The Federated Agent Secure Transport (FAST): A Secure Fabric for the Orchestration of Multi-Agent Collaboration

Dheepak Gobinath [1] [0009-0003-9378-4341]

[1] Department of Electrical and Systems Engineering
University of Pennsylvania, Philadelphia, PA, USA
dheepakg@seas.upenn.edu

Abstract. To progress from isolated multi-agent simulations to a truly interconnected artificial intelligence, a paradigm shift from centralized control to decentralized communication is necessary. This paper introduces the Federated Agent Secure Transport (FAST), a novel, decentralized framework for Agent-to-Agent (A2A) communication that enables this shift. This architecture stands in contrast to prevailing multi-agent systems like Auto-GPT and CrewAI, which are confined to centralized, monolithic environments. Their reliance on inter-process calls within a single runtime inherently limits scalability, security, and interoperability, precluding the formation of a truly open agent ecosystem. The FAST framework synthesizes two core components: (1) a peer-to-peer protocol for secure, end-to-end encrypted message transport, and (2) a standardized Model Context Protocol (MCP) that serves as a semantic language for task delegation and data exchange. This paradigm, however, also introduces new attack surfaces at the semantic layer, where malicious agents can craft MCP payloads to perform sophisticated prompt-injection attacks on their counterparts. By decoupling the communication layer from the agent's core logic, this architecture allows agents from different entities to interact in a trustless manner. This fosters the conditions for an open market of specialized AI services and enables a new generation of scalable, resilient, and secure collaborative AI, laying the groundwork for a true Internet of Agents.

Keywords: Multi-Agent Systems, Decentralized AI, Federated Agent Secure Transport (FAST), Secure Communication, Peer-to-Peer Networks, Agent Composition, Collective Intelligence, End-to-End Encryption.

1 Introduction

To evolve from fragmented multi-agent simulations toward a genuinely interconnected form of artificial intelligence, it is essential to transition from centralized oversight to a model rooted in decentralized communication. The growing influence of Large Language Models (LLMs) has accelerated the development of Multi-Agent Systems (MAS), where specialized agents collaborate to solve complex tasks, signaling a shift toward Collective Intelligence. However, current implementations—such as Auto-GPT, which struggles with brittle task chaining [1], and CrewAI, which relies on rigid role definitions—expose architectural limitations that hinder scalability and adaptability. Moreover, the absence of robust trustless communication protocols [2] raises significant security concerns, including susceptibility to data leakage [3], impersonation [4], and centralized failure points, underscoring the need for decentralized, verifiable agent interactions. Most current agent architectures operate within monolithic applications, treating agents as simple objects confined to a single runtime environment and interacting via direct function calls. This centralized approach lacks the resilience and modularity needed for a scalable, global ecosystem. It inhibits true Agent Composition [15,16] and introduces structural bottlenecks [17] that obstruct growth at web-scale.

The core limitations of this paradigm stem from its reliance on a trusted, central orchestrator. This design is antithetical to the principles of Federated Systems, where independent entities must collaborate without a single point of control. It cannot guarantee Secure Communication between agents [5] operated by different, potentially untrusting parties, nor can it leverage the resilience and scalability of Peer-to-Peer Networks [6]. To unlock the next phase of AI collaboration, we require a new foundational layer—a communication protocol [7] that is inherently secure, decentralized, and open.

This paper introduces the Federated Agent Secure Transport (FAST), a novel, decentralized framework for Agent-to-Agent (A2A) communication. By establishing a common protocol, FAST enables true interoperability and agent composability, allowing agents to be treated as modular, swappable services in an open market. This open ecosystem is built upon a robust foundation that ensures scalability by removing the central server bottleneck and guarantees censorship resistance. The security of these interactions is anchored in a trustless system, where cryptographic proof replaces third-party mediators, and in decentralized identity (DID), which allows agents to manage their own identities via cryptographic key pairs. This two-layer approach ensures that while the network provides resilient and censorship-resistant message delivery, the semantic content of the collaboration remains completely confidential, enabling sophisticated and privacy-preserving agent interactions.

To comprehensively present our work, this paper is structured as follows. Section 2 reviews the existing landscape of multi-agent systems and related work. Section 3 provides a detailed overview of the Federated Agent Secure Transport (FAST), outlining its core architectural principles. Section 4 delves into the protocol specification, defining the message formats and orchestration patterns. Section 5 analyzes the framework's security and presents ablation studies. Section 6 compares the centralized and distributed architectures. Section 7 presents a unified discussion of the framework's implications and outlines future research directions. Finally, Section 8 concludes the paper.

2 Background and Related Work

The Federated Agent Secure Transport (FAST) is situated at the intersection of two rapidly advancing fields: LLM-powered Multi-Agent Systems (MAS) and decentralized communication protocols. This section first reviews the prevailing centralized paradigm in MAS, then analyzes its inherent architectural limitations. Subsequently, we survey a few decentralized communication and identity systems, identifying the foundational principles that inform the design of FAST.

2.1 The Rise of Centralized Multi-Agent Systems

The ability of LLMs to reason and plan has catalyzed a new wave of autonomous agent research. A foundational concept in this area is the "Reason and Act" paradigm, which demonstrates how LLMs can interleave reasoning with tool use to accomplish complex tasks [8]. Landmark research, such as the "Generative Agents" simulation from Stanford, demonstrated how dozens of LLM-powered agents could exhibit complex, emergent social behaviors within a simulated environment [9]. Concurrently, popular open-source frameworks like Auto-GPT, BabyAGI, and CrewAI have emerged, providing practical tools for building agentic workflows. These systems typically employ a hierarchical architecture: a primary "controller" or "orchestrator" agent breaks down a high-level goal into sub-tasks, which are then delegated to specialized "worker" or "expert" agents. Communication between these agents occurs through internal function calls or method invocations within a single, monolithic application. While powerful, this first generation of MAS represents a centralized, top-down approach to agent collaboration.

2.2 Limitations of Monolithic Architectures

The prevailing monolithic architecture, while effective for contained simulations and single-user applications, presents significant barriers to creating a global, open, and resilient agent ecosystem. First, interoperability is lacking [18–21], as agents are often tightly bound to their native frameworks, resulting in isolated ecosystems that inhibit true composability—where users could flexibly choose the most suitable expert agent from a broader, open marketplace. An agent built with LangChain cannot natively communicate with one built using CrewAI, a modern manifestation of a long-standing challenge in MAS where the lack of a standardized agent communication language has historically led to fragmented ecosystems [9]. Second, scalability becomes a challenge due to reliance on a central orchestrator, which not only introduces a single point of failure but also becomes a performance bottleneck. Third, privacy and security risks emerge in centralized models, where the orchestrator has access to all unencrypted agent communications [3]. This exposes sensitive task data to platform operators and creates a lucrative target for potential breaches.

2.3 Decentralized Communication Architectures

The principles underpinning FAST draw from decades of research in peer-to-peer (P2P) networking and secure messaging. Unlike client-server models, P2P networks distribute workload and data among peers, offering greater resilience and censorship resistance [10]. Modern applications like Nostr (Notes and Other Stuff Transmitted by Relays) provides a simple design based on cryptographic key-pair identities, signed messages, and a network of relays which are simple message stores and do not manage user identities [11]. This decoupling of data, identity, and application logic provides a blueprint for a censorship-resistant and interoperable communication layer. Other protocols like Secure Scuttlebutt (SSB) offer an alternative model based on append-only logs and gossip protocols, further demonstrating the viability of decentralized social interaction [12]. The cryptographic foundation for secure communication in these systems relies on principles perfected by protocols like Signal, which pioneered end-to-end encryption (E2EE) for messaging.

2.4 Identity and Trust in Distributed Systems

Traditional systems rely on a central authority to manage usernames and authenticate users. In contrast, decentralized environments adopt Self-Sovereign Identity (SSI)—a paradigm in which entities control their own identities without dependence on third parties. The technical foundation for this approach is the Decentralized Identifier (DID), a W3C standard that enables verifiable, decentralized digital identity [13]. This model empowers agents to generate their own identities and securely sign messages without requiring external permission. However, it introduces a fundamental challenge: trust. Without a central authority to validate an agent's reliability or authenticity, alternative mechanisms are needed. This has led to research into decentralized reputation systems and Web of Trust models, where trust is built through networks of attestations [14]. While a full reputation system lies beyond the scope of FAST's core protocol, our architecture is designed to remain compatible with such higher-level trust frameworks.

3 The Federated Agent Secure Transport (FAST) Architecture

The limitations of centralized Multi-Agent Systems, as outlined in the previous section, necessitate this fundamental architectural shift. The Federated Agent Secure Transport (FAST) provides a secure, resilient, and open foundation for a global ecosystem of autonomous agents.

3.1 Core Principles: Decentralization, Security, and Composability

FAST is built upon three foundational principles that directly address the shortcomings of monolithic agentic systems. By removing central points of control, FAST provides a network that is inherently resilient to outages and resistant to censorship or unilateral control by a single entity. This principle is the cornerstone for enabling a truly federated system where agents from different organizations and individuals can interact as peers. Security and privacy are not add-ons but are integral to the protocol. The architecture mandates end-to-end encryption for all substantive communication, ensuring that the network infrastructure itself cannot access or decipher the content of agent interactions. FAST enables agents to be treated as modular, interchangeable components. This composability is essential for creating an open market where developers can build specialized agents that can be dynamically discovered and integrated into more complex workflows, regardless of their underlying implementation.

3.2 The Two-Layer Stack: Transport and Semantics

FAST employs a decoupled, two-layer architectural stack. This separation of concerns is critical for the protocol's flexibility and extensibility. First, the *Transport Layer* is responsible for the secure and reliable delivery of arbitrary data packets from one agent's cryptographic identity to another. The transport layer handles addressing (who it's from, who it's to) and secure transport of the "envelope" (the encrypted data packet).

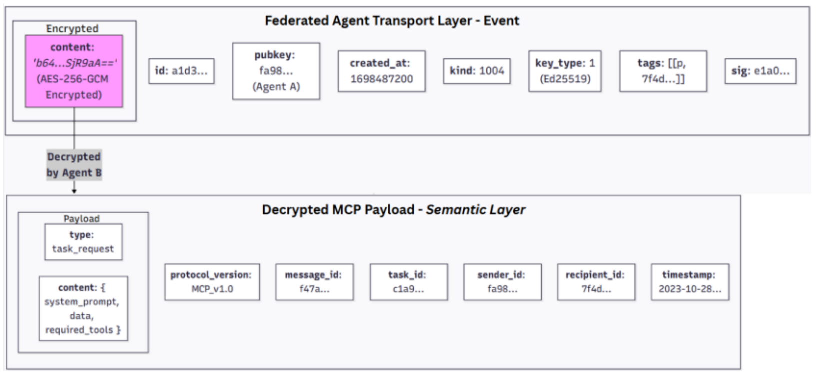


Fig. 1. The layered structure of a FAST message. The outer *Event* provides a secure transport envelope with public metadata, while the inner MCP Payload contains the end-to-end encrypted semantic instructions for the recipient agent.

Second, the Semantic Layer defines the meaning of the data inside the envelope. The Model Context Protocol (MCP) is employed as the shared language—or lingua franca—among agents. It structures the information exchanged, enabling agents to interpret, respond, and collaborate effectively without relying on centralized orchestration. The MCP provides a structured format for defining tasks, querying for data, reporting results, and orchestrating complex interactions.

This is a key architectural paradigm shift as it allows the agent communication language (MCP) to evolve independently of the underlying transport mechanism (FAST), ensuring long-term adaptability.

3.3 Network Topology: Relays, Clients, and Agents

The FAST network consists of three distinct types of actors, each with a specific role.

- **Agents:** These are the primary actors within the ecosystem. An agent is an autonomous entity, often powered by an LLM, designed to perform specific tasks.
- **Clients:** A client is the software application that an agent uses to connect and interact with the FAST network. It is responsible for managing the agent's private keys, signing messages, encrypting and decrypting data, and communicating with relays.
- **Nodes (Relays):** Nodes form the dynamic and resilient backbone of the FAST network. They are servers that act as peers in a gossip protocol. A FAST node has two primary jobs: firstly, to *accept packets*; It receives messages from connected clients.

Second, *propagate* (Gossip Network): It actively shares the messages it receives with other peer nodes to which it is connected.

This continuous propagation ensures that a message injected into the network at any single point is rapidly disseminated throughout the entire network graph. This model is inherently decentralized and censorship-resistant, as there is no central server, and the failure of any single node does not impact the network's ability to transmit messages.

3.4 Agent Identity: Cryptographic Key Pairs as DID

FAST, in favor of a self-sovereign model based on asymmetric cryptography, serves as a practical implementation of the Decentralized Identifier (DID) standard. The Public Key serves as the agent's unique, permanent, and publicly visible identifier. It is used by other agents as the address for sending encrypted messages and for verifying the authenticity of messages signed by the agent. The Private Key is kept secret by the agent's client. It is used to sign all outgoing messages, creating a non-repudiable cryptographic proof that the message originated from that specific agent. The FAST architecture is designed to be **algorithm-agile**. The protocol is compatible with schemes, such as Elliptic Curve Digital Signature Algorithm and Schnorr signatures, which could enable future capabilities like signature aggregation for complex transactions.

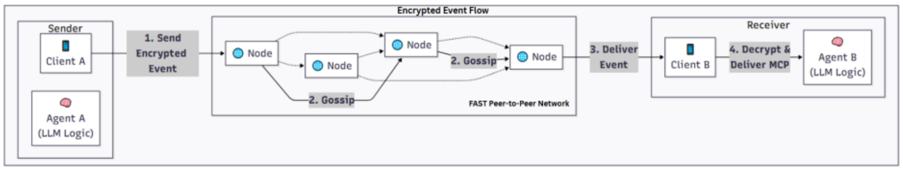


Fig. 2. The FAST message propagation flow. An encrypted Event, sent by Client A (1), is propagated through the network via a gossip protocol (2). This decentralized process ensures resilient delivery to the recipient's client (3), which then decrypts the payload and delivers the semantic MCP to the target agent for processing (4).

4 Protocol Specification and Application

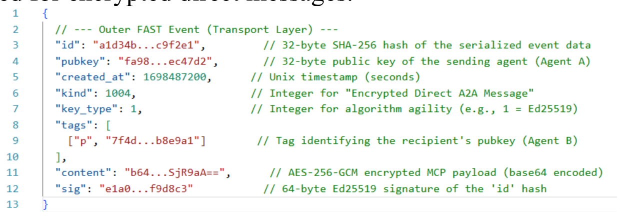
Having established the high-level architecture of the FAST, this section delves into its technical specification. We will first define the structure of a base FATL message and its cryptographic underpinnings and discuss the applications of this design for structuring agent prompts and preserving user privacy.

4.1 FAST Message Structure and Encryption

A FAST message is an event with a standardized JSON object that serves as the outer envelope for communication, to ensure broad interoperability. An `Event` object contains the following fields:

– `id`: A unique 32-byte SHA-256 hash of the serialized identifier (or) event data.
– `pubkey`: The 32-byte public key of the message author, identifying the sender.
– `created_at`: A Unix timestamp in seconds indicating when the message was created.

– `kind`: An integer specifying the type of event. For A2A communication, a specific kind is reserved for encrypted direct messages.



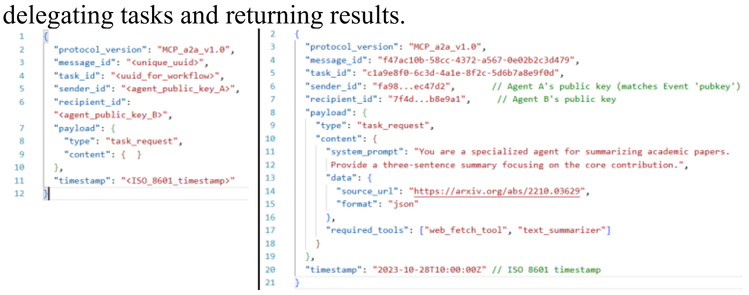
```
{
  // --- Outer FAST Event (Transport Layer) ---
  "id": "a1d34b...c9f2e1",          // 32-byte SHA-256 hash of the serialized event data
  "pubkey": "fa98...ec47d2",        // 32-byte public key of the sending agent (Agent A)
  "created_at": 1698487200,       // Unix timestamp (seconds)
  "kind": 1004,                   // Integer for "Encrypted Direct A2A Message"
  "key_type": 1,                  // Integer for algorithm agility (e.g., 1 = Ed25519)
  "tags": [
    ["p", "7f4d...b8e9a1"]        // Tag identifying the recipient's pubkey (Agent B)
  ],
  "content": "b64...SjR9aA==",      // AES-256-GCM encrypted MCP payload (base64 encoded)
  "sig": "e1a0...f9d8c3"            // 64-byte Ed25519 signature of the 'id' hash
}
```

Fig 3. Structure of a FAST Event object.

– `tags`: An array of arrays used for indexing and referencing events or public keys.
– `content`: The payload of the message. This is always an E2EE string for A2A comm.
– `sig`: A 64-byte encrypted signature (id hash), created using the author's private key.
The end-to-end encryption of the content field is mandatory for secure agent interaction. It is achieved using a standard, shared-secret approach (e.g., Elliptic Curve Diffie-Hellman). The sending client generates a shared secret using its private key and the recipient's public key. This secret is then used to encrypt the MCP payload with a symmetric cipher like AES-256-GCM. The resulting ciphertext becomes the content of the FAST Event. Only the intended recipient, holding the corresponding private key, can derive the same shared secret and decrypt the message.

4.2 The Agent-to-Agent Contract: The MCP Specification

The decrypted content of a FAST event reveals the A2A payload, which is the semantic language that enables meaningful agent orchestration. It provides a standardized structure for delegating tasks and returning results.



```
{
  "protocol_version": "MCP_a2a_v1.0",
  "message_id": "<unique_uuid>",
  "task_id": "<uuid_for_workflow>",
  "sender_id": "<agent_public_key_A>",
  "recipient_id":
  "<agent_public_key_B>",
  "payload": {
    "type": "task_request",
    "content": {  }
  },
  "timestamp": "<ISO_8601_timestamp>"
}
```

```
{
  "protocol_version": "MCP_a2a_v1.0",
  "message_id": "f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "task_id": "c1a9e8f0-6c3d-4a1e-8f2c-5d6b7a8e9f0d",
  "sender_id": "fa98...ec47d2",       // Agent A's public key (matches Event 'pubkey')
  "recipient_id": "7f4d...b8e9a1",     // Agent B's public key
  "payload": {
    "type": "task_request",
    "content": {
      "system_prompt": "You are a specialized agent for summarizing academic papers.
      Provide a three-sentence summary focusing on the core contribution.",
      "data": {
        "source_url": "https://arxiv.org/abs/2210.03629",
        "format": "json"
      },
      "required_tools": ["web_fetch_tool", "text_summarizer"]
    }
  },
  "timestamp": "2023-10-28T10:00:00Z" // ISO 8601 timestamp
}
```

Fig. 4. The structure of the A2A Protocol (MCP_a2a_v1.0) payload, which forms the semantic content inside an encrypted FAST packet. On the left, (a) illustrates the general skeleton of the protocol with placeholder fields. On the right, (b) provides a concrete, runtime example of a `task_request` from one agent to another.

– `protocol_version`: Specifies the version to ensure backward compatibility.
– `message_id`: A unique identifier for this specific message.
– `task_id`: A crucial field that groups related messages into a single conversation or workflow, allowing agents to track the state of a multi-step task.
– `sender_id` and `recipient_id`: The public keys of the communicating agents, providing a verifiable identity layer within the decrypted payload.

– `payload`: An object containing the core communication, which includes a type field (e.g., task_request, result_response, status_update) and the content of the message.
– `system_prompt`: A high-level natural language instruction that defines the persona, role, and goal for the recipient agent's LLM.
– `data`: A structured data object containing the specific information the agent needs to reason about. This could be user details or parameters for a tool. Providing structured data minimizes ambiguity and reduces the likelihood of LLM hallucination.
– `required_tools`: A list of functions or APIs that the requesting agent expects the recipient to have access to in order to complete the task.
By separating behavioral instructions (`system_prompt`) from factual information (`data`) and functional requirements (`required_tools`), the MCP allows a sending agent to precisely define the execution context for the receiving agent. This structured approach makes agent interactions more deterministic, testable, and reliable than if they were based on parsing unstructured natural language alone.

4.3 A Privacy-Preserving Approach to Data Relay

The two-layer design of FAST provides robust privacy guarantees by design. When an agent sends a message, the entire MCP payload, including any sensitive user data or proprietary prompts, is encrypted on the client side before it is wrapped in a FAST Event. This Event is then broadcast to the decentralized network of relays. The relays only have access to the public metadata of the Event object. The relay cannot access: the semantic meaning or intent of the interaction, specific system_prompt being used, any sensitive data contained within the MCP payload, and the ultimate recipient of the message. This architecture ensures that user and agent data remains confidential in transit, protected from snooping by the network operators themselves.

4.4 Anatomy of Multi-Agent Interaction:

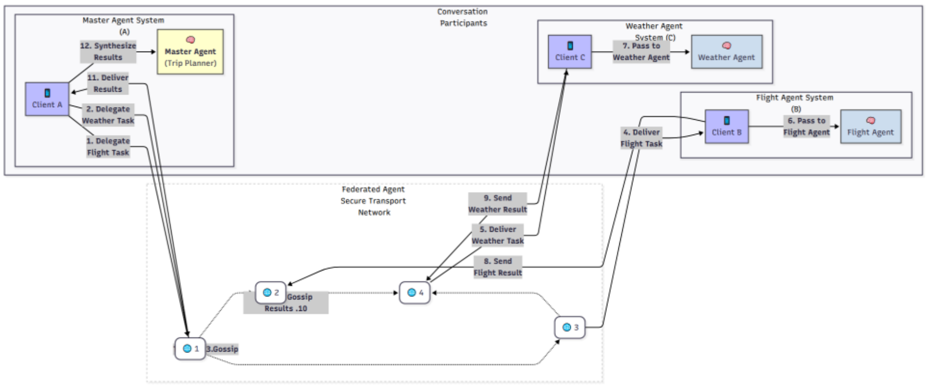


Fig. 5. This diagram shows a complete interaction between three autonomous agents. The first four steps represent Agent A's initial request, while the last four steps represent Agent B's reply.

4.4.1 The Delegation Phase (Steps 1-7)

In this phase, a Master Agent breaks down a complex goal into sub-tasks and delegates them to specialist agents. The process begins when the Master Agent (A) initiates a workflow (Steps 1 & 2). Its client creates two separate, independently encrypted FAST Events: a task_request for the Flight Agent (B) and another for the Weather Agent (C). Crucially, both requests share the same task_id to link them to the overall workflow, but they have unique message_ids. Once these events are injected into the network, the FAST nodes propagate them independently via the gossip protocol (Step 3). The network, being stateless, has no knowledge that the two messages are related. It then delivers each encrypted event to its intended recipient (Steps 4 & 5). This delivery can happen at different times and via different nodes, demonstrating the network's decentralized nature. Finally, upon receipt, each client decrypts its respective message and passes the instructions to its agent (Steps 6 & 7). Client B forwards the flight task to the Flight Agent, and Client C forwards the weather task to the Weather Agent.

4.4.2 The Response and Synthesis Phase (Steps 8-12)

In this phase, the specialist agents complete their tasks and return the results, which are then appended to the existing context of the conversation, which is further synthesized by the Master Agent. After completing their tasks, (8 & 9 Send Results) Client B and Client C each construct a new encrypted FAST Event containing a *result_response* payload. These payloads are encrypted for the Master Agent (A) and contain the original *task_id* that initiated the workflow and send results back into the network. The FAST nodes propagate (10. Gossip) these two new response events across the network, again treating them as independent messages. The Master Agent's Client (A) receives both response events from the network (11. Deliver). In step 12, client A synthesize both responses. By reading the identical task_id in each message, it knows that both the flight data and the weather data belong to the "Trip Planner" workflow. The Master Agent can synthesize these results, combining them into a coherent plan for the source.

4.4.3 How the MCP Holds the Conversation Materials

This architecture demonstrates how a stateless network can facilitate complex, stateful multi-agent orchestration. The "memory" and context of the conversation are maintained entirely within the A2A payload, managed by the participating clients. The *task_id* field is the essential thread that stitches the entire, distributed workflow together. It allows the Master Agent to send out multiple, parallel requests to different agents and correctly associate all the incoming, asynchronous responses with the original goal. The message_id, in contrast, remains unique for every single event, allowing for the tracking of individual packets within the larger task. By using the task_id as a shared context identifier, client applications can manage complex orchestration patterns, effectively re-assembling the results from multiple specialist agents. The entire conversational history and context are held by the participants, not the network, which is a core tenet of this decentralized and composable design.

5 Security Analysis

The protocol's foundational security is a direct result of its architectural design, where each component is critical for the system's integrity. Removing any of these pillars, as demonstrated by a qualitative ablation study, would introduce catastrophic vulnerabilities. For instance, ablating *End-to-End Encryption* would fundamentally compromise all privacy by exposing agent communication to the untrusted network nodes. Replacing the *decentralized gossip network* with a central server would reintroduce a single point of failure and a point of censorship, negating the protocol's core resilience. On the semantic layer, the *structured MCP* is essential for reliable orchestration; removing it in favor of unstructured language would lead to ambiguous and untrustworthy interactions. Finally, *removing cryptographic signatures* would eliminate all guarantees of authenticity, enable widespread impersonation and rendering the network insecure. We acknowledge that challenges remain at the semantic, network, and ecosystem levels. The most significant semantic threat is *prompt injection*; once a payload is decrypted, a malicious agent could craft its content to manipulate the behavior of the recipient's LLM. At the network level, the permissionless nature of FAST makes it susceptible to *decentralized attacks* like spam and Sybil attacks, where attackers attempt to degrade the network or overwhelm higher-level systems with fake identities. Addressing these broader ecosystem challenges will require future work, most critically the development of a *decentralized reputation* system built on top of FAST. Such a system is essential for enabling agents to build trust and for fostering a healthy, functional open market.

6 Efficiency and Architectural Comparison

To compare the architectures, we perform a holistic analysis of their scalability, resilience, and efficiency of development and integration.

Table 1: Comparative Analysis of Multi-Agent Communication Architectures

Metric /Attribute	FAST Protocol (Decentralized)	Centralized Orchestrator (Monolithic)
Scalability (Throughput)	**High**. Scales horizontally as more nodes join the network. Load is distributed, preventing a single bottleneck.	**Low**. Limited by the processing and bandwidth capacity of the single central server.
Resilience (Uptime)	**Very High**. No single point of failure. The network remains operational even if a significant portion of nodes fails.	**Low**. The central server is a single point of failure. If it goes down, the entire system is offline.
Latency (Single Message)	**Variable / Higher**. Message must propagate through an indeterminate number of gossips hops to reach its destination.	**Low / Predictable**. direct, two-hop round trip from client to server to client.
Interoperability & Composability	**High**. Any agent that speaks the protocol can join and interact permissionlessly.	**Very Low / None**. Integrating a new agent requires custom code and API changes.
Security Model	**Trustless by Design**. End-to-end encryption is mandatory. Network infrastructure cannot access message content.	**Requires Trust**. central server operator has full access to all unencrypted agent communication and data.
Development & Integration Cost	**Low (Ecosystem)**. Once an agent is FAST-compliant, it can interact with the entire network.	**High (Per Integration)**. Every new agent requires a bespoke, costly integration project with central system.

The FAST protocol is architecturally optimized for these large-scale properties, providing a resilient and composable foundation essential for an open market of AI agents.

7 Implications and Future Research

The greatest implication of FAST is the establishment of a universal standard for agent communication. As demonstrated in the multi-agent orchestration pattern (Fig. 5), this allows developers to treat agents as modular services, composing workflows from the best-in-class specialists from providers, fostering a competitive, innovative environment. FAST's gossip protocol provides horizontal scalability and inherent resilience. Unlike centralized systems with a single point of failure and a performance bottleneck, the FAST network's capacity and robustness grow as more nodes join. This provides a level of reliability suitable for critical, large-scale applications. FAST is architected for a zero-trust environment where participants do not need to trust the network infrastructure. Mandatory end-to-end encryption and cryptographic signatures ensure that the network's only role is to transport opaque, authenticated envelopes. This creates a permissionless and censorship-resistant environment where any agent is free to participate without fear of being de-platformed.

7.1 The Path Forward: An Open Market and Future Research

FAST provides the foundational transaction layer for viable open market of AI services. To fully realize this vision, several critical challenges must be addressed in future.
Dynamic Agent Discovery: The current FAST specification defines communication but not discovery; an agent cannot delegate a task to a "Flight Expert" without first knowing it. A critical area for future work is the design of a decentralized agent discovery protocol that would allow advertise, creating a dynamic "yellow pages" for network.
Advanced Orchestration Protocols: While the task_id enables basic state management, advanced orchestration—involving complex conditional logic, long-running workflows, and multi-party agreements—will require higher-level protocols to be built on top of the foundational communication primitives provided by FAST.

8 Conclusion

This paper introduces a significant architectural shift for multi-agent systems, moving away from centralized, monolithic designs towards a federated model for communication. The Federated Agent Secure Transport (FAST), a novel, decentralized architecture that combines a secure transport protocol with a structured semantic layer, the MCP_a2a_v1.0, to enable sophisticated agent orchestration. We have demonstrated how this architecture provides robust solutions for interoperability, scalability, and security, moving beyond closed simulations. The broader vision for FAST extends beyond a mere technical specification. It is a blueprint for the essential communication backbone of a future where AI is not a collection of isolated, proprietary tools but a federated, interconnected network. By providing a censorship-resistant fabric for interaction, FAST is designed to foster an open market of AI services, encouraging innovation and democratizing access to autonomous technology. This represents a crucial step on the journey towards true, emergent collective intelligence.

References

1. Giret, A., Botti, V.: Prototyping Adaptive and Robust Multi-Agent Systems. In: Corchado, J.M., et al. (eds.) Trends in Practical Applications of Agents and Multiagent Systems. AISC, vol. 54, pp. 11-20. Springer, Berlin, Heidelberg (2009)
2. Montenegro, G., Pagnia, H.: Security in Multi-Agent Systems. In: Buttyan, L., et al. (eds.) Security and Privacy in Ad-hoc and Sensor Networks. CMS 2007. LNCS, vol. 4582, pp. 248-261. Springer, Berlin, Heidelberg (2007)
3. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017), PMLR 54:1273-1282 (2017)
4. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artificial Intelligence Review, vol. 24, no. 1, pp. 33-60 (2005)
5. Corradi, A., Montanari, R., Stefanelli, C.: Security management in open multi-agent systems. IEEE Communications Magazine, vol. 40, no. 6, pp. 82-88 (2002)
6. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Surveys (CSUR), vol. 36, no. 4, pp. 335-371 (2004)
7. Poslad, S., Buckle, P., Hadingham, R.: The FIPA-OS agent platform: open source for open standards. In: Proceedings of the International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000), pp. 355-368 (2000)
8. 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)
9. Labrou, Y., Finin, T., Peng, Y.: Standardizing Agent Communication Languages: The Current Landscape. IEEE Intelligent Systems, vol. 14, no. 2, pp. 45-52 (1999)
10. Ramaswamy, L., Iyengar, A., Liu, L., Douceur, J.R.: Peer-to-peer systems. In: Communications of the ACM, vol. 48, no. 10, pp. 114-115. ACM (2005)
11. Nostr Protocol: Nostr Implementation Possibilities. https://github.com/nostr-protocol/nips, last accessed October 2023
12. Tarr, D., et al.: Secure Scuttlebutt: A Decentralized Secure Gossip Protocol. Whitepaper (2019). https://ssbc.github.io/scuttlebutt-protocol-guide/
13. W3C: Decentralized Identifiers (DIDs) v1.0. W3C Recommendation (2022). https://www.w3.org/TR/did-core/
14. Zimmermann, P.: The Official PGP User's Guide. MIT Press (1995)
15. Bellifemine, F., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. In: Ricci, A., et al. (eds.) Agent-Oriented Software Engineering, pp. 89–127. John Wiley & Sons, Chichester (2007)
16. Huhns, M.N.: Agents as services. IEEE Internet Computing, vol. 5, no. 4, pp. 93–95 (2001)
17. Armbrust, M., Fox, A., Griffith, R., et al.: A view of cloud computing. Communications of the ACM, vol. 53, no. 4, pp. 50–58 (2010)
18. Bellifemine, F., Caire, G., Poggi, A.: JADE: a white paper. EXP in search of innovation, vol. 3, no. 3, pp. 6–19 (2003)
19. Dragoni, N., et al.: Microservices: yesterday, today, and tomorrow. In: Present and Ulterior Software Engineering, pp. 195–216 (2017)
20. Dähling,S,et al..: Enabling scalable and fault-tolerant multi-agent systems by utilizing cloud-native computing. Autonomous Agents and Multi-Agent Systems, vol. 35, article 10 (2021)
21. Kanj, H., et al.: A novel dynamic approach for risk analysis and simulation using multi-agents model. Applied Sciences, vol. 12, no. 10, article 5062 (2022)