# CROSSMAS: A DEEP DIVE INTO MULTI-AGENT COLLABORATION ACROSS DISTRIBUTED ENVIRONMENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Language-model agents can complete complex tasks with tools, and multi-agent systems demonstrate collaborative execution. These advances motivate delegating tasks spanning multiple systems and participants to agents. Many systems rely on human-designed roles and workflows, with studies focusing on overall performance. Yet dependencies and failures during execution may require handoffs beyond these plans. Greater autonomy thus requires agents to organize cooperation themselves, but how they do so with local access and no predefined task assignments remains poorly understood. CrossMAS studies task completion and autonomous cooperation when information, mutable state, and action authority are distributed across nodes. Across four task families and four models, we combine matched comparisons of six organizations with trajectory analysis. Across these tasks, autonomous MAS average 0.394–0.437 completion, below the single-decision references Merge (0.604) and Central (0.538); organizational support yields no consistent gain. Agents exchange contributions and act on them, but adopting incorrect contributions, confusing evidence sources or action targets, and leaving downstream dependencies unresolved prevent local progress from becoming joint completion.

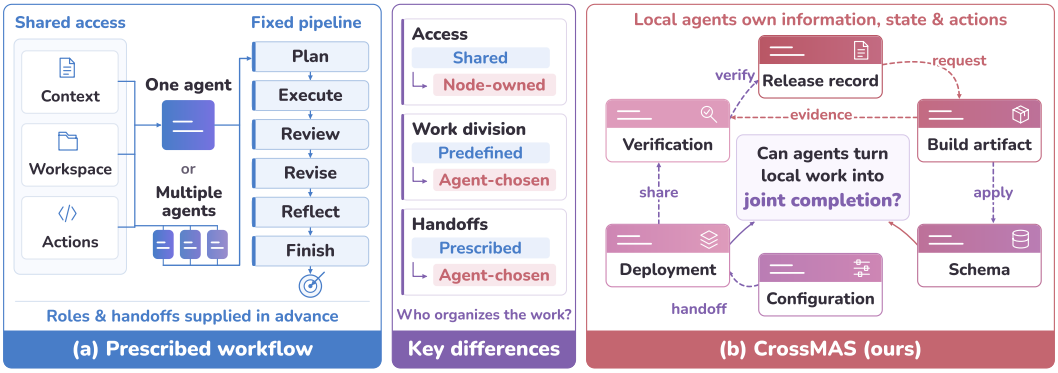



Figure 1: **From prescribed workflows to autonomous cooperation.** We study how agents with node-local information, state, and authority organize a shared task without predefined task-level roles.

## 1 INTRODUCTION

Language-model agents can plan, use tools, and execute workflows with limited supervision (Yao et al., 2023; Wu et al., 2024). These capabilities motivate delegating shared tasks across systems and participants to agents. With all required information and actions available, a single agent can maintain a unified history, execute the workflow, and check completion. However, many real-world tasks span multiple systems, environments, or participants, so the information and actions required for completion are not always available in one place. A single agent may lack the context, tools, or action authority needed to complete the workflow.



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.

Multi-agent execution provides a natural way to handle such tasks. Agents use local information, tools, and actions to pursue a shared objective. Joint completion requires identifying cross-agent dependencies, exchanging information or intermediate results, assigning actions, and checking the shared outcome. Existing LLM multi-agent systems (MAS) organize interaction through roles, dialogue, and staged workflows (Li et al., 2023; Hong et al., 2024; Wu et al., 2024; Chen et al., 2024). These systems typically specify roles, task decomposition, or interaction structure in advance and evaluate the resulting pipeline. However, dependencies discovered during execution or local failures may require work assignments and handoffs beyond these plans. Greater autonomy therefore requires agents to organize such work themselves, not only execute prescribed steps. Recent studies analyze cooperation failures (Cemri et al., 2025; Yang et al., 2026), while distributed-task benchmarks separate information, workspaces, or action access (Zhang et al., 2026; Khatua et al., 2026; Yu et al., 2026). These advances motivate a closer account of how agents organize cross-node execution and turn local contributions into jointly completed work.

We study autonomous cooperation under distributed access through two questions. First, how does task completion vary with access boundaries, the placement of task-level decisions, and organizational support? Second, when roles and decomposition are not prescribed, what cooperation do agents form, and how does a local contribution enter—or fail to enter—the final result? We introduce CrossMAS as an empirical study of these questions. Across Core and three adapted task families, we compare four models under six organizations. Matched comparisons hold the task, model, underlying task operations, and evaluator fixed while varying access boundaries, decision placement, and organizational support. Alongside completion and resource costs, we trace local contributions through cross-node actions to evaluated outcomes. Single-decision execution leads overall, while organizational support yields no consistent gain. Trajectory analysis locates gaps between using contributions and completing the work that depends on them. Figure 1 contrasts prescribed workflows with the distributed, role-unprescribed setting studied here.

**Our contributions are threefold:**

- We formulate a distributed, role-unprescribed execution setting in which observations, mutable state, and action authority remain node-owned, together with a task collection and evaluation protocol that measures their joint completion.
- We construct matched comparisons among six organizations to examine completion under distributed boundaries, centralized task-level decisions, and explicit cooperation support.
- We provide a fine-grained analysis of autonomous cooperation by reconstructing how information, artifacts, actions, and responsibilities cross agent boundaries, change downstream execution, and enter the final evaluated result.

## 2 Related Work

**Language-model agents and multi-agent organization.** Toolformer and ReAct integrate external actions into model inference (Schick et al., 2023; Yao et al., 2023), and Generative Agents studies persistent multi-actor environments (Park et al., 2023). CAMEL, AutoGen, MetaGPT, and AgentVerse organize model instances through roles, dialogue, and staged workflows (Li et al., 2023; Wu et al., 2024; Hong et al., 2024; Chen et al., 2024). Magentic-One coordinates specialized agents through a planning and progress-tracking orchestrator (Fourney et al., 2024).

**Evaluation and analysis of agent collaboration.** COOP$^2$, MAST, and scaling studies analyze cooperation and execution failures (Yang et al., 2026; Cemri et al., 2025; Kim et al., 2025; Xu et al., 2024; Zhu et al., 2025); Who&When attributes failures to agents and execution steps (Zhang et al., 2025). Benchmarks with separated information, workspaces, or authority make cross-agent execution part of the task (Zhang et al., 2026; Khatua et al., 2026; Yu et al., 2026). Other work compares single-agent reasoning and multi-agent discussion (Wang et al., 2024), coordination cost, diversity loss under matched budgets, and shared-workspace coding (Sun et al., 2026; Ann et al., 2026; Cho & Lee, 2026). The Collaboration Gap examines unguided cooperation (Davidson et al., 2026), HiddenBench distributed-information reasoning (Li et al., 2026), and Pappu et al. (2026) self-organizing teams' use of expertise. These studies show that individual strength need not yield effective cooperation. MAESTRO analyzes trace-level reliability and cost (Ma et al., 2026). CrossMAS pairs organizational

comparisons with tracing contributions into actions, persistent progress, and joint outcomes under distributed access. Appendix A.2 compares these settings and analysis targets.

# 3 Preliminaries and Problem Setup

## 3.1 Distributed Agent Setting

We first formalize the problem motivating our study: when the observations, mutable state, and action authority required for a shared task are distributed across nodes, how do actors organize their local capabilities to achieve joint completion?

**Distributed State and Authority.** Following local-state and message-passing abstractions (Lamport, 1978), let $V = \{1, \ldots, n\}$ denote the nodes and $S^t = (s_1^t, \ldots, s_n^t)$ the environment state at time $t$. Node $i$'s local access is defined by an observation function $\Omega_i$ and an action set $\mathcal{A}_i$:

$$o_i^t = \Omega_i(S^t), \qquad a_i^t \in \mathcal{A}_i. \tag{1}$$

Here, $s_i^t$ may be a workspace, a business object, or another mutable state, and $\mathcal{A}_i$ specifies the task operations available at that node. Tool calls act on the actual environment and produce state updates or error feedback. Information can cross interfaces, while authority to modify node state is governed by the execution contract. Figure 1 contrasts this node-local access with a shared execution environment.

**Cross-Node Coordination.** The relation between local access and the shared objective creates cross-node dependencies. We write $i \to j$ when node $j$'s next step requires information, an artifact, or a completed operation from node $i$. Actors must discover and satisfy these dependencies to turn local work into a joint result. Let $\tau$ denote an execution trajectory containing tool calls, messages, feedback, and state changes for task $\mathcal{T}$. Its evaluation function assigns completion from formal submissions, delivered artifacts, and final state:

$$c(\tau) = E_{\mathcal{T}}(\tau) \in [0, 1], \tag{2}$$

where $c = 1$ means that all required completion atoms are satisfied; Equation 4 specifies this evaluation function.

**Agent Organizations.** To turn this question into comparable execution processes, we examine common forms of unified decision making, autonomous peer interaction, and explicit organizational support. An organization $\Gamma$ specifies decision makers' access scopes, decision histories, and communication interfaces. Under autonomous execution, a node agent chooses a task action or message from its local history:

$$u_i^t \sim \pi_\theta(\cdot \mid h_i^t; \Gamma), \qquad h_i^t = (o_i^{\leq t}, u_i^{<t}, f_i^{<t}, m_{\to i}^{\leq t}), \tag{3}$$

Here $\pi_\theta$ is the task model's policy, $f_i$ is tool feedback, and $m_{\to i}$ denotes received messages; $u_i$ is either a local action $a_i \in \mathcal{A}_i$ or a communication action. We instantiate these common forms as six conditions (Table 1).

Merge–Central contrasts merged and node-separated execution; Central–P2P contrasts global and local decision histories. P2P leaves task division to agents; Relay, Role, and Control add message forwarding, responsibility information, and coordination support, respectively. Figure 2 summarizes the study design.

Table 1: **Organization conditions.** Mechanism dimensions of the six matched conditions.

| Condition | Decision locus | Cross-node interface | Work organization |
|---|---|---|---|
| **Merge** | Single decision-maker | Merged environment | Unified |
| **Central** | Single decision-maker | Separated node interfaces | Centralized |
| **P2P** | Local agents | Direct peer messaging | Self-organized |
| **Relay** | Local agents | Selected messages relayed verbatim | Self-organized |
| **Role** | Local agents | Direct peer messaging | Responsibility-informed |
| **Control** | Coordinator + workers | Coordinator-mediated messages | Coordinator-organized |

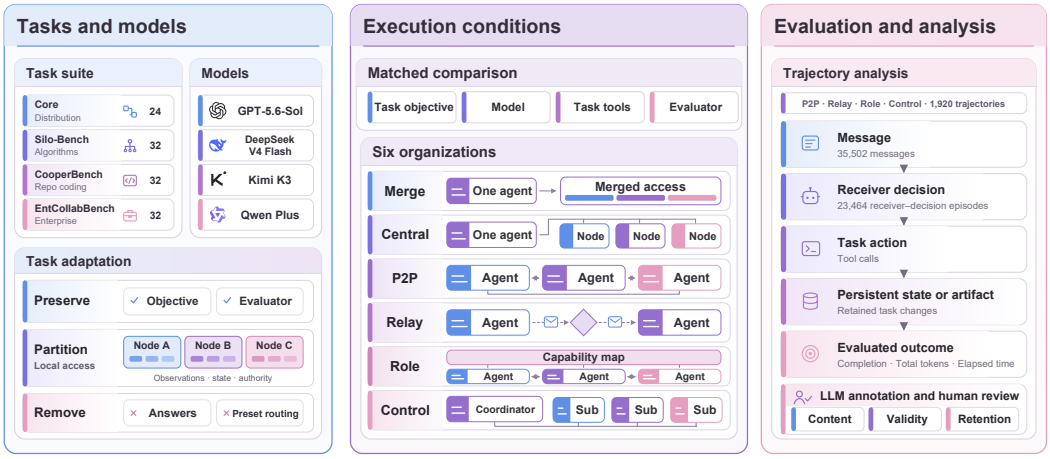



Figure 2: **CrossMAS study design.** Tasks with node-local access run under six organizations. We compare completion and resource costs, and reconstruct information transfer and subsequent behavior in the four MAS conditions to trace local contributions into joint outcomes.

**Cooperation Measurement.** Organizations specify interaction opportunities and access constraints; what agents exchange, who takes responsibility, and how contributions are used emerge during execution. We therefore link information transferred between agents to subsequent receiver behavior. Messages available to the same receiver before one model decision form a receiver–decision episode; messages with no later receiver decision remain inactive. We connect messages, actions, and results through task objects and their evidence.

We then code each episode at three linked levels. At the information level, we classify what is transferred and determine whether it is valid, invalid, or unresolved under task-native evidence. At the receiver level, activation requires a subsequent model decision, uptake requires that decision to use the transferred task object, and execution requires a corresponding task action, changed target or responsibility, or grounded relay.

At the outcome level, persistent progress requires a verifiable change in task state, artifact, submission, dependency, or closure; final retention requires that the contribution remain in the result observed by the evaluator. These labels are grounded in tests, submissions, dependency references, artifacts, or authorized state changes rather than message text alone. The shared episode unit supports comparison across task environments; Appendices B.4 and B.5 define scoring, matching windows, and verification. Together, these organizational contrasts and process measurements address two questions:

**Q1. Completion under distributed execution (§4.1)**
How do distributed boundaries, decision placement, and organizational support affect task completion and execution cost?

**Q2. Cooperation dynamics in autonomous MAS (§4.2)**
What cooperation do agents form, and how do local contributions enter subsequent execution and affect joint outcomes?

### 3.2 Experimental Setup

**Task Suite and Adaptation.** We instantiate the problem in Section 3.1 as executable tasks. Core covers job recovery, replica repair, and rollout recovery: local operations depend on evidence produced at other nodes before the task can close. It varies node count, workload, and dependencies. To examine organizational behavior across domains, we also adapt Silo-Bench, CooperBench, and EntCollabBench (Zhang et al., 2026; Khatua et al., 2026; Yu et al., 2026). Figure 3 summarizes the four families and their 120 configurations.

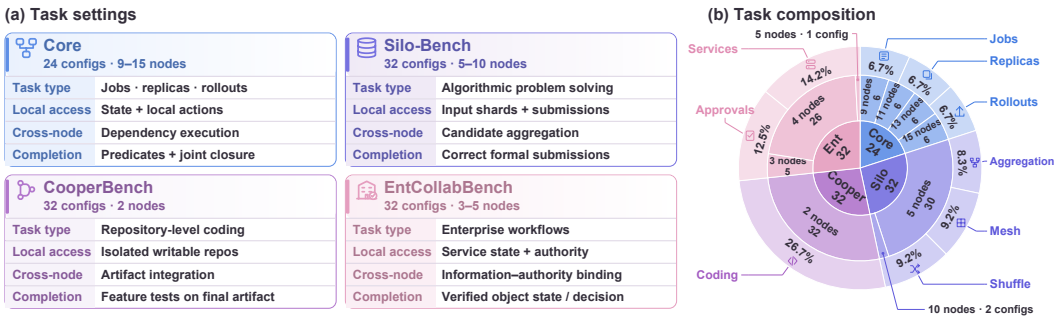



Figure 3: **Task-family taxonomy.** (a) Task settings and completion evidence. (b) Rings independently summarize family, node count, and task type (inner to outer), grouped by family color. Angles and percentages reflect shares of 120 configurations.

Adaptation retains task objectives and native evaluation evidence while mapping inputs, workspaces, services, and permissions to node interfaces. We remove model-visible answers and preset communication routes while retaining local task requirements and permissions. Agents choose the execution workflow. Each external family contains 30 main tasks and two preselected structural variants. Execution contracts and adaptation details are given in Appendix B.1.

**Matched Organizational Comparison.** Each model runs the same task under all six organizations, forming a matched model–task panel. Within a panel, the task objective, underlying task tools, and evaluator are fixed at the task level; the organization changes access scope, decision placement, and communication together. Table 1 summarizes these organization-level contrasts; instructions and interfaces are described in Appendix B.3.

**Models and Execution Protocol.** We use GPT-5.6-Sol, DeepSeek V4 Flash, Kimi K3, and Qwen Plus, producing 480 matched panels and 2,880 trajectories. All conditions receive the same public task objective. Distributed workers additionally receive their node identity, local observations, and tools, and interact across nodes through the exposed interfaces. Conditions within each task family share its execution boundaries; model configurations, scheduling, continuation, and stopping rules are given in Appendix B.2.

**Evaluation and Measurements.** Native evaluators supply test, submission, or state evidence. GPT-6-Astra maps this evidence and the trajectory to completion atoms under a fixed family-specific schema and annotates cooperation processes, with sampled human verification. Native binary success is recorded separately as Full. For a task with $K_{\mathcal{T}}$ completion atoms, the evaluation function in Equation 2 is the proportion of passed atoms:

$$c(\tau) = \frac{1}{K_{\mathcal{T}}} \sum_{k=1}^{K_{\mathcal{T}}} z_k(\tau), \qquad z_k(\tau) \in \{0, 1\}. \tag{4}$$

Here $z_k = 1$ denotes a passed atom. We also record total tokens, elapsed time, model decisions, messages, and tool calls. The 1,920 trajectories from the four MAS conditions contain 35,502 messages, forming 23,464 receiver–decision episodes. Completion scoring, event matching, annotation rules, and human-review coverage are documented in Appendices B.4 and B.5; Figure 2 summarizes how the cooperation-specific records are linked.

# 4 Results

## 4.1 Completion under distributed execution

We compare completion within matched model–task panels using the organizational contrasts in Section 3.1. Tokens and elapsed time measure resource use separately. Trajectory inspection then locates the earliest outcome-relevant difference in each non-tied pair.

Table 2: **Completion by model, task family, and organization.** Cells average 24 Core tasks or 32 tasks in each adapted family; Avg. covers all 120 configurations. Bold marks column maxima within each model.

| GPT-5.6-Sol | | | | | DeepSeek V4 Flash | | | | | Kimi K3 | | | | | Qwen Plus | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Mode | Core | Silo | Coop. | Ent | Avg. | Mode | Core | Silo | Coop. | Ent | Avg. | Mode | Core | Silo | Coop. | Ent | Avg. | Mode | Core | Silo | Coop. | Ent | Avg. |
| Merge | .515 | **.861** | **.781** | .680 | **.722** | Merge | **.444** | **.638** | .734 | **.703** | **.642** | Merge | .667 | **.802** | .641 | .824 | **.738** | Merge | .093 | .475 | .172 | **.455** | .312 |
| Central | **.649** | .773 | .406 | .701 | .631 | Central | .222 | .406 | .500 | .658 | .461 | Central | **.749** | .770 | .359 | .822 | .670 | Central | **.486** | **.598** | .047 | .454 | **.390** |
| P2P | .245 | .234 | .688 | **.751** | .495 | P2P | .002 | .292 | .719 | .599 | .430 | P2P | .373 | .592 | .625 | .780 | .607 | P2P | .010 | .090 | .312 | .294 | .188 |
| Relay | .289 | .287 | .719 | .589 | .483 | Relay | .000 | .272 | .703 | .641 | .431 | Relay | .339 | .576 | **.750** | .766 | .625 | Relay | .007 | .090 | .406 | .278 | .208 |
| Role | .138 | .423 | .750 | .670 | .519 | Role | .001 | .177 | **.750** | .604 | .408 | Role | .358 | .325 | .703 | .771 | .551 | Role | .015 | .126 | .422 | .415 | .260 |
| Control | .118 | .328 | .734 | .641 | .478 | Control | .000 | .096 | .734 | .598 | .381 | Control | .053 | .386 | .719 | **.831** | .527 | Control | .031 | .029 | **.453** | .202 | .189 |

| Aggregate | Merge | Central | P2P | Relay | Role | Control |
|---|---|---|---|---|---|---|
| **Completion** | **.604** | .538 | .430 | .437 | .435 | .394 |
| Top-2 inclusion (%) | 90.6 | 81.7 | 63.3 | – | – | 59.6 |

Top-2 is computed among Merge, Central, P2P, and Control, retaining ties.

**Observation 1.** Merge and Central provide the strongest aggregate completion.

Mean completion and tie-inclusive Top-2 frequency give the same ordering (Table 2). Merge leads overall and for three models; Central leads for Qwen Plus. Across 480 matched panels, overall mean completion is 0.604 for Merge and 0.538 for Central, compared with 0.430 for P2P and 0.394 for Control.

**Observation 2.** Merge leads overall, but its advantage over Central is task-dependent.

Merge and Central tie in 301/480 panels. Merge leads by 0.254 in Cooper and 0.057 in Silo, nearly ties in Ent (+0.007), and trails by 0.097 in Core (Table 10). Across the 179 non-tied panels shown in Figure 4, artifact convergence is the earliest outcome-relevant difference in 34/50 Cooper panels, and submission coverage in 19/35 Silo panels (Appendix B.6). Cooper contributes most of the overall Merge advantage (Appendix B.3).

In Core, initial target mismatch occurs in 62/96 Merge and 64/96 Central trajectories; recovery follows in 28/62 and 35/64, respectively. These traces show how node-local feedback enters later retries; model-specific outcomes vary (Table 2). Ent shows no consistent process advantage. The task-specific contrasts connect artifact convergence and feedback-guided revision to completion.

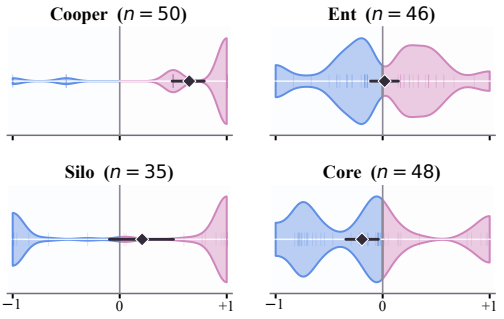



Figure 4: **Non-tied Merge–Central gaps.** Smoothed distributions of $c_{\text{Merge}} - c_{\text{Central}}$; diamonds mark means and bars show 95% panel-bootstrap intervals.

**Observation 3.** Single-decision execution leads in Core and Silo; Cooper shows a different ordering.

Figure 5 separates the aggregate result by task family. The ordering changes most clearly in Cooper, where P2P and Control match or exceed Merge and Central in mean completion; Ent keeps all four conditions closer, while Silo and Core separate the single-decision-maker conditions from P2P and Control. The model-specific values in Table 2 show the variation behind these family means.

**Observation 4.** Central exceeds P2P at every tested Core node count.

Mean completion is lower at 15 than at 9 Core nodes under Merge, Central, and P2P (Figure 6). Node count varies together with workload and dependency structure. Within each scale, Central exceeds P2P by 0.446, 0.390, 0.316, and 0.323 at 9, 11, 13, and 15 nodes. The plotted panel-bootstrap intervals remain above zero, as do the task-clustered regression intervals and task-level bootstrap

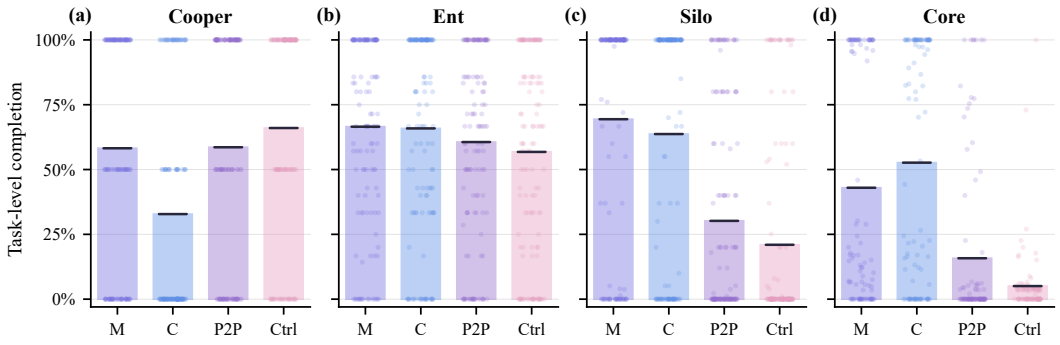



Figure 5: **Completion across task families.** Bars show mean completion and points show individual model–task results under Merge (M), Central (C), P2P, and Control (Ctrl).

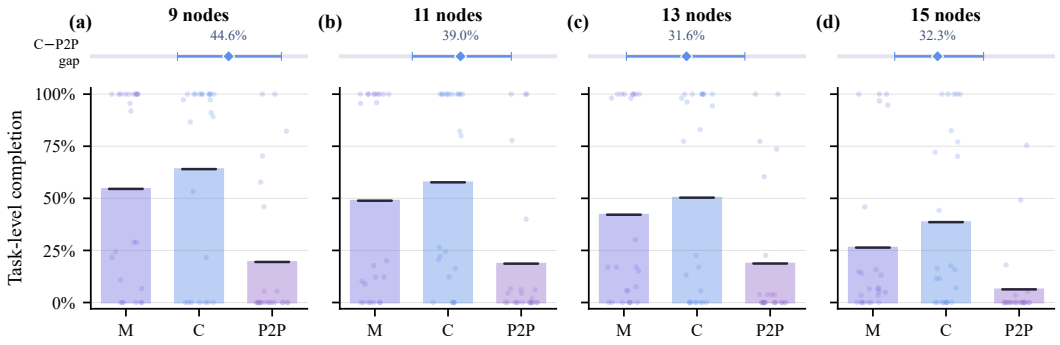



Figure 6: **Core node scaling.** Bars show mean completion and points show the 24 matched model–task results at each node count. The upper strips show paired Central–P2P completion differences with 95% panel-bootstrap intervals.

checks (Table 13; Appendix B.7). The Central–P2P gap persists across the tested scales despite their different workloads and dependencies.

**What single-decision execution preserves.** Table 2 provides the aggregate reference for this comparison. The matched comparisons locate where completion differs between single and local decision histories. In 374/480 panels, Merge and Central both match or exceed P2P; both strictly exceed it in 155 panels, whereas P2P strictly exceeds both in 27. The two single-decision references give different comparisons in the remaining 79 panels.

The contrast remains within tasks that the two single-decision conditions demonstrably solve: among 26 Core panels completed by both Merge and Central, P2P averages 0.498 completion, falls below them in 20, and completes nothing in 6. This is a gap within the jointly solvable subset.

The same history can also carry an invalid interpretation into later actions. Across 192 Core single trajectories, we count cases in which an early target mismatch is followed by an exact repetition of the rejected action, no valid recovery, and no subsequent business progress. Mismatch exposure is almost identical in Merge and Central, yet the strict sequence occurs in 22/96 Merge trajectories and 12/96 Central trajectories; the discordant matched pairs split 14–4 (exact McNemar $p = .031$; Tables 11 and 12). This combination of execution continuity and path-dependent error motivates the next analysis: we examine how autonomous agents interact to turn their local work into a joint result.

### 4.2 Cooperation dynamics in autonomous MAS

Under P2P, Relay, Role, and Control, we examine how local contributions enter other agents' actions and become joint outcomes during execution.

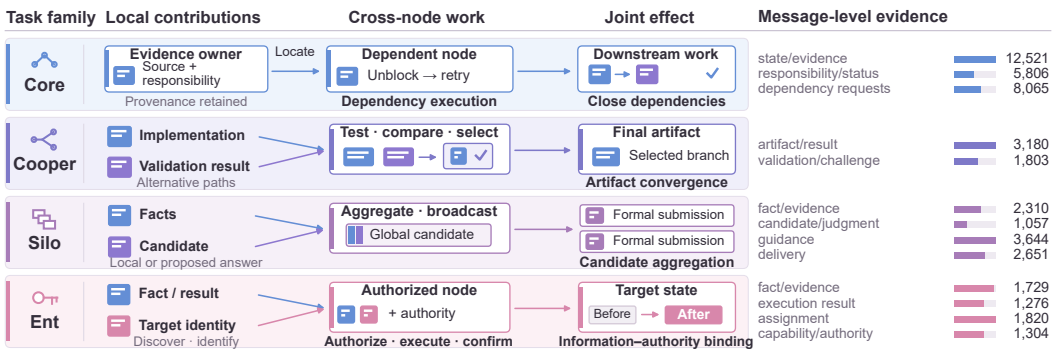



Figure 7: **Task-native cooperation.** Message counts aggregate all models and autonomous MAS modes. Categories may overlap; bar lengths are scaled within each task family.

The tasks require four kinds of cross-node work: artifact convergence in Cooper, candidate aggregation in Silo, dependency execution in Core, and information–authority binding in Ent (Figure 7). They share a reception-to-action chain, but differ in the contributions that must cross it.

**Observation 5.** P2P, Relay, and Role have similar aggregate completion, while Control completes less despite making more decisions.

Table 3 shows that P2P, Relay, and Role lie within 0.007 mean completion, with 120, 119, and 121 native-evaluator successes. Control falls to 0.394 completion and 109 native-evaluator successes. Its mean token use (1.389M) and decision count (91.1) are also the largest, whereas Role exchanges the fewest messages (14.3 per trajectory). These resource measures do not enter the completion score; they distinguish the execution profiles behind the outcome.

Table 3: **Autonomous MAS outcomes and resources.** 480 runs per mode. Full counts native successes; resources show means, with medians for tokens/time.

| Metric | P2P | Relay | Role | Control |
|---|---|---|---|---|
| **Completion** | 0.430 | **0.437** | 0.435 | 0.394 |
| **Full** | 120 | 119 | **121** | 109 |
| Tokens (M) | **1.141** (0.457) | **1.221** (0.497) | **1.227** (0.438) | **1.389** (0.414) |
| Time (s) | **586** (226) | **563** (218) | **545** (199) | **603** (199) |
| Decisions | 80.0 | 84.4 | 85.4 | 91.1 |
| Messages | 19.8 | 20.5 | 14.3 | 19.4 |
| Tools | 80.1 | 84.4 | 82.2 | 82.8 |

**Observation 6.** No autonomous MAS organization dominates across models and task families.

In the 16 observed model–family strata, P2P and Role each rank first in 5, and Relay and Control each in 3 (Figure 8). These sample rankings vary by task. Averaged over models, Control and Role are nearly tied in Cooper (0.660 versus 0.656), whereas Control is lowest in Core and Silo; in Ent it is nearly tied with Relay (0.568 versus 0.569).

The execution stages distinguish action from lasting progress (Appendix B.5). Out of 480 trajectories per mode, P2P and Relay each reach task action in 163, with 113 and 103 reaching persistent progress. Role raises task-action entry to 186 but only 98 reach persistent progress. Control leaves 160 trajectories after a receiver decision but before observable uptake, and only 81 reach persistent progress. Completion effects reverse in the same way: relative to P2P, Control changes completion by $+0.074$ in Cooper and $-0.107$ in Core, while Role changes it by $+0.070$ in Cooper and $-0.039$ in Silo. Thus, more action entry or explicit support does not uniformly coincide with more persistent progress or higher completion.

**Observation 7.** Errors are not automatically eliminated through cooperation; subsequent action and internal convergence can both preserve incorrect contributions.

In Silo, with a post-receive decision, 72/76 correct and 25/28 incorrect candidate exposures are followed by matching submissions. Table 9 reports the full accounting: across all 49 incorrect exposures, 26 match submissions, 23 end without submission, and none yields a correct submission

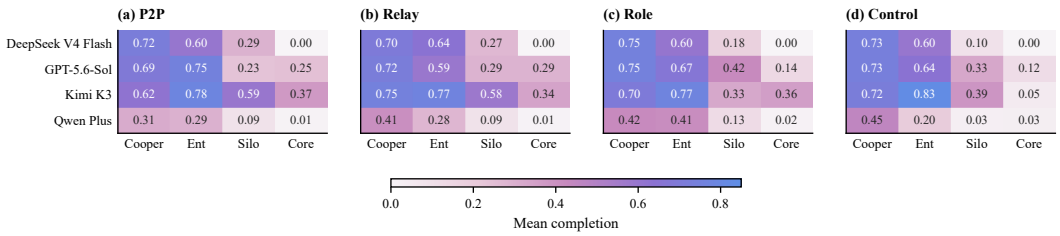



Figure 8: **MAS heterogeneity.** Each cell shows mean completion over 32 tasks for Cooper, Ent, and Silo, and 24 for Core.

(Appendix B.5). In Cooper, 151 of 351 internally converged artifacts still fail evaluation. Convergence means identical patches or a cleanly combined tree among 480 main-task MAS runs.

**The gap after reception.** Figure 7 summarizes the task-native relations that must survive reception. The trajectories locate three distinct failures after a contribution reaches another agent. Selection fails when continuation or agreement substitutes for a correctness check, as Observation 7 shows. Binding fails when the relation that makes a contribution executable is not preserved. In Core Control, forwarding assigns a source inconsistent with 791 previously valid evidence references across 21/96 trajectories. This failure involves the runtime source-assignment rule (Appendix B.5). In one Ent task, three of four Control models replace the target identity with a locally executable object, producing seven persistent state changes outside the target. Losing provenance makes correct evidence unusable; losing identity redirects execution to the wrong object.

Closure fails when an activated contribution starts work without carrying its downstream obligations. In Core, Control completes the root obligation in 49/96 trajectories versus 34/96 for P2P. The ordering reverses at 25% business work (7 versus 19) and at final closure (1 versus 7). Control starts more local work but closes fewer of the dependencies that work creates. Agents often receive a contribution and continue acting; the main gap is converting activation into correct, executable, closed progress.

**Value before interaction.** Complete artifacts can retain value on independent paths (Figure 7). Across 64 MAS main-task mode-runs that strictly exceed both single-decision references in Cooper, 58 win through preserving, converging, or selecting among complete execution paths. In 12 audited lead-only strict wins, the selected branch owner (agent1) supplies the final artifact through the adapter's fallback; none retains a peer artifact (Appendix B.5). These are wins from branch selection, not evidence of peer correction. In Silo, receiver inactivity similarly contains errors without correcting them. Independent paths can preserve an alternative result and limit the reach of a local error even before cooperation improves either path.

**Reconnecting paths.** Independent paths must reconnect to produce a joint result (Figure 7). The observed handoffs and downstream errors highlight the importance of validation at this transition.

Core also provides successful reconnection examples. Among all 384 MAS trajectories, 59 contain direct peer information followed by a receiver action change and a legal state change. These chains provide node-exclusive feedback while retaining the target, producer provenance, and executor authority needed for subsequent joint work.

## 5 Conclusion

CrossMAS studies how node-owned observations, state, and authority become joint completion. Matched comparisons across four task families and six organizations compare merged access, centralized decisions, and local cooperation. Single-decision execution leads overall, while organizational support yields no consistent gain. MAS can preserve alternatives, but turning local contributions into joint completion requires validation and preservation of targets, provenance, authority, and downstream obligations. Tracking contributions through actions to evaluated outcomes reveals where that transition succeeds and where cooperation breaks down.

### ETHICS STATEMENT

All experiments are conducted in controlled benchmark environments and involve no human subjects or personal data. Agent actions affect only task-local states and artifacts. The observed failures highlight risks in autonomous multi-agent execution, including error propagation and actions on incorrect task objects. Deployments in consequential settings should therefore retain least-privilege access, independent validation, human oversight, and auditable execution logs.

### AI USE STATEMENT

Beyond the language models evaluated in our experiments, we used generative AI tools to assist with research framing, analysis and visualization code, qualitative trajectory analysis, literature search, and manuscript drafting and editing. GPT-6-Astra mapped native evaluation evidence to completion-atom labels and annotated the 23,464 receiver–decision episodes. Analysis scripts computed completion scores and aggregate statistics from the recorded labels. Human verification is described in Appendix B.4. The authors inspected the experiment configurations, source trajectories, numerical results, citations, figures, and AI-assisted text, made all final research and editorial decisions, and take responsibility for the complete content of this work.

### REPRODUCIBILITY STATEMENT

Appendix B documents task construction, organization conditions, execution and scoring procedures, trajectory annotation, and human verification. The supplementary material provides the task configurations, organization protocols, evaluator and annotation specifications, frozen metrics and process annotations, and analysis scripts used to reconstruct the reported results.

## REFERENCES

Summer Eunhyung Ann, Haokun Liu, and Chenhao Tan. The interaction tax: When communication erases diversity in multi-agent teams. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2608.23541.

Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4): 819–840, 2002. doi: 10.1287/moor.27.4.819.297.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A. Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. doi: 10.52202/085713-4082. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b1041e52d3be19f0a9bc491657488e4a-Paper-Datasets_and_Benchmarks_Track.pdf.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. ChatEval: Towards better LLM-based evaluators through multi-agent debate. In *International Conference on Learning Representations*, pp. 9079–9093, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/25cc3adf8c85f7c70989cb8a97a691a7-Paper-Conference.pdf.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations*, pp. 20094–20136, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/578e65cdee35d00c708d4c64bce32971-Paper-Conference.pdf.

Seonglae Cho and Donghyun Lee. AgentRoom: Concurrent multi-agent coding in a CRDT-backed shared workspace, 2026. URL https://arxiv.org/abs/2608.23740.

Tim R. Davidson, Adam Fourney, Saleema Amershi, Robert West, Eric Horvitz, and Ece Kamar. The collaboration gap: Exploration and benchmarking of open-world agentic cooperation. In *Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2511.02687.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-One: A generalist multi-agent system for solving complex tasks, 2024. URL https://arxiv.org/abs/2411.04468.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, pp. 23247–23275, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6507b115562bb0a305f1958ccc87355a-Paper-Conference.pdf.

Md. Ashraful Islam, Mohammed Eunus Ali, and Md Rizwan Parvez. MapCoder: Multi-agent code generation for competitive problem solving. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4912–4944. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.269. URL https://aclanthology.org/2024.acl-long.269/.

Arpandeep Khatua, Hao Zhu, Peter Tran, Arya Prabhudesai, Frederic Sadrieh, Johann K. Lieberwirth, Xinkai Yu, Yicheng Fu, Michael J. Ryan, Jiaxin Pei, and Diyi Yang. CooperBench: Why coding agents cannot be your teammates yet, 2026. URL https://arxiv.org/abs/2601.13295.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae Won Park. MDAgents: An adaptive collaboration of LLMs for medical decision-making. In *Advances in Neural Information Processing Systems*, volume 37, pp. 79410–79452. Curran Associates, Inc., 2024. doi: 10.52202/079017-2522. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/90d1fc07f46e31387978b88e7e057a31-Paper-Conference.pdf.

Yubin Kim, Ken Gu, Chanwoo Park, Chunjong Park, Samuel Schmidgall, A. Ali Heydari, Yao Yan, Zhihan Zhang, Yuchen Zhuang, Yun Liu, Mark Malhotra, Paul Pu Liang, Hae Won Park, Yuzhe Yang, Xuhai Xu, Yilun Du, Shwetak Patel, Tim Althoff, Daniel McDuff, and Xin Liu. Towards a science of scaling agent systems, 2025. URL https://arxiv.org/abs/2512.08296.

Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978. doi: 10.1145/359545.359563. URL https://lamport.azurewebsites.net/pubs/time-clocks.pdf.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51991–52008. Curran Associates, Inc., 2023. doi: 10.52202/075280-2264. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a3621ee907def47c1b952ade25c67698-Paper-Conference.pdf.

Yuxuan Li, Aoi Naito, and Hirokazu Shirado. Systematic failures in collective reasoning under distributed information in multi-agent LLMs. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2505.11556.

Tie Ma, Yixi Chen, Vaastav Anand, Alessandro Cornacchia, Amândio R. Faustino, Guanheng Liu, Shan Zhang, Hongbin Luo, Suhaib A. Fahmy, Zafar A. Qazi, and Marco Canini. MAESTRO: Multi-agent evaluation suite for testing, reliability, and observability, 2026. URL https://arxiv.org/abs/2601.00481.

Aneesh Pappu, Batu El, Hancheng Cao, Carmelo di Nolfo, Yanchao Sun, Meng Cao, and James Zou. Multi-agent teams hold experts back. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.01011.

Joon Sung Park, Joseph C. 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. ACM, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.810. URL https://aclanthology.org/2024.acl-long.810/.

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. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Gleb Vitalevich Solovev, Alina Borisovna Zhidkovskaya, Anastasia Orlova, Nina Gubina, Anastasia Vepreva, Rodion Golovinskii, Ilya Tonkii, Ivan Dubrovsky, Ivan Gurev, Dmitry Gilemkhanov, Denis Chistiakov, Timur A. Aliev, Ivan Poddiakov, Galina Zubkova, Ekaterina V. Skorb, Vladimir Vinogradov, Alexander Boukhanovsky, Nikolay Nikitin, Andrei Dmitrenko, Anna Kalyuzhnaya, and Andrey Savchenko. MADD: Multi-agent drug discovery orchestra. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 6956–6998. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.367. URL https://aclanthology.org/2025.findings-emnlp.367/.

Haoyang Su, Renqi Chen, Shixiang Tang, Zhenfei Yin, Xinzhe Zheng, Jinzhe Li, Biqing Qi, Qi Wu, Hui Li, Wanli Ouyang, Philip Torr, Bowen Zhou, and Nanqing Dong. Many heads are better than one: Improved scientific idea generation by a LLM-based multi-agent system. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28201–28240. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1368. URL https://aclanthology.org/2025.acl-long.1368/.

Weixiang Sun, Zehong Wang, Hong Huang, Colby Nelson, Yijun Ma, and Yanfang Ye. The collaboration tax: How much LLM multi-agent systems pay to coordinate. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026. URL https://arxiv.org/abs/2608.22152.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Y. Zou. Mixture-of-agents enhances large language model capabilities. In *International Conference on Learning Representations*, pp. 33944–33963, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5434be94e82c54327bb9dcaf7fca52b6-Paper-Conference.pdf.

Qineng Wang, Zihao Wang, Ying Su, Hanghang Tong, and Yangqiu Song. Rethinking the bounds of LLM reasoning: Are multi-agent discussions the key? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6106–6131, 2024. doi: 10.18653/v1/2024.acl-long.331. URL https://aclanthology.org/2024.acl-long.331/.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. In *Proceedings of the First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, Kurt Keutzer, See-Kiong Ng, and Jiashi Feng. MAgIC: Investigation of large language model powered multi-agent in cognition, adaptability, rationality and collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7315–7332. Association for Computational Linguistics,

2024. doi: 10.18653/v1/2024.emnlp-main.416. URL https://aclanthology.org/2024.emnlp-main.416/.

Hanqing Yang, Narjes Nourzad, Shiyu Chen, Marie Siew, Jingdi Chen, and Carlee Joe-Wong. COOP$^2$: Defining, observing, and repairing cooperation in LLM multi-agent systems, 2026. URL https://arxiv.org/abs/2603.00349.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Tao Yu, Hao Wang, Changyu Li, Shenghua Chai, Minghui Zhang, Zhongtian Luo, Yuxuan Zhou, Haopeng Jin, Zhaolu Kang, Jiabing Yang, YiFan Zhang, Xinming Wang, Hongzhu Yi, Zheqi He, Jing-Shu Zheng, Xi Yang, Yan Huang, and Liang Wang. Beyond the all-in-one agent: Benchmarking role-specialized multi-agent collaboration in enterprise workflows, 2026. URL https://arxiv.org/abs/2605.08761.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. ProAgent: Building proactive cooperative agents with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17591–17599, 2024. doi: 10.1609/aaai.v38i16.29710. URL https://doi.org/10.1609/aaai.v38i16.29710.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599, 2025. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Yuzhe Zhang, Feiran Liu, Yi Shan, Xinyi Huang, Xin Yang, Yueqi Zhu, Xuxin Cheng, Cao Liu, Ke Zeng, Terry Jingchen Zhang, and Wenyuan Jiang. Silo-Bench: A scalable environment for evaluating distributed coordination in multi-agent LLM systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026. URL https://arxiv.org/abs/2603.01045.

Jun Zhao, Can Zu, Xu Hao, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. LONGAGENT: Achieving question answering for 128k-token-long documents through multi-agent collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16310–16324. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.912. URL https://aclanthology.org/2024.emnlp-main.912/.

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 *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8580–8622. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.421. URL https://aclanthology.org/2025.acl-long.421/.

# A Discussion, Future Work, and Limitations

## A.1 Discussion and future work

**Cooperation as organizational capability.** Under a distributed execution contract, cooperation is more than an inference strategy or an ensemble of opinions. It is the mechanism by which information, state, and authority owned by different actors become jointly executable. This view shifts the unit of analysis from individual messages to the operational relations they establish across nodes. A cooperative system should therefore be judged by whether it can turn collectively available capability into a shared outcome, not simply by whether its agents communicate.

**Communication and correlated error.** Communication changes the dependency structure of an MAS. Before a judgment is shared, different actors may still pursue independent trajectories. Once that judgment becomes a premise for downstream action, their future behavior becomes coupled. We conjecture that error diffusion depends less on communication volume than on the point at which communicated information acquires execution authority. This makes the boundary between receiving information and acting on it a central object for both system design and safety analysis.

**Managing independence and interdependence.** Preserving independent paths can retain alternatives while important claims remain unresolved. These paths must eventually connect strongly enough to satisfy joint obligations. This motivates explicit rules for when contributions remain provisional, when they become actionable, and which execution relations must survive a handoff. More communication and greater solution diversity are both incomplete objectives without this transition.

**Future work.** These hypotheses invite causal interventions on validation gates, handoff semantics, and the assignment of action authority. Parameter- and representation-level analysis could examine how models encode source identity, trust, and responsibility. Specialized training through supervised fine-tuning, process supervision, or reinforcement learning could then target selection, validation, handoff, and closure as distinct capabilities. The same questions should also be studied under asynchronous execution, dynamic authority, node failures, and human–agent collaboration.

## A.2 Extended comparison with empirical studies of multi-agent collaboration

Table 4 locates CrossMAS among empirical studies by comparing their execution settings, organizational variation, role assumptions, process evidence, and primary analysis targets. Local observations and joint objectives also appear in decentralized partially observable decision processes (Bernstein et al., 2002). Our formalization specifies the observation, action, and communication interfaces used to study execution rather than a policy-optimization problem. Related organization designs also span software development, coding, and long-context QA (Qian et al., 2024; Islam et al., 2024; Zhao et al., 2024), evaluation, medicine, aggregation, games, and scientific discovery (Chan et al., 2024; Kim et al., 2024; Wang et al., 2025; Zhang et al., 2024; Su et al., 2025; Solovev et al., 2025). The Collaboration Gap, HiddenBench, and studies of self-organizing expert teams already establish that strong individual performance need not translate into effective cooperation. CrossMAS examines how transferred contributions enter valid actions, persistent progress, and joint outcomes under local access.

Table 4: **Empirical study comparison.** Execution setting, organization, roles, process evidence, and analysis target.

| Study | Task / execution setting | Organization contrast | Roles / decomposition | Process evidence | Primary analysis target |
|---|---|---|---|---|---|
| MAgIC (Xu et al., 2024) | Social deduction and game-theoretic interaction | Base agents and PGM enhancement | Game roles are task-defined | Social and cognitive metrics plus game outcomes | Cognition, rationality, and cooperation |
| MultiAgentBench (Zhu et al., 2025) | Six collaborative and competitive scenarios | Star, chain, tree, and graph topologies; cognitive planning | Scenario roles and coordination protocols are configured | Milestone, planning, communication, and task metrics | Broad collaboration and competition evaluation |

Continued on next page

**Table 4 continued**

| Study | Task / execution setting | Organization contrast | Roles / decomposition | Process evidence | Primary analysis target |
|---|---|---|---|---|---|
| MAST (Cemri et al., 2025) | Coding, math, and general-agent traces from seven MAS frameworks | Existing framework configurations | Roles follow the evaluated frameworks | Human and LLM failure labels over complete traces | Cross-system failure taxonomy |
| COOP$^2$ (Yang et al., 2026) | Tasks with verifiable cooperative requirements in two environments | Three communication structures and targeted repair | Cooperation is expressed through group plans and constraints | Plans, grounded progress, constraint failures, and repair events | Observing and repairing cooperation |
| Scaling Agent Systems (Kim et al., 2025) | Six agentic benchmarks | Single-agent and four canonical MAS architectures | Roles follow the architecture and task | Outcomes, tools, compute, and task and system factors | Scaling and architecture–task alignment |
| Collaboration Tax (Sun et al., 2026) | 32 solo-tractable two-player tasks with private information | Solo and decentralized two-agent cooperation | Fixed two-player relationship | Scores and a four-stage conversational cascade | Team-decentralization loss |
| Interaction Tax (Ann et al., 2026) | 11 verifier-scored optimization tasks | Independent generation, full-solution interaction, and critique under matched budgets | Independent proposal generation | Verifier scores and solution diversity and convergence | Diversity loss through interaction |
| MAESTRO (Ma et al., 2026) | Twelve MAS implementations across agentic frameworks and interaction patterns | Repeated variation of architecture, model, and tools | Roles follow each system implementation | Framework-agnostic traces, call graphs, latency, cost, and failures | Reliability and systems observability |
| Collaboration Gap (Davidson et al., 2026) | Collaborative maze solving with partial observations | Solo with complete or split maps; homogeneous or heterogeneous teams | Unguided natural communication | Task outcomes and interaction behavior | Individual capability versus cooperative performance |
| HiddenBench (Li et al., 2026) | Hidden-profile decisions with distributed information | Complete-information solo and distributed teams; communication protocols | Each agent holds part of the information | Information exploration, shared evidence, and final decisions | Collective reasoning under information asymmetry |
| Expert teams (Pappu et al., 2026) | Human-inspired and frontier ML tasks | Individual experts and self-organizing teams | Interaction is unconstrained; expertise varies | Team outcomes and use of expert judgments | Expertise identification versus effective use |
| Silo-Bench (Zhang et al., 2026) | Algorithmic tasks with partitioned inputs | Single and distributed execution | Input ownership follows the task | Formal submitted answers | Problem solving across information boundaries |
| CooperBench (Khatua et al., 2026) | Feature implementation in isolated repositories | Solo and collaborative coding | Local feature requirements and workspaces | Patches, integration, and feature tests | Cooperative software implementation |
| EntCollab-Bench (Yu et al., 2026) | Enterprise services, objects, and approvals | Role-local execution and collaboration | Task-defined context and authority | Service state and review outcomes | Enterprise task completion across local access |
| **CrossMAS** | Distributed execution contracts over node-owned observations, mutable state, and action authority across four task families | Six matched conditions spanning merged, centralized, peer, relay, responsibility-informed, and coordinator-supported execution | Task-level decomposition is unprescribed in P2P; capability guidance and coordination are explicit conditions | Message-to-decision-to-action chains linked to task-native states, artifacts, and evaluated outcomes | Organizational completion differences and contribution-to-outcome processes |

### A.3 LIMITATIONS

**Behavioral and training scope.** Our analysis uses externally observable execution records—messages, decisions, tool feedback, and state changes—rather than model parameters, internal representations, or training dynamics. The resulting mechanisms locate where cooperation succeeds or fails during execution; they do not identify how these behaviors are represented within a model. We also evaluate general-purpose models without task- or organization-specific supervised fine-tuning, so the results characterize their existing collaborative behavior rather than the performance ceiling after specialized training.

**Empirical scope.** The study covers four models, four task families, and six organizations, with one trajectory per model–task–organization combination. Repeated trials would further assess the stability of small completion differences. The observed execution relations—selection, provenance, authority, and closure— provide hypotheses for the targeted interventions described above.

**System scope.** The task contracts enforce node-local observations, mutable states, and action authority, while execution remains controlled and turn-based. The experiments do not cover asynchronous scheduling, adversarial nodes, network partitions, unbounded delays, or dynamic membership. CrossMAS consequently studies model-generated coordination under distributed execution contracts, not the fault tolerance of distributed protocols. Token and elapsed-time comparisons are matched within the same model and task; their absolute values remain dependent on the serving and execution environment.

# B Experimental Details

## B.1 Task construction and adaptation

We preserve each task's objective and evaluator while exposing its information, mutable state, and actions through node-local interfaces. Every local action is executed against the underlying task state, and the original evaluator reads the resulting artifact, submission, or service state. The 120 task configurations span four execution contracts (Table 5).

Core contains eight job-execution, eight replica-repair, and eight rollout-recovery configurations, crossing 9, 11, 13, and 15 nodes with two task termination variants. A root operation produces an input manifest, commit certificate, or release record. Downstream operations require predecessor evidence at their owning nodes; completion checks required work, readiness, commitment, and closure in the final state. Node count changes together with the amount and structure of this work. For the three adapted benchmarks, we map the benchmark's shards, repositories, services, and permissions to the same node interface without changing the requested outcome or its evaluator. We remove model-visible answers and preset communication routes while retaining local task requirements and action permissions. Each adapted family contains 30 base tasks and two structural variants; variants are distinct configurations, not repeated samples of the same execution. The detailed Silo candidate and Cooper artifact analyses use the 30 base tasks, giving 480 MAS runs per family (30 tasks $\times$ four models $\times$ four organizations).

## B.2 Models and common execution protocol

Table 6 separates the models that act in the task environments from the model used for trajectory annotation. Task-model calls do not set a common sampling seed or top-$p$ value. Core permits up to 16,384 output tokens; Ent task calls use temperature 0; the Cooper Control coordinator uses temperature 0. Remaining decoding fields follow the corresponding task-adapter defaults. For a fixed model–task panel, all six organization conditions use the same task-model ID and task adapter.

Table 5: **Task execution contracts.** The two additional configurations for each adapted benchmark are structural variants included in that task family.

| Family | Tasks | Nodes | Distributed execution | Completion evidence |
|---|---|---|---|---|
| **Core (ours)** | 24 | 9–15 | Node-owned state, actions, predecessor evidence, and joint closure | Required work, readiness, commitment, and closure predicates in final state |
| Silo-Bench | 32 | 5–10 | Sharded inputs, local computation, and formal submissions | Required scalar leaves in the submitted answer |
| CooperBench | 32 | 2 | Isolated writable repositories and evaluated code artifacts | Two requested feature suites on the delivered artifact |
| EntCollabBench | 32 | 3–5 | Role-local service context, business state, and action authority | Required object changes or supported role decisions |

Table 6: **Execution and annotation models.** Models used for task execution and trajectory annotation.

| Role | Paper name | API request ID |
|---|---|---|
| **Task execution** | GPT-5.6-Sol | `gpt-5.6-sol` |
| | DeepSeek V4 Flash | `deepseek-v4-flash` |
| | Kimi K3 | `kimi-k3` |
| | Qwen Plus | `qwen-plus-2025-12-01` |
| Trajectory annotation | GPT-6-Astra | `gpt-6-astra` |

Every condition receives the same public task objective. In the decentralized conditions, every worker receives that objective at initialization together with its node identity, local observations, local tools, and addressable peers. Private node state, peer workspaces, and peer action authority are not copied into its context. A worker can affect peer-owned state only through the exposed task and communication interfaces; workers cannot enter or take over a peer's runtime.

The task harness records model decisions, tool calls, messages, tool feedback, and persistent state changes as separate events. Tool failures are returned to the acting model as observations, and a natural-language completion claim does not override the task evaluator. The six conditions within a task family share the same family-specific execution boundary. Core and Silo use 1,800-second checkpoint segments, Cooper uses 3,600-second segments, and Ent uses 420-second request and 1,800-second cell segments. A continuation preserves the environment and appends events to the same logical trajectory; it is not counted as another trial. Because the native harnesses expose different units of work, we do not impose one cross-family limit on model decisions, tokens, or tool calls.

An author monitored running processes and reviewed execution reports to identify faults and assess whether execution should continue or stop. This operational monitoring accompanied the automated scheduling and stopping rules described below.

**Core scheduling and stopping.** Core uses a ready queue and checks the native evaluator after each decision. A tool call re-enqueues its actor; a message re-enqueues its sender and recipient; wait and finish make an actor passive until it is activated again. Evaluator success ends execution. All 576 Core runs use the same event-triggered continuation rule: collective inactivity, three repeated no-progress action signatures by an actor, or a no-progress message sequence of length three times the actor count triggers a system prompt to continue. The repeated-action signature includes the action and current task progress. Repetition counters are reset at this intervention. Renewed inactivity or no-progress repetition ends the run; API errors have a separate terminal status. The recorded endings are 79 successes, 385 persistent no-progress states, 109 persistent inactivity states, and three API errors. None is marked as budget-censored. These statuses are distinct from completion: the evaluator and completion schema read the final state reached by the run.

### B.3 Organization conditions

Each model–task pair is run under all six conditions. The task, model, underlying task operations, and evaluator remain fixed within the resulting panel; the conditions change where information resides, who makes task-level decisions, and how nodes coordinate. We abbreviate All-merge as Merge throughout the results.

Table 7: **Organization protocols.** Information, action, and coordination scope under each condition.

| Condition | Decision makers | Information and action scope | Coordination |
|---|---|---|---|
| **Merge** | One agent | Merged node observations and unified access to all task actions | No delegation or peer messaging |
| **Central** | One agent | Separated node interfaces and state; access to every node's actions | The agent selects and operates node interfaces directly |
| P2P | Local agents | Each agent observes local state and uses local actions | Direct messaging; no prescribed roles, routing, or work division |
| Relay | Local agents | Same local observations and actions as P2P | Verbatim forwarding of agent-selected messages |
| Role | Local agents | Local actions plus capability or responsibility information | Agents retain direct communication and choose the execution workflow |
| Control | Coordinator and local agents | Workers retain local actions; the coordinator has communication access but no task tools | All cross-node traffic is routed through the coordinator |

The implementation makes these differences visible at the agent interface. Merge supplies one agent with a node-namespaced snapshot of the initial local observations and access to every task tool. Central receives the union of node-local tool specifications, each tagged with its owning `target_node_id`; tool results and mutable state remain node-scoped. P2P workers choose their work division, message recipients, and content. Relay retains these choices and copies messages verbatim, without task reasoning, summarization, or reassignment. Core records relay metadata; Silo, Cooper, and Ent also describe the relay transport in their task prompts.

Role provides capability or responsibility information while preserving local tool ownership. Core and Ent disclose node roles or capabilities. Silo names a global integrator and local shard analysts; Cooper identifies each worker's feature implementation and verification responsibility. Agents choose how to carry out these functions and coordinate subsequent work. In Control, workers address a communication-only coordinator, which organizes work through messages while workers retain task tools, workspaces, and service authority.

Merge and Central provide diagnostic single-decision references with access to all task actions. Their global decision history relaxes the local-history constraint imposed on workers. Consequently, the comparisons concern these complete execution organizations, including their context and communication interfaces, rather than changing decision ownership in isolation.

Cooper evaluates both requested feature suites on the delivered artifact. Central publishes patches from its two node workspaces and calls the native evaluator in its cooperative setting, also used for MAS artifacts. Central and MAS therefore share artifact reconciliation, including the single-branch fallback; Merge delivers one workspace patch. The cooperative adapter can deliver an identical patch, a cleanly combined tree, or a selected single branch. We retain these routes in the process analysis so that selecting one branch is distinguishable from integrating peer work.

Cooper accounts for much of the aggregate Merge–Central gap. Across the other 352 model–task panels, Merge and Central average 0.611 and 0.615. The Core contrast also varies by model: excluding Qwen Plus, its mean Merge–Central difference is approximately +0.002 (Table 2). These decompositions motivate the task-specific analysis of artifact delivery and feedback-guided revision.

### B.4 Measurements and human verification

Each native evaluator first produces task-specific evidence from the executed artifact, submission, or persistent state. GPT-6-Astra applies a fixed family-specific schema to this evidence and the canonical trajectory, recording each required completion atom as passed or failed together with its supporting test, state field, submission position, or event. Equation 4 computes completion as the fraction of passed completion atoms. Cooper uses requested feature suites, Silo uses scalar submission leaves, Core uses required final-state predicates, and Ent uses requested business objects or role-resolved reviews. We first compute $c(\tau)$ for each trajectory, then average these scores with equal weight within the reported group; atoms are not pooled across tasks. A family mean covers its configurations and models, and an organization's overall mean covers all 480 model–task pairs. Thus

the overall mean weights the 120 configurations equally, not the four families equally. We separately record total model tokens, runner start-to-end time, model decisions, tool calls, and sent and received messages. In Table 3, Full counts trajectories that pass the task-family native evaluator, out of 480 runs per organization. This binary field is reported separately from graded completion, not obtained by thresholding $c(\tau)$ at 1. In Ent, business-object atoms check the requested object identity and fields against recorded tool evidence; approval atoms check the role-resolved reviews. Full retains the native evaluator's aggregate verdict rather than recomputing it from these atoms, so it need not coincide with $c(\tau) = 1$.

The analysis contains 2,880 trajectories: 120 task configurations, four task-execution models, and six conditions. This forms 480 complete model–task panels. Completion, total tokens, and wall time are present for every trajectory. GPT-6-Astra labels completion atoms and cooperation events from recorded evidence; it does not act in the task environments or execute the native tests. Full is the native evaluator's binary result, whereas graded completion is computed from the evidence-backed atom labels. Cooper has two feature-suite atoms, yielding completion values of 0, 0.5, or 1; other families have task-dependent numbers of required leaves, predicates, or objects.

A human author independently re-scored a stratified blind sample of 96 outcome trajectories, one from every model $\times$ task-family $\times$ condition cell, and obtained exact agreement on all completion atoms. All 2,880 trajectory summaries were screened. For process attribution, we returned to the complete trajectories of the candidate divergences, matched cases, and counterexamples used in the analysis, recording the relevant feedback, subsequent decisions, state changes, and final delivery.

For cooperation analysis, the same author checked 144 stratified outcome-blind episodes and 96 claim-bearing episodes against the underlying messages, decisions, and task-native state. Agreement below refers to the categorical and Boolean fields used by the analysis; free-form rationales were not scored for exact wording. Table 8 reports these checks.

Table 8: **Human annotation verification.** Verification of outcome and cooperation labels against recorded task evidence.

| Audit set | Reviewed | Exact agreement | Changed |
|---|---|---|---|
| **Task completion** | 96 | 96/96 (100%) | 0 |
| Stratified cooperation episodes | 144 | 144/144 (100%) | 0 |
| Claim-bearing cooperation episodes | 96 | 96/96 (100%) | 0 |

### B.5 Cooperation trajectory annotations

The cooperation analysis covers all 1,920 trajectories from P2P, Relay, Role, and Control. We align each message to the first subsequent model decision by its receiver and follow at most one additional receiver decision when an immediate response only routes the contribution onward. Messages consumed by the same receiver decision form one episode, preventing multiple simultaneous messages from receiving duplicate behavioral credit. A message with no later receiver decision forms an inactive episode. This procedure maps 35,502 messages to 23,464 receiver–decision episodes. This window measures immediate uptake and one routing continuation; information used only after later intervening decisions is outside this episode-level measure. Final retention is checked separately against the delivered result.

GPT-6-Astra applies a fixed episode schema to the canonical events and retains the supporting message, decision, action, and state-event identifiers for each label. Message content is multi-label: requests, state or evidence, methods or plans, assignments, candidates or artifacts, validation or challenge, status, and closure information can co-occur. Behavior is coded sequentially. Activation requires a post-receive model decision. Uptake requires that decision to use a transferred task object, identifier, evidence reference, candidate, artifact, or assignment; acknowledgment alone does not qualify. Execution requires a corresponding task action, changed target or responsibility, or relay grounded in the transferred object. Persistent progress requires a recorded change in task state, artifact, submission, dependency, or closure, and final retention requires the contribution to remain in the outcome read by the evaluator.

Validity is coded as valid, invalid, or unresolved and is determined only from task-native evidence. Core uses producer provenance, accepted evidence references, and persistent business state; Silo uses candidate ground truth and the formal submission; Cooper uses the evaluated artifact, patch route, and feature tests; Ent binds each action to its service object, identity, authority, and read-back state. Temporal proximity without a structural or content match is not counted as uptake or execution. The mechanism cases reported in the paper are manually checked by returning to the underlying messages, receiver decisions, task actions, and final evaluator state.

The trajectory-stage counts in Section 4.2 record whether a run reaches each stage at least once, out of 480 trajectories per organization. A receiver decision without uptake is activation only; it does not show that the incoming task object informed execution. Counts of messages, episodes, trajectories, and final task outcomes therefore use different units.

**Core source and progress relations.** For coordinator messages, the runtime assigns the original-source field from the last incoming node sender and validates each cited evidence reference against that source. Across 96 Control runs, 791 forwarding rejections in 21 runs concern references previously accepted when received by the coordinator. These are reference-level events under this source-assignment rule, not 791 independent task failures. The count does not estimate how many tasks would succeed under a different forwarding rule. Root completion means executing the initial manifest, certificate, or release operation. Work coverage is completed required work items divided by all such items; joint closure additionally requires the final readiness, commitment, and closure predicates. Direct-peer chains are counted among all 384 Core MAS runs, rather than interpreted as an adoption rate conditional on peer exposure.

**Silo candidate-exposure counts.** The candidate analysis identifies 178 recipient–candidate exposures across 480 main-task MAS runs. An exposure is a distinct candidate asserted in a message received before that recipient's final submission; repeated mentions of the same candidate to the same recipient within a run count once. Correctness is checked against that recipient's required output. Adoption records a match between the candidate and the recipient's final submission. A post-receive decision lies between receipt and submission, or after receipt when no submission occurs. Among exposures with such a decision, 72/76 correct and 25/28 incorrect candidates match submissions.

Table 9: **Silo error-exposure accounting.** The 49 incorrect recipient–candidate exposures, grouped by post-receive decision availability. No exposure ends in a correct submission.

| Post-receive decision | Exposures | Matching submission | No submission |
|---|---|---|---|
| Yes | 28 | 25 | 3 |
| No | 21 | 1 | 20 |
| **Total** | **49** | **26** | **23** |

Thus, 25 counts matching submissions with an intervening decision, while 26 also includes one matching submission without a new decision between receipt and submission. Of the 23 exposures ending without submission, 20 have no post-receive decision and three have one. These outcome matches and decision counts are recorded separately; a match alone does not establish that the received message changed the receiver's answer.

**Cooper lead-only artifacts.** In the 12 audited lead-only strict wins, the lead label identifies agent1, whose branch the adapter selects through its `solo-agent1` fallback. This designation describes final artifact selection. Agents still form their task-level work arrangements under the corresponding organization condition. The audit checks whether peer-specific solution content enters the selected branch: all 12 final artifacts come from agent1, with no retained peer artifact. In the 480 main-task MAS runs, 351 artifacts are internally converged: the workers produce identical filtered patches or their patches form a clean combined tree. Of these, 200 pass the native evaluator and 151 fail. This definition separates artifact compatibility from functional correctness.

### B.6 PROCESS AUDITS FOR THE SINGLE-DECISION CONTRASTS

The Merge–Central audit covers every non-tied panel: 179 matched panels and 358 trajectories. Each comparison holds the model and task fixed. We extract the execution fields native to each family and locate the earliest axis on which the two trajectories differ in a completion atom. Table 10 reports the resulting audit; these family-specific labels retain their own semantics and denominators. An earliest divergence is an observed difference between paired trajectories, not a counterfactual estimate of the isolated effect of that event.

For Cooper, the extracted record contains the evaluated artifact, patch route, merge status, feature tests, and runtime or evaluator errors. Silo contributes the formal submission location, missing or wrong leaves, candidate correctness, and output envelope. Core contributes business-state paths, work and closure coverage, tool name, node, arguments, error feedback, and later successful retries. Ent contributes business calls, object identities, catalog or file reads, decisions, state changes, and approval evidence.

Table 10: **Merge–Central divergence.** $\Delta$ is the paired mean completion difference (Merge$-$Central); W/T/L denotes Merge wins/ties/losses against Central across 128 panels per external family and 96 in Core. Process evidence covers all 179 non-tied panels.

| Family | $\Delta$ | W/T/L | Execution requirement | Earliest outcome-relevant divergence |
|---|---|---|---|---|
| **Cooper** | **+.254** | 44/78/6 | Converged evaluated artifact | Artifact convergence 34/50; evaluator/runtime 2/50; implementation semantics 14/50 |
| Silo | +.057 | 22/93/13 | Complete formal submission | Submission coverage 19/35; candidate correctness 16/35 |
| Ent | +.007 | 23/82/23 | Context-resident actions | No process variable favors one condition consistently |
| **Core** | **−.097** | 12/48/36 | Feedback-driven dependencies | Business-work coverage 42/48; closure 5/48; other 1/48; mismatch recovery is 28/62 for Merge and 35/64 for Central |

The 179 non-tied panels comprise 50 Cooper, 46 Ent, 35 Silo, and 48 Core panels; Merge wins 101 and Central wins 78. Ent's non-ties comprise 28 business, 13 approval, and five external outcomes. All non-tied panels receive a task-native divergence label, without pooling the family-specific axes.

**Ent context and resource use.** The context-residency proxy sums, over actors, returned service-catalog characters multiplied by that actor's decision count. In the 60-panel process subset, 57 pairs have positive P2P/Merge token and residency ratios. Their Spearman correlation is 0.938; among the 26 usable pairs with equal completion it is 0.907. This is a resource association for the P2P–Merge contrast, separate from the Merge–Central outcome audit.

**Repeated invalid action after explicit feedback.** Core exposes a direct environmental contradiction through `target_mismatch`, along with the tool, node, arguments, persistent business state, and later recovery. In an exploratory trajectory analysis, we apply the following strict rule to all 192 Core single trajectories:

$$\text{early mismatch} \ \wedge \ \text{exact action repeated} \ \wedge \ \text{not recovered} \ \wedge \ \text{no subsequent business progress}.$$

An exact repetition matches the tool, node, and arguments of the rejected action. An early mismatch precedes every business-state creation. Recovery requires a later valid execution of the corresponding tool at that node, and progress is measured from persistent business state rather than the agent's self-report.

Table 11: **Repeated invalid-action screening.** All Core single trajectories after explicit target feedback. Early mismatch and repeated action are separate filters within the 126 exposed trajectories.

| Screening criterion | Trajectories |
|---|---|
| All Core Merge and Central trajectories | 192 |
| Explicit `target_mismatch` | 126 |
| Early mismatch among the 126 exposed | 98 |
| Exact repetition among the 126 exposed | 45 |
| Both early mismatch and exact repetition | 42 |
| Later recovered among these 42 | 8 |
| **All four conditions satisfied** | **34** |

All 63 mismatch-exposed trajectories that never recovered also produced no new business state after the first contradiction. The strict rule selects the 34 that additionally encountered the contradiction before any progress and later repeated the same invalid action. Labels are determined from structured event fields; key matched cases were checked against complete trajectories, and a 20-case field-to-label consistency check obtained exact agreement.

Table 12: **Paired persistent-error occurrence.** Occurrence of all four conditions across 96 Core model–task pairs (24 tasks $\times$ four models). The discordant directions are 14 Merge-only versus 4 Central-only pairs; the two-sided exact McNemar test gives $p = .0309$.

| | Central: yes | Central: no | Total |
|---|---|---|---|
| Merge: yes | 8 | 14 | 22 |
| Merge: no | 4 | 70 | 74 |
| **Total** | **12** | **84** | **96** |

### B.7 Complete outcome and resource results

There is one recorded trajectory per model–task–organization combination. The panel-bootstrap intervals resample matched model–task differences. The regression in Table 13 uses 24 task clusters and a $t_{23}$ reference distribution; each node count contains six task configurations evaluated with four models. Its point estimates equal the raw mean gaps in this balanced design. A task-level bootstrap that keeps the four models together gives 95% intervals of [0.260, 0.635], [0.292, 0.499], [0.131, 0.522], and [0.175, 0.467] at 9, 11, 13, and 15 nodes, respectively (100,000 resamples; seed 20260926). All remain positive. Both calculations summarize variation across the evaluated tasks; repeated executions would measure within-task sampling variation.

Figure 9 shows every task-level completion value rather than only the aggregate means used in the main text. Figures 10 and 11 summarize the corresponding model-, task-family-, and node-scale patterns. Tables 14–19 provide the exact completion, token, and time values.

Table 13: **Regression-adjusted Core gaps.** Central–P2P completion gaps in Core. The OLS model includes organization-by-node-count interactions and fixed effects for model, task kind, and termination protocol. Standard errors are clustered by task configuration (192 trajectories; 24 clusters); intervals and tests use a $t_{23}$ reference distribution.

| Nodes | Gap | Clustered SE | 95% CI | $p$ |
|---|---|---|---|---|
| 9 | **0.446** | 0.104 | [0.231, 0.661] | $< .001$ |
| 11 | 0.390 | 0.058 | [0.271, 0.510] | $< .001$ |
| 13 | 0.316 | 0.106 | [0.096, 0.536] | .007 |
| 15 | 0.323 | 0.081 | [0.156, 0.489] | $< .001$ |

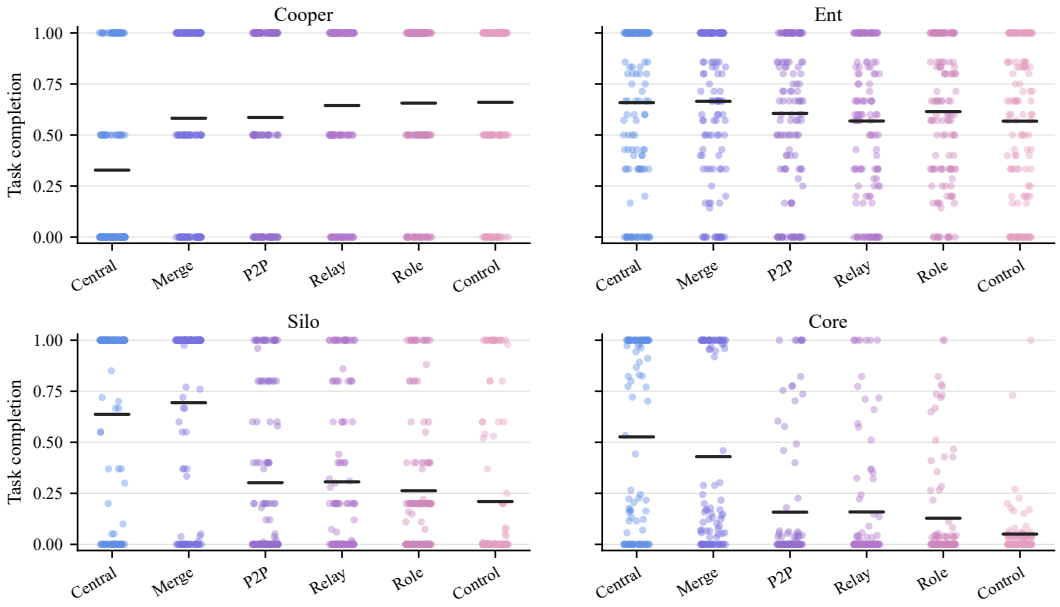



Figure 9: **Task-level completion distributions.** Each point is one model–task result under the indicated condition (128 points per condition in Cooper, Ent, and Silo; 96 in Core). Horizontal black marks show means; horizontal jitter is visual only.

Table 14: **Completion by model, family, and mode.** Mean completion under each execution mode. Bold marks the highest completion in each model–family row.

| Task family | Model | Central | Merge | P2P | Relay | Role | Control |
|---|---|---|---|---|---|---|---|
| **Cooper** | DeepSeek V4 Flash | 0.500 | 0.734 | 0.719 | 0.703 | **0.750** | 0.734 |
| | GPT-5.6-Sol | 0.406 | **0.781** | 0.688 | 0.719 | 0.750 | 0.734 |
| | Kimi K3 | 0.359 | 0.641 | 0.625 | **0.750** | 0.703 | 0.719 |
| | Qwen Plus | 0.047 | 0.172 | 0.312 | 0.406 | 0.422 | **0.453** |
| **Ent** | DeepSeek V4 Flash | 0.658 | **0.703** | 0.599 | 0.641 | 0.604 | 0.598 |
| | GPT-5.6-Sol | 0.701 | 0.680 | **0.751** | 0.589 | 0.670 | 0.641 |
| | Kimi K3 | 0.822 | 0.824 | 0.780 | 0.766 | 0.771 | **0.831** |
| | Qwen Plus | 0.454 | **0.455** | 0.294 | 0.278 | 0.415 | 0.202 |
| **Silo** | DeepSeek V4 Flash | 0.406 | **0.638** | 0.292 | 0.272 | 0.177 | 0.096 |
| | GPT-5.6-Sol | 0.773 | **0.861** | 0.234 | 0.287 | 0.423 | 0.328 |
| | Kimi K3 | 0.770 | **0.802** | 0.592 | 0.576 | 0.325 | 0.386 |
| | Qwen Plus | **0.598** | 0.475 | 0.090 | 0.090 | 0.126 | 0.029 |
| **Core** | DeepSeek V4 Flash | 0.222 | **0.444** | 0.002 | 0.000 | 0.001 | 0.000 |
| | GPT-5.6-Sol | **0.649** | 0.515 | 0.245 | 0.289 | 0.138 | 0.118 |
| | Kimi K3 | **0.749** | 0.667 | 0.373 | 0.339 | 0.358 | 0.053 |
| | Qwen Plus | **0.486** | 0.093 | 0.010 | 0.007 | 0.015 | 0.031 |

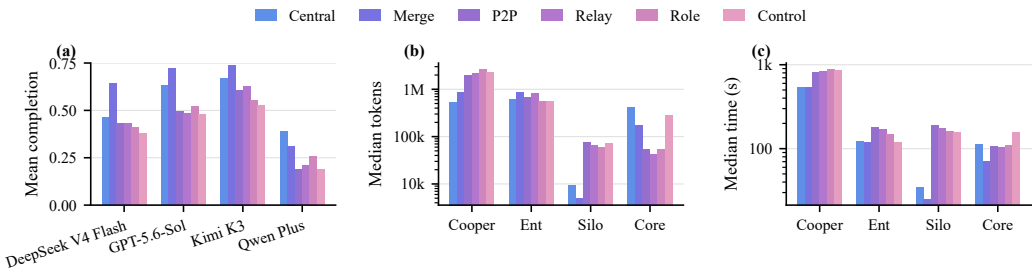



Figure 10: **Overall outcome and resource patterns.** Aggregate patterns across all six organizations. (a) Mean completion by model. (b–c) Median total tokens and wall time by task family, pooling matched model–task panels within each family. Resource panels use logarithmic axes.

Table 15: **Token use by model, family, and mode.** Median total tokens.

| Task family | Model | Central | Merge | P2P | Relay | Role | Control |
|---|---|---|---|---|---|---|---|
| **Cooper** | DeepSeek V4 Flash | 1,496,624 | 1,438,660 | 3,041,984 | 3,494,608 | 3,782,116 | 3,465,136 |
| | GPT-5.6-Sol | 204,176 | 248,658 | 441,394 | 448,846 | 607,374 | 493,950 |
| | Kimi K3 | 575,732 | 1,159,614 | 2,601,190 | 2,469,772 | 3,402,360 | 2,539,836 |
| | Qwen Plus | 474,187 | 802,670 | 1,308,057 | 1,797,028 | 1,916,517 | 1,428,890 |
| **Ent** | DeepSeek V4 Flash | 657,028 | 914,976 | 1,285,200 | 1,043,058 | 675,412 | 1,034,901 |
| | GPT-5.6-Sol | 737,413 | 823,935 | 1,798,670 | 1,585,804 | 593,882 | 1,116,382 |
| | Kimi K3 | 566,106 | 782,025 | 508,417 | 573,900 | 398,029 | 516,518 |
| | Qwen Plus | 537,778 | 570,590 | 215,513 | 421,686 | 611,759 | 79,716 |
| **Silo** | DeepSeek V4 Flash | 14,904 | 4,608 | 146,148 | 238,454 | 120,880 | 64,004 |
| | GPT-5.6-Sol | 6,098 | 4,680 | 29,207 | 38,239 | 46,737 | 59,686 |
| | Kimi K3 | 6,072 | 3,058 | 42,462 | 49,384 | 48,066 | 82,284 |
| | Qwen Plus | 16,228 | 9,766 | 115,836 | 115,606 | 54,189 | 50,131 |
| **Core** | DeepSeek V4 Flash | 109,212 | 191,796 | 47,750 | 38,072 | 50,958 | 8,584 |
| | GPT-5.6-Sol | 401,930 | 246,780 | 51,766 | 86,028 | 18,338 | 1,677,568 |
| | Kimi K3 | 581,658 | 564,068 | 1,495,428 | 1,531,905 | 1,421,412 | 704,698 |
| | Qwen Plus | 504,236 | 74,462 | 8,788 | 8,758 | 9,829 | 214,106 |

Table 16: **Wall time by model, family, and mode.** Median elapsed time in seconds.

| Task family | Model | Central | Merge | P2P | Relay | Role | Control |
|---|---|---|---|---|---|---|---|
| **Cooper** | DeepSeek V4 Flash | 549.1 | 367.6 | 885.1 | 848.3 | 793.6 | 836.2 |
| | GPT-5.6-Sol | 276.8 | 290.8 | 373.1 | 302.0 | 384.5 | 408.0 |
| | Kimi K3 | 1462.4 | 994.7 | 1733.8 | 1689.2 | 1692.9 | 1562.0 |
| | Qwen Plus | 522.1 | 734.6 | 675.5 | 844.0 | 880.5 | 844.2 |
| **Ent** | DeepSeek V4 Flash | 94.0 | 97.3 | 171.2 | 177.4 | 138.5 | 124.8 |
| | GPT-5.6-Sol | 132.1 | 154.8 | 244.1 | 240.1 | 141.5 | 156.8 |
| | Kimi K3 | 100.7 | 129.2 | 153.6 | 139.7 | 124.0 | 124.4 |
| | Qwen Plus | 130.2 | 124.4 | 144.4 | 163.0 | 191.3 | 96.8 |
| **Silo** | DeepSeek V4 Flash | 40.5 | 17.4 | 262.4 | 272.5 | 186.3 | 118.3 |
| | GPT-5.6-Sol | 27.3 | 20.5 | 108.9 | 135.4 | 189.7 | 253.3 |
| | Kimi K3 | 18.8 | 18.7 | 127.8 | 136.3 | 126.2 | 165.6 |
| | Qwen Plus | 93.6 | 102.3 | 227.0 | 189.2 | 166.5 | 102.2 |
| **Core** | DeepSeek V4 Flash | 35.3 | 60.5 | 79.3 | 60.2 | 72.8 | 13.4 |
| | GPT-5.6-Sol | 316.8 | 312.8 | 365.9 | 448.5 | 160.2 | 1959.7 |
| | Kimi K3 | 81.0 | 72.4 | 299.6 | 275.9 | 255.4 | 182.8 |
| | Qwen Plus | 153.2 | 34.6 | 10.3 | 10.1 | 10.7 | 128.7 |

Table 17: **Core completion by node count.** Mean completion by model and execution mode.

| Model | Nodes | Central | Merge | P2P | Relay | Role | Control |
|---|---|---|---|---|---|---|---|
| DeepSeek V4 Flash | 9 | 0.333 | 0.646 | 0.000 | 0.000 | 0.004 | 0.000 |
| | 11 | 0.333 | 0.319 | 0.003 | 0.000 | 0.000 | 0.000 |
| | 13 | 0.195 | 0.494 | 0.000 | 0.000 | 0.000 | 0.000 |
| | 15 | 0.026 | 0.319 | 0.006 | 0.000 | 0.000 | 0.000 |
| GPT-5.6-Sol | 9 | 0.981 | 0.755 | 0.342 | 0.378 | 0.242 | 0.267 |
| | 11 | 0.731 | 0.717 | 0.351 | 0.432 | 0.201 | 0.100 |
| | 13 | 0.679 | 0.440 | 0.280 | 0.340 | 0.091 | 0.079 |
| | 15 | 0.204 | 0.147 | 0.009 | 0.005 | 0.017 | 0.028 |
| Kimi K3 | 9 | 0.667 | 0.667 | 0.427 | 0.232 | 0.609 | 0.146 |
| | 11 | 0.833 | 0.833 | 0.373 | 0.469 | 0.216 | 0.018 |
| | 13 | 0.664 | 0.667 | 0.456 | 0.415 | 0.252 | 0.041 |
| | 15 | 0.833 | 0.500 | 0.238 | 0.240 | 0.355 | 0.006 |
| Qwen Plus | 9 | 0.580 | 0.113 | 0.009 | 0.000 | 0.024 | 0.033 |
| | 11 | 0.410 | 0.086 | 0.019 | 0.007 | 0.014 | 0.036 |
| | 13 | 0.475 | 0.085 | 0.013 | 0.016 | 0.013 | 0.031 |
| | 15 | 0.479 | 0.088 | 0.000 | 0.006 | 0.011 | 0.023 |

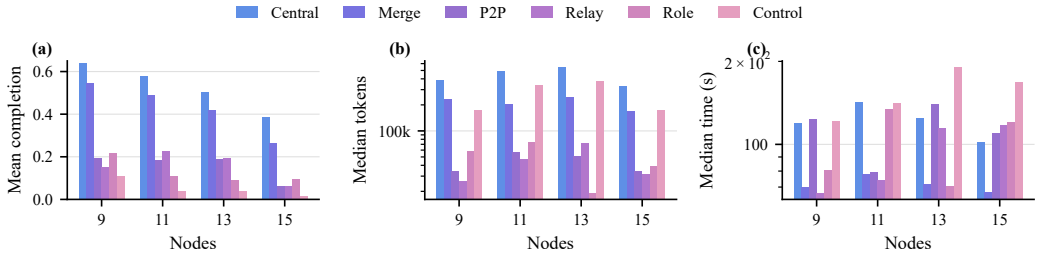



Figure 11: **Core scaling in completion and resources.** Each node count aggregates 24 matched model–task panels: mean completion in (a), and median total tokens and wall time in (b–c). Resource panels use logarithmic axes.

Table 18: **Core token use by node count.** Median total tokens by model and execution mode.

| Model | Nodes | Central | Merge | P2P | Relay | Role | Control |
|---|---|---|---|---|---|---|---|
| DeepSeek V4 Flash | 9 | 208,030 | 373,362 | 8,554 | 13,976 | 34,196 | 23,338 |
| | 11 | 98,010 | 132,502 | 51,256 | 31,801 | 57,435 | 8,362 |
| | 13 | 135,610 | 465,251 | 18,600 | 40,695 | 15,558 | 27,433 |
| | 15 | 37,786 | 168,220 | 63,502 | 69,860 | 69,592 | 9,630 |
| GPT-5.6-Sol | 9 | 401,930 | 310,718 | 945,834 | 919,936 | 493,200 | 4,574,503 |
| | 11 | 451,137 | 453,123 | 341,053 | 1,191,350 | 681,274 | 3,645,482 |
| | 13 | 569,680 | 374,031 | 163,090 | 1,008,986 | 9,573 | 1,677,568 |
| | 15 | 225,226 | 134,332 | 10,864 | 8,802 | 14,060 | 391,590 |
| Kimi K3 | 9 | 391,871 | 395,334 | 1,230,634 | 905,272 | 1,446,197 | 1,098,304 |
| | 11 | 555,988 | 564,970 | 1,369,704 | 1,200,887 | 1,255,040 | 402,946 |
| | 13 | 757,393 | 670,641 | 2,673,108 | 2,961,392 | 1,481,482 | 840,661 |
| | 15 | 1,010,148 | 656,170 | 1,336,353 | 1,644,338 | 1,136,038 | 822,012 |
| Qwen Plus | 9 | 406,694 | 84,336 | 8,368 | 8,368 | 26,736 | 83,898 |
| | 11 | 435,904 | 74,486 | 8,537 | 8,542 | 9,358 | 175,844 |
| | 13 | 625,786 | 71,146 | 10,824 | 12,524 | 9,584 | 464,359 |
| | 15 | 713,822 | 108,352 | 8,862 | 8,912 | 12,142 | 141,000 |

Table 19: **Core wall time by node count.** Median elapsed time in seconds by model and execution mode.

| Model | Nodes | Central | Merge | P2P | Relay | Role | Control |
|---|---|---|---|---|---|---|---|
| **DeepSeek V4 Flash** | 9 | 48.0 | 73.9 | 9.6 | 13.4 | 43.7 | 34.2 |
| | 11 | 32.0 | 39.4 | 79.3 | 44.9 | 80.5 | 12.7 |
| | 13 | 39.5 | 100.6 | 16.2 | 63.2 | 15.2 | 68.4 |
| | 15 | 28.9 | 60.5 | 109.7 | 113.4 | 107.1 | 9.5 |
| **GPT-5.6-Sol** | 9 | 318.5 | 340.3 | 1168.3 | 1347.0 | 1594.0 | 3034.3 |
| | 11 | 306.0 | 286.1 | 1362.5 | 1542.6 | 1927.0 | 3786.5 |
| | 13 | 372.7 | 374.5 | 804.9 | 1107.0 | 121.8 | 1959.7 |
| | 15 | 361.4 | 304.1 | 93.4 | 117.5 | 121.0 | 744.3 |
| **Kimi K3** | 9 | 58.6 | 61.0 | 235.6 | 184.2 | 270.0 | 208.0 |
| | 11 | 202.6 | 222.0 | 276.0 | 257.5 | 246.7 | 141.5 |
| | 13 | 81.0 | 72.4 | 352.7 | 362.5 | 267.3 | 176.2 |
| | 15 | 81.2 | 61.1 | 297.5 | 275.9 | 220.2 | 213.0 |
| **Qwen Plus** | 9 | 140.8 | 41.7 | 11.6 | 9.8 | 27.0 | 62.5 |
| | 11 | 129.5 | 28.6 | 12.2 | 9.9 | 11.0 | 91.3 |
| | 13 | 159.4 | 32.6 | 12.1 | 14.7 | 10.0 | 359.0 |
| | 15 | 170.0 | 34.7 | 9.4 | 11.3 | 11.2 | 154.0 |