, C.: Adaplanner: Adaptive planning from feedback with language models. Advances in neural information processing systems **36**, 58202–58245 (2023) https://doi.org/10.52202/075280-2537

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) https://doi.org/10.48550/arXiv.2501.06322

Tillmann, A.: Literature review of multi-agent debate for problem-solving. arXiv preprint arXiv:2506.00066 (2025) https://doi.org/10.48550/arXiv.2506.00066

Talebirad, Y., Nadiri, A.: Multi-agent collaboration: Harnessing the power of intelligent LLM agents. arXiv preprint arXiv:2306.03314 (2023) https://doi.org/10.48550/arXiv.2306.03314

Tanenbaum, A.S., Wetherall, D.J.: Computer Networks, 5th edn. Prentice Hall, Upper Saddle River, NJ (2011)

Valmeekam, K., Marquez, M., Olmo, A., Sreedharan, S., Kambhampati, S.: Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. Advances in Neural Information Processing Systems **36**, 38975–38987 (2023) https://doi.org/10.52202/075280-1693

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., *et al.*: Autogen: Enabling next-gen LLM applications via multi-agent conversations. In: First Conference on Language Modeling (2024). https://doi.org/10.48550/arXiv.2308.08155

Wang, Y., Guo, S., Pan, Y., Su, Z., Chen, F., Luan, T.H., Li, P., Kang, J., Niyato, D.: Internet of agents: Fundamentals, applications, and challenges. arXiv preprint arXiv:2505.07176 (2025) https://doi.org/10.1109/TCCN.2025.3623369

Wildcard: agents.json. GitHub repository. Commit: d1be22cfa453a323aa5adc135989f0fe381bae71. Accessed 12 March 2026. (2025). https://github.com/wild-card-ai/agents-json

Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. The knowledge engineering review **10**(2), 115–152 (1995) https://doi.org/10.1017/S0269888900008122

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., *et al.*: A survey on large language model based autonomous agents. Frontiers of Computer Science **18**(6), 186345 (2024) https://doi.org/10.1007/s11704-024-40231-1

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., *et al.*: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022) https://doi.org/10.52202/068431-1800

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 **68**(2), 121101 (2025) https://doi.org/10.1007/s11432-024-4222-0

Xu, W., Wang, T., Xia, Y., Zhang, S., Liew, S.C.: Agent-OSI: A layered protocol stack toward a decentralized internet of agents. arXiv preprint arXiv:2602.13795 (2026) https://doi.org/10.48550/arXiv.2602.13795

Xie, J., Zhang, K., Chen, J., Yuan, S., Zhang, K., Zhang, Y., Li, L., Xiao, Y.: Revealing the barriers of language agents in planning. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pp. 1872–1888 (2025). https://doi.org/10.18653/v1/2025.naacl-long.93

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) https://doi.org/10.48550/arXiv.2504.16736

Yu, J., Wang, X., Tu, S., Cao, S., Zhang-Li, D., Lv, X., Peng, H., Yao, Z., Zhang, X., Li, H., *et al.*: Kola: Carefully benchmarking world knowledge of large language models. In: International Conference on Learning Representations, vol. 2024, pp. 44594–44637 (2024). https://doi.org/10.48550/arXiv.2306.09296

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. Advances in neural information processing systems **36**, 11809–11822 (2023) https://doi.org/10.52202/075280-0517

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.: ReAct: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629 (2022) https://doi.org/10.48550/arXiv.2210.03629

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) https://doi.org/10.48550/arXiv.2502.14321

Zhang, Z., Bo, X., Ma, C., Li, R., Chen, X., Dai, Q., Zhu, J., Dong, Z., Wen, J.-R.: A survey on the memory mechanism of large language model based agents. arXiv preprint arXiv:2404.13501 (2024) https://doi.org/10.1145/3748302

Zeng, R., Fang, J., Liu, S., Meng, Z.: On the structural memory of LLM agents. arXiv preprint arXiv:2412.15266 (2024) https://doi.org/10.48550/arXiv.2412.15266

Zhang, C., Zhang, C., Li, C., Qiao, Y., Zheng, S., Dam, S.K., Zhang, M., Kim, J.U., Kim, S.T., Choi, J., *et al.*: One small step for generative AI, one giant leap for AGI: A complete survey on ChatGPT in AIGC era. arXiv preprint arXiv:2304.06488 (2023) https://doi.org/10.48550/arXiv.2304.06488

Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., *et al.*: A survey of large language models. arXiv preprint arXiv:2303.18223 (2023) https://doi.org/10.1007/s11704-026-60308-3

Zou, Z., Zhao, L., Liu, Z., Zhan, Q.: BlockA2A: Towards secure and verifiable agent-to-agent interoperability. arXiv preprint arXiv:2508.01332 (2025) https://doi.org/10.48550/arXiv.2508.01332