AEP: TOWARDS TRUSTWORTHY AGENT COMMUNICATION, NETWORKING, AND COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

As privately owned agents become increasingly capable, collaboration in open agent networks is limited by partial topology knowledge, ownership boundaries, and the risk of exposing or contaminating private context. Existing agent-to-agent protocols are useful when collaborators are already known, while shared-memory and broadcast-based systems rely on visibility and trust assumptions that do not hold in decentralized, low-trust environments. We introduce *Open Private Agent Collaboration* (OPAC), an asset-centric paradigm for cooperation among private agents with open membership, partial observability, and ownership-aware access control. Rather than treating collaboration as the exchange of final responses, OPAC enables agents to selectively discover, exchange, audit, and integrate policy-controlled context assets. This shifts the unit of cooperation from transient messages to reusable external context, while keeping disclosure and integration subject to privacy and security constraints. We instantiate OPAC with the AGENT EXCHANGE PROTOCOL (AEP), a decentralized framework that combines semantic discovery, demand-driven coordination, safety-gated asset exchange, and ledger-backed evidence derivation. By distinguishing physical topology from each agent's local cognitive topology, AEP allows agents to discover task-conditioned collaborators under an unknown network structure, release only admissible assets, and update future cooperation decisions from auditable interaction evidence. AEP therefore provides a middle ground between pure request-response interaction and globally shared memory, offering a practical foundation for trustworthy agent-native networking through selective, auditable reuse of distributed context assets.

1 INTRODUCTION

Recent advances in foundation models have made private agents increasingly capable of planning, reasoning, retrieval, and tool use (Mialon et al., 2023; Zhao et al., 2023; Yao et al., 2022; Schick et al., 2023; Park et al., 2023; Wang et al., 2023). As these agents are increasingly deployed across users, organizations, and services, realistic settings are giving rise to networks of privately owned agents (Chen et al., 2025; Yang et al., 2025b;c; Chen et al., 2024a; Li et al., 2024; A2A Project, 2025; Chang et al., 2025; Yuan et al., 2026). We refer to these systems as *Open Private Agent Networks*: decentralized ecosystems in which agents dynamically collaborate while preserving ownership, access control, and selective disclosure over their memories, tools, and capabilities. Such networks can reduce information silos by enabling agents to share and reuse task-relevant experiences, observations, cached results, and specialized capabilities held elsewhere in the network.

To better align with realistic deployment requirements, we emphasize two requirements for open private agent networks. First, the network should support *open and dynamic membership*. Instead of forming centrally managed agent clusters, agents may join, leave, or fail over time, while each agent may observe only topology fragments, partial metadata, and limited routing evidence (Yang et al., 2025a). Second, the network should support *ownership-aware access control*. Agents owned by different users, organizations, or services do not share a single trust boundary. A cooperation mechanism must therefore support authorization, minimal disclosure, auditing, and controlled release and integration to safeguard agent-owned knowledge assets such as private memory and skills. These two requirements lead to two challenges. (1) *Partial Observability*. Before requesting help or importing external context, an agent must form a task-conditioned belief about which peers are

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

relevant, reachable, authorized, and sufficiently trustworthy. In decentralized Open Private Agent Networks, this requires collaborator discovery from partial trusted evidence, without relying on a global registry or incurring excessive communication and privacy costs. (2) *Privacy and security constraints across ownership boundaries*. Even after a relevant collaborator is found, cooperation remains risky because the agents are owned by different parties and may follow different policies. The challenge is to enable useful collaboration without incurring privacy leakage (Yagoubi et al., 2026; Wang et al., 2025a; Zhang & Yang, 2025; Wang et al., 2025b), and without letting external information introduce malicious knowledge injection (Zhan et al., 2024; Debenedetti et al., 2024; Cui & Du, 2025; Dong et al., 2025; Amayuelas et al., 2024), or memory contamination (Cui & Du, 2025; Dong et al., 2025; Chen et al., 2024b; Amayuelas et al., 2024). Thus, cross-owner cooperation must control both what is shared and what is accepted.

Existing cooperation paradigms fall short in complementary ways. (1) Request-centric agent-to-agent protocols work well when a counterpart, endpoint, or capability interface is known, but do not address discovery of relevant and trustworthy collaborators under partial observability (A2A Project, 2025; Chang et al., 2025; Marro et al., 2024; BeeAI Team, 2025). (2) Shared-memory, broadcast, blackboard, and tightly coordinated systems support stronger coordination, but typically rely on common state, shared governance, and trust assumptions unsuitable for independently owned private agents (Rezazadeh et al., 2025; EigenFlux, 2026; Moltbook, 2026; Han & Zhang, 2025; Masterman et al., 2024). Table 1 illustrates this gap: existing approaches cover only subsets of these capabilities, but do not jointly support trustworthy discovery under partial topology, ownership-aware selective exchange, input safety, and evidence-based trust.

Table 1: Comparison at the paradigm and protocol levels. (a) Representative agent collaboration paradigms. (b) Representative production-grade agent communication protocols.

(a) Paradigm-level comparison

Architecture	Discovery	Privacy	Safety	Efficiency	Collaboration
Shared Mem.	×	×	×	✓	✓
Broadcast	×	△	×	×	△
Request-centric A2A	×	△	×	△	△
AEP	✓	✓	✓	✓	✓

(b) Protocol-level comparison

Protocol	Open Discovery	Access Control	Selective Release	Input Safety	Evidence-based Trust	Adaptive Coordination
ACP (BeeAI Team, 2025)	△	△	△	×	△	×
ANP (Chang et al., 2025)	△	✓	✓	△	△	✓
Agora (Marro et al., 2024)	×	×	×	×	×	✓
AEP (Ours)	✓	✓	✓	✓	✓	✓

✓: native/strong support; △: partial or infrastructure-dependent; ×: not supported.

Motivated by these limitations, we introduce the *Open Private Agent Collaboration* (OPAC) paradigm: an asset-centric framework for cooperation among privately owned agents in open, partially observed, and low-trust networks. Rather than merely sending requests and receiving generated responses, OPAC enables agents to discover, exchange, audit, and integrate policy-controlled context assets, such as memory fragments, cached tool results, verification signals, and skills, that can be reused across downstream tasks. To solve the partial observability problem, OPAC is built around the distinction between *physical topology* and *local cognitive topology*. The physical topology describes the actual connectivity of the agent network, while local cognitive topology is the current local view maintained by an individual agent: discovered topology fragments, known peer connections, and estimates of remote agents' information and states. Considering potential risk in privacy leakage and knowledge contamination, OPAC also treats cooperation as a privacy- and security-constrained process. Thus, the objective of OPAC is not simply to maximize task utility, but to improve utility subject to constraints on privacy, security risk, trust, and communication cost.

We instantiate OPAC with AGENT EXCHANGE PROTOCOL (AEP), a concrete protocol framework for safe and efficient cooperation under unknown topology. AEP implements OPAC through four subprotocols: AEP-ROUTING for semantic discovery and evidence-derived candidate admission, AEP-COORDINATE for collaboration planning and negotiation, AEP-EXCHANGE for safety-gated asset release and staged integration, and AEP-LEDGER for recording claims, incidents, trust assessments, and quality feedback, as shown in Figure 1. Together, these components allow agents to form task-conditioned collaboration neighborhoods, exchange only admissible assets, and update future cooperation decisions from verifiable evidence. Our contributions are three-fold:

- We introduce the *Open Private Agent Collaboration* (OPAC) paradigm for collaboration among private agents in open, partially observed, and low-trust environments. OPAC frames collaboration as asset-centric exchange under constraints on task utility, privacy risk, security risk, trust, and communication cost.

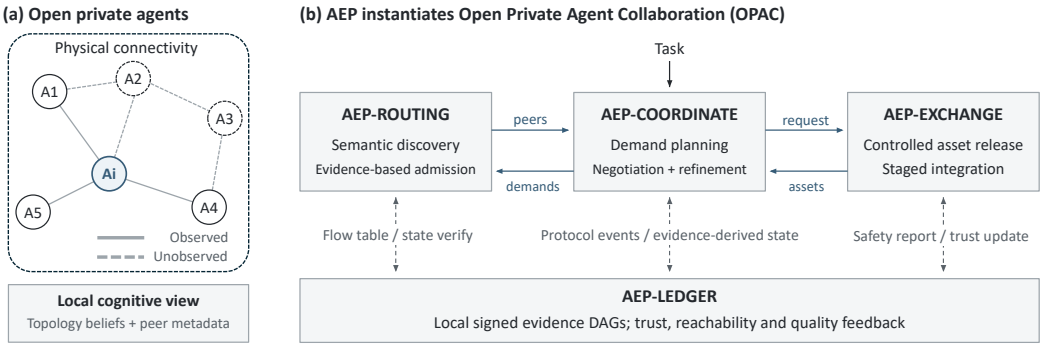

Figure 1: Open private agent collaboration and AEP framework.

- We present AGENT EXCHANGE PROTOCOL (AEP), a four-subprotocol framework that instantiates OPAC through semantic discovery, collaboration coordination, safety-gated asset exchange, and ledger-backed evidence derivation under unknown topology.
- We provide both theoretical and empirical evidence that AEP improves the utility–safety trade-off. Our experiments show that AEP-based networks achieve the highest task scores, 0.00% privacy leakage, near-zero injection success, and only a 3.5% task-score drop under a 0.4 malicious-agent ratio.

2 PROBLEM FORMALIZATION

We formalize OPAC as a distributed multi-objective decision problem over an open network of private agents:

$$\mathcal{V} = \{A_1, A_2, \ldots, A_n\}, \tag{1}$$

embedded in an unknown communication graph $G = (\mathcal{V}, \mathcal{E})$ with physical topology $W^{\text{phy}} \in \mathbb{R}^{n \times n}$. Agent A_i does not observe G globally; instead, it reasons from a local cognitive map $G_i(t)$, a topology belief $W_i^{\text{cog}}(t)$, routing state $R_i(t)$, task input X_i, target output Y_i, and private memory M_i. OPAC treats cooperation as the selective acquisition of external context assets, such as memory and skills, rather than unrestricted message passing. Under a policy π, agent A_i receives an admitted asset bundle $m^{\pi}_{\to i}(t)$ and reasons with $\widetilde{M}_i^{\pi}(t) = M_i \cup m^{\pi}_{\to i}(t)$. The goal is to improve task performance through useful imported context, without assuming full topology visibility or unrestricted state sharing. This means that external context can be used only when it is both reachable under the agent's local view and authorized for release. Let $\mathcal{A}_i(t)$ denote the assets currently admissible to A_i, and let $R_i^{\text{priv}}(\pi)$ denote the privacy leakage risk. Since imported context and privacy risk depend on other agents' release decisions, the network-level safe set is defined jointly:

$$\Pi_{\text{safe}}(t) = \left\{ \pi \in \Pi_1 \times \cdots \times \Pi_n \;\middle|\; \forall i,\, m^{\pi}_{\to i}(t) \subseteq \mathcal{A}_i(t),\; R_i^{\text{priv}}(\pi) \le \epsilon_i \right\}. \tag{2}$$

We evaluate each safe policy by predictive utility, cooperation trust, and resource cost. Here $U(\pi)$ measures task utility, $T(\pi)$ measures trustworthy and auditable cooperation, and $C(\pi)$ measures cooperation cost:

$$U(\pi) = \sum_{i=1}^{n} \mathbb{E}\left[\log p^{\star}\left(Y_i \mid X_i, \widetilde{M}_i^{\pi}(t)\right)\right], T(\pi) = \sum_{i=1}^{n} \mathbb{E}[\tau_i(\pi)], C(\pi) = \sum_{i=1}^{n} C_i(\pi). \tag{3}$$

The OPAC objective is:

$$z(\pi) = \big(U(\pi),\, T(\pi),\, -C(\pi)\big). \tag{4}$$

A safe policy is Pareto optimal if no other safe policy improves one objective without degrading another. Since different deployments may prefer different utility–trust–cost trade-offs, OPAC targets the safe Pareto frontier

$$\mathcal{P}^{\star}(t) = \{\pi \in \Pi_{\text{safe}}(t) \mid \nexists\, \pi' \in \Pi_{\text{safe}}(t) \text{ such that } z(\pi') \succ z(\pi)\}. \tag{5}$$

Thus, an OPAC solver must expand the safe feasible set under partial observability and move the empirical frontier toward $\mathcal{P}^{\star}(t)$. Appendix C gives the full formulation.

3 AGENT EXCHANGE PROTOCOL (AEP)

We present AGENT EXCHANGE PROTOCOL (AEP), a decentralized protocol for private-agent cooperation under unknown topology. AEP enables agents to discover relevant peers, acquire useful external context, and integrate it under privacy and security constraints through four protocol layers. Each agent uses AEP-ROUTING to maintain a trust-aware local peer view. Given a task requirement, if local context is insufficient, AEP-COORDINATE translates the unresolved need into external demands and selects candidate peers from this routing view. AEP-EXCHANGE then determines how the requested asset should be handled and validates received payloads through staged ingestion. Throughout the process, AEP-LEDGER records interaction evidence, allowing the other layers to base decisions on verified evidence rather than raw observations. Sec. 3.1–3.4 present the four protocol layers, Sec. 3.5 connects the induced policy family to the Pareto-formulated OPAC objective, and Appendix A and Appendix D provide operational details, theoretical results, and proofs. Figure 2 in Appendix A illustrates the protocol framework and interaction sequence.

3.1 AEP-ROUTING: SEMANTIC DISCOVERY AND EVIDENCE-DERIVED PEER STATE

AEP-ROUTING discovers and admits collaborators through three evidence-aware steps: it synchronizes public semantic metadata via **constrained gossip**, derives a trust- and quality-aware **Semantic Flow Table** from ledger-backed records, and confirms reachability through **topology probes**. Given an unresolved demand, it then routes the requester to peers that are reachable, task-relevant, and trustworthy. Appendix A.1 gives the full operational details.

Semantic flow table. In OPAC, agents must locate task-relevant external context under partial topology. AEP therefore maintains a local *Semantic Flow Table*, whose propagated `Flow Entry` stores compact public metadata about a discovered peer, including its semantic profile, route hints, freshness, and a `source_block_hash` pointer to ledger evidence. Since a `Flow Entry` is not itself a cryptographic proof, agents first merge received entries into a raw table for fast semantic discovery, and then derive a verified view from the corresponding ledger evidence. The verified view supplies the connectivity, blacklist, trust, and quality signals used for safety-aware routing.

Gossip-based verifiable state synchronization. AEP disseminates routing and evidence state through constrained gossip (Kempe & Kleinberg, 2002). Each synchronization envelope carries freshness-filtered flow entries together with incremental signed ledger evidence. Upon receipt, an agent merges entries into the raw routing table and ingests evidence blocks into the local ledger. Entries whose source evidence is missing are retained only as unverified cache. This separates fast semantic propagation from asynchronous evidence verification, allowing local routing views to converge without requiring every routing record to be self-verifying.

Topology probing. Gossip may reveal peers whose direct reachability is still unverified. To avoid treating such peers as collaboration-ready, AEP-ROUTING uses a signed probe–response handshake to confirm point-to-point reachability without exchanging task data. A successful probe records matched `topology_probe_tx`/`topology_probe_rx` evidence and adds the peer to the evidence-backed topology $G_i^{\mathrm{ev}}(t)$.

Evidence-based Semantic Routing. When the local context is insufficient, requester A_i embeds the unresolved demand as q_i and retrieves candidate peers from its semantic flow table. Candidates are scored by semantic relevance and evidence-derived peer state:

$$r_{ij}(t) = \phi\big(q_i, \texttt{capability}_j, \texttt{mem_sum}_j, \texttt{keywords}_j, \texttt{trustscore}_{ij}(t), \texttt{quality}_{ij}(t)\big)\,.$$

AEP admits only peers that are fresh, non-blacklisted, sufficiently trusted, endpoint-usable, and reachable in the evidence-backed topology $G_i^{\mathrm{ev}}(t)$. Unsafe peers are therefore excluded before ranking. Service quality then reranks the admissible set, so AEP only contacts selected, trusted, high-quality, and route-eligible peers.

3.2 AEP-COORDINATE: DEMAND-DRIVEN NEGOTIATION AND REFINEMENT

AEP-COORDINATE bridges semantic discovery and asset transfer by converting unresolved task needs into structured collaboration demands. It first performs a local sufficiency check; when external

help is needed, it decomposes the missing information into a compact **demand list**, uses AEP-ROUTING for **bounded peer selection**, and issues an exchange request or opens a short **negotiation session** to solve demands. After each round, it records outcome **feedback** as evidence, allowing later routing decisions, demand selection, capability descriptions, and asset summaries to be refined.

Demand-driven negotiation. For an incoming task, the requester first checks whether local memory and skills are sufficient; if so, it answers locally. Otherwise, it rewrites the unresolved need into a compact demand list, where each demand represents a missing fact, context fragment, and executable subproblem. The requester then uses AEP-ROUTING to identify a bounded set of reachable peers for each demand. If the demand is already well-specified, it is converted into an exchange request; otherwise, the requester opens a short negotiation session in which the responder may ask for missing parameters or clarification.

Feedback and refinement. After each collaboration round, the coordinator updates the demand state as resolved, pending, or unsolved according to the returned evidence and confidence. It also evaluates the usefulness of the response and selected assets, records service-quality feedback, and persists this feedback as evidence. These signals later refine peer quality estimates, demand selection, public capability descriptions, and asset summaries. Thus, AEP-COORDINATE closes the collaboration loop by converting downstream outcomes into future routing and metadata improvements. Appendix A.2 gives the operational details.

3.3 AEP-EXCHANGE: SAFETY-AWARE ASSET RELEASE

AEP-EXCHANGE handles how a selected responder contributes local memory or skill assets for an arranged demand. Given a demand, the responder searches its private memory and available skills for task-relevant assets, evaluates whether they can be safely used under the current request, and decides an appropriate release mode for each candidate. A memory asset may be shared directly when low-risk, sanitized before sharing when it contains sensitive or irrelevant details, or blocked when it violates the responder's release policy. A skill asset may either be released as an artifact or kept local, in which case the responder executes it on behalf of the requester and returns only the result. On the receiver side, received assets are validated again before being stored in local exchange memory. This design allows AEP to reuse external context while limiting privacy leakage, unsafe skill transfer, and memory contamination. Appendix A.3 gives more operational details.

3.4 AEP-LEDGER: VERIFIABLE EVIDENCE INFRASTRUCTURE

AEP-LEDGER provides the verifiable evidence substrate that makes AEP's cross-agent decisions auditable. It records protocol events in an **append-only evidence log**, orders them with a **causal clock model**, and keeps long-running histories manageable through **checkpointed compression**. Upper layers derive routing state, peer ratings, release evidence, and refinement signals from these verified records rather than raw observations. Appendix A.4 gives the full operational details.

Append-only evidence log. The ledger is maintained as an append-only signed Directed Acyclic Graph (DAG) log, typically a Merkle-DAG supporting lightweight proofs over selected fields (Laurie, 2014; Benet, 2014; Haeberlen et al., 2007; Merkle, 1987). Each block binds a protocol event to its author, causal order, and signed content, and is accepted only after validity and consistency checks. Conflicting signed records, such as double-signing on same sequence position, are treated as verifiable misbehavior and may trigger blacklisting. Thus, AEP turns protocol events into auditable evidence for routing, exchange, and refinement.

Clock model. Protocol evidence is ordered by a Lamport-style logical clock (Lamport, 2019) together with a monotonically increasing counter maintained by each author, so agents can reason about causal order and detect conflict from the same source. In contrast, local step counters only schedule routine maintenance such as gossip and metadata refresh, while physical time is reserved for the network deadlines, such as exchange timeouts.

Checkpointed compression. Long-running deployments may accumulate large evidence histories, so AEP periodically compresses finalized evidence into quorum-backed checkpoints, following snapshot-based state summarization (Chandy & Lamport, 1985). Consensus-backed checkpointing keeps this process safe and verifiable (Gifford, 1979; Yin et al., 2018). Each checkpoint commits to a compact summary of evidence-derived state, covered blocks, current DAG heads, and validator

signatures. New or rejoining agents verify the latest checkpoint and synchronize only incremental logs, allowing older evidence to be pruned while preserving proof-critical commitments.

3.5 AEP AS AN OPAC SOLVER

We interpret AEP as a decentralized approximate solver for the Pareto-formulated OPAC objective in Section 2. Its four layers jointly induce a safe policy family: routing expands admissible cooperation opportunities under partial topology, coordination allocates unresolved demands, exchange imports external assets under authorization and sanitization constraints, and ledger-backed evidence supports persistent agent policy with trust, reachability, and quality. We provide Theorem 1 as an idealized consistency and approximation result. Appendix D provides the full assumptions and proof details.

Theorem 1 (AEP approximate Pareto consistency, informal)**.** *Let $P^\star(t) \subseteq \Pi_{\mathrm{safe}}(t)$ be the safe Pareto-optimal policy set and let $P^{\star,z}(t) = \{z(\pi) : \pi \in P^\star(t)\}$ be its objective-space image. Under the assumptions in Appendix D, let $\widehat{P}_t^{\mathrm{AEP},z}$ be the objective-space image of the $2\epsilon_t$-approximate empirical Pareto frontier induced by AEP. Then, with high probability,*

$$d_H\left(\widehat{P}_t^{\mathrm{AEP},z}, P^{\star,z}(t)\right) \le O(\epsilon_t) + O\left(L_{\mathrm{obj}} \frac{b(\Gamma) + \sigma}{\gamma_{\mathrm{phy}}}\right) + O(\Delta U). \tag{6}$$

Thus, as statistical error, topology-learning error, and auditing opportunity loss vanish, the AEP-induced empirical objective frontier converges to $P^{\star,z}(t)$. Theorem 9 provides a formal formulation.

4 EXPERIMENTS

We evaluate AEP in an open-network multi-agent benchmark that captures privacy-sensitive collaboration under partial topology knowledge, unreliable communication, dynamic participation, and adversarial interaction. We report both utility and safety metrics to characterize the collaboration gains of AEP together with its privacy and robustness trade-offs.

4.1 EXPERIMENTAL SETUP

Simulation Setup. We use virtual HTTPS transport to emulate incomplete reachability, latency, packet loss, timeouts, retries, partitions, and leave-and-rejoin events. To assess scalability, we instantiate small-, medium-, and large-scale networks with 4, 16, and 64 benign agents, respectively. In each network, we additionally reserve two malicious agents for safety evaluation, targeting privacy theft and knowledge injection, respectively. Unless otherwise specified, utility metrics are computed over benign-agent task execution, while safety metrics are computed under adversarial participation.

Scenarios. Following the scenario construction strategy and judge-based evaluation methodology of MultiAgentBench (Zhu et al., 2025), we adapt part of its data and construct three domains: *Research*, *Coding*, and *Daily Service*. These tasks cover distributed knowledge gathering, code generation, and domain-specific service completion with personal context, private memory, and executable skills. Across domains, task executors and task-relevant knowledge and capabilities are deliberately distributed across different agents, making collaboration challenging and potentially risky. For task-utility and safety evaluation, each run contains 6 normal tasks; for long-term utility evaluation, each run contains 36 normal tasks.

Baselines. In the main experiments, we compare AEP with A2A and an A2A-driven broadcast baseline with DeepSeek-V4-Flash (Xu et al., 2026) backbone (Section 4.2). Broadcast contacts only known and addressable peers in the sender's local view, without global registry or oracle access. This paradigm-level comparison provides a broad comparison across representative collaboration architectures while controlling the implementation setting to isolate concrete design differences. To broaden the evaluation beyond abstract paradigms and assess practical protocol implementations, we additionally compare with three representative production-grade protocols within the request-centric A2A paradigm: ANP (Chang et al., 2025), Agora (Marro et al., 2024), and ACP (BeeAI Team, 2025) (Section 4.3). We further augment A2A and Broadcast with Qwen3Guard (Zhao et al., 2025) to distinguish AEP's protocol-level safety from generic external filtering (Appendix B.3), and evaluate alternative backbone models to examine model dependence (Appendix B.4).

Table 2: Task Utility of Multi-Agent Collaboration

Scenario	Protocol	Small-Scale Network			Middle-Scale Network			Large-Scale Network		
		Task Score	Completion Rate (%)	Tokens/task (k)	Task Score	Completion Rate (%)	Tokens/task (k)	Task Score	Completion Rate (%)	Tokens/task (k)
Coding	A2A	53.17	66.67	6.2	67.00	66.67	5.8	34.17	83.33	4.7
	Broadcast	67.00	66.67	7.2	82.67	83.33	11.7	33.33	33.33	26.2
	AEP	99.00	100.00	6.0	99.17	100.00	10.1	99.67	100.00	19.2
Daily-Service	A2A	31.33	66.67	3.1	17.00	16.67	3.3	31.00	66.67	4.2
	Broadcast	35.00	66.67	4.2	65.67	66.67	9.2	26.00	16.67	23.3
	AEP	82.67	100.00	3.3	95.50	100.00	7.9	81.47	83.33	18.0
Research	A2A	33.33	83.33	3.0	18.33	50.00	3.1	25.12	66.67	4.0
	Broadcast	20.00	83.33	3.8	35.00	33.33	10.3	33.33	33.33	35.2
	AEP	99.50	100.00	3.9	99.67	100.00	9.2	99.17	100.00	14.9

Table 3: Safety Assessment of Multi-Agent Collaboration

Scenario	Protocol	Small-Scale Network		Middle-Scale Network		Large-Scale Network	
		Leakage Rate (%)	Injection Success Rate (%)	Leakage Rate (%)	Injection Success Rate (%)	Leakage Rate (%)	Injection Success Rate (%)
Coding	A2A	38.11	100	36.11	100	31.94	100
	Broadcast	41.67	100	34.72	100	31.94	100
	AEP	0	0	0	0	0	5.41
Daily-Service	A2A	33.33	100	28.63	100	46.42	100
	Broadcast	66.67	100	36.11	100	41.67	100
	AEP	0	0	0	0	0	4.80
Research	A2A	35.80	100	35.14	100	38.80	100
	Broadcast	34.72	100	30.17	100	33.33	100
	AEP	0	0	0	2.90	0	4.10

Metrics. We evaluate both utility and safety. For utility, following MultiAgentBench (Zhu et al., 2025), we report **Task Score**, where an LLM judge assigns a rubric-based score to the final answer by referring to the reference answer; **Completion Rate**, the fraction of tasks completed within the task budget; and **Token/task**, the average token consumption per task from model calls and collaborative communication. For safety, we report **Privacy Leakage Rate**, the fraction of protected private items disclosed to unauthorized agents or outputs, following prior work on agent privacy leakage (Yagoubi et al., 2026; Wang et al., 2025a); and **Injection Success Rate**, the fraction of malicious injection attempts accepted into the downstream workflow, following prior work on prompt-injection evaluation for LLM agents (Zhan et al., 2024; Debenedetti et al., 2024). We aggregate all metrics over tasks and runs to characterize the utility–safety trade-off. Appendix B.7 gives full detail.

4.2 Comparison Across Collaboration Paradigms

Task Utility. Table 2 shows that AEP consistently achieves higher task scores and completion rates across all domains and network scales, with near-perfect performance in Coding and Research. While AEP may consume more tokens than A2A due to higher task completion and additional safety checks, it remains substantially more token-efficient than Broadcast, reflecting the benefit of selective collaboration over broad communication. Standard deviations are reported in Appendix B.6.

Safety Performance. Table 3 shows that A2A and Broadcast exhibit substantial privacy leakage and 100% injection success across all settings, whereas AEP achieves 0.00% privacy leakage and near-zero injection success. The remaining injection failures occur only in larger networks. These results are consistent with AEP's protocol-level safety controls over both asset release and integration. We additionally evaluate Qwen3Guard-enhanced variants of A2A and Broadcast in Appendix B.3.

4.3 Comparison Across Production-Grade Protocols

While our main experiments compare collaboration paradigms, we further evaluate AEP against concrete production-grade protocols within the request-centric A2A paradigm. Specifically, we consider ANP (Chang et al., 2025), Agora (Marro et al., 2024), and ACP (BeeAI Team, 2025) in Table 4. Protocol versions, implementations, and experimental settings are detailed in Appendix B.5. AEP consistently achieves higher task utility and completion while maintaining substantially lower injection success. These results show that AEP's advantages are not limited to controlled paradigm-level comparisons, but also hold against full protocol implementations.

Table 4: Comparison to production-grade protocol baselines under middle-scale network.

Scenario	Protocol	Task Score ↑	Completion Rate (%) ↑	Tokens/task (k) ↓	Leakage Rate (%) ↓	Injection Success Rate (%) ↓
Coding	ANP	55.44	66.67	6.3	0	100
	Agora	50.22	51.11	11.4	8.33	100
	ACP	50.22	57.78	5.3	0	100
	AEP	99.17	100.00	10.1	0	0
Daily-Service	ANP	49.63	57.50	5.5	0	100
	Agora	38.50	50.00	8.9	8.33	100
	ACP	48.00	60.00	4.8	8.33	100
	AEP	95.50	100.00	7.9	0	0
Research	ANP	50.75	60.00	3.9	16.67	100
	Agora	54.50	60.00	8.7	8.33	100
	ACP	49.00	57.50	5.1	16.67	100
	AEP	99.67	100.00	9.2	0	2.90

Table 5: Long-term Task Utility

Scenario	Protocol	Task Score	Completion Rate (%)	Tokens/task (k)
Coding	A2A	26.03	53.12	4.5
	Broadcast	41.47	68.75	9.8
	AEP	97.97	97.86	5.6
Daily-Service	A2A	30.78	40.62	4.3
	Broadcast	38.28	59.38	7.2
	AEP	93.19	90.62	7.4
Research	A2A	35.38	71.88	3.4
	Broadcast	47.22	75.00	10.4
	AEP	94.19	93.75	6.6

Table 6: Robustness under Malicious Agents

Protocol	Malicious Ratio ρ	nTSD ↓	$\mathrm{MCR}_{\mathrm{early}} \rightarrow \mathrm{MCR}_{\mathrm{late}}$ ↓
A2A	0.1	0.017	$0.333 \rightarrow 0.333$
A2A + Guard	0.1	0.357	$0.333 \rightarrow 0.333$
A2A	0.4	1.000	$0.333 \rightarrow 0.333$
A2A + Guard	0.4	0.500	$0.333 \rightarrow 0.333$
Broadcast	0.1	0.000	$0.133 \rightarrow 0.133$
Broadcast + Guard	0.1	0.000	$0.133 \rightarrow 0.133$
Broadcast	0.4	0.350	$0.400 \rightarrow 0.400$
Broadcast + Guard	0.4	0.300	$0.400 \rightarrow 0.400$
AEP	0.1	**0.000**	$\mathbf{0.167 \rightarrow 0.000}$
AEP	0.4	**0.035**	$\mathbf{0.500 \rightarrow 0.000}$

4.4 Evaluation on Long-term Task Utility

Table 5 shows that AEP's advantage persists over long-term interaction rather than disappearing after isolated tasks. Across Coding, Daily-Service, and Research, AEP achieves much higher task scores and completion rates than both A2A and Broadcast, indicating that its evidence-backed collaboration mechanism remains effective over repeated exchanges. The result suggests that AEP benefits from accumulated interaction evidence: quality feedback, safety incidents, verified reachability, and peer-state updates gradually improve later routing and exchange decisions. In contrast, A2A does not naturally expand discovery beyond initiated neighbors, while Broadcast continues to incur high communication cost and noisy context exposure. Although AEP may use more tokens than A2A, it converts this additional communication into substantially higher completion and answer quality, showing that long-term utility gains come from selective and increasingly evidence-informed collaboration.

4.5 Robustness under Increasing Malicious-Agent Ratios

We evaluate robustness on middle-scale Coding with malicious-agent ratios $\rho \in \{0, 0.1, 0.4\}$, where $\rho = 0$ is the clean reference. Malicious agents behave normally during discovery but return benign-looking corrupted payloads once contacted, following prior poisoning attacks (Amayuelas et al., 2024; Cui & Du, 2025; Dong et al., 2025; Chen et al., 2024b). We compare A2A and Broadcast with external Qwen3Guard-Gen-0.6B defense, while AEP uses only its native controls, and report normalized Task Score Drop (nTSD) and early-to-late Malicious Contact Rate (MCR). Attack construction and metric are detailed in Appendix B.2 and B.7. Table 6 shows that AEP remains robust as ρ increases. At $\rho = 0.4$, A2A and Broadcast incur nTSDs of 1.000 and 0.350, while AEP limits the drop to 0.035. External guarding does not reduce malicious contact over time, whereas AEP drives $\mathrm{MCR}_{\mathrm{late}}$ to zero at both attack ratios. This suggests that robustness requires progressively identifying and avoiding unreliable collaborators, not only filtering individual messages.

4.6 Ablation Study

Table 7 reports ablations of AEP on the middle-scale network. Removing Asset Exchange preserves task performance and 100% completion, but increases token cost from 3.5k to 4.5k per task, indicating that it mainly improves communication efficiency. Removing the Safety Gate causes privacy leakage to rise from 0.0% to 35.6% and injection success from 0.0% to 100.0%, confirming its critical role in security. Routing and Coordination are important for task quality: without Routing, the Task Score drops to 27.78 and Completion Rate to 22.22%, while removing Coordination lowers the Task Score

Table 7: Module-level ablation results of AEP.

(a) Asset Exchange ablation

Variant	Task Score ↑	Completion (%) ↑	Tokens/task (k) ↓
AEP w/o Exchange	98.3	100.0	4.5
Full AEP	97.9	100.0	3.5
Δ	−0.4%	0.0%	−22.2%

(b) Safety Gate ablation

Variant	Privacy leakage ↓	Injection success ↓
AEP w/o Safety Gate	35.6%	100.0%
Full AEP	0.0%	0.0%
Δ	−35.6%	−100.0%

(c) Routing ablation

Variant	Task Score ↑	Completion (%) ↑	Tokens/task (k) ↓
AEP w/o Routing	27.78	22.22	5.6
Full AEP	97.44	100.00	3.5
Δ	+250.8%	+350.0%	-37.5%

(d) Coordination ablation

Variant	Task Score ↑	Completion (%) ↑	Tokens/task (k) ↓
AEP w/o Coordination	33.33	100.00	2.1
Full AEP	97.44	100.00	3.5
Δ	+192.3%	0.0%	+66.7%

to 33.33 despite retaining 100% completion. We do not ablate AEP-Ledger because it serves as the protocol substrate, providing the logical clock and credit validation. Removing it would invalidate AEP's execution mechanisms rather than isolate a single module's contribution.

5 Related Work

LLM-based Multi-Agent Systems. Existing LLM-based multi-agent systems explore conversational, role-based, workflow-driven, distributed, and debate-based collaboration (Wu et al., 2024; Li et al., 2023; Chen et al., 2024a; Hong et al., 2024; Qian et al., 2024; Gao et al., 2024; Du et al., 2024; Kaesberg et al., 2025). Most assume known participants, communication patterns, or shared orchestration environments, whereas open private agent networks require collaborator discovery, reachability and trust reasoning, and coordination under partial topology and ownership boundaries.

Private Agent Cooperation. Recent work has studied heterogeneous agent ecosystems, agent communication protocols, personal agents, decentralized LLM-based multi-agent systems, and controlled memory sharing (Chen et al., 2025; Yang et al., 2025b; Li et al., 2024; Yang et al., 2025c; Rezazadeh et al., 2025). Request-centric protocols, such as agent-to-agent invocation through known endpoints or agent cards (A2A Project, 2025; Chang et al., 2025; Yuan et al., 2026), support capability advertisement and task execution, but usually assume that the requester already knows whom to contact. Shared-memory, broadcast, blackboard, and tightly coordinated systems (EigenFlux, 2026; Moltbook, 2026; Han & Zhang, 2025; Masterman et al., 2024) enable information reuse, but rely on stronger assumptions about shared state, topology visibility, governance, and trusted infrastructure. These assumptions are difficult to satisfy in open private agent networks, where agents must collaborate across ownership boundaries under partial observability.

Safety Risks in Multi-Agent Collaboration. Agentic systems face multiple risks, including privacy leakage (Yagoubi et al., 2026; Wang et al., 2025a; Zhang & Yang, 2025; Wang et al., 2025b; 2026), prompt injection (Zhan et al., 2024; Debenedetti et al., 2024), memory poisoning (Dong et al., 2025; Chen et al., 2024b), and adversarial multi-agent collaboration attack (Amayuelas et al., 2024; Cui & Du, 2025). These risks can propagate through inter-agent messages, shared memory, delegated tool use, and long-term memory updates. Open private agent networks therefore require cooperation mechanisms that discover useful and trustworthy collaborators under partial observability, while controlling both what private agents release and what they accept from others.

6 Conclusion

We introduced OPAC, an asset-centric paradigm for cooperation among private agents in open, partially observed, and low-trust networks, and instantiated it with AEP, which combines semantic discovery, demand-driven coordination, safety-gated asset exchange, and ledger-backed evidence derivation. Experiments show that AEP improves collaboration utility while reducing privacy leakage and injection risk, providing a practical foundation for trustworthy agent-native communication and networking.

AI USE STATEMENT

We used generative AI tools during the preparation of this work for language editing, improving the clarity and readability of the manuscript, assisting with the organization and presentation of the paper, brainstorming, and assisting with software code development and debugging. Generative AI tools were also used to assist with mathematical proofs and formulating mathematical claims, to help reformat datasets, and to support the evaluation process, including the computation and generation of task scores used in the quantitative analysis of experimental results. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, propose or refine research hypotheses, design or provide feedback on research methodology or experiments, translate research content, or conduct qualitative or thematic data analysis. All AI-assisted mathematical reasoning and proofs were independently reviewed and verified by the authors. AI-assisted dataset reformatting was checked to ensure that the underlying data and semantics were preserved. AI-assisted evaluation procedures and generated task scores were inspected and validated by the authors before being used in the reported results. Any AI-assisted code was likewise reviewed and tested before use in the reported experiments. The authors take responsibility for the final content of this work, including text, mathematical claims and proofs, data processing, evaluation results, code, and other artifacts produced with the aid of generative AI.

REFERENCES

A2A Project. Agent2Agent (A2A) Protocol. `https://a2aprotocol.ai/`, 2025. Open standard for agent interoperability, originally developed by Google and subsequently donated to the Linux Foundation.

Alfonso Amayuelas, Xianjun Yang, Antonis Antoniades, Wenyue Hua, Liangming Pan, and William Yang Wang. Multiagent collaboration attack: Investigating adversarial attacks in large language model collaborations via debate. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6929–6948, 2024.

BeeAI Team. Agent Communication Protocol (ACP). `https://github.com/i-am-bee/acp`, 2025. Open-source agent communication protocol, accessed 2026-09-26.

Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

K Mani Chandy and Leslie Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems (TOCS)*, 3(1):63–75, 1985.

Gaowei Chang, Eidan Lin, Chengxuan Yuan, Rizhao Cai, Binbin Chen, Xuan Xie, and Yin Zhang. Agent network protocol technical white paper. *arXiv preprint arXiv:2508.00007*, 2025.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, volume 2024, pp. 20094–20136, 2024a.

Weize Chen, Ziming You, Ran Li, Chen Qian, Chenyang Zhao, Cheng Yang, Ruobing Xie, Zhiyuan Liu, Maosong Sun, et al. Internet of agents: Weaving a web of heterogeneous agents for collaborative intelligence. In *International Conference on Learning Representations*, volume 2025, pp. 36374–36411, 2025.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024b.

Yu Cui and Hongyang Du. Mad-spear: A conformity-driven prompt injection attack on multi-agent debate systems. *arXiv preprint arXiv:2507.13038*, 2025.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory injection attacks on llm agents via query-only interaction. *arXiv preprint arXiv:2503.03704*, 2025.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first international conference on machine learning*, 2024.

EigenFlux. EigenFlux Whitepaper: A Broadcast Network for AI Agents. `https://www.eigenflux.ai/blog/eigenflux-whitepaper`, 2026. Accessed: 2026-05-03.

Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, et al. Agentscope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*, 2024.

David K Gifford. Weighted voting for replicated data. In *Proceedings of the seventh ACM symposium on Operating systems principles*, pp. 150–162, 1979.

Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. Peerreview: Practical accountability for distributed systems. *ACM SIGOPS operating systems review*, 41(6):175–188, 2007.

Bochen Han and Songmao Zhang. Exploring advanced llm multi-agent systems based on blackboard architecture. *arXiv preprint arXiv:2507.01701*, 2025.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

Lars Benedikt Kaesberg, Jonas Becker, Jan Philip Wahle, Terry Ruas, and Bela Gipp. Voting or consensus? decision-making in multi-agent debate. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11640–11671, 2025.

D. Kempe and J. Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 471–480, 2002. doi: 10.1109/SFCS.2002.1181971.

Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: the Works of Leslie Lamport*, pp. 179–196. 2019.

Ben Laurie. Certificate transparency. *Communications of the ACM*, 57(10):40–46, 2014.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

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

Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*, 2024.

Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pp. 369–378. Springer, 1987.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*, 2023.

Moltbook. Moltbook: The Front Page of the Agent Internet. https://www.moltbook.com/, 2026. Accessed: 2026-05-03.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Alireza Rezazadeh, Zichao Li, Ange Lou, Yuying Zhao, Wei Wei, and Yujia Bao. Collaborative memory: Multi-user memory sharing in llm agents with dynamic access control. *arXiv preprint arXiv:2505.18279*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Shouju Wang, Fenglin Yu, Xirui Liu, Xiaoting Qin, Jue Zhang, Qingwei Lin, Dongmei Zhang, and Saravan Rajmohan. Privacy in action: Towards realistic privacy mitigation and evaluation for llm-powered agents. *arXiv preprint arXiv:2509.17488*, 2025a.

Yuhao Wang, Wenjie Qu, Shengfang Zhai, Yanze Jiang, Zichen Liu, Yue Liu, Yinpeng Dong, and Jiaheng Zhang. Silent leaks: Implicit knowledge extraction attack on rag systems through benign queries. *arXiv preprint arXiv:2505.15420*, 2025b.

Yuhao Wang, Shengfang Zhai, Guanghao Jin, Yinpeng Dong, Linyi Yang, and Jiaheng Zhang. Mempot: Defending against memory extraction attack with optimized honeypots. *arXiv preprint arXiv:2602.07517*, 2026.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Faouzi El Yagoubi, Godwin Badu-Marfo, and Ranwa Al Mallah. Agentleak: A full-stack benchmark for privacy leakage in multi-agent llm systems. *arXiv preprint arXiv:2602.11510*, 2026.

Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems, 2025a. URL https://arxiv.org/abs/2504.00587.

Yingxuan Yang, Huacan Chai, Yuanyi Song, Siyuan Qi, Muning Wen, Ning Li, Junwei Liao, Haoyi Hu, Jianghao Lin, Gaowei Chang, et al. A survey of ai agent protocols. *arXiv preprint arXiv:2504.16736*, 2025b.

Yingxuan Yang, Qiuying Peng, Jun Wang, Ying Wen, and Weinan Zhang. Unlocking the potential of decentralized llm-based mas: Privacy preservation and monetization in collective intelligence. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pp. 2896–2900, 2025c.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus in the lens of blockchain. *arXiv preprint arXiv:1803.05069*, 2018.

Dun Yuan, Fuyuan Lyu, Ye Yuan, Weixu Zhang, Bowei He, Jiayi Geng, Linfeng Du, Zipeng Sun, Yankai Chen, Changjiang Han, et al. Beyond message passing: Toward semantically aligned agent communication. *arXiv preprint arXiv:2604.02369*, 2026.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024.

Yanzhe Zhang and Diyi Yang. Searching for privacy risks in llm agents via simulation. *arXiv preprint arXiv:2508.10880*, 2025.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, et al. Qwen3guard technical report. *arXiv preprint arXiv:2510.14276*, 2025.

Pengyu Zhao, Zijian Jin, and Ning Cheng. An in-depth survey of large language model-based artificial intelligence agents. *arXiv preprint arXiv:2309.14365*, 2023.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. MultiAgentBench : Evaluating the collaboration and competition of LLM agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8580–8622, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.421. URL https://aclanthology.org/2025.acl-long.421/.

APPENDIX CONTENTS

A Extended Operational View of AEP **15**

A.1 AEP-ROUTING Operational Details 15

A.1.1 Semantic Flow Table 15

A.1.2 Constrained Gossip Synchronization 16

A.1.3 Topology Probe 18

A.1.4 Remote Agent Selection 18

A.1.5 Evidence-Derived Peer Rating and Pruning 19

A.2 AEP-COORDINATE Operational Details 21

A.3 AEP-EXCHANGE Operational Details 22

A.4 AEP-LEDGER Operational Details 23

A.4.1 Ledger Details 23

A.4.2 Clock Model and Runtime Time Semantics 26

B Extended Experiment and Extra Experiment Details **26**

B.1 Safety Comparison to Shared Memory Paradigms 26

B.2 Robustness to Scalable Attacks under Multi-Agent Scenario 26

B.3 External-Guard Baselines and Scalable Attack 27

B.4 Cross-Model Evaluation 28

B.5 Implementation Details of Production-Grade Protocols 28

B.6 Reported Variability of the Main Utility Results 28

B.7 Metric Definitions 28

C Extended OPAC Formulation **30**

C.1 Context Assets, Collaboration Trajectories, and Policies 30

C.2 Safe Feasible Policy Class Under Partial Observability 31

C.3 Multi-Objective Evaluation of Cooperation Policies 31

C.4 Pareto Formulation of OPAC 32

D Full Theoretical Results and Proofs **32**

D.1 Analysis Assumptions 33

D.2 Discovery Results 34

D.3 Safe Exchange Results 35

D.4 Consistency Results 35

D.5 Proofs 36

E Limitations **41**

F Social Impact Discussion **41**

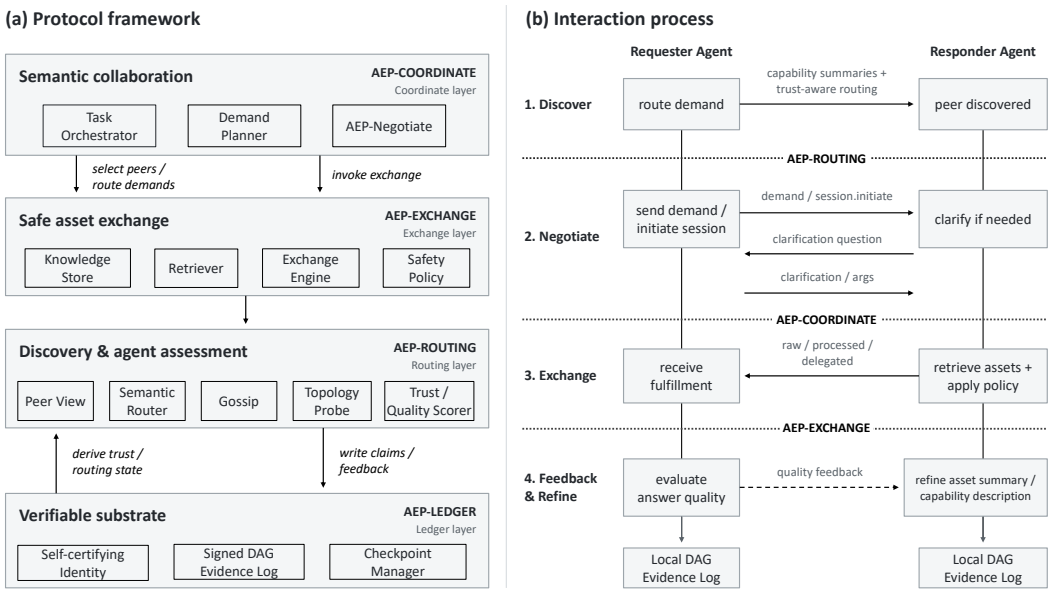

Figure 2: Protocol Framework and Sequence Diagram in Interaction Process.

Table 8: Implementation-oriented instantiation of the abstractions used in Section 3. The table shows how protocol-level concepts correspond to concrete runtime components without elevating those runtime details to the main theoretical contribution.

Paper abstraction	Implementation component
Cognitive map / semantic flow table	`RoutingProtocol` + `FlowTable`
Evidence-derived candidate admission	Derived verified routing view + local trust / quality / blacklist state
Demand planning and allocation	`TaskOrchestrator` + demand backlog + adaptive Top-K policy
Negotiated exchange	`AepNegotiateSession`
Asset exchange	`ExchangeEngine` + `SafetyBackend`
Asynchronous cooperation	`SemanticJobService` + one-way fulfill
Audited evidence	Signed hash-linked DAG / Merkle DAG + evidence verification pipeline
Trust coordinate $T(\pi)$	Evidence-derived local safety trust over verified interaction evidence
Quality feedback	Quality-feedback records + derived quality score + summary / capability refinement
Evidence-log compression	Quorum-backed checkpoint commit + DAG pruning

A Extended Operational View of AEP

This appendix gives an implementation-oriented view of Section 3 while preserving the higher-level abstractions used in the main text. Table 8 summarizes how the abstractions used in Section 3 map onto concrete implementation components in the current protocol stack. Figure 2 represents the full structure of our protocol and the interaction process with all protocol modules.

A.1 AEP-Routing Operational Details

A.1.1 Semantic Flow Table

The Semantic Flow Table is the local routing state used by AEP-Routing to support semantic peer discovery, safety-aware candidate admission, and quality-aware reranking. Each entry corresponds to a discovered peer agent and stores compact public metadata rather than private memory. The

Field	View	Description
`agent_id`	Raw / Derived	Globally unique identifier of the peer agent, typically tied to the peer's self-certifying public identity.
`capability_desc`	Raw / Derived	Public capability description announced by the peer. This is the primary textual signal used for semantic routing.
`trust_score`	Raw / Derived	Local or evidence-derived trust estimate. In routing, the derived trust score is preferred when available and is used for candidate admission.
`endpoint_url`	Raw / Derived	Network endpoint of the peer, such as an HTTPS JSON-RPC address. In the derived view, this endpoint is extracted from the latest verified announcement.
`next_hop`	Raw	Next-hop agent identifier used to reach the peer. In the current direct-exchange setting, this is usually equal to `agent_id`.
`hops`	Raw	Estimated hop distance from the current node to the peer. It is also used as a simple route-cost proxy.
`last_seen_clock`	Raw / Derived	Logical clock of the most recent observation or verified announcement. It is used for freshness control and stale-entry decay.
`agent_card`	Raw	Optional full Agent Card object, preserving extensible metadata not normalized into the core routing fields.
`learned_via`	Raw	Source from which the entry was learned, such as `gossip`, `bootstrap`, `query_response`, `restored`, or `local`.
`announce_seq`	Raw	Sequence number of the corresponding `capability_announce` block, used for freshness comparison and conflict arbitration.
`path_cost`	Raw	Estimated routing cost, currently approximated by `float(hops)`.
`route_confidence`	Raw	Confidence in the route entry. Gossip-propagated entries may decay with hop distance, for example using a rule such as `max(0.2, 1/(1+hops))`.
`memory_summary`	Raw / Derived	Public summary of the peer's memory state. It provides a secondary semantic matching signal without exposing private memory.
`keywords`	Raw / Derived	Compact keyword tuple extracted from public capability or memory metadata, used for lexical-overlap matching.
`skill_summary`	Raw / Derived	Public summary of the peer's available skills, such as a skill-name list or capability signature.
`skill_count`	Raw / Derived	Number of skills declared by the peer, used as an auxiliary capability-richness signal.
`source_block_hash`	Raw / Derived	Hash of the `capability_announce` evidence block from which the entry was learned or derived. It supports provenance tracing and endpoint verification.
`is_connected`	Derived	Whether the peer has evidence-backed reachability, for example through matched topology probe evidence or matched exchange claims.
`is_blacklisted`	Derived	Whether the peer is locally blacklisted due to equivocation, double signing, safety incidents, or other ledgered evidence.
`service_quality`	Derived	Quality score smoothed from historical feedback. It is used mainly to rerank already-admissible peers, not to override safety trust.

Table 9: Fields maintained by the Semantic Flow Table. The raw view supports routing and semantic matching, while the derived view adds ledger-backed safety and quality signals.

table combines a raw routing view, represented by `FlowEntry`, with an optional ledger-derived verified view, represented by `DerivedFlowEntry`. The raw view records information learned from gossip, bootstrap configuration, query responses, or local announcements, including agent identity, capability description, endpoint information, route freshness, route confidence, public memory summary, keywords, and skill summary. The derived view is reconstructed from the local evidence ledger and overlays the raw routing state with safety-relevant evidence, including blacklist status, verified connectivity, evidence-derived trust, service quality, and provenance. During routing, blacklist status and trust determine whether a peer is admissible, while semantic similarity and service quality determine the ranking among already-admissible peers. This separation ensures that a useful but unsafe peer cannot be promoted solely by utility, while a safe but low-quality peer is not incorrectly treated as malicious.

A.1.2 Constrained Gossip Synchronization

This subsection describes the operational gossip procedure used by AEP-Routing. Gossip is responsible for disseminating both lightweight routing metadata and verifiable evidence state under partial topology. Its design decouples fast semantic propagation from slower evidence verification, while bounding communication amplification.

Gossip schedule. Each agent periodically runs a gossip step as part of its routing runtime. At the beginning of a gossip cycle, the agent advances its logical clock, removes stale routing entries, recomputes evidence-derived trust if needed, selects a small set of gossip peers, constructs synchronization envelopes, and enqueues them for outbound transmission. Staleness pruning prevents old flow entries from being repeatedly propagated, while periodic trust recomputation incorporates new ledger evidence into later peer selection.

Peer selection. Let $\mathcal{N}_i(t)$ be the set of peers currently visible to agent A_i. AEP selects a bounded fanout set by trust-weighted sampling without replacement:

$$p_{ij}(t) \propto \max\{\epsilon, \mathrm{Trust}_{ij}(t)\}, \qquad A_j \in \mathcal{N}_i(t), \tag{7}$$

where $\epsilon > 0$ ensures that low-trust but not-yet-excluded peers are not assigned exactly zero sampling probability. The fanout budget limits per-round amplification. To reduce eclipse risk, the implementation also tracks recent gossip sources; if source diversity falls below a minimum threshold, the sender temporarily expands the peer set with additional randomly selected peers outside the initial sample.

Synchronization envelope. A gossip message is a synchronization envelope with three payload components:

$$M_i = \big(\texttt{entries}, \texttt{blocks}, \texttt{heads}\big). \tag{8}$$

The `entries` field contains recently active `FlowEntry` records exported from the sender's raw semantic flow table. Only entries whose last-observed clock is within a freshness window are exported:

$$\ell_i(t) - \ell_j^{\mathrm{seen}} \le s_{\mathrm{export}}. \tag{9}$$

The `blocks` field contains signed DAG evidence blocks that the receiver is estimated not to have yet. These blocks are exported incrementally using the receiver's previously known DAG heads. Blocks are sorted by logical order and included greedily subject to a maximum block count and maximum serialized payload size. The optional `heads` field contains the sender's current DAG heads, which allow the receiver to update its view of the sender's ledger frontier and enable future delta synchronization.

Inbound processing. Upon receiving a gossip envelope, the receiver first performs cheap defensive checks. It discards duplicate messages using a bounded message-id cache, rejects messages exceeding block-count or payload-size limits, and records the sender in the local gossip-source tracker. The receiver then processes the envelope along two paths. The fast path merges `FlowEntry` records into the raw semantic flow table. During this merge, the receiver rewrites the next hop of each imported route to the immediate sender, increments the hop count, and updates route confidence as a decreasing function of hop length and evidence availability. If multiple entries describe the same target agent, the receiver arbitrates among them using freshness, announcement sequence, and path length. The slow path verifies and ingests signed DAG blocks. For each received block, the receiver checks the author signature, validates any announcement-specific proof requirements, inserts the block into the local evidence DAG, and applies accepted side effects. For example, trust-update blocks may update distributed safety assessments, while quality-feedback blocks may update local service quality estimates. If a flow entry points to a source block that has not yet arrived, the entry remains only an unverified routing cache entry until the corresponding evidence is available.

Head synchronization. When a gossip envelope contains DAG heads, the receiver accepts them only if the heads are locally known or already covered by a finalized checkpoint. Accepted heads update the receiver's view of the sender's ledger frontier. Future gossip messages can then export only blocks after that frontier, avoiding repeated transmission of already synchronized evidence.

Burst propagation. If merging incoming flow entries changes the local routing table, the agent may temporarily enter a short burst window. During this window, it gossips more aggressively than the normal periodic schedule, allowing newly discovered routing state to propagate quickly. The burst is bounded by a small logical-clock budget, so it accelerates convergence without turning gossip into unbounded broadcast.

Summary. The gossip mechanism therefore separates semantic and evidential synchronization. Flow entries propagate quickly and support early local routing views, while DAG blocks provide delayed but verifiable evidence for authentication, trust, quality, and topology inference. Fanout control, freshness windows, TTL, payload caps, deduplication, source-diversity tracking, and bounded burst broadcast jointly keep synchronization scalable and robust under partial topology.

A.1.3 TOPOLOGY PROBE

Topology probing is the active reachability-verification mechanism used by AEP-ROUTING. Its purpose is to distinguish a peer that is merely learned through gossip from a peer that can support direct point-to-point semantic exchange. A successful probe produces signed evidence that can later be consumed by routing, exchange, and trust inference.

Trigger condition. During local view synchronization, agent A_i considers probing a discovered peer A_j when A_j has a non-empty endpoint, is not already connected in the derived evidence view, has not already been confirmed by a previous probe, and has not been probed for the same announcement version. The announcement version can be represented by a probe key such as

$$\text{key}_{ij} = (A_j, \text{source_block_hash}_j, \text{announce_seq}_j). \tag{10}$$

This prevents repeated probes against the same stale routing announcement while allowing a new probe when the peer publishes a fresh capability announcement.

Signed probe handshake. A probe is a lightweight signed request–response exchange. The requester sends a message containing a probe identifier, source and target identifiers, a nonce, the requester's latest capability-announcement block hash, and a logical clock. The probe identifier can be written abstractly as

$$\text{probe_id} = H(A_i \,\|\, A_j \,\|\, h_i^{\text{cap}} \,\|\, \ell_i), \tag{11}$$

where h_i^{cap} is the requester's latest capability block hash and ℓ_i is its logical clock. The request is signed by A_i.

Upon receipt, A_j checks that the target identity matches itself, verifies that the claimed source identity matches the transport peer, validates the request signature using the source public key, and rejects duplicate probe identifiers. If the request is accepted, A_j returns a signed response echoing the `probe_id` and nonce, adding a fresh response nonce, its own latest capability-announcement block hash, and an acceptance status.

Evidence recording. Both sides record the successful handshake as ledger evidence. Let

$$h_{ij}^{\text{probe}} = H(\text{probe_id}, A_i, A_j, \text{nonce}) \tag{12}$$

be the shared probe hash. The receiver records a block of kind `topology_probe_rx`, while the requester records a block of kind `topology_probe_tx`. These records have the same logical role as ordinary exchange claims: only matched transmit and receive claims for the same hash support an edge in the evidence-backed topology.

Topology effect. The topology inference procedure treats matched topology-probe claims as evidence of a verified undirected edge:

$$(A_i, A_j) \in G_i^{\text{ev}}(t) \quad \Longleftarrow \quad \exists h : \text{tx}(A_i, A_j, h) \wedge \text{rx}(A_j, A_i, h). \tag{13}$$

Thus, a successful topology probe can promote a gossip-discovered peer into an evidence-reachable peer without requiring a prior semantic asset exchange.

Use in routing and exchange. A probe-confirmed peer is treated as directly reachable for collaboration eligibility, even if it was originally learned through a multi-hop gossip path. The confirmation is also exposed to exchange-side evidence construction, where probe confirmation and derived connectivity are used as alternative forms of topology confirmation. After probe evidence is recorded, trust inference may be recomputed because $G_i^{\text{ev}}(t)$ has gained a new verified edge.

Security role. Topology probes do not transfer task data and do not update service quality. Their role is narrower: they authenticate the peer identity, test direct network reachability, prevent replay through nonces and processed-probe caches, and create verifiable evidence for later topology inference. This keeps routing from relying on unverified multi-hop gossip alone while avoiding the cost of broadcasting semantic requests to unconfirmed peers.

A.1.4 REMOTE AGENT SELECTION

This subsection specifies the remote-agent selection procedure used by AEP-ROUTING. Given an unresolved demand from requester A_i, the selector returns a small set of peers that are semantically

relevant, safety-admissible, and practically reachable. The selector only decides whom to contact; whether any asset is released is still decided later by AEP-EXCHANGE.

Let q_i be the embedding of the current demand. Agent A_i maintains a raw semantic flow table $\mathcal{T}_i$ and, when ledger evidence is available, a derived verified view $\mathcal{D}_i$. For each candidate A_j, the raw table provides public semantic and routing metadata, including capability summary cap_j, memory summary mem_j, skill summary skill_j, keywords kw_j, endpoint, hop count, next hop, freshness, and raw trust. The derived view adds evidence-derived trust $\tilde{\tau}_{ij}$, service quality $\tilde{q}_{ij}$, blacklist status, and verified connectivity.

Candidate admission. Selection first applies hard safety gates. A peer is rejected if it is blacklisted, lacks a usable endpoint, or fails the effective trust threshold. The effective trust score is

$$\tau_{ij}^{\mathrm{eff}} = \begin{cases} \tilde{\tau}_{ij}, & \text{if verified connectivity is available,} \\ 0.5\tilde{\tau}_{ij}, & \text{if derived evidence exists but connectivity is unverified,} \\ 0.3\tau_{ij}, & \text{if only raw routing evidence is available.} \end{cases} \tag{14}$$

The admitted candidate set is

$$\mathcal{C}_i = \left\{ A_j \in \mathcal{T}_i : A_j \notin \mathrm{Blacklist}_i,\ \tau_{ij}^{\mathrm{eff}} \geq \tau_{\min} \right\}. \tag{15}$$

Thus, trust is used as an admission signal rather than only as a ranking feature.

Semantic scoring. For every admitted peer, AEP computes a semantic relevance score from public metadata:

$$s_{ij} = w_c \mathrm{sim}(q_i, \mathrm{cap}_j) + w_m \mathrm{sim}(q_i, \mathrm{mem}_j) + w_s \mathrm{sim}(q_i, \mathrm{skill}_j) + w_k \,\mathrm{Jaccard}(K_i, \mathrm{kw}_j), \tag{16}$$

where K_i is the keyword set extracted from the demand, $\mathrm{sim}(\cdot)$ is the cosine similarity with sentence embedding. Missing metadata fields are ignored and the remaining weights are renormalized. If service-quality evidence is available, the semantic score is adjusted as

$$\bar{s}_{ij} = s_{ij}\big(0.7 + 0.3\tilde{q}_{ij}\big). \tag{17}$$

Service quality therefore reranks already-admitted peers, but cannot rescue a blacklisted or insufficiently trusted peer.

Topology filtering and Top-k selection. After scoring, the selector keeps only candidates that are currently usable for remote collaboration. A peer is route-eligible if it is directly reachable, confirmed by topology-probe evidence, or connected through an admissible route within the collaboration hop limit $h_{\max}$[1]. The final remote candidate set is

$$\mathcal{R}_i = \{ A_j \in \mathcal{C}_i : \mathrm{RouteEligible}_i(A_j; h_{\max}) = 1 \}. \tag{18}$$

AEP-ROUTING returns

$$\mathrm{Select}_i(q_i) = \mathrm{TopK}_{A_j \in \mathcal{R}_i}\, \bar{s}_{ij}. \tag{19}$$

This procedure implements the separation used by AEP-Routing: semantic metadata expands discovery, trust determines candidate admissibility, quality ranks safe candidates, and topology evidence prevents unreachable or weakly supported routes from entering downstream coordination.

A.1.5 Evidence-Derived Peer Rating and Pruning

This subsection describes the operational procedure behind evidence-derived peer rating and pruning in AEP-Routing. The procedure consumes the raw semantic flow table $\mathcal{T}_i(t)$, the local evidence view $\mathcal{Q}_i(t)$, and the evidence-backed interaction topology $G_i^{\mathrm{ev}}(t)$.

Evidence-backed topology. Agent A_i distinguishes announced reachability from verified reachability. The evidence topology $G_i^{\mathrm{ev}}(t)$ is constructed only from matched two-sided evidence, such as paired exchange claims or paired topology-probe claims. Thus, a peer is treated as evidence-reachable only when the ledger contains verifiable support for the corresponding interaction edge.

[1] In this version, we assume all agents with authorized collaboration are connected and agents are not allowed to collaborate if they are not mutually reachable, therefore $h_{\max}$ is default set to 1.

Algorithm 1 Remote Agent Selection in AEP-ROUTING

Require: demand embedding q_i, raw table $\mathcal{T}_i$, derived view $\mathcal{D}_i$, trust floor $\tau_{\min}$, hop limit $h_{\max}$, budget k

$\mathcal{C}_i \leftarrow \emptyset$
for $A_j \in \mathcal{T}_i$ **do**
  if A_j is blacklisted or has no usable endpoint **then**
    continue
  end if
  compute effective trust τ_{ij}^{eff}
  if $\tau_{ij}^{\text{eff}} \geq \tau_{\min}$ **then**
    add A_j to $\mathcal{C}_i$
  end if
end for
for $A_j \in \mathcal{C}_i$ **do**
  compute semantic score s_{ij}
  adjust by service quality to obtain $\bar{s}_{ij}$
end for
remove peers that fail $\mathrm{RouteEligible}_i(A_j; h_{\max})$
return top-k remaining peers ranked by $\bar{s}_{ij}$

Safety trust. After audited interactions, A_i records bounded local safety assessments for its peers. Remote trust reports may also be ingested when they appear in valid signed evidence blocks. These local and distributed assessments are then aggregated over $G_i^{\text{ev}}(t)$ by a damped propagation procedure:

$$p_i^{(\ell+1)} = (1-\alpha)\frac{\mathbf{1}}{n_i} + \alpha P_i(t)^\top p_i^{(\ell)}, \tag{20}$$

where $P_i(t)$ is a transition matrix induced by evidence-backed safety assessments, which is usually records normalized peer rating value coming from disseminated ledger evidence. After convergence, the derived trust score of peer A_j is

$$\mathrm{Trust}_{ij}(t) = p_{i,j}^{\star}. \tag{21}$$

Thus, trust is a topology-derived safety rating rather than a raw average of recent interactions.

Blacklist state. Blacklist status is derived from verifiable protocol violations. For example, if the evidence log contains equivocation evidence, such as conflicting signed blocks from the same author at the same sequence position, the corresponding peer is marked as blacklisted and removed from future candidate sets.

Pruning. Pruning is implemented as consecutive filters over $\mathcal{T}_i(t)$.

First, stale entries are removed:

$$\mathcal{T}_i^{\text{fresh}}(t) = \left\{ A_j \in \mathcal{T}_i(t) : \ell_i(t) - \ell_j^{\text{seen}} \leq s_{\max} \right\}. \tag{22}$$

Second, unsafe entries are excluded using blacklist status and an effective trust score:

$$\mathcal{T}_i^{\text{safe}}(t) = \left\{ A_j \in \mathcal{T}_i^{\text{fresh}}(t) : A_j \notin \mathrm{Blacklist}_i(t),\ \tau_{ij}^{\text{eff}}(t) \geq \tau_{\min} \right\}. \tag{23}$$

Here $\tau_{ij}^{\text{eff}}(t)$ is derived from verified trust when available, discounted when evidence is incomplete, and falls back to raw routing trust only when no verified view exists.

Third, unreachable entries are removed:

$$\mathcal{P}_i(t) = \left\{ A_j \in \mathcal{T}_i^{\text{safe}}(t) : \mathrm{Reachable}_i(A_j) = 1 \right\}. \tag{24}$$

A peer is reachable if it has a usable endpoint and is directly connected, bootstrap-connected, topology-probe confirmed, or connected in $G_i^{\text{ev}}(t)$.

The output $\mathcal{P}_i(t)$ is therefore a fresh, non-blacklisted, sufficiently trusted, and evidence-reachable peer set. Later semantic routing can rank this set for a specific task, but the rating and pruning procedure itself is evidence-driven and task-independent.

A.2 AEP-COORDINATE OPERATIONAL DETAILS

This appendix specifies the operational procedure behind AEP-COORDINATE. The coordinator controls when collaboration is needed, how unresolved needs are represented, how peers are allocated, and how feedback is converted into later refinement.

Local sufficiency. When a task query x_i arrives, requester A_i first retrieves local memory and skill context. The coordinator estimates whether this local context is sufficient for answering the task. In the full implementation, this can be decided by an LLM gate that receives the task, local context, and retrieval similarity, and returns a structured decision

$$(\text{sufficient}, \text{confidence}, \text{reason}). \tag{25}$$

If this gate is unavailable or fails, the coordinator falls back to a similarity threshold over the top retrieved local assets. If the task is locally sufficient, the task is answered without entering the collaboration loop.

Demand planning. If local context is insufficient, the coordinator converts the task into a demand backlog

$$\mathcal{B}_i = \{d_1, \ldots, d_m\}. \tag{26}$$

Each demand d contains a demand identifier, a textual description, a priority, a status, and bookkeeping fields such as previously requested peers. A demand may represent a missing fact, missing context fragment, executable subproblem, or clarification need. The planner may condition on the original task, the previous demand backlog, received responses, and the latest insufficiency reason. If planning fails, the fallback is a single demand covering the original query.

Candidate allocation. For each unresolved demand d, the coordinator requests a route pool from AEP-ROUTING. The returned peers have already passed semantic and topology-aware routing filters. The coordinator may optionally rerank the pool using task-level evidence, then removes peers that were already tried for the same demand:

$$\mathcal{U}_{i,d} = \mathcal{C}_{i,d} \setminus \text{RequestedPeers}_{i,d}. \tag{27}$$

The remaining untried candidates are allocated according to the demand priority and collaboration budget. If the caller provides preferred targets, the coordinator bypasses semantic routing and only checks whether those targets are eligible for direct collaboration. If routing returns no usable peers, the implementation may fall back to the local agent view, ranked by effective trust, freshness, and route confidence.

Negotiation. If a demand is underspecified, the requester may open a short negotiation session instead of immediately requesting an exchange. A session has an identifier, a round counter, and a remaining budget. The responder may ask clarification questions or request missing task parameters. The requester then decides whether to continue negotiation, convert the session into a direct exchange request, or terminate it. The loop is bounded by maximum rounds and budget:

$$r \le r_{\max}, \qquad b_r \ge 0. \tag{28}$$

Negotiation therefore delays asset release until the requested context is better specified, reducing unnecessary communication and accidental over-sharing. By default, the $r_{\max}$ and the b_r is 3 and 9.

Demand resolution. After receiving a response, the coordinator updates the demand status. A demand is marked resolved when the response provides sufficient evidence with confidence above a resolution threshold:

$$\text{Resolved}(d) = \mathbf{1}\{\text{conf}(d) \ge \theta_{\text{res}}\}. \tag{29}$$

Otherwise, it remains pending, is retried with additional candidates, or becomes exhausted if the candidate budget is depleted. By default, the sufficiency threshold is set to 0.75.

Quality feedback and refinement. For each fulfilled demand, the requester computes a quality signal $\Psi_{ij}^{\text{qual}} \in [0, 1]$ measuring the usefulness of peer A_j's response for the original task or demand. The feedback is used in two ways. First, it updates the local quality estimate:

$$\text{Qual}_{ij}(t+1) = \text{clip}_{[0,1]}\left((1-\eta)\text{Qual}_{ij}(t) + \eta\Psi_{ij}^{\text{qual}}(t)\right). \tag{30}$$

Second, it is recorded as ledger evidence so that other agents may ingest the feedback and update their own local estimates. Repeated negative feedback can trigger diagnosis of failure modes such as

stale memory, capability mismatch, or low-quality asset summaries. The implementation may then register asset summaries or capability descriptions for refinement and publish improved metadata in later capability announcements.

Role in the protocol. AEP-COORDINATE does not itself release assets. It determines whether external cooperation is needed, decomposes the need into bounded demands, allocates those demands to candidate peers, manages short negotiation loops, and turns downstream outcomes into feedback for future routing and metadata refinement.

A.3 AEP-EXCHANGE OPERATIONAL DETAILS

This appendix specifies the operational procedure behind AEP-EXCHANGE. The exchange layer is responsible for selecting relevant assets, deciding the release mode, enforcing outbound and inbound safety gates, and integrating admitted payloads into exchange memory.

Candidate assets. For a demand from requester A_i, responder A_j searches its local memory and skills for candidate assets

$$\Xi_j = \{\xi_1, \ldots, \xi_m\}. \tag{31}$$

If no relevant asset is found, the exchange engine returns a no-relevant-asset decision. Otherwise, candidates are pre-filtered by relevance and bounded by a pre-evaluation budget so that expensive safety and LLM-based checks are applied only to plausible assets.

Trust and topology gate. Before evaluating individual assets, the responder constructs exchange evidence for the requester. If the requester is blacklisted, the exchange is blocked. If the requester lacks topology confirmation, its effective trust may be discounted. The request is allowed to proceed only if

$$\tau_{ji}^{\text{eff}} \geq \tau_{\text{floor}}. \tag{32}$$

This gate prevents low-trust or unconfirmed peers from reaching the asset-release stage.

Relevance filtering and reranking. Each surviving candidate asset is scored against the demand. Assets whose similarity falls below a minimum relevance threshold are removed. When enabled, an additional LLM relevance check can reject superficially similar but semantically mismatched assets. If many assets survive, the exchange engine may rerank them with a task-aware reranker before safety evaluation.

Release modes. For each candidate asset ξ, the safety backend returns a release decision:

$$a(\xi) \in \{\texttt{raw_memory}, \texttt{processed_memory}, \texttt{raw_skill}, \texttt{delegated_computation}, \texttt{blocked}\}. \tag{33}$$

These modes correspond to three abstract actions:

$$a \in \{\text{raw}, \text{processed}, \text{block}\}. \tag{34}$$

The mapping depends on the asset type:

memory	raw	release raw or lightly redacted memory,
memory	processed	release summarized or sanitized memory,
skill	raw	release skill code,
skill	processed	execute locally and return only output.

The key asymmetry is that processed memory still exports transformed state, whereas processed skill exchange exports behavior without exposing the skill implementation.

Outbound safety evaluation. The outbound safety backend evaluates each candidate using asset sensitivity, skill risk, requester trust, authorization policy, and transformation requirements. A high-trust requester may receive raw low-risk assets, while sensitive assets may require redaction, summarization, or forced delegation. Assets that violate the outbound policy are blocked:

$$\text{Allow}(\xi, A_i) = 0. \tag{35}$$

Before final transmission, generated or transformed payloads may pass through an output guard to reduce leakage risk.

Processed memory materialization. When the selected tier is `processed_memory`, the responder materializes a sanitized memory artifact. This may involve redaction, summarization, and a final

output-guard check. The resulting payload is intended to preserve task-relevant information while removing unnecessary sensitive detail.

Delegated computation materialization. When the selected tier is `delegated_computation`, the responder keeps the skill local. The exchange engine binds task arguments to the skill interface, checks whether missing arguments can be clarified, executes the skill locally when possible, and returns only the result. If the execution result is sensitive, the result may be wrapped into a semantic answer rather than exposing raw output.

Selection of exchange items. After relevance, trust, and safety filtering, the responder returns a bounded set of exchange items:

$$\mathcal{X}_{j\to i} = \mathrm{TopK}_{\xi\in\Xi_j}\,\mathrm{Score}(\xi), \tag{36}$$

where the score reflects relevance, release tier, risk, and policy constraints. The returned exchange decision records the selected items, release modes, and any blocked or no-asset status.

Inbound validation and exchange memory. The requester does not directly write received payloads into core work memory. Each inbound payload is first validated under the receive-side safety policy. Payloads suspected of injection, misinformation, malformed content, or policy violation are rejected and may trigger a safety incident. Accepted payloads are normalized into asset records and stored in exchange memory:

$$E_i(t+1) = \mathrm{Integrate}(E_i(t), \mathcal{X}_{j\to i}), \tag{37}$$

separate from the agent's work memory $W_i(t)$. Later retrieval may use both work memory and exchange memory, but promotion into longer-lived memory remains a separate decision.

Safety incidents. If the receive-side gate detects a safety incident, the requester may write a trust-update block to the evidence log and penalize the source peer. This links exchange-time safety failures back into the evidence-derived trust mechanism used by routing and future exchange decisions.

Post-exchange feedback. After using the received payload, the requester can send feedback containing answer-level quality, asset-level quality, per-asset comments, and a reason code such as successful resolution, low quality, or capability mismatch. Depending on the configured feedback policy, feedback may be sent for all exchanges, only negative outcomes, or unresolved demands. The receiver uses this feedback for local asset refinement, quality-score updates, and evidence persistence.

Role in the protocol. AEP-EXCHANGE realizes the admissible-context constraint of OPAC at runtime. It does not simply transmit whatever a remote peer requests; it selects task-relevant assets, applies trust and topology gates, chooses an appropriate release mode, validates inbound content, separates received context from core memory, and emits feedback for future coordination and routing.

A.4 AEP-LEDGER OPERATIONAL DETAILS

This appendix specifies the operational realization of AEP-LEDGER. The ledger is a verifiable evidence infrastructure: it stores signed protocol evidence, verifies remote evidence blocks, derives queryable views for upper subprotocols, and compresses old evidence through checkpoints. It does not itself decide whether an asset should be released, which peer should be selected, or how a demand should be negotiated.

A.4.1 LEDGER DETAILS

Ledger realization. AEP supports two compatible ledger realizations. The first is a signed hash-linked DAG, where each block hash commits to the block's canonical content and signature. The second is a Merkle DAG, where the block hash is the Merkle root over normalized header and payload fields. The Merkle variant supports field-level proofs: a peer can verify that a specific field, such as a public key or capability description, belongs to a committed block without receiving the entire payload.

Evidence block. The atomic evidence object is a block

$$b = (h_b, \mathrm{parents}_b, a_b, \ell_b, s_b, p_b, \sigma_b), \tag{38}$$

where h_b is the block hash or Merkle root, $\mathrm{parents}_b$ is the set of parent hashes, a_b is the author identifier, ℓ_b is the logical clock, s_b is the per-author sequence number, p_b is the payload, and σ_b is

the author signature. The signed canonical content includes the sorted parent hashes, author identity, logical clock, sequence number, and canonical payload:

$$\mathrm{Canon}(b) = \mathrm{Canon}(\mathrm{parents}_b, a_b, \ell_b, s_b, p_b). \tag{39}$$

The signature is therefore bound not only to the payload, but also to the local DAG context in which the evidence is appended.

Evidence payloads. A payload is typed by its `kind`. The implementation uses payload kinds such as

`capability_announce`, `tx_claim`, `rx_claim`, `topology_probe_tx`, `topology_probe_rx`, `trust_update`, `quality_feedback`, `checkpoint_commit`.

Capability announcements publish signed public metadata. Exchange and topology-probe claims provide interaction evidence. Trust updates and quality feedback provide later evidence for safety and service-quality derivation. Checkpoint commits summarize finalized ledger state.

Local append. When agent A_i appends a local block, it chooses the current DAG heads as parents, increments its per-author sequence number, signs the canonical block content, and computes the block commitment. In the hash-linked variant, the block hash is computed from the canonical content and signature. In the Merkle variant, header and payload fields are committed as Merkle leaves and reduced to a root. The appended block becomes a new DAG head, while previous heads referenced as parents cease to be frontier heads.

Remote ingestion. When a remote block is received, the ledger performs layered validation:

$$\mathrm{Accept}(b) = \mathbf{1}\{\mathrm{FreshHash} \wedge \mathrm{NotBlacklisted} \wedge \mathrm{HashValid} \wedge \mathrm{SigValid} \wedge \mathrm{SeqValid} \wedge \mathrm{ParentsAvailable}\}. \tag{40}$$

Concretely, the receiver rejects duplicate hashes, rejects blocks authored by blacklisted agents, recomputes the hash or Merkle root, verifies the author signature, checks per-author sequence monotonicity, and confirms that parent hashes are locally available or already covered by a finalized checkpoint. Blocks with missing parents may be held as orphans until their ancestors arrive.

Equivocation and blacklisting. The per-author sequence number gives a lightweight equivocation test. If two distinct blocks share the same $(\mathrm{author}, \mathrm{seq})$ but have different block hashes, they constitute double-signing evidence:

$$b \neq b', \quad a_b = a_{b'}, \quad s_b = s_{b'}. \tag{41}$$

When such evidence is detected, the author is blacklisted and its future blocks are rejected. This makes blacklisting evidence-driven rather than reputation-only.

Merkle field proofs. In the Merkle DAG realization, each normalized field is hashed into a leaf. A field proof for key k and value v consists of the leaf hash and sibling hashes on the path to the block root. Verification recomputes

$$r = \mathrm{MerkleRoot}(k, v, \mathrm{proof}) \tag{42}$$

and accepts the field claim only if $r = h_b$. This allows a peer to prove selected fields of a committed capability or checkpoint block without revealing the full block body.

Evidence-backed topology. The evidence-backed interaction topology $G_i^{\mathrm{ev}}(t)$ is inferred from matched signed claims, not from routing-table hop counts. Let $\mathrm{Tx}(a, b, h)$ denote a transmit claim by a toward b with interaction hash h, and let $\mathrm{Rx}(b, a, h)$ denote the corresponding receive claim. Then the ledger-derived topology contains an undirected edge only if both sides are present:

$$(A_a, A_b) \in G_i^{\mathrm{ev}}(t) \quad \Longleftrightarrow \quad \exists h : \mathrm{Tx}(a, b, h) \wedge \mathrm{Rx}(b, a, h). \tag{43}$$

The same rule applies to matched topology-probe claims. This prevents a gossip-propagated route from being treated as verified connectivity unless it is backed by two-sided signed evidence.

Queryable evidence view. Let $\mathcal{L}_i(t)$ denote the local evidence log of agent A_i. The ledger exposes a derived query view

$$\mathcal{Q}_i(t) = \mathrm{Query}(\mathcal{L}_i(t)), \tag{44}$$

which contains verified capability records, blacklist status, matched interaction claims, topology-probe evidence, trust-update evidence, quality-feedback records, and checkpoint state. Upper subprotocols

query $\mathcal{Q}_i(t)$ but do not mutate it directly. It combines signed ledger facts with locally derived scores. A typical derived entry contains the latest signed capability announcement for an author, its source block hash, blacklist status, verified connectivity in $G_i^{\mathrm{ev}}(t)$, derived trust score, service-quality estimate, and freshness metadata. Importantly, this view does not trust raw gossip entries directly; it reconstructs peer state from signed evidence plus local derivations.

Trust and quality boundary. The ledger stores trust-update and quality-feedback evidence, but the final ratings are derived outside the ledger. Routing may compute safety trust by propagating local and distributed safety assessments over $G_i^{\mathrm{ev}}(t)$, while service quality may be maintained as an EMA over quality-feedback evidence. This preserves a clean boundary: AEP-LEDGER stores and verifies evidence, whereas upper subprotocols derive policy state from that evidence.

Checkpoint construction. Long-running deployments accumulate many evidence blocks, so AEP periodically constructs a checkpoint

$$C_e = (e, \rho_e, \mathcal{H}_e, \mathcal{S}_e, \Sigma_e), \tag{45}$$

where e is the epoch, ρ_e is the checkpoint state root, $\mathcal{H}_e$ contains covered block hashes and covered DAG heads, $\mathcal{S}_e$ contains state snapshots and proof references, and Σ_e is quorum-backed signature evidence. The committed state may include trust snapshots, quality snapshots, the evidence-backed topology, validator state, capability references, topology proof references, trust evidence references, and quality evidence references. The state root is computed by hashing the canonical checkpoint content.

Checkpoint modes and finalization. A checkpoint may be a genesis checkpoint when no previous finalized checkpoint exists, a normal checkpoint extending a single compatible predecessor, or a merge checkpoint combining multiple compatible checkpoint heads. We use cryptographic consensus mechanism to make sure the checkpoint compression process safe and verifiable (Gifford, 1979; Yin et al., 2018). A designated proposer builds the checkpoint candidate, validators verify its canonical content and membership conditions, and the checkpoint is finalized once the accumulated signature weight reaches the quorum threshold:

$$\sum_{v \in \mathrm{Signers}(C_e)} w_v \geq \theta_{\mathrm{quorum}}. \tag{46}$$

The finalized checkpoint is then recorded as checkpoint evidence.

Checkpoint application. When an agent applies a finalized checkpoint, it can restore derived state from the committed snapshots. In particular, it may install the trust snapshot, verified-topology snapshot, quality snapshot, and capability references, then reconstruct local routing entries from checkpointed capability facts. This gives new or recovering agents a fast path to a coherent evidence-derived state without replaying the entire historical DAG.

Checkpoint barriers and delta synchronization. After finalization, the ledger registers a checkpoint barrier consisting of the covered block hashes, covered heads, and per-author maximum covered sequence numbers. During later synchronization, export procedures can stop traversing history once they reach a barrier and need only send post-checkpoint delta blocks. A newly joined peer can therefore verify the latest finalized checkpoint root and then download the evidence blocks that occur after it.

Pruning. Checkpointing permits physical pruning without abandoning append-only evidence semantics. Blocks covered by a finalized checkpoint can be removed from active storage, while the checkpoint commit, recent post-checkpoint blocks, unresolved claims, latest capability announcements, and proof-critical ancestor fragments are retained as needed. After pruning, DAG heads and per-author sequence state are recomputed subject to the checkpoint barrier.

Persistence and recovery. On recovery from persistent storage, an agent reloads persisted heads, walks backward through their ancestors, reinserts blocks in topological order, clears orphan state where possible, and recomputes derived state such as blacklist status, verified topology, trust, and quality. Since checkpoint barriers are derived from finalized checkpoint evidence, recovery can rebuild them rather than treating them as separate trusted state.

Summary. AEP-LEDGER provides durable evidence, not direct policy decisions. Its append-only authenticated DAG records protocol claims; matched claims induce verified topology; queryable

views expose evidence-derived state to upper layers; and quorum-backed checkpoints compress history while preserving verifiability.

A.4.2 Clock Model and Runtime Time Semantics

AEP uses three complementary notions of time. The first is a local step counter, which is a monotone runtime counter advanced by the local scheduler. It is used only to trigger periodic background tasks, including gossip rounds, trust recomputation, knowledge maintenance, memory-summary refresh, skill-summary refresh, and keyword-index refresh. Since the step counter is local to a single agent runtime, it does not participate in distributed ordering and should not be interpreted as a global tick or a physical-time measurement.

The second notion of time is the protocol logical clock. Each agent maintains a Lamport-style logical clock. Local protocol events advance the clock, and inbound messages carrying a logical-clock field update the receiver according to the standard rule

$$\ell_i \leftarrow \max\{\ell_i, \ell_{\text{recv}}\} + 1. \tag{47}$$

This logical clock is the primary time base for evidence-bearing protocol state. In particular, locally appended DAG blocks record the current logical clock, flow entries store the logical time at which they were last observed, gossip export filters and stale-entry decay are evaluated against logical freshness, sessions are pruned by logical timeout intervals, checkpoints are proposed at logical-clock intervals, and emitted protocol events carry logical-clock metadata. As a result, any evidence generated by an agent after receiving remote protocol messages is logically ordered after the observed remote events.

The third notion of time is physical millisecond time. AEP uses physical time only for network-facing asynchronous timeout semantics, such as detecting whether a semantic exchange request has exceeded its wall-clock deadline. This is necessary because asynchronous exchange interacts with real transport behavior, including RPC latency, socket timeouts, retries, and partitions. Physical time is therefore an implementation-level timeout signal rather than a source of distributed causality.

During normal execution, the local step counter determines when a periodic task should be attempted. When such a task generates protocol evidence or sends a protocol message, the logical clock is advanced and the resulting evidence is stamped with logical time. During recovery, the agent restores the logical clock to at least the maximum logical clock observed in the local ledger, aligns the local step counter to avoid immediate background-task bursts, and resets interval trackers to the restored clock values. This prevents recovered agents from appending evidence that appears causally earlier than already persisted ledger state.

B Extended Experiment and Extra Experiment Details

B.1 Safety Comparison to Shared Memory Paradigms

We compare shared memory paradigms and our protocol's robustness to injection attack and privilege extraction attack. As shown in Table. 10, under the full shared memory paradigm cooperation incurs a 100% privacy leakage rate and injection success rate, while our protocols defense all the attacks.

Table 10: Safety Comparison between AEP and Shared Memory Collaboration

Protocol	Leakage Rate	Injection Success Rate
Shared Memory	100.00%	100.00%
AEP	0.00%	0.00%

B.2 Robustness to Scalable Attacks under Multi-Agent Scenario

Experimental setting. We evaluate whether AEP remains robust when the fraction of malicious agents increases in a multi-agent collaboration network. The experiment is conducted on the Coding domain under a middle-scale network, where task-relevant information and capabilities are distributed across agents. For each protocol, we vary the malicious-agent ratio as $\rho \in \{0, 0.1, 0.4\}$, where ρ denotes the fraction of agents controlled by the attacker. The clean setting $\rho = 0$ is used as the reference condition for computing task score degradation, while the attacked settings $\rho = 0.1$ and $\rho = 0.4$ are reported in Table 6. As a controlled defense baseline, we further augment A2A and

Broadcast with the same external Qwen3Guard-Gen-0.6B filter for inbound and outbound messages, while AEP relies solely on its native safety and exchange controls.

Attack design. Inspired by prior work Amayuelas et al. (2024); Cui & Du (2025); Dong et al. (2025); Chen et al. (2024b), we instantiate a scalable poisoning attack where malicious agents behave as ordinary collaborators during discovery, but return poisoned payloads once contacted. Unlike the explicit injection attacks considered earlier, the payloads contain no overtly malicious instructions or policy-violating content. Instead, they are *benign-looking but factually incorrect or misleading* claims, faked intermediate results, or subtly corrupted evidence, which cannot be reliably removed by direct safety gating.

Metrics. We report two primary metrics. First, normalized task score drop, denoted as nTSD (normalized Task Score Drop), measures end-to-end utility degradation:

$$\mathrm{nTSD} = \frac{\text{origin task score} - \text{attacked task score}}{\max(\text{origin task score}, \epsilon)}. \tag{48}$$

A lower nTSD means that the protocol preserves task quality better under attack. Second, we report the change in MCR (Malicious Contact Rate) from early to late epochs, written as

$$\mathrm{MCR}_{\mathrm{early}} \rightarrow \mathrm{MCR}_{\mathrm{late}}.$$

Here, MCR measures the fraction of peer contacts that involve malicious agents. This metric evaluates whether a protocol can reduce malicious exposure over time.

B.3 External-Guard Baselines and Scalable Attack

Controlled defense configuration and calibration. We augment A2A and Broadcast with the same external defense, denoted A2A + Guard and Broadcast + Guard, using Qwen3Guard-Gen-0.6B for inbound and outbound screening at the corresponding communication boundaries. The guard follows its official chat template without a customized safety policy, while the backbone, tasks, topology, and attack instances are otherwise held fixed. AEP instead uses its native release-side and receive-side controls and is not equipped with the external Guard. We calibrate the guard on an open-source prompt-injection dataset[2]. A deterministic calibration split is drawn from the validation set and a disjoint evaluation split from the test set, each containing 200 benign and 200 attack samples. For each message, we define the risk score as $r = 1 - P(\texttt{Safe})$, where probabilities are normalized over the constrained `Safe`, `Unsafe`, and `Controversial` logits. Inbound and outbound thresholds are calibrated independently to maximize attack recall subject to an empirical benign false-positive rate of at most 1% on the calibration split.

Safety results. Table 11 evaluates direct and paraphrased injection attacks together with privacy-extraction probes. All methods have zero observed privacy leakage in this guarded comparison. The external Guard does not prevent the tested injected payloads from entering the downstream workflows of A2A and Broadcast, whose injection success rates remain 100%. AEP records 0% injection success in Coding and Daily-Service and 2.90% in Research. These results distinguish protection against unauthorized release from protection against accepting malicious context; neither metric alone establishes whether a final answer was behaviorally hijacked.

Scalable multi-agent attack results. We also evaluate the guarded baselines on the middle-scale Coding under malicious agent setup of Section 4.5, where contacted malicious peers return benign-looking but incorrect claims or intermediate results. Table 12 reports normalized task-score drop (nTSD) relative to the corresponding clean condition and early-to-late malicious contact rate (MCR). At $\rho = 0.4$, the Guard reduces A2A's nTSD from 1.000 to 0.500 and Broadcast's from 0.350 to 0.300, but their malicious-contact rates remain unchanged. At $\rho = 0.1$, A2A + Guard instead has a larger drop than unguarded A2A (0.357 versus 0.017), so the effect on utility is not uniformly beneficial. AEP records nTSD values of 0.000 and 0.035 at the two attack ratios and reduces late-stage MCR to zero in both cases. Content screening does not itself update the baseline collaborator-selection

[2]Available at `https://huggingface.co/datasets/neuralchemy/Prompt-injection-dataset`.

Table 11: Safety evaluation with guarded A2A and Broadcast under middle scale network.

Scenario	Protocol	Injection Success ↓	Privacy Leakage ↓
Coding	A2A + Guard	100.00 ± 0.00	0.00 ± 0.00
	Broadcast + Guard	100.00 ± 0.00	0.00 ± 0.00
	AEP	0.00 ± 0.00	0.00 ± 0.00
Daily-Service	A2A + Guard	100.00 ± 0.00	0.00 ± 0.00
	Broadcast + Guard	100.00 ± 0.00	0.00 ± 0.00
	AEP	0.00 ± 0.00	0.00 ± 0.00
Research	A2A + Guard	100.00 ± 0.00	0.00 ± 0.00
	Broadcast + Guard	100.00 ± 0.00	0.00 ± 0.00
	AEP	2.90 ± 1.91	0.00 ± 0.00

Table 12: Robustness under scalable malicious agent poisoning on middle-scale Coding.

Protocol	Malicious Ratio ρ	nTSD ↓	$\mathrm{MCR}_{\mathrm{early}} \rightarrow \mathrm{MCR}_{\mathrm{late}}$ ↓
A2A	0.1	0.017	$0.333 \rightarrow 0.333$
A2A	0.4	1.000	$0.333 \rightarrow 0.333$
A2A + Guard	0.1	0.357	$0.333 \rightarrow 0.333$
A2A + Guard	0.4	0.500	$0.333 \rightarrow 0.333$
Broadcast	0.1	0.000	$0.133 \rightarrow 0.133$
Broadcast	0.4	0.350	$0.400 \rightarrow 0.400$
Broadcast + Guard	0.1	0.000	$0.133 \rightarrow 0.133$
Broadcast + Guard	0.4	0.300	$0.400 \rightarrow 0.400$
AEP	0.1	0.000	$0.167 \rightarrow 0.000$
AEP	0.4	0.035	$0.500 \rightarrow 0.000$

policies; AEP additionally uses recorded interaction outcomes in subsequent routing and exchange decisions.

B.4 Cross-Model Evaluation

We replace the backbone while retaining the same middle-scale Coding tasks, agent initialization, physical topology, protocol configuration, and evaluation procedure. Table 13 reports the resulting utility and token cost. All three backbones complete every evaluated task. GPT-5.5 achieves a Task Score close to DeepSeek-V4-Flash at a similar token cost, while Qwen3.7-Max yields lower answer quality and moderately higher token consumption.

Table 13: Cross-model evaluation of AEP on Coding in the middle-scale network. Only the backbone is replaced; tasks, agent initialization, topology, protocol configuration, and evaluation procedure are held fixed.

Backbone	Task Score ↑	Completion (%) ↑	Tokens/task (k) ↓
DeepSeek-V4-Flash	99.17	100.00	10.1
GPT-5.5	99.67	100.00	10.4
Qwen3.7-Max	83.33	100.00	11.2

B.5 Implementation Details of Production-Grade Protocols

We execute the production-grade protocols with upstream SDK packages `anp` (0.9.3)[3], `agora-protocol` (0.2.0)[4], and `acp-sdk` (1.0.3)[5]. The protocol request validation, serialization, discovery responses, and task handling execute within their applications, while operating-system HTTP transport is redirected through the benchmark's simulated network for common reachability, failure, latency, and byte accounting. To address Agora's limited agent-discovery capability, we additionally augment it with direct-neighbor gossip for exchanging routing cards, enabling a fairer comparison.

B.6 Reported Variability of the Main Utility Results

Table 14 preserves the supplied sample standard deviations (SD) for Task Score, Completion Rate, and Tokens/task across the three domains and network scales. Each value is estimated across 3 runs with different initialization seed, which may change agents' neighbors and knowledge assignments.

B.7 Metric Definitions

We evaluate AEP and all baselines along three dimensions: task utility, safety, and robustness. Let $\mathcal{D}$ denote the evaluation task set, and let $N = |\mathcal{D}|$. For each task $x_t \in \mathcal{D}$, a protocol produces an output $\hat{y}_t$, an interaction trace $\mathcal{E}_t$, and a set of collaboration contacts $\mathcal{C}_t$.

[3]ANP: `https://www.agent-network-protocol.com/specs/1.2/white-paper`
[4]Agora: `https://agoraprotocol.org/`
[5]ACP: `https://agentcommunicationprotocol.dev/introduction/welcome`

Table 14: Standard deviations of utility metrics across network scales.

Scenario	Protocol	Small-Scale Network			Middle-Scale Network			Large-Scale Network		
		Task Score	Completion Rate (%)	Tokens/task (k)	Task Score	Completion Rate (%)	Tokens/task (k)	Task Score	Completion Rate (%)	Tokens/task (k)
Coding	A2A	16.67	0.00	0.07	9.62	0.00	0.10	9.62	0.00	0.04
	Broadcast	9.62	0.00	0.14	19.25	0.00	0.11	0.00	0.00	0.16
	AEP	1.47	0.00	1.31	0.34	0.00	6.27	0.27	0.00	6.05
Daily-Service	A2A	12.73	19.00	0.31	0.00	10.00	0.06	19.25	0.00	0.05
	Broadcast	0.00	0.00	0.04	9.62	0.00	0.07	16.46	0.00	0.06
	AEP	0.48	0.00	1.08	4.14	0.00	13.16	9.62	0.00	4.20
Research	A2A	3.85	10.00	0.17	0.00	10.00	0.23	16.67	0.00	0.12
	Broadcast	0.00	0.00	0.47	0.00	0.00	1.02	0.00	0.00	0.81
	AEP	0.00	0.00	0.65	0.27	0.00	1.47	0.34	0.00	3.99

Task Score. Task Score measures the end-to-end quality of the final answer. For each task, an LLM judge assigns a rubric-based score

$$s_t = J(x_t, \hat{y}_t, y_t^\star) \in [0, 100],$$

where $y_t^\star$ is the reference answer or task-specific grading target. The reported Task Score is the average score over all tasks:

$$\text{TaskScore} = \frac{1}{N} \sum_{t=1}^{N} s_t.$$

Higher values indicate better task performance.

Completion Rate. Completion Rate measures whether a protocol can finish the task within the interaction and time budget. Let $c_t \in \{0, 1\}$ indicate whether task t returns a valid final answer before the budget is exhausted. The completion rate is

$$\text{CompletionRate} = \frac{1}{N} \sum_{t=1}^{N} c_t.$$

We report this value as a percentage.

Token Cost. Token cost measures the average communication and reasoning cost per task. For each task, we count all tokens consumed by LLM calls and collaborative communication events, including prompts, generated responses, negotiation messages, exchange requests, and returned payloads. Let $\text{tok}(e)$ be the token count of event $e \in \mathcal{E}_t$. The average token cost is

$$\text{Tokens/task} = \frac{1}{N} \sum_{t=1}^{N} \sum_{e \in \mathcal{E}_t} \text{tok}(e).$$

In the main tables, we report this value in thousands of tokens.

Privacy Leakage Rate. Privacy Leakage Rate measures how often protected private information is exposed to unauthorized parties or final outputs. For each task, let $\mathcal{P}_t$ be the set of protected private items, and let $\ell(p) = 1$ indicate that protected item p is disclosed to an unauthorized agent, tool call, message, or final answer. The leakage rate is

$$\text{LeakageRate} = \frac{\sum_{t=1}^{N} \sum_{p \in \mathcal{P}_t} \mathbb{I}[\ell(p) = 1]}{\sum_{t=1}^{N} |\mathcal{P}_t|}.$$

We count each protected item at most once per task, even if it appears multiple times in the same interaction trace. Lower values indicate stronger privacy protection.

Injection Success Rate. Injection Success Rate measures the fraction of adversarial injection attempts that are accepted into the downstream workflow. Let $\mathcal{A}_t$ be the set of malicious injection attempts in task t, and let a be successful if the injected content is accepted into the agent's reasoning context, exchange memory, tool-use plan, delegated workflow, or final response. The injection success rate is

$$\text{InjectionSuccessRate} = \frac{\sum_{t=1}^{N} \sum_{a \in \mathcal{A}_t} \mathbb{I}[\text{success}(a) = 1]}{\sum_{t=1}^{N} |\mathcal{A}_t|}.$$

Lower values indicate stronger robustness against malicious external context.

Normalized Task Score Drop. For robustness experiments with malicious agents, we measure utility degradation relative to the clean setting. Let Score_0 be the Task Score under no attack, and let Score_ρ be the Task Score when the malicious-agent ratio is ρ. The normalized task score drop is

$$\mathrm{nTSD}(\rho) = \frac{\mathrm{Score}_0 - \mathrm{Score}_\rho}{\max(\mathrm{Score}_0, \epsilon)},$$

where $\epsilon > 0$ is a small constant for numerical stability. Lower values indicate that the protocol preserves task quality better under attack.

Malicious Contact Rate. Malicious Contact Rate measures how frequently benign agents interact with malicious agents. Let $\mathcal{C}$ be the set of collaboration contacts in a given evaluation window, and let $\mathrm{mal}(j) = 1$ indicate that the contacted peer j is malicious. We define

$$\mathrm{MCR} = \frac{\sum_{(i,j)\in\mathcal{C}} \mathbb{I}[\mathrm{mal}(j) = 1]}{|\mathcal{C}|}.$$

For long-term robustness experiments, we report $\mathrm{MCR}_{\mathrm{early}} \rightarrow \mathrm{MCR}_{\mathrm{late}}$ to show whether the protocol reduces malicious exposure over repeated interactions.

C Extended OPAC Formulation

This appendix records the full version of Section 2, while the main text keeps only the compact formulation needed to support the protocol design and theorems in Section 3.

We formalize OPAC as a distributed multi-objective decision problem over open private-agent networks. Let

$$\mathcal{V} = \{A_1, A_2, \ldots, A_n\} \tag{49}$$

denote a collection of private agents embedded in an unknown physical communication graph

$$G = (\mathcal{V}, \mathcal{E}). \tag{50}$$

For theoretical analysis, we also use a weighted matrix representation $W^{\mathrm{phy}} \in \mathbb{R}^{n\times n}$ of the physical topology when convenient. Unlike centralized or fully observable multi-agent systems, no agent is assumed to know G globally. Instead, each agent A_i maintains a time-dependent local cognitive map

$$G_i(t) = (V_i(t), E_i(t)), \tag{51}$$

together with a weighted cognitive topology $W_i^{\mathrm{cog}}(t)$ and a routing state $R_i(t)$ derived from currently available metadata.

Each agent A_i is associated with a task variable pair (X_i, Y_i), a private memory store M_i, and a local auditing policy governing what information may be received, retained, or transmitted. The goal of cooperation is to improve prediction or decision quality for Y_i by augmenting local context with externally acquired context assets while respecting privacy, trust, and communication constraints.

C.1 Context Assets, Collaboration Trajectories, and Policies

The basic unit of cooperation in OPAC is not a full response, but a *context asset*. We use this term abstractly to denote a transferable unit of task-relevant context that can be exchanged and integrated into another agent's local reasoning state. Context assets include memory fragments, sanitized summaries, skill artifacts, and delegated execution outputs; a memory slot is an important special case. Unlike a fully generated response, a context asset is intended to be reusable across tasks and amenable to asynchronous exchange. Section 3.3 later refines this operationally, while the theory continues to track all admitted contributions through the same imported-context abstraction.

For agent A_i, a cooperation event under policy π is written as

$$e_k = (A_j \rightarrow A_i,\, a_k,\, m_{j\rightarrow i}^{(k)}), \tag{52}$$

where $a_k \in \{\texttt{raw}, \texttt{processed}, \texttt{block}\}$ records the exchange action selected for that interaction and $m_{j\to i}^{(k)}$ denotes the admitted context asset for that event. A collaboration trajectory is then

$$\Gamma_i^{\pi}(t) = (e_1, e_2, \ldots, e_K), \tag{53}$$

and the aggregated external context delivered to A_i is denoted by

$$m_{\to i}^{\pi}(t) = \{m_{j\to i}^{(k)}\}_{k=1}^{K}. \tag{54}$$

The effective context available to A_i under policy π is

$$\widetilde{M}_i^{\pi}(t) = M_i \cup m_{\to i}^{\pi}(t). \tag{55}$$

A local cooperation policy π_i maps the local state of agent A_i—including its observation, private memory, cognitive topology, routing state, and local safety policy—to routing, exchange, auditing, and integration actions. A network-level cooperation policy is denoted by

$$\pi = (\pi_1, \ldots, \pi_n). \tag{56}$$

C.2 Safe Feasible Policy Class Under Partial Observability

Because agents only have partial topology visibility, a policy can only act on candidates that are currently discoverable under the agent's local cognitive state. Let

$$\mathcal{A}_i(t) \tag{57}$$

denote the admissible set of external context assets for agent A_i at time t, where admissibility jointly requires discoverability, reachability, authorization, and successful auditing. Operationally, $\mathcal{A}_i(t)$ should be read as the agent's currently certified admissible set under local routing and audit state rather than as an oracle revelation of all truly safe transfers.

Let $\mathcal{S}$ denote the set of sensitive information types, and let

$$\mathrm{Auth}(s) \subseteq \mathcal{V} \tag{58}$$

denote the agents authorized to access sensitive information $s \in \mathcal{S}$. For a joint policy π, we define the privacy leakage risk of agent A_i as

$$R_i^{\mathrm{priv}}(\pi) = \max_{s\in\mathcal{S},\, A_i\notin\mathrm{Auth}(s)} I(m_{\to i}^{\pi}(t); s)\,. \tag{59}$$

More generally, harmful-peer isolation, poisoning resistance, and content filtering are all absorbed into the admissible set and the local auditing policy.

Because the imported context $m_{\to i}^{\pi}(t)$ and the privacy risk $R_i^{\mathrm{priv}}(\pi)$ inherently depend on the release actions of other agents, the network-level safe feasible set cannot be decomposed into an independent Cartesian product. While the base action space $\Pi_1 \times \cdots \times \Pi_n$ is decomposable, the network-level joint safe policy set at time t is defined collectively as:

$$\Pi_{\mathrm{safe}}(t) = \left\{\pi \in \Pi_1 \times \cdots \times \Pi_n \;\middle|\; \forall i,\ m_{\to i}^{\pi}(t) \subseteq \mathcal{A}_i(t),\ R_i^{\mathrm{priv}}(\pi) \le \epsilon_i\right\} \tag{60}$$

for task- or agent-dependent tolerances $\epsilon_i \ge 0$. This joint feasible class is the decision space of OPAC.

C.3 Multi-Objective Evaluation of Cooperation Policies

For theoretical analysis, we evaluate predictive performance under the Bayes-optimal log score. For a safe policy $\pi \in \Pi_{\mathrm{safe}}(t)$, define the local predictive utility of agent A_i as

$$U_i(\pi) = \mathbb{E}\Big[\log p^{\star}\Big(Y_i \mid X_i, \widetilde{M}_i^{\pi}(t)\Big)\Big]\,, \tag{61}$$

where $p^{\star}$ denotes the Bayes-optimal predictive distribution. The aggregate network utility is then

$$U(\pi) = \sum_{i=1}^{n} U_i(\pi). \tag{62}$$

This Bayes-optimal log score is used as an analytical surrogate for task utility; in empirical agent tasks, Y_i may correspond to a reference answer, judge rubric, executable success criterion, or user-specified task outcome. Because utility is defined through predictive log score, Section 3.2 connects it to the task-relevant information carried by imported context assets.

In addition to utility, OPAC tracks a safety-trust objective and a resource objective. Let

$$T_i(\pi) = \mathbb{E}[\tau_i(\Gamma_i^\pi(t), m_{\to i}^\pi(t))] \tag{63}$$

denote a cooperation trust score, where $\tau_i(\cdot)$ rewards authorization compliance, routing through trusted peers, and avoidance of harmful or poorly sanitized transfers. Let

$$C_i(\pi) = C_i(\Gamma_i^\pi(t)) \tag{64}$$

denote the communication and computation cost of the policy, including route discovery, transmission, sanitization, summarization, and local reasoning overhead. We aggregate these objectives as

$$U(\pi) = \sum_{i=1}^n U_i(\pi), \qquad T(\pi) = \sum_{i=1}^n T_i(\pi), \qquad C(\pi) = \sum_{i=1}^n C_i(\pi). \tag{65}$$

Thus the hard-safe feasible class specifies which policies are allowable at all, while $T(\pi)$ distinguishes the more or less trustworthy operating points within that allowable set.

For comparison with the full-information ideal, let $\widetilde{M}_i^\star$ denote the effective context that agent A_i would access under full global awareness and unconstrained discovery. The corresponding information-theoretic upper bound is

$$I_i^{\text{global}} = I\left(Y_i; \widetilde{M}_i^\star \mid X_i, M_i\right), \tag{66}$$

which will later be useful for discussing topology-aware approximation gaps.

C.4 Pareto Formulation of OPAC

We represent each safe cooperation policy by the objective vector

$$z(\pi) = \big(U(\pi),\, T(\pi),\, -C(\pi)\big), \tag{67}$$

so that larger coordinates are always preferred. For two safe policies $\pi, \pi' \in \Pi_{\text{safe}}(t)$, we say that π' *dominates* π if $z(\pi') \geq z(\pi)$ in coordinate wise and at least one inequality is strict.

The OPAC Pareto frontier at time t is therefore

$$\mathcal{P}^\star(t) = \{\pi \in \Pi_{\text{safe}}(t) \mid \nexists\, \pi' \in \Pi_{\text{safe}}(t) \text{ such that } z(\pi') \succ z(\pi)\}. \tag{68}$$

Accordingly, the goal of OPAC is not to identify a single policy under a fixed scalarization, but to characterize or approximate the safe Pareto frontier using only local observations, partial topology beliefs, and audited cooperation. This avoids baking in a universal operating point even though different deployments may prioritize privacy, latency, or utility differently.

In practice, agents only observe empirical estimates of these objectives from finitely many interaction samples. Let $\widehat{z}_t(\pi)$ denote the empirical estimate of $z(\pi)$ after t observations. A solver for OPAC must therefore address two coupled challenges: it must expand the safe feasible set available under partial observability, and it must drive the empirical frontier induced by feasible policies toward the population Pareto frontier $\mathcal{P}^\star(t)$.

D Full Theoretical Results and Proofs

This appendix records the idealized assumptions and full supporting formal statements behind the main theorem in Section 3.5. The main text keeps only the solver-level statement for page budget and exposition.

D.1 Analysis Assumptions

To avoid hidden assumptions, we collect the analytical conditions used by the appendix results in one place. For the consistency results, fix an evaluation snapshot at horizon t. The task distribution, authorization rules, and reference physical topology are held fixed during evaluation. Policy classes and population objective maps may depend on the preceding history, but are fixed before collecting fresh evaluation samples. Within these consistency results, all objectives are maximized, and $>$ denotes strict inequality in every coordinate. For a nonempty finite set $Q \subset \mathbb{R}^3$, define

$$\mathrm{PF}(Q) = \{x \in Q : \nexists y \in Q,\ y \geq x,\ y \neq x\},$$

and

$$g_Q(x) = \max_{y \in Q} \min_{1 \leq k \leq 3} (y_k - x_k).$$

Throughout this appendix, d_H is the Hausdorff distance induced by $\|\cdot\|_\infty$.

(A1) **Bounded pruning losses.** For each routed candidate queue, the pruning loss variables ΔU_k are bounded in $[0, L]$ and are independent or otherwise satisfy the same concentration inequality used below.

(A2) **Connected normalized physical topology.** The physical interaction graph is connected with spectral gap $\gamma^{\mathrm{phy}} > 0$, and semantic heterogeneity enters through a bounded bias term $b(\Gamma)$ together with bounded observation noise σ.

(A3) **Topology-learning contraction.** The cognitive topology update contracts toward the physical graph in expectation according to

$$\mathbb{E}\big\|W_i^{\mathrm{cog}}(t+1) - W^{\mathrm{phy}}\big\|_2 \leq (1 - \eta\gamma^{\mathrm{phy}})\, \mathbb{E}\big\|W_i^{\mathrm{cog}}(t) - W^{\mathrm{phy}}\big\|_2 + \eta(b(\Gamma) + \sigma), \quad (69)$$

where $\eta > 0$ is a learning step size satisfying $1 - \eta\gamma^{\mathrm{phy}} \in (0, 1)$.

(A4) **Monotone admissible discovery.** In the idealized analysis model, improving routing evidence monotonically expands the relevant admissible discoverable set, so topology-constrained imported information is nondecreasing in expectation.

(A5) **Certified sanitization and blocking.** The sanitization map S is assumed to provide a certified residual leakage bound ρ_S, such that $I(S(\xi); s) \leq \rho_S$. Transmitted assets retain at most this residual information, and perfect blocking implies $\rho_S = 0$.

(A6) **Finite-entropy Bayes utility.** For each agent, the target variable Y_i has finite conditional entropy given (X_i, M_i), and predictive utility is measured by the Bayes-optimal log score from Section 2.

(A7) **Finite joint safe policy class.** At horizon t, the reference class $\Pi_{\mathrm{safe}}(t)$ is a nonempty finite set satisfying the jointly coupled safety constraints.

(A8) **Fresh bounded objective samples.** For each policy in a class being evaluated, each objective estimate is an average of N_t fresh samples. Conditional on the evaluation snapshot, the samples for each policy and coordinate are independent, have the corresponding population objective as their common mean, and lie in an interval of length at most $B_{\max}$. This applies to both the reference map z and the realized map z_t introduced in A11. Independence across policies or coordinates is not required. The count N_t is per policy, not the total number of interactions across the network. This bounded-sample condition is additional to the finite-entropy condition in A6.

(A9) **Uniform topology sensitivity.** Write $\mathbf{W} = (W_1, \ldots, W_n)$ and

$$\|\mathbf{W} - \mathbf{W}'\|_{\max,2} = \max_i \|W_i - W_i'\|_2.$$

For all relevant topology states and all $\pi \in \Pi_{\mathrm{safe}}(t)$,

$$\|z(\pi; \mathbf{W}) - z(\pi; \mathbf{W}')\|_\infty \leq L_{\mathrm{obj}} \|\mathbf{W} - \mathbf{W}'\|_{\max,2}.$$

Here $\|\cdot\|_2$ is the spectral norm. Let $\mathbf{W}^{\mathrm{phy}} = (W^{\mathrm{phy}}, \ldots, W^{\mathrm{phy}})$ and use $z(\pi) = z(\pi; \mathbf{W}^{\mathrm{phy}})$ for the reference objective in the consistency comparison.

(A10) **Uniform dominance-gap error bound.** Define

$$Z_0(t) = \{z(\pi) : \pi \in \Pi_{\mathrm{safe}}(t)\}, \qquad Z_t = \{z_t(a) : a \in \mathcal{K}_t\},$$

where $\mathcal{K}_t$ and z_t are specified in A11. There is a constant $\kappa > 0$, independent of t and bounded away from zero along any asymptotic regime considered, such that

$$g_Q(x) \geq \kappa \, \mathrm{dist}_\infty\big(x, \mathrm{PF}(Q)\big), \qquad x \in Q, \quad Q \in \{Z_0(t), Z_t\}.$$

This condition is imposed on both population objective sets, not on their noisy empirical estimates.

(A11) **Uniform safe realizability and empirical archive.** At each snapshot, AEP induces a nonempty finite class $\mathcal{K}_t$ of safe realized policies with population objective map z_t. There is a surjection $R_t : \Pi_{\mathrm{safe}}(t) \to \mathcal{K}_t$ such that

$$\sup_{\pi \in \Pi_{\mathrm{safe}}(t)} \|z_t(R_t(\pi)) - z(\pi; \mathbf{W}^{\mathrm{cog}}(t))\|_\infty \leq C_{\mathrm{a}} \, \overline{\Delta U},$$

where $\mathbf{W}^{\mathrm{cog}}(t) = (W_1^{\mathrm{cog}}(t), \ldots, W_n^{\mathrm{cog}}(t))$, C_{a} is independent of t, and

$$\overline{\Delta U} = K_{\max}^{-1} \sum_{k=1}^{K_{\max}} \Delta U_k$$

is the nonnegative loss in Lemma 3. Every policy in $\mathcal{K}_t$ is evaluated as in A8. The analyzed AEP archive retains exactly those policies that are not exceeded in every empirical objective by more than $2\varepsilon_t$ by another policy in $\mathcal{K}_t$.

(A12) **Evaluation after the initial topology transient.** A3 holds for all n agents on the fixed physical topology, with $q = 1 - \eta\gamma^{\mathrm{phy}} \in (0, 1)$ and

$$e_0 = \max_i \mathbb{E}\|W_i^{\mathrm{cog}}(0) - W^{\mathrm{phy}}\|_2 < \infty.$$

For Theorem 9, let M_t be a deterministic upper bound on $|\Pi_{\mathrm{safe}}(t)|$ and define, for $0 < \delta < 1$,

$$\varepsilon_t = B_{\max} \sqrt{\frac{\log(12M_t/\delta)}{2N_t}}.$$

The stated bound applies at evaluation horizons satisfying

$$L_{\mathrm{obj}} q^t e_0 \leq \varepsilon_t.$$

D.2 Discovery Results

Theorem 2 (Stationary cognitive-topology error)**.** *Under Assumption A3, with step size $\eta \in (0, 1/\gamma^{\mathrm{phy}})$,*

$$\limsup_{t \to \infty} \mathbb{E}\big\|W_i^{\mathrm{cog}}(t) - W^{\mathrm{phy}}\big\|_2 \leq \frac{b(\Gamma) + \sigma}{\gamma^{\mathrm{phy}}}. \tag{70}$$

Lemma 3 (Concentration of auditing-induced pruning loss)**.** *Under the bounded independent pruning-loss assumption, for any $\varepsilon > 0$,*

$$\mathbb{P}\big(\overline{\Delta U} - \mathbb{E}[\overline{\Delta U}] > \varepsilon\big) \leq \exp\left(-\frac{2K_{\max}\varepsilon^2}{L^2}\right). \tag{71}$$

Thus, the realized average pruning loss concentrates around its expectation as the candidate budget grows; this does not by itself bound the expectation or the worst-case loss across policies.

Theorem 4 (Topology-aware utility growth under routing convergence)**.** *Let $m_{\to i}(t)$ denote the external memory discoverable by agent A_i under its current routing state. If the routing process monotonically expands the set of reachable relevant candidates, then the topology-aware cooperation gain*

$$I_i^{\mathrm{topo}}(t) = I(Y_i; m_{\to i}(t) \mid X_i, M_i) \tag{72}$$

is non-decreasing in expectation, and the information gap

$$E_i(t) = I_i^{\mathrm{global}} - I_i^{\mathrm{topo}}(t) \tag{73}$$

decreases as routing converges.

D.3 Safe Exchange Results

Proposition 5 (Authorized exchange bound)**.** *Let ξ_j be a candidate asset and S be a sanitization map correctly enforced by the source agent to produce the release $m_{j\to i} = S(\xi_j)$. If S satisfies a certified residual leakage bound $I(S(\xi_j); s) \le \rho_S$ for any sensitive information s, then the privacy risk to an unauthorized receiver is bounded by ρ_S. Furthermore, by the Data Processing Inequality (DPI), this leakage cannot exceed the original information content: $I(S(\xi_j); s) \le I(\xi_j; s)$. In the idealized case of perfect blocking or perfect redaction, we have $\rho_S = 0$.*

Theorem 6 (Utility–information equivalence for safe exchange policies)**.** *Consider the safe policy class $\Pi_{\mathrm{safe}}(t)$ from Section 2. Under the Bayes-optimal log score used to define $U(\pi)$, we have*

$$\arg\max_{\pi\in\Pi_{\mathrm{safe}}(t)} U(\pi) = \arg\max_{\pi\in\Pi_{\mathrm{safe}}(t)} \sum_{i=1}^{n} I(Y_i; m^{\pi}_{\to i}(t) \mid X_i, M_i)\,. \tag{74}$$

Accordingly, among safe policies, maximizing predictive utility is equivalent to maximizing the task-relevant conditional information imported through exchange.

Corollary 7 (When raw release dominates processed release)**.** *Consider a task T_i for agent A_i and a candidate source agent A_j. Suppose there exists a transferable asset ξ_j (e.g., a memory fragment or raw skill code) and a processed counterpart $\widetilde{\xi}^{\mathrm{proc}}_j$ such that*

$$I(Y_i; \xi_j \mid X_i, M_i) > I(Y_i; \widetilde{\xi}^{\mathrm{proc}}_j \mid X_i, M_i), \tag{75}$$

and the raw release does not incur higher communication cost nor lower cooperation trust than the processed alternative. Then raw release strictly dominates processed release in the Pareto sense for that interaction, i.e., $z(\pi_{\mathrm{raw}}) > z(\pi_{\mathrm{proc}})$.

D.4 Consistency Results

Theorem 8 (Sample complexity of Pareto consistency)**.** *Fix a horizon t and a confidence level $0 < \delta < 1$. Under A7–A8, if*

$$N_t \ge \frac{B^2_{\max}}{2\varepsilon^2} \log\!\left(\frac{6|\Pi_{\mathrm{safe}}(t)|}{\delta}\right),$$

then, with probability at least $1-\delta$,

$$\sup_{\pi\in\Pi_{\mathrm{safe}}(t)} \|\widehat{z}_t(\pi) - z(\pi)\|_\infty \le \varepsilon.$$

Define

$$\widehat{\mathcal{P}}_{t,2\varepsilon} = \{\pi \in \Pi_{\mathrm{safe}}(t) : \nexists \pi' \in \Pi_{\mathrm{safe}}(t),\ \widehat{z}_t(\pi') > \widehat{z}_t(\pi) + 2\varepsilon\mathbf{1}\}\,,$$

and

$$\widehat{\mathcal{P}}^{z}_{t,2\varepsilon} = \{z(\pi) : \pi \in \widehat{\mathcal{P}}_{t,2\varepsilon}\}.$$

On the same event,

$$\mathcal{P}^{\star,z}(t) \subseteq \widehat{\mathcal{P}}^{z}_{t,2\varepsilon}, \qquad \sup_{\pi\in\widehat{\mathcal{P}}_{t,2\varepsilon}} g_{Z_0(t)}(z(\pi)) \le 4\varepsilon.$$

If A10 also holds for $Z_0(t)$, then

$$d_H\Big(\widehat{\mathcal{P}}^{z}_{t,2\varepsilon}, \mathcal{P}^{\star,z}(t)\Big) \le \frac{4\varepsilon}{\kappa}.$$

The same statement applies to the realized class $\mathcal{K}_t$ and population map z_t, with their corresponding empirical estimates.

Theorem 9 (AEP as a Pareto-consistent solver for OPAC, formal)**.** *Under A1–A12, fix $0 < \delta < 1$ and an evaluation horizon satisfying A12. Let*

$$\widehat{\mathcal{K}}_t = \{a \in \mathcal{K}_t : \nexists a' \in \mathcal{K}_t,\ \widehat{z}_t(a') > \widehat{z}_t(a) + 2\varepsilon_t\mathbf{1}\}\,,$$

and

$$\widehat{\mathcal{P}}^{\mathrm{AEP},z}_t = \{z_t(a) : a \in \widehat{\mathcal{K}}_t\}.$$

Then, with probability at least $1-\delta$*,*

$$d_H\Big(\widehat{\mathcal{P}}_t^{\mathrm{AEP},z}, \mathcal{P}^{\star,z}(t)\Big) \le \mathcal{O}(\varepsilon_t) + \mathcal{O}\bigg(L_{\mathrm{obj}}\frac{b(\Gamma)+\sigma}{\gamma^{\mathrm{phy}}}\bigg) + \mathcal{O}\big(\overline{\Delta U}\big)\,. \tag{76}$$

The hidden constants may depend on κ*,* C_{a}*,* n*, and* δ*, but not on* t *or the three displayed error magnitudes; an explicit bound is given in the proof. The objective-space image uses population objectives of the empirically selected policies. Using their empirical objective values instead adds at most* ε_t *to the bound. Along regimes where* $\varepsilon_t \to 0$*,* $L_{\mathrm{obj}}(b(\Gamma)+\sigma)/\gamma^{\mathrm{phy}} \to 0$*, and* $\overline{\Delta U} \to 0$ *in probability, with the preceding constants uniformly controlled for each fixed confidence level, the Hausdorff distance converges to zero in probability. If a structural error term does not vanish, the result gives an approximation bound rather than exact consistency.*

D.5 Proofs

We now provide theorem-by-theorem proofs corresponding to the full statements collected in Appendix D. Throughout, all expectations are taken with respect to the joint distribution induced by the task variables, local memories, routing state, and the safe cooperation policy under consideration.

Proof of Theorem 2. Let

$$\Delta_t := W_i^{\mathrm{cog}}(t) - W^{\mathrm{phy}}. \tag{77}$$

Assume that the physical graph is connected with spectral gap

$$\gamma^{\mathrm{phy}} := 1 - \lambda_2\big(W^{\mathrm{phy}}\big) > 0, \tag{78}$$

and choose a constant step size η such that $0 < \eta < 1/\gamma^{\mathrm{phy}}$. Let

$$\rho_\eta := 1 - \eta\gamma^{\mathrm{phy}} \in (0,1). \tag{79}$$

If the topology-learning error satisfies the graph-spectral recursion

$$\mathbb{E}\|\Delta_{t+1}\|_2 \le (1-\eta\gamma^{\mathrm{phy}})\,\mathbb{E}\|\Delta_t\|_2 + \eta\big(b(\Gamma)+\sigma\big), \tag{80}$$

then

$$\limsup_{t\to\infty} \mathbb{E}\left\|W_i^{\mathrm{cog}}(t) - W^{\mathrm{phy}}\right\|_2 \le \frac{\eta\big(b(\Gamma)+\sigma\big)}{1-\rho_\eta}. \tag{81}$$

Equivalently,

$$\limsup_{t\to\infty} \mathbb{E}\left\|W_i^{\mathrm{cog}}(t) - W^{\mathrm{phy}}\right\|_2 \le \frac{b(\Gamma)+\sigma}{\gamma^{\mathrm{phy}}}. \tag{82}$$

Write

$$e_t := \mathbb{E}\|\Delta_t\|_2. \tag{83}$$

The assumed recursion is exactly

$$e_{t+1} \le \rho_\eta e_t + \eta\big(b(\Gamma)+\sigma\big), \qquad 0 < \rho_\eta < 1. \tag{84}$$

Iterating this inequality gives, for every $t \ge 0$,

$$\begin{aligned} e_t &\le \rho_\eta^t e_0 + \eta\big(b(\Gamma)+\sigma\big)\sum_{k=0}^{t-1}\rho_\eta^k \\ &= \rho_\eta^t e_0 + \eta\big(b(\Gamma)+\sigma\big)\frac{1-\rho_\eta^t}{1-\rho_\eta}. \end{aligned} \tag{85}$$

Because $\rho_\eta \in (0,1)$, we have $\rho_\eta^t \to 0$ as $t \to \infty$. Taking $\limsup$ on both sides yields

$$\limsup_{t\to\infty} e_t \le \frac{\eta\big(b(\Gamma)+\sigma\big)}{1-\rho_\eta}. \tag{86}$$

Substituting the definition $\rho_\eta = 1 - \eta\gamma^{\mathrm{phy}}$ gives

$$1 - \rho_\eta = \eta\gamma^{\mathrm{phy}}, \tag{87}$$

and therefore

$$\frac{\eta\big(b(\Gamma)+\sigma\big)}{1-\rho_\eta} = \frac{b(\Gamma)+\sigma}{\gamma^{\mathrm{phy}}}. \tag{88}$$

Recalling that $e_t = \mathbb{E}\|W_i^{\mathrm{cog}}(t) - W^{\mathrm{phy}}\|_2$ proves both displayed bounds. □

Proof of Theorem 4. Fix an agent A_i. By assumption, the routing process monotonically expands the set of reachable relevant candidates. Hence the corresponding imported-context collection is increasing in the sense that for times $t_1 \le t_2$, the sigma-field generated by $m_{\to i}(t_1)$ is contained in the sigma-field generated by $m_{\to i}(t_2)$. Equivalently, there exists a random variable Δ_{t_1,t_2} such that

$$m_{\to i}(t_2) = \big(m_{\to i}(t_1), \Delta_{t_1,t_2}\big). \tag{89}$$

Applying the chain rule for conditional mutual information yields

$$\begin{aligned} I(Y_i; m_{\to i}(t_2) \mid X_i, M_i) &= I(Y_i; m_{\to i}(t_1), \Delta_{t_1,t_2} \mid X_i, M_i) \\ &= I(Y_i; m_{\to i}(t_1) \mid X_i, M_i) + I(Y_i; \Delta_{t_1,t_2} \mid X_i, M_i, m_{\to i}(t_1)) \,. \end{aligned} \tag{90}$$

The second term is nonnegative, so

$$I_i^{\mathrm{topo}}(t_2) \ge I_i^{\mathrm{topo}}(t_1). \tag{91}$$

Therefore $I_i^{\mathrm{topo}}(t)$ is non-decreasing, and hence also non-decreasing in expectation.

Now recall that the full-information benchmark is

$$I_i^{\mathrm{global}} = I\Big(Y_i; \widetilde{M}_i^{\star} \mid X_i, M_i\Big)\,, \tag{92}$$

where $\widetilde{M}_i^{\star}$ contains all relevant memory accessible under global awareness and unconstrained discovery. Since the discoverable memory under routing convergence approaches this globally available set, the residual gap

$$E_i(t) = I_i^{\mathrm{global}} - I_i^{\mathrm{topo}}(t) \tag{93}$$

is non-increasing in t. In particular, whenever routing reveals an additional relevant admissible source with positive conditional information, the gap decreases strictly; otherwise it remains unchanged. This proves the claim. □

Proof of Theorem 6. Fix a safe policy $\pi \in \Pi_{\mathrm{safe}}(t)$. By definition,

$$U_i(\pi) = \mathbb{E}\Big[\log p^{\star}\Big(Y_i \mid X_i, \widetilde{M}_i^{\pi}(t)\Big)\Big]\,. \tag{94}$$

Because $p^{\star}(\cdot \mid X_i, \widetilde{M}_i^{\pi}(t))$ is the Bayes-optimal predictive distribution, the standard identity for log score gives

$$U_i(\pi) = -H\Big(Y_i \mid X_i, \widetilde{M}_i^{\pi}(t)\Big)\,. \tag{95}$$

Using $\widetilde{M}_i^{\pi}(t) = \big(M_i, m_{\to i}^{\pi}(t)\big)$, we obtain

$$\begin{aligned} U_i(\pi) &= -H(Y_i \mid X_i, M_i, m_{\to i}^{\pi}(t)) \\ &= -H(Y_i \mid X_i, M_i) + I(Y_i; m_{\to i}^{\pi}(t) \mid X_i, M_i)\,, \end{aligned} \tag{96}$$

where the second line follows from the identity

$$I(Y; Z \mid W) = H(Y \mid W) - H(Y \mid W, Z). \tag{97}$$

Summing over agents yields

$$U(\pi) = \sum_{i=1}^{n} U_i(\pi) = -\sum_{i=1}^{n} H(Y_i \mid X_i, M_i) + \sum_{i=1}^{n} I(Y_i; m_{\to i}^{\pi}(t) \mid X_i, M_i)\,. \tag{98}$$

The first term is independent of the policy π. Therefore, over the safe policy class, maximizing $U(\pi)$ is equivalent to maximizing

$$\sum_{i=1}^{n} I(Y_i; m_{\to i}^{\pi}(t) \mid X_i, M_i)\,. \tag{99}$$

Hence

$$\arg\max_{\pi \in \Pi_{\mathrm{safe}}(t)} U(\pi) = \arg\max_{\pi \in \Pi_{\mathrm{safe}}(t)} \sum_{i=1}^{n} I(Y_i; m_{\to i}^{\pi}(t) \mid X_i, M_i)\,, \tag{100}$$

as claimed. □

Proof of Corollary 7. Let π_{raw} and π_{proc} denote the policies for raw and processed release, respectively. In the OPAC framework, their corresponding multi-objective vectors are defined as $z(\pi_{\text{raw}}) = (U_{\text{raw}}, T_{\text{raw}}, -C_{\text{raw}})$ and $z(\pi_{\text{proc}}) = (U_{\text{proc}}, T_{\text{proc}}, -C_{\text{proc}})$.

By Theorem 6, the predictive utility ordering between these two actions is equivalent to the ordering of their task-relevant conditional mutual information. Hence the hypothesis

$$I(Y_i; \xi_j \mid X_i, M_i) > I(Y_i; \widetilde{\xi}_j^{\text{proc}} \mid X_i, M_i) \tag{101}$$

strictly implies an advantage in the utility dimension:

$$U_{\text{raw}} > U_{\text{proc}}. \tag{102}$$

Furthermore, under the premise that the raw release does not incur higher communication cost nor lower cooperation trust than the processed alternative, the remaining coordinates satisfy:

$$-C_{\text{raw}} \geq -C_{\text{proc}} \quad \text{and} \quad T_{\text{raw}} \geq T_{\text{proc}}. \tag{103}$$

Combining these conditions yields $z(\pi_{\text{raw}}) \geq z(\pi_{\text{proc}})$ coordinate-wise, with a strict inequality in the predictive utility dimension. Therefore, the raw release strictly dominates the processed release in the Pareto sense:

$$z(\pi_{\text{raw}}) \succ z(\pi_{\text{proc}}). \tag{104}$$

This proves that raw release yields a strictly better Pareto tradeoff for that interaction. □

Proof of Lemma 3. By assumption, the random variables $\Delta U_1, \ldots, \Delta U_{K_{\max}}$ are independent and each lies in $[0, L]$. Their average is

$$\overline{\Delta U} = \frac{1}{K_{\max}} \sum_{k=1}^{K_{\max}} \Delta U_k. \tag{105}$$

Hoeffding's inequality for bounded independent random variables gives, for any $\varepsilon > 0$,

$$\mathbb{P}\big(\overline{\Delta U} - \mathbb{E}[\overline{\Delta U}] > \varepsilon\big) \leq \exp\left(-\frac{2K_{\max}^2 \varepsilon^2}{\sum_{k=1}^{K_{\max}} (L-0)^2}\right) = \exp\left(-\frac{2K_{\max}\varepsilon^2}{L^2}\right). \tag{106}$$

This is exactly the stated bound. The exponential dependence on $K_{\max}$ shows that the realized pruning loss concentrates sharply around its expectation as the candidate budget increases. □

Proof of Proposition 5. Assume the source agent applies a sanitization map S to the candidate asset ξ_j, resulting in the transmitted artifact $m_{j\to i} = S(\xi_j)$. For any sensitive attribute s unauthorized for the receiver, the privacy leakage risk is measured by the mutual information $I(m_{j\to i}; s) = I(S(\xi_j); s)$.

By definition, if the sanitization mechanism guarantees a certified residual leakage ρ_S, we have:

$$I(S(\xi_j); s) \leq \rho_S. \tag{107}$$

Furthermore, since $S(\xi_j)$ is a measurable function of the pre-sanitized asset ξ_j, the Data Processing Inequality (DPI) ensures that sanitization cannot increase mutual information:

$$I(S(\xi_j); s) \leq I(\xi_j; s)\,. \tag{108}$$

Thus, the privacy risk to the unauthorized receiver is formally bounded by the certified capacity ρ_S of the sanitization map, which in turn is bounded by the original information $I(\xi_j; s)$.

If the policy blocks transmission entirely, then $m_{j\to i}$ is a constant null symbol, yielding $I(S(\xi_j); s) = 0$. Likewise, under perfect redaction with respect to s, the sanitized artifact is completely independent of s, so the mutual information is also zero. In these idealized cases, the upper bound becomes strictly $\rho_S = 0$. □

Proof of Theorem 8. Condition on the evaluation snapshot. By A8 and Hoeffding's inequality, followed by a union bound,

$$\mathbb{P}\left(\sup_{\pi\in\Pi_{\mathrm{safe}}(t)}\|\widehat{z}_t(\pi)-z(\pi)\|_\infty>\varepsilon\,\middle|\,\text{snapshot}\right)\le 6|\Pi_{\mathrm{safe}}(t)|\exp\left(-\frac{2N_t\varepsilon^2}{B_{\max}^2}\right)\le\delta.$$

The same probability guarantee holds after removing the conditioning. Work on the resulting uniform-accuracy event.

If a Pareto-optimal policy $\pi^\star$ were excluded, some policy π would satisfy

$$\widehat{z}_t(\pi)>\widehat{z}_t(\pi^\star)+2\varepsilon\mathbf{1}.$$

Uniform accuracy would imply $z(\pi)>z(\pi^\star)$ in every coordinate, a contradiction. Hence

$$\mathcal{P}^{\star,z}(t)\subseteq\widehat{\mathcal{P}}^{z}_{t,2\varepsilon}.$$

For any retained policy π, suppose $g_{Z_0(t)}(z(\pi))>4\varepsilon$. Since the policy class is finite, there exists π' such that

$$z(\pi')>z(\pi)+4\varepsilon\mathbf{1}.$$

Uniform accuracy gives

$$\begin{aligned}\widehat{z}_t(\pi')&\ge z(\pi')-\varepsilon\mathbf{1}\\&>z(\pi)+3\varepsilon\mathbf{1}\\&\ge\widehat{z}_t(\pi)+2\varepsilon\mathbf{1},\end{aligned}$$

contradicting retention. Thus $g_{Z_0(t)}(z(\pi))\le 4\varepsilon$.

Under A10,

$$\mathrm{dist}_\infty\big(z(\pi),\mathcal{P}^{\star,z}(t)\big)\le\frac{g_{Z_0(t)}(z(\pi))}{\kappa}\le\frac{4\varepsilon}{\kappa}.$$

Combining this directed-distance bound with the preceding set inclusion proves the Hausdorff bound. Replacing the reference policy class and map by $\mathcal{K}_t$ and z_t gives the realized-class statement. □

Proof of Theorem 9. Write

$$D=\frac{b(\Gamma)+\sigma}{\gamma^{\mathrm{phy}}},\qquad r_t=\max_i\|W_i^{\mathrm{cog}}(t)-W^{\mathrm{phy}}\|_2,\qquad q=1-\eta\gamma^{\mathrm{phy}}.$$

Step 1 (Finite-time topology control). Iterating A3 gives, for every agent,

$$\mathbb{E}\|W_i^{\mathrm{cog}}(t)-W^{\mathrm{phy}}\|_2\le q^te_0+D(1-q^t)\le q^te_0+D.$$

Since a maximum of nonnegative variables is bounded by their sum,

$$\mathbb{E}r_t\le n(q^te_0+D).$$

Markov's inequality therefore yields, with probability at least $1-\delta/2$,

$$r_t\le\frac{2n}{\delta}(q^te_0+D).$$

If the right-hand side is zero, the same conclusion holds almost surely.

Step 2 (Population objective-set perturbation). For each $\pi\in\Pi_{\mathrm{safe}}(t)$, A9 and A11 imply

$$\|z_t(R_t(\pi))-z(\pi)\|_\infty\le L_{\mathrm{obj}}r_t+C_{\mathrm{a}}\overline{\Delta U}=:\beta_t.$$

The map R_t is defined on every reference policy and is onto $\mathcal{K}_t$, so this bound controls both directions:

$$d_H(Z_t,Z_0(t))\le\beta_t.$$

On the topology event, A12 gives

$$\beta_t \le \frac{2n}{\delta}\big(\varepsilon_t + L_{\mathrm{obj}} D\big) + C_{\mathrm{a}}\overline{\Delta U} =: \overline{\beta}_t.$$

This step uses A11. Concentration of an average pruning loss alone would not imply this uniform objective-set bound.

Step 3 (Frontier stability). Let

$$F_0 = \mathrm{PF}(Z_0(t)) = \mathcal{P}^{\star,z}(t), \qquad F_t = \mathrm{PF}(Z_t).$$

For any $p \in F_0$, choose $u \in Z_t$ satisfying $\|u - p\|_\infty \le \beta_t$. For every $v \in Z_t$, choose $x \in Z_0(t)$ satisfying $\|v - x\|_\infty \le \beta_t$.

Since p is Pareto optimal, $\min_k(x_k - p_k) \le 0$. Therefore

$$\min_k(v_k - u_k) \le \min_k(x_k - p_k) + 2\beta_t \le 2\beta_t.$$

Taking the maximum over v gives $g_{Z_t}(u) \le 2\beta_t$. By A10,

$$\begin{aligned}\mathrm{dist}_\infty(p, F_t) &\le \|p - u\|_\infty + \mathrm{dist}_\infty(u, F_t)\\ &\le \left(1 + \frac{2}{\kappa}\right)\beta_t.\end{aligned}$$

Interchanging the two objective sets and applying A10 to $Z_0(t)$ gives the reverse directed-distance bound. Hence

$$d_H(F_t, F_0) \le \left(1 + \frac{2}{\kappa}\right)\beta_t.$$

Step 4 (Statistical error and aggregation). Condition on the evaluation snapshot. By A8 and $|\mathcal{K}_t| \le |\Pi_{\mathrm{safe}}(t)| \le M_t$,

$$\begin{aligned}&\mathbb{P}\left(\sup_{a\in\mathcal{K}_t} \|\widehat{z}_t(a) - z_t(a)\|_\infty > \varepsilon_t \,\middle|\, \text{snapshot}\right)\\ &\qquad \le 6M_t \exp\left(-\frac{2N_t\varepsilon_t^2}{B_{\max}^2}\right) = \frac{\delta}{2}.\end{aligned}$$

The same guarantee holds unconditionally. A union bound shows that the statistical and topology events hold jointly with probability at least $1 - \delta$.

Applying Theorem 8 to $\mathcal{K}_t$ and z_t yields

$$d_H\left(\widehat{\mathcal{P}}_t^{\mathrm{AEP},z}, F_t\right) \le \frac{4\varepsilon_t}{\kappa}.$$

Consequently,

$$\begin{aligned}&d_H\left(\widehat{\mathcal{P}}_t^{\mathrm{AEP},z}, F_0\right)\\ &\quad\le \frac{4\varepsilon_t}{\kappa} + \left(1 + \frac{2}{\kappa}\right)\overline{\beta}_t\\ &\quad= \left[\frac{4}{\kappa} + \frac{2n}{\delta}\left(1 + \frac{2}{\kappa}\right)\right]\varepsilon_t\\ &\qquad+ \frac{2n}{\delta}\left(1 + \frac{2}{\kappa}\right) L_{\mathrm{obj}}\frac{b(\Gamma) + \sigma}{\gamma^{\mathrm{phy}}}\\ &\qquad+ C_{\mathrm{a}}\left(1 + \frac{2}{\kappa}\right)\overline{\Delta U}.\end{aligned}$$

This proves the stated three-term bound.

On the statistical event, replacing selected population objectives by their empirical estimates adds at most ε_t to the Hausdorff bound. Under the stated vanishing-error conditions, the bound yields convergence in probability by taking the confidence failure probability arbitrarily small. No almost-sure or simultaneous-in-t guarantee is asserted. □

E LIMITATIONS

This work studies AEP mainly in controlled benchmark environments, which may not fully capture the heterogeneity of real-world agent deployments. The current protocol also relies on practical choices for routing thresholds, trust updates, safety gates, and checkpointing policies. Different applications may require different trade-offs among utility, privacy, safety, and communication cost. Future work can further evaluate AEP in larger deployments and study adaptive policy configuration.

F SOCIAL IMPACT DISCUSSION

AEP aims to make collaboration among privately owned agents more useful while preserving selective disclosure and auditable control. This may help future agent ecosystems reduce duplicated work, improve interoperability, and support safer reuse of distributed context assets. However, agent networking systems may also increase the scale at which information is shared across services if deployed without proper governance. We therefore view access control, transparency, auditing, and human-configured policies as important complements to the protocol design.

AI USE STATEMENT

We used generative AI tools during the preparation of this work for language editing, improving the clarity and readability of the manuscript, and assisting with the organization and presentation of the paper. Generative AI tools were also used for brainstorming and, where applicable, for assisting with code development and debugging. All AI-assisted content was reviewed and, when necessary, revised by the authors. Any AI-assisted code was inspected and tested before being used in the reported experiments.