ROUTING MESSAGES BY WHAT TEAMMATES KNOW FOR EMBODIED MULTI-AGENT COOPERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied agents built on large language models (LLMs) coordinate by exchanging natural-language messages, and each message must be delivered to some subset of teammates. Most existing methods choose these recipients from structure fixed in advance, such as roles, a communication graph, or a learned policy. However, how useful a message is to a teammate depends on what that teammate knows when the message is sent, and this changes at every step. We derive an approximation of a message's expected information gain for each recipient that depends on the recipient only through one fact: whether it already holds valid information about the goal-relevant entities the message mentions. Choosing the recipients of each message therefore reduces to tracking which agent knows about which entity, and the messages the agents exchange provide the evidence for this. We present CREAC (Cooperative Router for Embodied Agent Communication), a training-free router that infers from the team's messages which agents know about which entities, delivers each message only to the teammates for whom it is expected to be informative, and asks a teammate directly when it can no longer tell who knows. CREAC does not modify the agents' planners or prompts; it changes only which messages each agent receives. On household tasks with three to six agents and wildfire response with fifteen heterogeneous agents, CREAC outperforms broadcasting to all teammates in all ten combinations of planner, benchmark, and team size, reducing steps by up to 28% on household tasks and damage by 21–30% on wildfire response. The precision of its recipient selection increases with team size. In a user study, participants preferred teams that used CREAC over teams that broadcast every message and teams that did not communicate.

1 INTRODUCTION

Picture a team of fifteen engineers in the middle of a sudden outage. Every engineer is busy restoring their own part of the system. One of them traces some of the errors to a configuration change in the payment service. She can post this to the whole team channel, or she can send it to the two engineers who maintain the payment service. Which should she do?

Most readers will choose the second option, and the reasons behind that choice are the subject of this paper. First, the same message has a different value for each recipient. The two maintainers need the report to repair the payment service; the other thirteen must stop their own repairs, read it, and set it aside. Second, she can choose only because she knows who is working on what. Nobody gave her a list of responsibilities; she learned it from the channel itself, where the two engineers said earlier that they were taking over the payment service. Third, if she were unsure who had made the change, she would still not write to everyone. She would ask the one person most likely to know.

Teams of language model agents face the same choice every time an agent sends a message. We call this choice of recipients *routing*. A wrong choice in either direction affects the agents' decisions. An agent has to read a message before it can decide that the message does not help, and a message meant for someone else competes with the observations and requests that bear on the agent's own next action (Yan et al., 2025; Zhang et al., 2024a). Sending every message to every teammate, as in Zhang et al. (2023), therefore fills each agent's context with messages meant for others, and the problem grows with team size. Withholding a message has the opposite cost: the agent that needs it acts without the information.

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.

Existing methods address this problem in three ways. Planner-coupled systems such as CoTS (Zu et al., 2025) and REVECA (Seo et al., 2025) tie communication to joint planning or to repeated belief synchronization, so selective communication becomes part of the planner and adds to its cost. Learned targeted-communication policies (Liu et al., 2020; Ding et al., 2020) and fixed-topology methods (Zhang et al., 2024a;b; Liu et al., 2025) decide who talks to whom through a trained network, a communication graph, or an assignment of roles. These methods fix their structure in advance, but the value of a message depends on what its recipient knows when the message is sent, and this knowledge changes at every step. Decision-theoretic methods come closest to the engineer's reasoning. They send a message when its value to the team exceeds its cost (Roth et al., 2005; Becker et al., 2009), but they compute this value from joint beliefs over the team's states, which language agents, whose knowledge is stored in free text, do not provide. What is missing is a way to estimate, from the agents' messages alone, what each recipient currently knows.

We treat routing as the engineer did, as a decision about each recipient made at the moment a message is sent, and turn each of her three reasons into one part of our router. The first reason becomes the objective. A message m should go to a recipient i when its *expected information gain* for i is high, that is, when m would reduce i's uncertainty about what the shared goal depends on. This quantity is intractable for free-text messages. Cooperative agents, however, mostly send two kinds of message: reports of what the sender knows, and requests for what it does not know. For these two kinds, and in a team whose members do not withhold what they know, we approximate the gain by asking, for each entity the message mentions, whether the recipient already holds valid information about it. A report should therefore go to agents that do not, and a request to agents that do.

The second reason determines what the router must track. Research on transactive memory (Wegner, 1987) finds that teams coordinate well when each member knows who knows what. CREAC (Cooperative Router for Embodied Agent Communication) keeps this knowledge as a table over agents and goal-related entities, which we call its *meta-belief*. Each entry estimates whether an agent currently holds valid information about an entity. Like the engineer, CREAC fills this table by reading the messages the agents already send each other. The third reason becomes a fallback. When the meta-belief no longer indicates who knows about an entity, CREAC asks the agent most likely to know. CREAC only reads messages, chooses their recipients, and sends these queries, so it needs neither access to the agents' internal states nor training; it changes the agents' behavior only through the messages they receive. We keep the router simple, with three hyperparameters and no learned components, so that its gains can be attributed to the formulation rather than to model capacity.

We evaluate CREAC on Communicative Watch-And-Help (C-WAH) (Zhang et al., 2023), with three to six agents in household tasks, and on CREW-Wildfire (Hyun et al., 2025), with fifteen heterogeneous agents in wildfire response. Our contributions are as follows.

1. A reduction of message routing to tracking who knows what. We approximate the expected information gain of reports and requests by a quantity that depends on the agents only through whether each recipient holds valid information about each entity the message mentions.
2. CREAC, a training-free router that maintains this estimate from the agents' messages alone, without modifying the agents' planners or prompts.
3. A direct test of whether CREAC's delivery decisions help individual agents. Adding eight messages that CREAC withheld from an agent lowers the rate at which its next action makes progress by 10.4 points, and removing the messages CREAC delivered lowers it by 7.4 points.
4. CREAC performs best among the compared methods and outperforms broadcasting in all ten combinations of planner, benchmark, and team size, with up to 28% fewer steps on C-WAH and 21–30% lower damage on CREW-Wildfire. In a user study, participants preferred CREAC teams.

2 Related Work

Embodied large language model (LLM) multi-agent frameworks use language to handle partial observability, memory, dialogue, and joint planning (Zhang et al., 2023; Seo et al., 2025; Zu et al., 2025; Liu et al., 2024), but typically assume small teams, broadcast-style exchange, or planner-coupled communication. Decision-theoretic work decides at execution time whether a message is worth its cost, computing this value from joint beliefs in multi-agent partially observable Markov decision processes (POMDPs) (Roth et al., 2005; Becker et al., 2009); CREAC keeps this value-of-

information criterion but replaces the joint belief with a meta-belief over who knows which entity. Other methods reduce message volume with learned gates, attention, or targeted communication, or with graph constraints and role-based routing (Foerster, 2018; Das et al., 2019; Liu et al., 2020; Zhang et al., 2024a;b; Liu et al., 2025). These methods fix a trained policy, a topology, or a role prior in advance, whereas CREAC updates its estimate of who knows what at every step. To track where task-relevant knowledge resides in a team over time, CREAC draws on transactive memory, overhearing-based monitoring, and explicit belief tracking in LLM teams (Wegner, 1987; Kaminka et al., 2002; Li et al., 2023). Appendix A.1 gives an extended discussion.

3 Problem Definition

Environment. We model the team as a decentralized partially observable Markov decision process with communication (DEC-POMDP-COM) (Bernstein et al., 2002; Oliehoek et al., 2016; Spaan et al., 2006). Each of the n agents sees only part of the world. At each step it either acts in the world or sends a natural-language message $m_{t,i} \in \Sigma$ to its teammates. All agents share one goal G and a fixed set of goal-related entities $\mathcal{E}$, the entities whose state the goal specifies, such as the objects the goal asks the team to place. Messages can refer to these entities. The team receives a shared reward R_t, with discount factor γ and horizon h. Appendix A.2 gives the full specification.

Task objective. A router mediates all communication between the agents. It reads each message an agent sends, decides who should receive it, and delivers it. It may also send its own queries to individual agents. It never acts in the environment. The team objective is the standard cooperative return $J_{\text{task}}(\pi, \mu) = \mathbb{E}_{\pi,\mu}\left[\sum_{t=0}^{h-1} \gamma^t R_t\right]$, where π denotes the agents' policies and μ the router policy. We hold the task, the language model backbone, and the agent prompts fixed. The router can then change J_{task} only through the messages each agent receives: which teammates' messages reach it, and which queries the router sends it. Any difference in team return between CREAC and another communication policy is due to these two factors, and Section 6.4 separates their effects.

Assumptions. Our claim that selective routing raises J_{task} rests on three assumptions.

- **(A1) Messages differ in value across recipients.** A message sent at a given step has different expected information gain (Section 4.1) for different agents, because the agents have different observation histories and are working on different subtasks.
- **(A2) A message with low expected information gain is harmful.** Delivering such a message makes the recipient act worse. The message lengthens the context the agent has to read and competes with the observations and messages that matter for its decision.
- **(A3) Individual gains transfer to the team.** When individual agents act better, team return rises, as long as the router still delivers the messages that coordination requires.

Section 4 makes (A1) precise, and Section 6.3 tests it by comparing CREAC's recipient precision with that of delivering to every teammate. Section 6.4 tests (A2) at a fixed decision point and then in the team. Section 6.2 tests (A3) indirectly, through team performance.

Cooperative premise. We assume throughout that agents pursue the shared goal, ask only about what they do not know, answer a request with what they know when they can, and do not fabricate information they lack. This premise lets the router treat a message as evidence about what its sender knows. Agents that strategically withhold information are outside the scope of this paper (Section 7).

4 Method

4.1 Expected Information Gain of a Message

A message is useful to an agent when two conditions hold: the message is about something the goal depends on, and the agent does not already know it. Let z_e be the task-relevant state of a goal-related entity $e \in \mathcal{E}$, such as an object's location or the agent holding it, and let $Z = (z_e)_{e \in \mathcal{E}}$. At step t, agent i acts on its context $c_{t,i} = (P_i, \eta_{t,i}, o_{t,i})$: its prompt P_i, its memory $\eta_{t,i}$, and its observation $o_{t,i}$. We write $H_\phi(Z \mid c)$ for the entropy of the belief over Z that the frozen backbone, with parameters ϕ,

forms from context c. The expected information gain of a message m for agent i, $\mathrm{EIG}_t(m \to i)$, is how much m reduces the agent's uncertainty about Z:

$$\mathrm{EIG}_t(m \to i) \;=\; \mathbb{E}\Big[\, H_\phi(Z \mid c_{t,i}) - H_\phi(Z \mid c_{t,i}, m) \Big], \tag{1}$$

where the expectation is over the router's uncertainty about $c_{t,i}$, since the router does not observe the agent's context. The definition captures both conditions. Because Z contains only goal-related entities, the gain is zero for content the goal does not depend on. Because the entropy is conditioned on $c_{t,i}$, the gain is zero for content the agent already holds.

Equation 1 is intractable for free-text messages, so we approximate it for the two common message types, reports and requests, which we call SHARE and QUERY.

A SHARE message passes on information the sender holds: something it observed, its own state, or the subgoal it is pursuing, sent on its own initiative or in reply to a request. It has little gain for an agent that already knows its content, and a larger gain for an agent that does not. A QUERY message asks another agent for information or for an action. Under the cooperative premise, the recipient replies with what it knows or does what it can, so a QUERY is useful only if its recipient knows the relevant fact. We therefore estimate the expected information gain of a QUERY by that of the SHARE its recipient will send in reply.

Let $M_t(i, e) \in [0, 1]$ be the router's estimate that agent i currently holds valid information about entity e, and let $\mathcal{E}(m)$ be the set of entities that message m mentions. We treat the reduction in uncertainty that valid information about an entity brings to an agent that lacks it as a constant H_0. An agent that already holds information about e may still be uncertain about z_e, for example because its observation is old or another agent has since acted on e. These cases enter through $M_t(i, e)$, which estimates whether the agent's information is still valid. An agent whose information is valid gains little from being told again; we approximate this gain by zero. The two observations above give

$$\mathrm{EIG}_t^{\textsc{Share}}(m \to i) \;\approx\; H_0 \sum_{e \in \mathcal{E}(m)} \big(1 - M_t(i, e)\big), \tag{2}$$

$$\mathrm{EIG}_t^{\textsc{Query}}(m \to j) \;\approx\; H_0 \sum_{e \in \mathcal{E}(m)} M_t(j, e)\,\big(1 - M_t(s, e)\big), \tag{3}$$

where s is the sender of the QUERY. The gain for a SHARE recipient is large when it probably does not know the entities in m; the gain for a QUERY recipient is large when it probably knows them and the sender probably does not. A QUERY for an action has the same form (Appendix A.4).

The approximation keeps the second condition of Equation 1 and reduces the first to membership in $\mathcal{E}$: an entity counts if the goal depends on it and not otherwise. The derivation shows what a router must track. For both message types, the gain depends on the agents only through $M_t(i, e)$, whether each agent holds valid information about each entity. Routing needs only to know whether each term is large or small, so CREAC thresholds M_t directly (Section 4.4) instead of computing the sums.

4.2 INFERRING AGENT KNOWLEDGE FROM MESSAGES

The derivation leaves one quantity to estimate, $M_t(i, e)$, which we call the router's meta-belief. The router cannot read the agents' internal states, because that would break the DEC-POMDP-COM setting. Instead, it infers what each agent knows by *overhearing* the messages the agents send and receive. For each agent i and entity e the router stores an evidence bit $B(i, e) \in \{0, 1\}$, which records whether the messages so far indicate that agent i knows e, and the step $\ell(i, e)$ at which this entry was last updated. It sets the bits with two rules. **(R1) The sender and every recipient of a SHARE message about entity e are treated as knowing e. (R2) Every recipient of a QUERY message about e is treated as not knowing e until it answers.** The sender is also treated as not knowing e, unless the QUERY itself gives information about e, as in "bring the wine from the kitchen".

The sender clause of (R2) follows from the cooperative premise, under which an agent asks only about what it lacks; it sets $1 - M_t(s, e)$ in Equation 3 to one, so the answer, a SHARE about e, is routed back to the sender. The recipient clause of (R2) is temporary: a recipient that knows e answers, and (R1) sets its entry back to one.

Because the meta-belief is built only from what the agents say, it can drift from what they actually know. The next section keeps it current.

4.3 MAINTAINING THE ROUTER'S META-BELIEF

First, information goes out of date. An entry may record that an agent knows an entity after that agent's information has stopped being valid. We therefore weight each bit by a confidence $K_t(i,e) = \exp\big(-\lambda\,(t - \ell(i,e))\big)$ that decays with the number of simulation steps since the router last updated the entry, and use $M_t(i,e) = B(i,e)\,K_t(i,e)$ as the meta-belief.

Decay, however, creates a second problem. If no agent mentions an entity for long enough, the meta-belief entries of every agent for that entity approach zero, and the router can no longer choose recipients for messages about it. The router therefore watches for entities whose largest meta-belief has fallen below a query threshold τ_{query}. For such an entity it sends a query to the agent most likely to know it, that is, the agent with the highest confidence among $\mathcal{S}_e = \{j : B(j,e) = 1\}$. If $\mathcal{S}_e$ is empty, no agent has positive evidence, and the router sends the query to the agent whose entry is stalest, the one with the smallest $\ell(j,e)$. Appendix A.3 shows why sending the query is better than suppressing it in this case. We call these router-initiated queries *active queries*, to distinguish them from the QUERY messages that agents send.

4.4 SELECTIVE MESSAGE ROUTING

The router turns the meta-belief into delivery decisions with a routing threshold τ_{routing} on each entity the message mentions. It assumes that agent i knows e if $M_t(i,e) \geq \tau_{\text{routing}}$ and not otherwise. **(i) A SHARE message goes to every agent assumed not to know at least one of its entities. (ii) A QUERY message goes to every agent assumed to know at least one of its entities. (iii) A teammate the sender addresses by name always receives the message.** If these rules select no recipient, the message goes to every agent except the sender.

Testing each entity separately and taking the union of recipients means that an agent receives a SHARE when the largest term of Equation 2 exceeds $H_0(1 - \tau_{\text{routing}})$; this equals thresholding the sum when a message mentions a single entity. Once the recipients are fixed, the router updates the meta-belief with (R1) and (R2). All entries start at zero at the beginning of the simulation. Algorithm 1 in Appendix A.13 gives the complete loop.

5 IMPLEMENTATION

For each emitted message, a single call to gpt-oss-20b (Agarwal et al., 2025) classifies the message as SHARE, QUERY, or OTHERS, the last for messages that mention no goal-related entity, and extracts the entities it refers to (Appendix A.20); a message stating that the apple is in the kitchen is a SHARE about the apple. The entity set $\mathcal{E}$ is obtained once per task by prompting the same model with the goal description (Figure 6). OTHERS messages go to all agents except the sender. Routed messages are delivered at the beginning of the following step. We set $\lambda = 0.01$, $\tau_{\text{routing}} = 0.3$, and $\tau_{\text{query}} = 0.5$, chosen once on the smallest C-WAH configuration and used unchanged for every team size and for CREW-Wildfire (Appendices A.3 and A.9). Routing is a threshold comparison and active queries use a fixed template, so the classification and extraction call is the router's only model usage; Appendix A.12 measures its cost and latency.

6 EXPERIMENTS

We ask five questions: whether CREAC outperforms other communication strategies (Q1), how accurate its meta-belief is (Q2), whether messages with low expected information gain hurt agents and what routing adds to querying (Q3), whether decay and active querying keep the meta-belief accurate (Q4), and whether people prefer the result (Q5). Agents run on gpt-oss-20b (Agarwal et al., 2025) at medium reasoning effort, and all simulation experiments on both benchmarks use five random seeds.

6.1 BENCHMARKS AND BASELINES

The two benchmarks cover small homogeneous teams with structured goals and large heterogeneous teams with stochastic dynamics (Appendix A.5). Unless a table states otherwise, values after $\pm$ are standard errors over five seeds.

Table 1: Steps and tokens on C-WAH with three to six agents. Values after ± are standard errors over five seeds. Bold: best value in each column.

	3 agents		4 agents		5 agents		6 agents	
Method	*Step*↓	*Usage*↓	*Step*↓	*Usage*↓	*Step*↓	*Usage*↓	*Step*↓	*Usage*↓
Planner + CREAC router (ours)								
CoELA+CREAC	**55.45** ±1.9	**40k** ±2k	**48.27** ±1.9	**53k** ±3k	**44.90** ±2.0	**68k** ±4k	**43.28** ±2.0	85k ±5k
ReAct+CREAC	61.40 ±2.2	46k ±3k	55.70 ±2.4	55k ±3k	51.30 ±2.2	69k ±5k	49.76 ±2.3	**83k** ±4k
Planner + CREAC w/o active queries								
CoELA+CREAC	58.70 ±2.8	62k ±4k	55.75 ±2.2	67k ±3k	47.43 ±3.1	79k ±6k	47.05 ±3.1	97k ±7k
ReAct+CREAC	69.31 ±3.8	48k ±3k	68.65 ±2.8	60k ±5k	63.67 ±3.5	71k ±4k	63.32 ±2.7	93k ±7k
Planner-coupled systems								
CoTS	74.20 ±3.9	242k ±14k	73.07 ±4.2	343k ±20k	65.90 ±3.6	414k ±27k	73.16 ±4.7	679k ±43k
REVECA	63.45 ±1.8	108k ±5k	63.20 ±1.7	133k ±6k	63.48 ±1.9	195k ±10k	61.40 ±2.0	241k ±12k
Planner + Broadcast								
CoELA	62.26 ±2.4	59k ±3k	58.83 ±2.2	66k ±4k	50.63 ±2.1	76k ±5k	51.33 ±2.3	94k ±6k
ReAct	76.32 ±3.6	46k ±3k	73.72 ±3.4	56k ±4k	69.65 ±3.1	70k ±5k	69.33 ±3.2	89k ±6k
Planner + RCR-Router								
CoELA	62.83 ±2.5	58k ±3k	59.38 ±2.4	64k ±4k	56.09 ±2.6	69k ±4k	55.80 ±2.8	87k ±5k
ReAct	66.80 ±3.0	63k ±4k	60.34 ±2.6	72k ±4k	53.80 ±2.4	84k ±5k	52.18 ±2.5	90k ±6k

Communicative Watch-And-Help (C-WAH). C-WAH (Zhang et al., 2023) is a household coordination benchmark. We use the ten official test episodes and extend the original two-agent setting to teams of three to six agents. Each episode has three to eight object-centric subgoals and a horizon of $h = 250$ simulator steps. We report *Step*, the number of steps to completion, and *Usage*, the tokens consumed by agents and router. Every method completes all ten episodes within the horizon.

CREW-Wildfire. CREW-Wildfire (Hyun et al., 2025) is a wildfire response benchmark on a 200×200 map. We use the Full Environment with fifteen heterogeneous agents (ten firefighters, one bulldozer, two drones, two helicopters) and a horizon of $h = 200$ steps. We report the benchmark's $\mathit{Damage\ Cost} = \mathrm{TreesDestroyed} + 20 \cdot \mathrm{AgentsLost} + 100 \cdot \mathrm{CiviliansLost}$ and *Usage*.

Baselines. We compare against two groups of baselines. The first consists of planner-coupled systems: REVECA (Seo et al., 2025), which communicates selectively using beliefs held inside the planner, and CoTS (Zu et al., 2025), which puts communication inside tree search over joint plans. The second consists of communication policies on a fixed planner, CoELA (Zhang et al., 2023) or ReAct (Yao et al., 2022): **No-Comm**; **Broadcast**, the planners' standard channel; **Recency** and **Distance**, routing by who last mentioned the entity and by proximity; **RCR-Router** (Liu et al., 2025), role-aware routing; **Random**, each message to as many randomly chosen teammates as CREAC would select, with CREAC's active queries unless stated otherwise; and **Capped Broadcast**, Broadcast with each agent's message history truncated to a fixed token budget. Random controls how many agents receive each message, and Capped Broadcast how much message history each agent reads. We also compare with the graph-based AgentPrune (Zhang et al., 2024a) and GDesigner (Zhang et al., 2024b), and with per-message language model reasoning in place of the meta-belief (Appendix A.14).

6.2 Q1: Does CREAC outperform other communication strategies?

Small homogeneous teams. On C-WAH (Table 1), CoELA+CREAC takes the fewest steps at every team size and uses far fewer tokens than the planner-coupled systems (85k against 241k for REVECA and 679k for CoTS at six agents). CREAC takes fewer steps than RCR-Router at every team size under CoELA, and comparable or fewer steps under ReAct. Under ReAct, CREAC takes 20–28% fewer steps than Broadcast at every team size, with a comparable number of tokens. Under CoELA, CREAC takes 11–18% fewer steps than Broadcast at every size, 20–32% fewer tokens at three and four agents, and a comparable number of tokens at five and six. Without active queries, CREAC takes comparable or fewer steps than Broadcast at every team size under both planners; Section 6.4 shows

Table 2: Damage cost and tokens on CREW-Wildfire with fifteen heterogeneous agents; within each planner block only the communication policy changes. Values after ± are standard errors over five seeds. Bold: best value in each column.

	CoELA		**ReAct**	
Communication	*Damage Cost*↓	*Usage*↓	*Damage Cost*↓	*Usage*↓
No-Comm	142.4 ±9.5	**4.90M ±0.07M**	150.0 ±10.2	**4.70M ±0.08M**
Broadcast	108.2 ±7.8	5.55M ±0.11M	112.4 ±8.0	5.40M ±0.11M
Capped Broadcast	112.3 ±8.6	5.31M ±0.08M	116.6 ±7.4	5.19M ±0.10M
Random	101.7 ±9.4	5.41M ±0.11M	108.9 ±9.1	5.13M ±0.08M
Recency	115.0 ±8.4	5.18M ±0.09M	124.2 ±9.0	4.96M ±0.08M
Distance	121.0 ±8.7	5.22M ±0.09M	128.8 ±9.3	4.98M ±0.08M
RCR-Router	104.8 ±7.3	5.44M ±0.10M	117.8 ±8.5	5.21M ±0.09M
CREAC w/o active queries	97.0 ±8.1	5.52M ±0.12M	103.6 ±8.7	5.27M ±0.09M
CREAC	**76.2 ±7.1**	5.36M ±0.09M	**89.2 ±7.5**	5.15M ±0.09M

Table 3: Accuracy of CREAC's recipient choices by team size, in % (TP: target precision; RS: request success). The base-rate row gives the target precision of delivering to every teammate at the same delivery events.

	C-WAH				**CREW-Wildfire**
Agents	3	4	5	6	15
TP ↑	69.48 ±4.1	77.94 ±4.3	80.04 ±2.8	84.26 ±2.1	85.89 ±2.4
TP, base rate	55.18 ±3.8	54.60 ±3.1	51.93 ±2.5	51.30 ±1.7	47.53 ±2.2
RS ↑	81.61 ±3.9	91.06 ±3.4	90.06 ±2.6	92.22 ±1.7	92.95 ±1.9

how queries and routing combine. Appendix A.6 adds the graph-based baselines and the remaining policies, and Appendix A.15 gives routing traces.

Large heterogeneous teams. On CREW-Wildfire (Table 2), No-Comm gives the highest damage, so information has to be pooled across roles and places. Recency and Distance give damage comparable to or higher than Broadcast, so selecting recipients helps only when the selection reaches the agents that can use the message. CREAC gives the lowest damage under both planners (76.2 with CoELA, 89.2 with ReAct), 30% and 21% below Broadcast and 27% and 24% below RCR-Router, with 3–5% fewer tokens than Broadcast. Random, which delivers each message to as many agents as CREAC does, has higher damage than CREAC under both planners, so the gain comes from which agents receive each message. Capped Broadcast, which truncates each agent's message history to a fixed budget, also has higher damage than CREAC under both planners.

6.3 Q2: How accurate is the meta-belief built by overhearing?

We measure the accuracy of the meta-belief through the recipients it selects, judged against the ground-truth world state at delivery time (Table 3). *Target precision* is the fraction of selected recipients that are the intended kind of agent: for a SHARE, an agent that lacked valid information about the entity; for a QUERY, an agent that held it. *Request success* is whether at least one selected recipient can answer a QUERY. Target precision rises with team size in C-WAH, from 69.48% at three agents to 84.26% at six, and reaches 85.89% with fifteen agents in CREW-Wildfire. Request success stays above 90% from four agents on. The base rate falls as teams grow, from 55.18% at three agents to 47.53% at fifteen, so CREAC's margin over it grows from 14 to 38 points. If a message were worth about the same to every recipient, no rule for choosing recipients could exceed this base rate, so the margin supports (A1). These numbers include the extractor's errors, which Appendix A.12 measures directly. Replacing the meta-belief with a

Table 4: The meta-belief against direct reasoning on six-agent C-WAH. In the lower two rows a language model reads the dialogue and chooses recipients and queries itself. TP: target precision; RS: request success. Bold: best value in each column.

Router	*Step*↓	*TP*↑	*RS*↑	*Usage*↓
CREAC	**43.28** ±2.0	**84.26** ±2.1	**92.22** ±1.7	**85k** ±5k
gpt-oss-20b	61.02 ±4.6	71.78 ±6.4	81.94 ±3.6	124k ±10k
GPT-5	55.63 ±4.2	78.34 ±2.9	88.58 ±2.8	130k ±11k

Table 5: Delivery rule × active queries (CoELA): six-agent C-WAH steps↓, CREW-Wildfire damage cost↓. Random matches CREAC's recipient count.

	C-WAH		CREW-Wildfire	
Delivery	w/ queries	w/o	w/ queries	w/o
CREAC	**43.28** ±2.0	47.05 ±3.1	**76.2** ±7.1	97.0 ±8.1
Random	54.11 ±3.4	55.60 ±2.6	101.7 ±9.4	109.6 ±8.3
Broadcast	58.30 ±3.0	51.33 ±2.3	114.7 ±8.1	108.2 ±7.8

Table 6: Maintenance ablation, six-agent C-WAH (TP: target precision; RS: request success). Bold: best in each column.

Setting	*Step*↓	*TP*↑	*RS*↑	*Usage*↓
CREAC	**43.28** ±2.0	**84.26** ±2.1	**92.22** ±1.7	**85k** ±5k
w/o active queries	47.05 ±3.1	76.50 ±2.9	80.66 ±3.4	97k ±7k
w/o decay (λ=0)	51.85 ±2.7	74.80 ±3.1	84.10 ±2.8	92k ±6k

language model that reads the dialogue and decides recipients and queries itself takes more steps and more tokens, both with gpt-oss-20b and with GPT-5 (Singh et al., 2025; Table 4; Appendix A.6).

6.4 Q3: Do messages with low expected information gain hurt agents, and what does routing add?

Assumption (A2) concerns a single agent, so we test it one decision at a time, then for the team.

Single decisions. At 500 decision points from CREAC's six-agent C-WAH runs we freeze the agent's context and ask it to choose its next action again under four conditions: the delivered context; plus k messages CREAC withheld from it; plus k length-matched messages about entities outside the goal; and minus the messages CREAC delivered. An action is *productive* if it advances a goal object that no other agent is handling. Adding withheld messages lowers the productive-action rate, and the drop grows with the number added (Figure 1): by 1.4 points for one message (95% interval includes zero), 3.3 for two, 5.9 for four, and 10.4 for eight (95% interval 6.9–14.0). Filler of the same length lowers it by 8.2 points at $k = 8$, so most of the harm comes from the added length, as (A2) states. Removing the messages CREAC delivered lowers the rate by 7.4 points, so the messages CREAC delivers help their recipients, while those it withholds would hurt them. Appendix A.11 gives the protocol and the values at every k, and Appendix A.8 repeats the test at $k = 8$ with a second backbone and finds the same pattern.

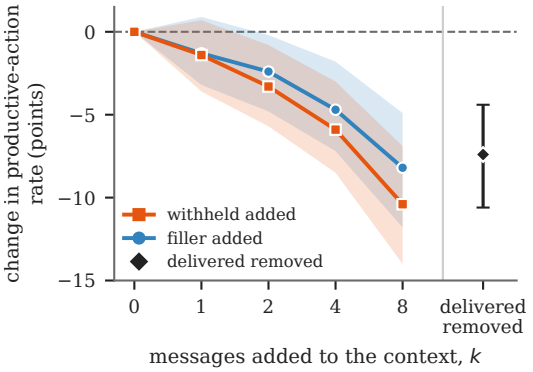

Figure 1: Exposure to messages with low expected information gain on six-agent C-WAH: change in the productive-action rate of one agent's next action when k messages that CREAC withheld from it, or k filler messages of the same length, are added. The separate point shows the effect of removing the messages CREAC delivered. Bands and the error bar are 95% intervals.

Routing and querying. CREAC changes what agents receive by choosing recipients and by sending queries; Table 5 crosses the two. We call Broadcast with active queries Broadcast+ActiveQuery: the router runs unchanged, but every message goes to everyone, so it delivers everything CREAC delivers plus exactly the messages the router withheld. The two factors interact. Under CREAC's delivery, queries cut steps from 47.05 to 43.28 on C-WAH and damage from 97.0 to 76.2 on CREW-Wildfire. Under Broadcast, the same queries raise steps from 51.33 to 58.30 on C-WAH and leave damage comparable on CREW-Wildfire (114.7 against 108.2). A query elicits information that no agent had volunteered; selective delivery sends the query to the agents likely to know and the answer to the agents that lack it, while Broadcast sends both to everyone. Broadcast+ActiveQuery sends about as many queries as CREAC, to similar targets (Appendix A.7). With queries, CREAC takes 26% fewer steps than Broadcast+ActiveQuery and gives 34% lower damage, the largest gaps in the table. Random, matched to CREAC's recipient count, is worse than CREAC in three cells and comparable in the fourth (CREW-Wildfire without queries), so the gain comes from who receives each message rather than from how many receive it.

Table 7: User study with 30 participants. Likert ratings 1–7, mean ± standard deviation (median). Bold: highest rating on each dimension.

Condition	*Trust* ↑	*Appropriateness* ↑	*Usefulness* ↑	*Efficiency* ↑
CREAC	**5.30 ± 1.74 (6.0)**	**5.27 ± 1.68 (6.0)**	**6.07 ± 1.28 (6.5)**	**5.97 ± 1.54 (7.0)**
Broadcast	4.13 ± 1.81 (4.0)	3.70 ± 1.68 (4.0)	4.50 ± 1.80 (5.0)	4.63 ± 1.96 (5.0)
No-Comm	4.23 ± 1.92 (4.0)	4.33 ± 1.86 (4.0)	5.33 ± 1.63 (6.0)	5.33 ± 1.73 (6.0)

6.5 Q4: Do decay and active querying keep the meta-belief accurate?

Removing either mechanism lowers target precision and request success (Table 6). Without active queries, step count rises from 43.28 to 47.05; without decay ($\lambda = 0$), it rises to 51.85. Each ablation hurts most on the metric tied to the error its mechanism corrects. Active queries refresh entries whose confidence has fallen, including those for entities that no agent has reported, so later QUERY messages about those entities reach an agent that can answer them. Without active queries, request success falls by 11.6 points, the largest drop in Table 6, while target precision falls by 7.8. Decay discounts old evidence, and by lowering its confidence it also triggers the queries that refresh it: with $\lambda = 0$ the router issues 1.6 active queries per episode instead of 2.5. Without decay, stale entries are neither discounted nor refreshed, and target precision falls by 9.5 points, more than request success (8.1). Neither variant uses fewer tokens than CREAC (97k and 92k against 85k).

6.6 Q5: Do people prefer the resulting communication pattern?

Thirty participants worked with six CoELA-based agents in C-WAH under a within-subject protocol, playing CREAC, Broadcast and No-Comm without being told which was which, and rated each on 7-point scales for trust, appropriateness, usefulness, and efficiency. CREAC is rated highest on every dimension (Table 7). The Friedman test across the three conditions is significant on trust, appropriateness and usefulness, and on these three CREAC is rated significantly above both Broadcast and No-Comm. Appendix A.16 gives the protocol and all tests.

7 Limitations

CREAC reads each message as evidence about what its sender knows, so it assumes cooperative agents that report what they observe and ask only about what they lack. Both benchmarks are fully cooperative, with a shared goal and reward; teams whose members withhold information strategically would need a different model of the sender. The meta-belief rests on one evidence bit per agent and entity, which does not separate knowing where an object is from holding it or planning to fetch it; a finer state per entity is a direct extension. The entity set is derived once per task from the goal description, which suits goal-specified tasks such as ours but not open-ended discovery. All messages pass through one router, and a decentralized version in which each agent keeps its own meta-belief is left to future work (Appendix A.17).

8 Conclusion

We framed message routing in embodied language-model teams as sending each message to the recipients for whom its expected information gain is high, and showed that, for SHARE and QUERY messages, this gain can be approximated by a quantity that depends on the agents only through a meta-belief over who holds valid information about which goal-related entity. CREAC maintains it from the agents' messages, without training and without modifying the planner. It performs best on C-WAH and CREW-Wildfire, outperforms broadcasting in all ten combinations of planner, benchmark and team size, and gains most when its queries are combined with selective delivery.

AI Use Statement

In this work, we used generative AI tools to assist with translation of the manuscript, to assist in implementing parts of the code, and to provide feedback on the research methodology and experiments.

We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for proofs or write proofs, propose or refine hypotheses, design the research methodology or experiments, clean or reformat datasets, support qualitative or thematic data analysis, or interpret results. The authors reviewed all AI-assisted work by manual inspection. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Language models are also components of the systems we study. gpt-oss-20b powers the agents and the router's classification and extraction call, and GPT-5 serves as a backbone for the direct-reasoning comparison in Appendix A.6. The main text and the appendix describe this use as part of the method.

Ethics Statement

This work includes a user study with human participants (Section 6.6 and Appendix A.16). The study was reviewed and approved by the Institutional Review Board (IRB) at our institution. Thirty adults aged 19 to 36 took part; they provided informed consent prior to participation and were compensated. Each participant played all three conditions with a team of agent teammates in C-WAH, without being told which communication policy was in use, and then answered a questionnaire. Responses were collected anonymously and used only for statistical analysis and research. All other experiments are simulator rollouts that involve no people and no physical robots (Appendix A.18), and we use existing benchmarks, simulators and models under their licenses (Appendix A.19).

Reproducibility Statement

Section 4 specifies the router, and Appendix A.13 gives the complete router loop as pseudocode. Appendix A.20 gives the prompt for the classification and extraction call, and Appendix A.3 and Appendix A.9 give the decay rate, the two thresholds, and the sensitivity of results to them. Appendix A.5 and Appendix A.14 describe the benchmarks and the configuration of every baseline, and Appendix A.12 and Appendix A.11 give the protocols of the robustness and exposure experiments. The agents and the router use the open-weight gpt-oss-20b model, all simulation experiments use five random seeds, and Appendix A.18 lists the hardware and the inference setup. GPT-5 is accessed through an API and appears only in the direct-reasoning comparison. Appendix A.16 gives the full survey instruction and the questionnaire items. The source code is included in the supplementary material, and we will release it upon acceptance.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Raphen Becker, Alan Carlin, Victor Lesser, and Shlomo Zilberstein. Analyzing myopic approaches for multi-agent communication. *Computational Intelligence*, 25(1):31–50, 2009.

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.

Alan Carlin and Shlomo Zilberstein. Value of communication in decentralized pomdps. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, pp. 16–21, 2009.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on machine learning*, pp. 1538–1546. PMLR, 2019.

Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. *Advances in neural information processing systems*, 33:22069–22079, 2020.

Prashant Doshi and Piotr J Gmytrasiewicz. A particle filtering based approach to approximating interactive pomdps. In *AAAI*, pp. 969–974, 2005.

J Foerster. *Deep multi-agent reinforcement learning*. PhD thesis, University of Oxford, 2018.

Claudia V Goldman and Shlomo Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 137–144, 2003.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Jonathan Hyun, Nicholas R Waytowich, and Boyuan Chen. Crew-wildfire: Benchmarking agentic multi-agent collaborations at scale. *arXiv preprint arXiv:2507.05178*, 2025.

Gal A Kaminka, David V Pynadath, and Milind Tambe. Monitoring teams by overhearing: A multi-agent plan-recognition approach. *Journal of artificial intelligence research*, 17:83–135, 2002.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. *arXiv preprint arXiv:1902.01554*, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Huao Li, Yu Chong, Simon Stepputtis, Joseph P Campbell, Dana Hughes, Charles Lewis, and Katia Sycara. Theory of mind for multi-agent collaboration via large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 180–192, 2023.

Jie Liu, Pan Zhou, Yingjun Du, Ah-Hwee Tan, Cees GM Snoek, Jan-Jakob Sonke, and Efstratios Gavves. Capo: Cooperative plan optimization for efficient embodied multi-agent cooperation. *arXiv preprint arXiv:2411.04679*, 2024.

Jun Liu, Zhenglun Kong, Changdi Yang, Fan Yang, Tianqi Li, Peiyan Dong, Joannah Nanjekye, Hao Tang, Geng Yuan, Wei Niu, et al. Rcr-router: Efficient role-aware context routing for multi-agent llm systems with structured memory. *arXiv preprint arXiv:2508.04903*, 2025.

Yen-Cheng Liu, Junjiao Tian, Chih-Yao Ma, Nathan Glaser, Chia-Wen Kuo, and Zsolt Kira. Who2com: Collaborative perception via learnable handshake communication. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6876–6883. IEEE, 2020.

Zhao Mandi, Shreeya Jain, and Shuran Song. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 286–299. IEEE, 2024.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8494–8502, 2018.

Maayan Roth, Reid Simmons, and Manuela Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 786–793, 2005.

Maayan Roth, Reid Simmons, and Manuela Veloso. What to communicate? execution-time decision in multi-agent pomdps. In *Distributed autonomous robotic systems 7*, pp. 177–186. Springer, 2006.

SeungWon Seo, SeongRae Noh, Junhyeok Lee, SooBin Lim, Won Hee Lee, and HyeongYeop Kang. Reveca: Adaptive planning and trajectory-based validation in cooperative language agents using information relevance and relative proximity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23295–23303, 2025.

SeungWon Seo, SooBin Lim, SeongRae Noh, Haneul Kim, and HyeongYeop Kang. From assumptions to actions: Turning llm reasoning into uncertainty-aware planning for embodied agents. *arXiv preprint arXiv:2602.04326*, 2026.

HaoYang Shang, Xuan Liu, Zi Liang, Jie Zhang, Haibo Hu, and Song Guo. United minds or isolated agents? exploring coordination of llms under cognitive load theory. *arXiv preprint arXiv:2506.06843*, 2025.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Matthijs TJ Spaan, Geoffrey J Gordon, and Nikos Vlassis. Decentralized planning under uncertainty for teams of communicating agents. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 249–256, 2006.

Garold Stasser, Dennis D Stewart, and Gwen M Wittenbaum. Expert roles and information exchange during discussion: The importance of knowing who knows what. *Journal of experimental social psychology*, 31(3):244–265, 1995.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Daniel M. Wegner. Transactive memory: A contemporary analysis of the group mind. 1987.

Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online planning for multi-agent systems with bounded communication. *Artif. Intell.*, 175:487–511, 2011.

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

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

Luke Zettlemoyer, Brian Milch, and Leslie Pack Kaelbling. Multi-agent filtering with infinitely nested beliefs. In *Neural Information Processing Systems*, 2008.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. *ArXiv*, abs/2410.02506, 2024a.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024b.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. *arXiv preprint arXiv:2307.02485*, 2023.

Andrew Zhu and Christopher Callison-Burch. Overhearing llm agents: A survey, taxonomy, and roadmap. *ArXiv*, abs/2509.16325, 2025.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Language agents as optimizable graphs. *ArXiv*, abs/2402.16823, 2024.

Lizheng Zu, Lin Lin, Song Fu, Na Zhao, and Pan Zhou. Collaborative tree search for enhancing embodied multi-agent collaboration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 29513–29522, 2025.

A Appendix

A.1 Extended Related Work

LLM-based embodied multi-agent systems. Recent advances in language models let embodied agents handle partial observability and long-horizon planning with language as a high-level interface (Achiam et al., 2023; Zhang et al., 2023). Frameworks such as CoELA (Zhang et al., 2023), REVECA (Seo et al., 2025), and PCE (Seo et al., 2026) coordinate in decentralized settings through communication, whereas RoCo (Mandi et al., 2024), CoTS (Zu et al., 2025), and CaPO (Liu et al., 2024) use communication for joint planning. These systems mostly assume small teams and rely on broadcast-style exchange (Zhang et al., 2023) or on fixed communication protocols with limited scalability (Seo et al., 2025; Liu et al., 2024). As team size grows, unmanaged exchange raises token consumption and exposes agents to messages irrelevant to their current decisions (Yan et al., 2025). REVECA partially addresses this by reducing message volume through belief synchronization inside the planner, but that ties the cost of selectivity to the planner. CREAC is instead an external router that decides message delivery from what it infers each agent knows, without modifying the planner.

Value of communication in decentralized control. The closest conceptual predecessors of CREAC decide at execution time whether a message is worth sending. Roth et al. (2005) plan centralized policies for multi-agent POMDPs under free communication, have each agent track a distribution over the team's possible joint beliefs, and communicate only when doing so changes the joint action for the better. Becker et al. (2009) analyze myopic rules that communicate when the expected value of the shared information exceeds its cost, and related work decides what and when to communicate in decentralized Markov decision processes (MDPs) and POMDPs from the same value-of-information criterion (Goldman & Zilberstein, 2003; Roth et al., 2006; Carlin & Zilberstein, 2009; Wu et al., 2011). These methods share CREAC's criterion, that a message should be sent when its value exceeds its cost, and differ in what the value is computed from. They reason over joint beliefs about the world state, which requires a model of the state space and grows quickly with team size. CREAC computes a simpler quantity, whether each agent holds valid information about each goal-related entity, from overheard natural-language messages, and uses it to choose recipients; the agents still decide whether and what to say.

Learned targeted communication. Deep multi-agent reinforcement learning learns when and with whom to communicate through gating and attention (Foerster, 2018; Das et al., 2019; Kim et al., 2019). Who2com learns a handshake that selects which teammates to request information from (Liu et al., 2020), and I2C learns a prior network that predicts, from an agent's own observation, which teammates' actions its message would influence (Ding et al., 2020). These methods choose recipients per message, as CREAC does, but through a trained network whose behavior is tied to the training distribution of team size and task. In the language-agent domain, recent work reduces aggregate message volume through static graph or cost constraints (Zhang et al., 2024a; Zhuge et al., 2024; Zhang et al., 2024b), or routes context by predefined role profiles, as in RCR-Router (Liu et al., 2025). These methods fix a structural prior, a graph topology or a role assignment, while CREAC estimates how task-relevant knowledge is spread across the team at each moment.

Team cognition and belief tracking. CREAC is motivated by transactive memory, a shared understanding of where knowledge resides in a team (Wegner, 1987; Stasser et al., 1995). CoThinker (Shang et al., 2025) also uses a transactive memory, which records each agent's expertise; CREAC records whether each agent's information about an entity is still valid, which changes during an episode. Interactive state estimation frameworks model teammates' beliefs formally (Doshi & Gmytrasiewicz, 2005; Zettlemoyer et al., 2008), but exact inference becomes intractable as teams grow (Bernstein et al., 2002). In LLM teams, Li et al. (2023) find that agents lose track of the task state over long horizons and that an explicit belief-state representation improves both collaboration and theory-of-mind inference. Monitoring by overhearing is a scalable and non-invasive way to infer knowledge states from observed dialogue (Kaminka et al., 2002; Zhu & Callison-Burch, 2025). CREAC adopts this view: it keeps an explicit meta-belief about agent knowledge outside the agents, which supports targeted routing and targeted queries without exact belief inference or access to private agent state.

A.2 Full DEC-POMDP-COM Specification

We model cooperative embodied multi-agent decision making as a decentralized partially observable Markov decision process with communication (DEC-POMDP-COM) (Bernstein et al., 2002; Oliehoek et al., 2016; Spaan et al., 2006). The environment is the tuple

$$(n,\ S,\ \{A_i\}_{i=1}^n,\ \{O_i\}_{i=1}^n,\ T,\ G,\ R,\ \gamma,\ h,\ \Sigma),$$

where n is the number of agents, S is the set of world states x, and Σ is the space of natural language messages. Each agent i has an action space $A_i = A_i^W \sqcup A_i^C$. Here A_i^W is the set of world actions, and $A_i^C = \Sigma \cup \{\perp\}$ is the set of communication actions, where $\perp$ stands for staying silent. At each step t, agent i takes one action $a_{t,i} \in A_i$: either a world action $a_{t,i}^W \in A_i^W$ or a communication action $a_{t,i}^C \in A_i^C$. If $a_{t,i}^C = m_{t,i} \in \Sigma$, the agent sends the message $m_{t,i}$; if $a_{t,i}^C = \perp$, it stays silent. The observation space of agent i is $O_i = O_i^W \times O_i^M$, where O_i^W contains its egocentric observations of the world and O_i^M contains the messages it receives. The router determines which messages enter O_i^M (Section 3). The world changes according to the transition model $T(x, a^W, x') = p(x' \mid x, a^W)$, where $a^W \in \prod_i A_i^W$ is the joint world action. All agents share one goal G and a fixed set of goal-related entities $\mathcal{E}$, the entities whose state the goal specifies, such as the objects the goal asks the team to place. Messages can refer to these entities. The team receives a shared reward R_t at each step, with discount factor γ and horizon h.

A.3 Interpretation of the Meta-Belief, the Decay, and the Thresholds

The following interpretation motivates the design of the meta-belief, the decay, and the thresholds. We use the notation of Sections 4.2 and 4.3: the evidence bit $B(i, e)$, the step $\ell(i, e)$ of the last update, the confidence $K_t(i, e)$, and the set $\mathcal{S}_e = \{j : B(j, e) = 1\}$.

Decay as knowledge survival. The exponential confidence decay can be read as a survival probability for stale knowledge. Suppose an agent's information about entity e becomes invalid according to a Poisson process with rate λ. The probability that it remains valid after Δt steps is

$$\Pr(V > \Delta t) = \exp(-\lambda \Delta t),$$

where V is the number of steps for which the information stays valid, so $K_t(i, e) = \exp(-\lambda(t - \ell(i, e)))$ approximates the probability that agent i's last observed knowledge about e is still valid at time t. Under this reading λ expresses how volatile the environment is assumed to be: smaller values keep old evidence longer, larger values forget it faster. With $\lambda = 0.01$ the expected validity time is 100 steps.

Thresholds as local decision gates. For a QUERY about entity e, recall that $M_t(j, e)$ is the router's estimate that agent j holds valid information about e. If routing the QUERY to j yields benefit b_e when j is a valid knower and costs κ_r to deliver, the local expected utility is $\mathbb{E}[U(j)] = M_t(j, e)b_e - \kappa_r$, so routing is worthwhile when

$$M_t(j, e) > \frac{\kappa_r}{b_e},$$

and τ_{routing} can be read as that break-even point rather than an arbitrary cutoff. Similarly, if $p^\star = \max_j M_t(j, e)$ is the confidence of the most likely knower, L_e the cost of failing to obtain information about e, and κ_q the overhead of a query, then querying is worthwhile when $(1 - p^\star)L_e > \kappa_q$, that is when

$$p^\star < 1 - \frac{\kappa_q}{L_e},$$

which is the rule $\max_j M_t(j, e) < \tau_{\text{query}}$.

Boundary case: no agent has positive evidence. When $\mathcal{S}_e = \emptyset$ we have $B(j, e) = 0$ for every agent, so $p^\star = 0$ and the query condition holds with the largest possible margin: the expected value of a query is greatest precisely when no agent has positive evidence, and suppressing it would contradict the analysis above. We call a query sent to the stalest agent in this case an *exploration query*. The target rule follows from the same model as the decay. K_t gives the probability that observed knowledge is still valid after elapsed time; symmetrically, agents keep acting and observing

between messages, so the probability that agent j holds unreported knowledge about e grows with the time since the router last updated (j, e). The stalest agent maximizes that probability under the same model of elapsed time, and the rule skips agents whose entries were just refreshed by a QUERY. In Appendix A.10 the exploration query takes fewer steps than both suppression and a broadcast question, as this analysis predicts.

A.4 ACTION QUERIES AS INFORMATION QUERIES

Section 4.1 derives the expected information gain of a QUERY that asks for information about an entity. A cooperative team also produces queries that ask a teammate to do something, such as bringing the pudding to the dinner table. This section shows that such a QUERY is covered by the same expression, states the assumption that makes this so, and identifies the one quantity that changes.

Setting. We reuse the notation of Section 4.1: z_e is the task-relevant state of entity $e \in \mathcal{E}$, the goal G specifies a target value z_e^G for each entity it constrains, H_0 is the constant reduction in uncertainty that valid information about an entity brings to an agent that lacks it, and a message changes the posterior only for the entities it mentions.

Definition 1 (Task)**.** *A task u is a pair $(\mathcal{E}(u), \{z_e^G\}_{e\in\mathcal{E}(u)})$: the entities the task touches and the configuration it is meant to produce. Every action query in our logs names a task of this form. C-WAH goals are object-placement predicates over the goal objects, and the CREW-Wildfire subtasks of suppression, clearing and rescue are defined over fire, obstacle and civilian entities.*

Assumption 1 (Resolution by either channel)**.** *Agent i comes to hold valid information about z_e in either of two ways: it learns the current value of z_e, or it sets z_e to its goal value z_e^G. In the second case the value is known because the agent produced it.*

Assumption 1 is the substantive step. It says that observing an entity and acting on an entity are two routes to the same position with respect to that entity, which is what lets one variable cover both cases.

Definition 2 (Extended knowledge variable)**.** *Let $\tilde{k}_t(j, e) = 1$ if and only if agent j is in a position to determine z_e at step t: either it holds valid information about z_e, or it is able and available to bring z_e to z_e^G. Write $\tilde{M}_t(j, e) = \Pr[\tilde{k}_t(j, e) = 1]$.*

Assumption 2 (Cooperative response)**.** *A recipient j of a* QUERY *about e sends a* SHARE *about z_e when $\tilde{k}_t(j, e) = 1$: an observation in the information case, a notice of completion in the action case. It does not fabricate such a* SHARE *when $\tilde{k}_t(j, e) = 0$.*

Proposition 1. *Under Assumptions 1 and 2, for any* QUERY *message m sent by agent s about the entities $\mathcal{E}(m)$, whether it requests information or action,*

$$\mathrm{EIG}_t(m \to j) \;\approx\; H_0 \sum_{e\in\mathcal{E}(m)} \tilde{M}_t(j, e)\,\big(1 - \tilde{M}_t(s, e)\big),$$

which is Equation 3 with $\tilde{M}$ in place of M.

Proof. Fix $e \in \mathcal{E}(m)$. The message m carries no content about z_e in either case: an information query states that the sender wants z_e, and an action query states that the sender wants z_e set to z_e^G. Conditioning on m therefore leaves $H_\phi(z_e \mid c)$ unchanged for every recipient, and the direct gain of m is zero. All of the gain is the gain of the response m'. By Assumption 2, m' is sent exactly when $\tilde{k}_t(j, e) = 1$, which has probability $\tilde{M}_t(j, e)$ under the router's estimate. By Assumption 1, in either case the responding agent knows z_e when it responds, so m' is a SHARE about e; applying Equation 2 to m' and the original sender gives a gain of $H_0(1 - \tilde{M}_t(s, e))$. Multiplying the probability that the response occurs by the gain it delivers, and summing over $\mathcal{E}(m)$, gives the stated expression. No step used which clause of $\tilde{k}$ holds, so the two cases are identical. □

Remark 1: the threshold is what differs. The analysis in Appendix A.3 routes a message to j when $\tilde{M}_t(j, e)\, b_e > \kappa_r$. For an information query κ_r is the cost of delivering one message. For an action query the recipient that accepts also pays the cost of executing u: it replans and moves. The

relevant cost is therefore $\kappa_r + \kappa_{\text{exec}}(u)$, with κ_{exec} usually much larger than κ_r, so the break-even threshold for action queries is

$$\tau^{\text{act}}_{\text{routing}} \;=\; \frac{\kappa_r + \kappa_{\text{exec}}(u)}{b_e} \;>\; \tau_{\text{routing}}.$$

Our implementation uses a single τ_{routing} for both kinds of QUERY and uses M in place of $\tilde{M}$; a separate threshold for action queries is a direct refinement.

Remark 2: delay. The response to an information query arrives within one step. The response to an action query arrives only after u has been executed, which spans many steps. Proposition 1 is unaffected, because it is an expectation over whether the response occurs; the delay changes only when. The confidence decay, however, runs over that longer interval, so $\tilde{M}_t(j, e)$ for an accepted but unfinished task decays before the SHARE announcing completion arrives. This is the mechanism behind the stale-entry case reported in Appendix A.10, where the router queries about a goal during the gap between its completion and the SHARE that announces it.

Remark 3: execution failure. Assumption 2 says a capable recipient completes the task and sends a SHARE. If execution fails, no such SHARE is sent, which reveals that $\tilde{k}_t(j, e)$ was zero. The update rules of Section 4.2 handle this unchanged: no SHARE means no (R1) update, so the entry decays and eventually triggers an active query. No separate failure-handling rule is needed.

A.5 BENCHMARK DETAILS

We follow the evaluation protocol of prior work (Zhang et al., 2023), where each agent receives observations only through the official environment APIs and acts under partial observability.

Communicative Watch-And-Help. C-WAH extends the original Watch-and-Help challenge with an explicit communication channel that lets agents coordinate on long-horizon household tasks. It is built on VirtualHome (Puig et al., 2018), which supports multi-agent simulation in indoor environments. We evaluate on ten episodes; in each, the team is assigned one shared household objective drawn from the goal families in Table 8.

Agents choose from a compact discrete action space covering navigation, orientation, object interaction, and message passing (Table 9). Observations are largely room-scoped: on entering a room an agent perceives objects not occluded by closed containers, while items inside appliances or cabinets become visible only after the agent opens them. To model limited bandwidth, each agent may send at most 500 characters per timestep. We fix the horizon to $h = 250$ simulator steps. Each episode involves three to eight target-object subgoals, and an episode counts as a failure if the shared objective is not completed within the step budget. A layout example is in Figure 2.

Table 8: C-WAH goal specifications.

Goals	Description
Prepare afternoon tea	put [*cupcake, pudding, apple, juice, wine*] on *coffeetable*
Wash dishes	put [*plate, fork*] inside *dishwasher*
Prepare a meal	put [*coffeepot, cupcake, pancake, poundcake, pudding, apple, juice, wine*] on *dinnertable*
Put groceries	put [*cupcake, pancake, poundcake, pudding, apple, juice, wine*] inside *fridge*
Set up a dinner table	put [*plate, fork*] on *dinnertable*

CREW-Wildfire Full Environment. CREW-Wildfire is a large-scale wildfire-response benchmark built on the CREW simulation platform (Hyun et al., 2025). We evaluate the Full Environment configuration (preset `Full_Game`), which combines the benchmark's subtasks into a single long-horizon scenario: agents must search for an initially unknown wildfire, contain or suppress its spread,

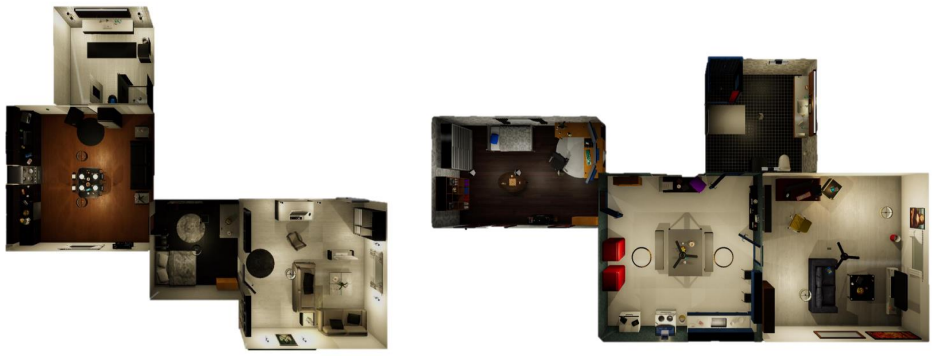

Figure 2: C-WAH example environments.

Table 9: C-WAH action space.

Action	Description
Walk towards	move to an object in the agent's current room, or to a room
Turn left	turn left by 30 degrees
Turn right	turn right by 30 degrees
Grasp	grasp an object
Open	open a closed container
Close	close an open container
Put	put the held objects into an open container or onto a surface
Send message	send a message to other agents

and rescue civilians. Unlike C-WAH, where success is a discrete household objective, the full setting is an open-ended damage-minimization problem under partial observability.

The environment is a procedurally generated 200×200 grid whose terrain and dynamics depend on vegetation, elevation, moisture, settlement, and wind layers. Fire spread is stochastic and depends on local terrain, so agents must keep revising plans as the fire front evolves. The full setting contains 15 heterogeneous agents (10 firefighters, 1 bulldozer, 2 drones, 2 helicopters) and 5 civilians. Fire and civilian locations are not revealed at the start, so the team must rely on exploration, observation sharing, and role-specialized coordination. The configuration is summarized in Table 10 and the role-specific actions in Table 11.

Agents receive local observations through the official CREW-Wildfire APIs. We use the high-level interface: agents reason over textual summaries of their local observations, communicate through the evaluated planning framework, and output natural-language action intents, which the Execution module translates into primitive actions. We fix the horizon to $h = 200$ high-level decision steps. Since the full environment has no binary completion condition, the horizon-limited score is used for evaluation.

Table 10: CREW-Wildfire full-environment specification.

Item	Description
Preset	`Full_Game`
Objective	locate and suppress the wildfire while rescuing civilians from danger
Map size	200×200 procedurally generated grid
Team composition	10 firefighters, 1 bulldozer, 2 drones, and 2 helicopters
Civilians	5 civilians placed in an initially unknown region
Initial knowledge	fire and civilian locations are not globally known; agents must discover them through local observations
Behavioral goals	task designation, agent capitalization, spatial reasoning, observation sharing, realtime coordination, plan adaptation, and objective prioritization
Evaluation objective	minimize damage cost: $N_{\text{trees destroyed}} + 20N_{\text{agents lost}} + 100N_{\text{civilians lost}}$; zero damage is optimal

Table 11: CREW-Wildfire role-specific primitive actions.

Agent type	Primitive actions and capabilities
Firefighter	move to a coordinate, cut one or more trees, pick up a civilian, drop off a civilian, spray water toward a target, and refill water at a water source
Bulldozer	drive to a coordinate without clearing vegetation, or drive while clearing a path to create firebreaks
Drone	fly to a coordinate for wide-area reconnaissance; drones cannot directly suppress fire or rescue civilians
Helicopter	fly to a coordinate, pick up firefighters, drop off firefighters, refill water, and drop a water payload

A.6 Additional Routing Baselines on C-WAH

Graph-based topology baselines. AgentPrune (Zhang et al., 2024a) fixes a sparse communication mask after a warm-up, and GDesigner (Zhang et al., 2024b) decodes a sparse topology for each message from task relevance and agent profiles. In C-WAH both take a number of steps comparable to Broadcast (Table 12). Both are our adaptations to the online recipient-selection interface used in our C-WAH routing experiments (Appendix A.14). The evaluation uses the six-agent C-WAH setting with five seeds and ten episodes per seed.

Table 12: Six-agent C-WAH comparison with graph-based topology baselines. *Usage* includes the tokens of router calls. Tokens per step is *Usage* divided by *Step*.

Method	Step ↓	Usage ↓	Tokens per step ↓
CoELA+CREAC	43.28 ±2.0	85k ±5k	1.96k
CoELA+Broadcast	51.33 ±2.3	94k ±6k	1.83k
AgentPrune	51.86 ±4.8	108k ±11k	2.08k
GDesigner	52.19 ±3.7	101k ±13k	1.94k

Table 12 reports the results. The two graph-based routers are close to each other and to Broadcast, and CREAC takes the fewest steps. Against AgentPrune, the stronger of the two, CoELA+CREAC takes 8.58 fewer steps and uses 23k fewer tokens, about 16.5% and 21.3% relative. Against all three other methods, CREAC uses comparable or fewer tokens (85k against 94k–108k).

The logs show why these two routers take more steps. AgentPrune behaves like a high-variance static topology: when its warm-up produces a useful sparse mask it routes efficiently, but a poor early mask

cannot be revised later in the episode. GDesigner benefits from task-adaptive topology scoring but still fails when the decoded sparse graph repeatedly assigns unresolved subtasks to an unproductive set of peers. CREAC instead keeps an explicit, staleness-aware estimate of which agents are likely to know each entity, so it can revise routing as the evidence changes.

Remaining communication policies across team sizes. Table 1 reports Broadcast and RCR-Router at every team size. Table 13 completes the set of communication policies of Table 2 on C-WAH, so that both benchmarks compare the same policies. CREAC takes fewer steps than all four at every team size and under both planners. Random matches CREAC's number of recipients per message, and Capped Broadcast limits each agent's message history to a fixed budget. CREAC takes 5.3 to 12.9 fewer steps than these two across team sizes and planners; the gap is largest against Capped Broadcast under ReAct.

Table 13: Further communication policies on C-WAH, steps and tokens. Values after ± are standard errors over five seeds.

	3 agents		4 agents		5 agents		6 agents	
Method	*Step*↓	*Usage*↓	*Step*↓	*Usage*↓	*Step*↓	*Usage*↓	*Step*↓	*Usage*↓
CoELA+Random	61.4 ±3.1	43k ±3k	55.9 ±2.2	54k ±2k	51.8 ±2.9	71k ±5k	54.1 ±3.4	90k ±6k
CoELA+Capped Broadcast	60.7 ±2.4	40k ±2k	58.6 ±3.3	55k ±4k	52.9 ±2.1	68k ±3k	53.2 ±2.7	86k ±5k
CoELA+Recency	64.8 ±3.6	58k ±4k	59.1 ±2.7	63k ±3k	53.4 ±2.5	74k ±5k	55.7 ±3.9	92k ±7k
CoELA+Distance	63.2 ±2.8	56k ±3k	62.3 ±4.1	67k ±5k	57.0 ±3.0	75k ±4k	55.2 ±2.6	90k ±6k
ReAct+Random	70.6 ±3.7	53k ±4k	63.1 ±2.5	63k ±3k	60.4 ±3.3	78k ±6k	56.2 ±2.4	88k ±5k
ReAct+Capped Broadcast	72.9 ±4.2	48k ±2k	68.4 ±3.0	61k ±4k	61.1 ±2.6	72k ±4k	62.7 ±3.8	87k ±6k
ReAct+Recency	71.5 ±3.1	46k ±3k	70.2 ±3.9	58k ±3k	65.3 ±3.4	69k ±5k	61.9 ±2.9	90k ±7k
ReAct+Distance	75.4 ±4.5	48k ±3k	69.7 ±3.2	56k ±4k	64.6 ±2.8	72k ±5k	66.3 ±3.5	88k ±6k

Direct reasoning as the router. Table 4 in Section 6.3 replaces the meta-belief with per-message language model reasoning over routing decisions, on the same six-agent C-WAH setting. The router model receives the message, its sender, the list of agents, the task goal, and the dialogue history the agents themselves see. It returns the recipient set and decides whether the router should send one query, and to whom; Appendix A.20 gives the prompt.

Replacing the meta-belief with per-message reasoning raises step count to 61.02 with gpt-oss-20b and 55.63 with GPT-5, and raises token usage to 124k–130k, about 1.5 times CREAC's 85k. Routing with the meta-belief takes fewer steps than direct reasoning both with the model CREAC itself uses for extraction and with a much stronger one. GPT-5 has higher target precision and request success and takes fewer steps than gpt-oss-20b on average, but these differences are within the standard errors, and both take more steps than CREAC. A more capable routing model therefore does not substitute for the meta-belief. Direct reasoning uses more tokens because its episodes are longer and every routing decision is a model call; GPT-5 spends many reasoning tokens on each call.

A.7 Router State Diagnostics

The analysis in this section is computed from the logs of the reported runs and requires no new rollouts.

Active queries and their targets. Table 14 counts the active queries under CREAC and under Broadcast+ActiveQuery, whom they reach, and how often the queried agent actually held the answer. Under Broadcast+ActiveQuery the router's meta-belief omits the deliveries beyond CREAC's selection (Appendix A.14). If these missing deliveries distorted its queries, its query count and targets would differ from CREAC's; Table 14 compares them. Broadcast+ActiveQuery issues slightly more queries than CREAC on both benchmarks (2.7 against 2.5 per episode on C-WAH, 6.1 against 5.9 on CREW-Wildfire), and its queried agents held the answer 83% and 71% of the time, against 87% and 76% for CREAC. The missing deliveries in its meta-belief therefore change its queries little and do not explain its higher step count.

Table 14: Active queries per episode and their targets, CoELA, six-agent C-WAH and CREW-Wildfire. Knower: highest-confidence agent in $\mathcal{S}_e$; stalest: fallback when $\mathcal{S}_e$ is empty.

Condition	Queries per episode	To knower (%)	To stalest (%)	Target held answer (%)
C-WAH, CREAC	2.5	71	29	87
C-WAH, Broadcast+ActiveQuery	2.7	66	34	83
CREW-Wildfire, CREAC	5.9	52	48	76
CREW-Wildfire, Broadcast+ActiveQuery	6.1	49	51	71

A.8 A Second Agent Backbone

Table 15 repeats the single-decision exposure test at $k = 8$ and the main six-agent C-WAH comparison with Qwen3-32B (Team, 2025) as the agent model, keeping gpt-oss-20b as the router's extraction model. Eight withheld messages lower the productive-action rate by 6.6 points and eight length-matched filler messages lower it by 5.6, so most of the harm comes from added length, as with gpt-oss-20b. CREAC takes 46.8 steps and 87k tokens, against 49.9 steps and 98k tokens for Broadcast.

Table 15: Six-agent C-WAH with Qwen3-32B as the agent model. Exposure columns: change in productive-action rate at $k = 8$, measured on CREAC's runs only, with 95% intervals. Bold: better value.

Method	*Step* ↓	*Usage* ↓	Withheld, $k = 8$	Filler, $k = 8$
CoELA+Broadcast	49.9 ±2.1	98k ±6k		
CoELA+CREAC	**46.8** ±2.1	**87k** ±5k	-6.6 $[-9.6, -3.6]$	-5.6 $[-8.4, -2.8]$

A.9 Hyperparameter Sensitivity and Update-Rule Ablation

The defaults ($\tau_{\text{routing}} = 0.3$, $\tau_{\text{query}} = 0.5$, $\lambda = 0.01$) were chosen on three-agent C-WAH and used unchanged in all other settings. Table 16 sweeps each hyperparameter over five values in the six-agent C-WAH setting with the others at their defaults, and Table 17 evaluates the joint extremes of the two thresholds. Every run uses five seeds; rows at the default values restate the six-agent CoELA+CREAC result of Table 1.

Table 16: Single-hyperparameter sensitivity on six-agent C-WAH. Ranges are min–max step counts across the five tested values.

Hyperparameter	Range tested	*Step*↓ (min–max)
Routing threshold τ_{routing}	0.1 to 0.9	43.28 – 53.82
Query threshold τ_{query}	0.1 to 0.9	43.28 – 59.43
Decay λ	0.001 to 0.1	43.28 – 50.49

Table 17: Joint extremes of the two thresholds on six-agent C-WAH.

$(\tau_{\text{routing}},\ \tau_{\text{query}})$	*Step* ↓
$(0.1,\ 0.9)$	48.5 ±3.8
$(0.9,\ 0.9)$	48.2 ±2.3
$(0.1,\ 0.1)$	53.7 ±3.8
$(0.9,\ 0.1)$	62.5 ±5.5

Table 18 reports all metrics for two decay rates near the default. A nearby rate such as $\lambda = 0.02$ gives similar results, while $\lambda = 0.05$ raises step count and lowers request success, since too fast a decay discards useful evidence.

In the single-hyperparameter sweeps, steps rise only at settings that reduce the information delivered to agents, which moves CREAC toward No-Comm; elsewhere performance stays at the level of the default. At the joint extremes, the two corners with a high query threshold degrade only mildly. The clearly harmful corner combines a high routing threshold with a low query threshold: the high routing threshold shrinks the recipient sets of QUERY messages while the low query threshold suppresses the compensating queries, so agents receive less information by both routes, which is the failure the analysis in Appendix A.3 predicts.

Table 18: Decay rates near the default ($\lambda = 0.01$) on six-agent C-WAH. TP: target precision; RS: request success. Bold: best value in each column, with ties all bold.

Decay	*Step* ↓	*TP* ↑	*RS* ↑	*Usage* ↓
$\lambda = 0.01$ (default)	**43.28** ±**2.0**	84.26 ±2.1	**92.22** ±**1.7**	**85k** ±**5k**
$\lambda = 0.02$	44.26 ±1.8	**84.84** ±**1.9**	91.52 ±2.3	**85k** ±**5k**
$\lambda = 0.05$	47.83 ±2.4	80.96 ±2.1	86.23 ±2.8	86k ±7k

Recipient-update rule. Table 19 tests a softer recipient update. It behaves like the default but enters a recipient's entry at 0.5 instead of 1 for information the router delivered; this gives a lower and flatter meta-belief and raises redundant delivery.

Table 19: Recipient-update variant on six-agent C-WAH. Bold: best.

Variant	*Step* ↓
CREAC (default)	**43.28** ±**2.0**
Softer recipient update	50.5 ±4.8

A.10 Active Queries: Target Policy, Cost, and Traces

When no agent has positive evidence. Table 20 compares the three policies available when the router has no positive entry for an entity. Suppressing the query raises step count to 59.6, because entities with no entry are exactly the ones no routing decision can help with. Broadcasting the question gives a comparable step count (56.8 against 59.6). The implemented exploration query takes fewer steps than both (43.28), because it turns a zero-evidence situation into evidence at the cost of one targeted message, as the analysis in Appendix A.3 predicts.

Table 20: Policies when no agent has positive evidence about an entity, six-agent C-WAH. The exploration-query row restates the default result of Table 1. Bold: best.

Policy when $\mathcal{S}_e$ is empty	*Step* ↓
Suppress the query	59.6 ±5.6
Broadcast fallback	56.8 ±3.2
Exploration query to the stalest agent (default)	**43.28** ±**2.0**

Cost of querying. We classified all 123 active queries issued across five seeds of six-agent C-WAH by hand. About 52% arrive while the target agent is in the middle of navigation or manipulation (*interruption*); in C-WAH an agent acts or speaks in a step but not both, so answering displaces that step's world action. About 13% ask for information the target does not in fact hold, because the meta-belief lags the world (*lagging meta-belief*). About 1% concern a goal already completed but not yet reported (*stale entry*). The remaining 34% recover information without significant interruption (*recovery*). The cost is therefore at most one displaced action per query, against the 16.3-step benefit of the exploration query over suppression in Table 20.

Traces. Three traces illustrate the categories. *Interruption:* the router queried Alice for the location of *pudding* while Alice was walking toward a kitchen cabinet to explore it; answering displaced that step's navigation action, and the plan resumed at the next step. *Stale entry:* the router re-queried *cutleryfork* after it had already been placed in the dishwasher, because the meta-belief still held a decayed positive entry and the completed goal had not yet been reported in dialogue. *Recovery:* with team-wide confidence about *apple* at zero, the exploration query to the stalest agent produced a SHARE reply at the next step, and the apple was placed within the following dozen steps. Linking satisfied goals to the meta-belief would remove the stale-entry case.

A.11 SINGLE-DECISION EXPOSURE EXPERIMENT

Single-decision protocol. Decision points are sampled uniformly from the six-agent C-WAH runs of CoELA+CREAC (five seeds, ten episodes). For a decision of agent i at step t, a message is *withheld* if another agent sent it in the window $[t-10, t]$, CREAC classified it as SHARE or QUERY, and CREAC did not deliver it to i. The k most recent withheld messages are inserted into the agent's message field in chronological order and in the same format as delivered messages. Filler messages come from episodes of a different goal family, mention no entity of the current goal, and match the total token length of the withheld messages they replace to within 10%. The removal condition deletes the messages CREAC delivered to i in the same window and is evaluated only at points where at least one such message exists. For each k, only points with at least k withheld messages are used. Each condition is sampled three times with the decoding settings of the main experiments. An action is productive if it moves toward, opens a container holding, grasps, or places a goal object that is not yet placed and that no other agent holds or is heading to, or if it explores a room or container that holds an undiscovered goal object. We fit a mixed-effects logistic regression of the productive indicator on condition and k with a random intercept per decision point. Table 21 gives the values plotted in Figure 1.

Table 21: Single-decision exposure on six-agent C-WAH: change in the productive-action rate, in points, relative to the delivered context, with 95% intervals.

Messages added	Withheld messages added	Length-matched filler added
$k = 1$	-1.4 [-3.6, 0.7]	-1.3 [-3.2, 0.9]
$k = 2$	-3.3 [-5.7, -0.8]	-2.4 [-4.8, -0.2]
$k = 4$	-5.9 [-8.5, -3.0]	-4.7 [-7.2, -1.8]
$k = 8$	-10.4 [-14.0, -6.9]	-8.2 [-11.8, -4.9]
Delivered messages removed	-7.4 [-10.6, -4.4]	

A.12 MEASURING THE EXTRACTION STEP

Extraction accuracy. We hand-labeled 300 messages from the main-experiment logs, with two independent annotators and adjudication of disagreements. Against these labels the information-content field reaches 0.827 accuracy, with 14 false negatives and 38 false positives out of 300; entity extraction reaches 0.707 precision, 0.906 recall, and 0.794 F1; and the intent field (`Share` versus `Request`) reaches 0.862 accuracy. The real error rate is therefore between about 9% and 29% depending on the field, close to the 10 to 30% range of the synthetic corruption study below, so the main results were obtained under real extraction noise of the level that the synthetic experiments cover.

Three-way classification. These numbers cover two binary fields of the extraction output. Routing uses the three-way label, and errors in that label differ in cost. SHARE and QUERY messages go to complementary recipient sets, so reading one as the other sends the message to the wrong agents. A SHARE or QUERY read as OTHERS is delivered to every agent, which costs efficiency but still reaches the agents that need it. We therefore report the full confusion matrix of the three-way label against adjudicated hand labels on the same 300 messages (Table 22).

Table 22: Three-way classification of the 300 hand-labeled messages, in message counts (rows: hand label; columns: router label). The SHARE–QUERY cells are the errors that send a message to the wrong agents.

	Router label			
Hand label	SHARE	QUERY	OTHERS	Recall
SHARE	113	27	14	0.734
QUERY	9	123	12	0.854
OTHERS	0	0	2	1.000
Precision	0.926	0.820	0.071	

The router assigns the correct three-way label to 238 of the 300 messages (79.3%; macro-F1 0.60). Almost every hand-labeled message is a SHARE or a QUERY; only 2 are OTHERS. The errors fall into two groups. A swap between SHARE and QUERY affects 36 messages (12.0%): 27 SHARE messages read as QUERY and 9 QUERY messages read as SHARE. A SHARE or QUERY read as OTHERS affects 26 messages (8.7%); these messages still reach every agent, including the ones that need them. By rule 4 of the prompt (Appendix A.20), a message receives the label OTHERS exactly when no evidence tuple is extracted, so these 26 errors come from missed entities. They also account for the low precision of OTHERS (0.071), which pulls the macro-F1 down, since only two messages are truly OTHERS. The next paragraph measures what all of these errors cost together.

Error propagation. To measure propagation directly we re-ran the same episodes under identical seeds with the extracted tuples replaced by the adjudicated gold tuples for every observed message: gold tuples reduce steps by 1.8 and raise target precision by 4.3 points.

Robustness protocol. All robustness variants use the same six-agent C-WAH setting, seed set, horizon, planner, and hyperparameters as the clean row. Tuple corruption is applied after extraction produces tuples and before the meta-belief is updated, so the forwarded message text is unchanged. For a corruption rate $\rho \in \{0.1, 0.2, 0.3\}$ we sample a fraction ρ of the non-empty tuples in each run. For each sampled tuple $(i, e, v_{\text{content}}, v_{\text{intent}})$ we corrupt exactly one field chosen uniformly at random: e is replaced with another goal entity from the same episode, v_{content} is flipped between INFO and NOINFO, or v_{intent} is flipped between the intent values `Request` and `Share`. Sender identity and timestep are preserved, which isolates the effect of extraction errors on the meta-belief.

For language perturbations we perturb the router-observed copy of selected entity-bearing messages before extraction, keeping the simulator state and agent policies fixed. The ambiguous/coreference condition rewrites explicit entity or location mentions into underspecified references such as pronouns, deictic phrases, or omitted room names. The misleading/stale correction condition injects a contradictory correction using an older entity-location or agent-entity association from the same episode log. Perturbed messages are processed by the same extraction and routing code as clean messages; no extra prompt or hand-coded correction is added.

Robustness to injected error. Table 23 reports performance as evidence tuples are corrupted at 10%, 20% and 30%, and under two language perturbations, following the protocol above. Target precision falls from 84.26 to 81.68, 78.25 and 74.79, and step count rises by 2.1, 4.4 and 4.8. At 30% corruption, above the real error rate of every field, CREAC takes comparable or fewer steps than CoELA with Broadcast (48.07 against 51.33). Ambiguous references raise step count to 51.90. A misleading or stale correction has the largest effect (target precision 68.30, 54.80 steps), because it directly contradicts what the meta-belief currently holds.

Table 23: Robustness of the router to injected extraction errors on six-agent C-WAH. TP: target precision; RS: request success. Bold marks the clean setting, which is best or tied for best in every column.

Setting	*Step* ↓	*TP* ↑	*RS* ↑	*Usage* ↓
CREAC (clean)	**43.28** ±2.0	**84.26** ±2.1	**92.22** ±1.7	**85k** ±5k
Tuple corruption				
Noise 10%	45.38 ±3.1	81.68 ±2.4	88.24 ±2.8	85k ±5k
Noise 20%	47.65 ±2.3	78.25 ±2.2	85.63 ±2.1	85k ±6k
Noise 30%	48.07 ±2.9	74.79 ±3.1	82.43 ±3.5	87k ±5k
Language perturbation				
ambiguous / coreference	51.90 ±3.0	73.80 ±3.6	84.92 ±3.0	90k ±5k
misleading / stale correction	54.80 ±3.6	68.30 ±4.2	80.20 ±4.3	93k ±8k

Cost and latency. Router calls account for 2.8% of total tokens, and the *Usage* numbers in all tables include them. No GPU is dedicated to the router: the single vLLM instance on one H100 serves the planning calls of all agents and the router's extraction calls, so removing the router would free 2.8% of the load on a server the deployment must provision anyway. Measured on CREW-Wildfire with

15 agents, extracting evidence from one message takes 0.62 s at the median and 1.4 s at the 95th percentile. Extractions for simultaneous messages are batched and overlap with the agent planning calls that dominate each step, so at 15 agents the router adds at most 1.9 s per decision step. Because messages are delivered at the next step, extraction runs in parallel with the environment step. End to end, an episode runs about 7% faster than Broadcast and within 3% of the heuristic baselines. Table 24 summarizes the measurements.

Table 24: Extraction measurements (gpt-oss-20b served by vLLM on one H100 shared by all agents and the router).

Quantity	Value
Information-content accuracy (300 hand-labeled messages)	0.827 (14 false negatives, 38 false positives)
Entity extraction precision / recall / F1	0.707 / 0.906 / 0.794
Intent accuracy (`Share` vs. `Request`)	0.862
Three-way label accuracy / macro-F1	0.793 / 0.60
Gold-tuple substitution under identical seeds: ΔStep / ΔTP	-1.8 steps / $+4.3$ points
Router share of total tokens (always included in *Usage*)	2.8%
Extraction latency per message, median / 95th percentile	0.62 s / 1.4 s
Added latency per decision step, 15 agents (batched, overlapped)	$\leq$ 1.9 s
Episode wall-clock vs. Broadcast / vs. heuristic routers	about 7% faster / within 3%

A.13 Router Pseudocode

Algorithm 1 states the complete router loop at one step. Agents, planners, and message contents are unchanged. The router issues at most one active query per step, with a cooldown of ten steps between consecutive queries. The query text q_e is the fixed template `[Router Query] {target}, what do you currently know about {entity}?`, with the target agent and the entity filled in; the direct-reasoning baseline uses the same text (Figure 5). When several entities below τ_{query} share the lowest $\max_j M_t(j, e)$, the router queries the entity whose entries were updated least recently, that is, the one with the smallest $\max_j \ell(j, e)$. When several agents tie as the target, the router picks one of them uniformly at random. Because all entries start at zero, the query condition holds for every entity at the first step, before any agent has spoken; the router makes an exception and sends no active query at that step. A message that names a teammate always reaches that teammate, whatever its entry in the meta-belief, so CREAC never withholds a message from an agent the sender addressed.

A.14 Baseline Details

CoTS. CoTS (Zu et al., 2025) is an end-to-end embodied multi-agent planning baseline built around cooperative tree search. It augments Monte Carlo tree search with LLM-guided discussion: agents explore candidate long-horizon plan branches, score them with reward and evaluation functions, and use a plan evaluation module to avoid unnecessary plan churn while still revising plans that become unsuitable. We run CoTS as a complete system. Its communication is coupled to joint tree search and follows the original assumption that communication does not consume environment time, so this comparison contrasts lightweight recipient selection with a planner-integrated method under the same task metrics.

REVECA. REVECA (Seo et al., 2025) is an LLM-based cooperative architecture for partially observable embodied tasks. It combines relevance estimation for memory filtering, relative-proximity-aware adaptive planning for collaborator-aware action selection, and trajectory-based validation to detect plans invalidated by other agents. We run it as a complete system because its communication, belief maintenance, and planning are tightly coupled. This separates its role from CREAC's: REVECA changes how an agent reasons and validates plans, whereas CREAC does not modify the planner and only decides which generated messages reach which teammates.

CoELA. CoELA (Zhang et al., 2023) provides the modular embodied-agent backbone used in our C-WAH experiments. Each agent combines partial perception, local memory, LLM planning, natural-language communication, and symbolic action execution. In the codebase, each agent

Algorithm 1 CREAC router loop at step t.

Require: overheard messages $\{m_{t,i}\}$; evidence bits B and last-update steps ℓ; thresholds $\tau_{\text{routing}}, \tau_{\text{query}}$; decay rate λ

for all messages $m_{t,i}$ **do** ▷ read and update the sender
  $(y_{t,i}, \mathcal{E}(m_{t,i}), v_{t,i}) \leftarrow \textsc{Extract}(m_{t,i})$ ▷ label, entities, content per entity
  if $y_{t,i} = \textsc{Share}$ **then**
    $B(i,e) \leftarrow 1$, $\ell(i,e) \leftarrow t$ for all $e \in \mathcal{E}(m_{t,i})$ ▷ (R1), sender
  else if $y_{t,i} = \textsc{Query}$ **then**
    $B(i,e) \leftarrow \mathbb{1}[v_{t,i}(e) = \textsc{Info}]$, $\ell(i,e) \leftarrow t$ for all $e \in \mathcal{E}(m_{t,i})$ ▷ (R2), sender
  end if
end for
$K_t(j,e) \leftarrow \exp(-\lambda\,(t - \ell(j,e)))$; $\quad M_t(j,e) \leftarrow B(j,e)\,K_t(j,e) \quad$ for all j, e
for all messages $m_{t,i}$ **do** ▷ choose recipients
  $\mathcal{N}_{t+1}(i) \leftarrow \emptyset$
  if $y_{t,i} = \textsc{Others}$ **then**
    $\mathcal{N}_{t+1}(i) \leftarrow \{j : j \neq i\}$ ▷ no selective routing
  else
    for all $e \in \mathcal{E}(m_{t,i})$ **do**
      if $y_{t,i} = \textsc{Query}$ **then**
        $\mathcal{R}_e \leftarrow \{j \neq i : M_t(j,e) \geq \tau_{\text{routing}}\}$ ▷ likely knowers
      else
        $\mathcal{R}_e \leftarrow \{j \neq i : M_t(j,e) < \tau_{\text{routing}}\}$ ▷ likely needers
      end if
      $\mathcal{N}_{t+1}(i) \leftarrow \mathcal{N}_{t+1}(i) \cup \mathcal{R}_e$
    end for
    $\mathcal{N}_{t+1}(i) \leftarrow \mathcal{N}_{t+1}(i) \cup \textsc{Named}(m_{t,i})$ ▷ teammates the sender addresses
    if $\mathcal{N}_{t+1}(i) = \emptyset$ **then**
      $\mathcal{N}_{t+1}(i) \leftarrow \{j : j \neq i\}$ ▷ fallback
    end if
    $B(j,e) \leftarrow \mathbb{1}[y_{t,i} = \textsc{Share}]$, $\ell(j,e) \leftarrow t$ for all $e \in \mathcal{E}(m_{t,i})$, $j \in \mathcal{N}_{t+1}(i)$ ▷ (R1)/(R2)
  end if
  deliver the original $m_{t,i}$ to $\mathcal{N}_{t+1}(i)$ at step $t{+}1$
end for
if t is not the first step **and** at least 10 steps have passed since the last active query **then**
  $\mathcal{E}_q \leftarrow \{e \in \mathcal{E} : \max_j M_t(j,e) < \tau_{\text{query}}\}$
  if $\mathcal{E}_q \neq \emptyset$ **then**
    $e^\star \leftarrow \arg\min_{e \in \mathcal{E}_q} \max_j M_t(j,e)$ ▷ ties: smallest $\max_j \ell(j,e)$
    $\mathcal{S}_{e^\star} \leftarrow \{j : B(j,e^\star) = 1\}$
    $j^\star \leftarrow \arg\max_{j \in \mathcal{S}_{e^\star}} K_t(j,e^\star)$ if $\mathcal{S}_{e^\star} \neq \emptyset$, else $\arg\min_j \ell(j,e^\star)$ ▷ ties: random
    send templated query $q_{e^\star}$ to $j^\star$; $\quad B(j^\star,e^\star) \leftarrow 0$, $\ell(j^\star,e^\star) \leftarrow t$ ▷ (R2), recipient
  end if
end if

tracks room-level progress, unchecked containers, ungrabbed target objects, action history, and recent dialogue; it first generates a short QUERY or SHARE message and then chooses among executable high-level actions such as exploration, checking containers, grabbing, putting, or sending the generated message. Its standard channel delivers every message to every teammate, which is the Broadcast condition. With a router, the planner, prompts, action parser, and message generator are unchanged; only the recipient set for each message is replaced by the routing policy.

ReAct. ReAct (Yao et al., 2022) interleaves natural-language reasoning with task-specific actions so the model can update its plan while interacting with the environment. We instantiate it as an alternative planner over the same C-WAH symbolic action space and message format used by CoELA. The implementation conditions the model on the current progress summary, dialogue history, and available actions, elicits reasoning before the final action choice, and constrains the output to one executable option. This gives a planner less tied to CoELA's cooperation assumptions, which lets us test whether the routing gains persist beyond a single agent architecture.

RCR-Router. RCR-Router (Liu et al., 2025) is a role-aware context-routing baseline with structured shared memory. The original method routes semantically relevant memory subsets to each agent by role, task stage, and token budget. Our C-WAH adapter preserves this without extra LLM calls: roles are inferred from recently mentioned entities, task stage is the current step divided by the episode horizon, shared memory is a rolling message window, and the budget maps to a maximum recipient count. For each message the router scores candidate recipients by role relevance, recency, and a stage weight that favors SHARE messages earlier and QUERY messages later; recipients are selected greedily under the budget.

No-Comm. No-Comm disables communication and uses no router. Each agent is initialized with the communication flag disabled, so it neither consumes incoming message fields nor receives prompt instructions for sending QUERY or SHARE messages. The action prompt is reduced to selecting a single environment action and no router is instantiated, so agents coordinate only through their private observations and the environment dynamics.

Broadcast. Broadcast uses the same agent and message-generation interface as the routed conditions but replaces selective routing with a fully connected rule. Whenever an agent outputs an action beginning with `[send_message]`, the environment extracts the text and passes it to the baseline router, which returns all non-sender agents as recipients, without entity parsing, meta-belief, active queries, or extra LLM calls. The environment records these deliveries with sender names and injects them into the next-step message field. This is the planners' standard channel, reported as Planner + Broadcast in Table 1.

Broadcast+ActiveQuery. Broadcast+ActiveQuery (Section 6.4) keeps the full router running, including extraction, meta-belief updates, and active queries under the default thresholds, and replaces only the delivery rule with the fully connected one: every message, including query replies, goes to all non-sender agents, so its delivery set is a superset of CREAC's. The recipient-side update of Section 4.2 is applied only to the recipient set that CREAC's rule selects, so the meta-belief evolves as it would under CREAC and the queries are triggered by the same rule. The condition uses the same episodes as the main experiments.

CREAC without active queries. This ablation runs CREAC's extraction, meta-belief updates and routing unchanged and disables the active-query step of Algorithm 1.

Random. Random keeps CREAC running in the background to obtain, for each message, the number of recipients CREAC would select, and delivers the message to that many non-sender agents drawn uniformly at random. Active queries are kept in Table 2 and Table 13 and kept or disabled as stated in Table 5. Random therefore matches CREAC's number of recipients per message and differs only in who receives each message.

Capped Broadcast. Capped Broadcast delivers every message to every teammate, as Broadcast does, but keeps in each agent's message field only the most recent messages whose total length fits a fixed budget on the accumulated dialogue history: 4096 tokens on C-WAH, at every team size, and 8192 tokens on CREW-Wildfire. It changes how much message history each agent reads and leaves who receives each message unchanged.

Recency. Recency is a lightweight entity-history router that approximates "who last mentioned what" without the meta-belief. Messages are parsed for explicit `<name> (id)` entity mentions, with a plain-word fallback. For QUERY messages the router sends the message to the most recent non-sender agent that mentioned each referenced entity; with no such history it falls back to broadcast. For all other messages it routes to agents that have not yet mentioned the referenced entities, then updates the entity-to-last-sender and entity-to-sender-set tables. Recency therefore tests whether mention order alone suffices, without decay, spatial reasoning, or router-side LLM calls.

Distance. Distance routes using last-known spatial locations. Before each routing step the environment provides the last-known positions of agents and entities. The router extracts mentioned entities and averages the known positions of all referenced entities. For QUERY messages it sends to the closest candidate agent to that averaged location, top-1 by default. For all other messages it sends to

the farthest half of the non-sender agents, on the heuristic that distant agents are least likely to already know local information. If entity mentions or position data are missing, it falls back to broadcast.

AgentPrune. AgentPrune (Zhang et al., 2024a) treats a multi-agent system as a spatial-temporal communication graph and prunes redundant or harmful edges with a low-rank-guided one-shot mask. The released code optimizes the masks on static question answering (QA) and coding benchmarks such as MMLU (Hendrycks et al., 2020), HumanEval (Chen et al., 2021), and GSM8K (Cobbe et al., 2021), where labeled answer quality is available. Because C-WAH is an online embodied-control setting without labels during an episode, our adapter reformulates it as an online router: a short warm-up broadcasts messages while sender-receiver edge scores are updated from entity relevance, recipient novelty, and goal-object overlap; after warm-up it applies a low-rank approximation and one-shot pruning to fix a sparse mask for the rest of the episode, with a top-scored fallback if a sender would otherwise have no outgoing recipient.

GDesigner. GDesigner (Zhang et al., 2024b) designs task-aware communication topologies using role/profile representations and a graph neural network (GNN) graph designer to decode sparse graphs. The official code evaluates this on static QA and coding benchmarks. In C-WAH all agents share the same base prompt, so our adapter replaces static role profiles with online profiles built from each agent's recent entity mentions, and the C-WAH goal objects act as the task node. For every message the adapter decodes a sparse topology by scoring candidate edges on task relevance, receiver-profile compatibility, sender-receiver complementarity, recipient novelty, and an anchor ring/layer prior. The number of recipients grows with task complexity, episode stage, and team size.

Direct reasoning. This baseline replaces the meta-belief with per-message language model reasoning over routing decisions. The model also decides when to send a query and to whom. The prompt states CREAC's query criteria and limits in words (at most one query per step, a cooldown of ten steps, a fixed query text), and the model applies them by its own reasoning, with no computed rule. GPT-5 runs at medium reasoning effort. We run it with gpt-oss-20b and with GPT-5 (Singh et al., 2025); results are in Appendix A.6 and the prompt is in Appendix A.20.

A.15 Qualitative Case Study

We describe three representative routing patterns observed in C-WAH.

Case 1: targeted QUERY routing. Broadcasting a QUERY to all possible knowers raises recipient-level context even when only a few replies are needed. In one episode Alice requested the location of *wine* (383). Although all teammates held the correct location at that time, only Bob and David were above the routing threshold in the meta-belief, so CREAC routed the QUERY to them and still reached an agent that could answer.

Case 2: proactive sharing. When an agent shares a newly discovered object, forwarding to agents who already hold the same evidence adds duplicate context without changing their decisions. In one trace David generated a SHARE about a target object, and CREAC delivered it only to Carol, the one teammate whose inferred state lacked that evidence.

Case 3: implicit assignment. QUERY messages asking for object interaction can create redundant motion if every agent receives them and several travel toward the same target. In one episode Frank requested transport of *pudding* (373). CREAC delivered the QUERY to Carol, the only teammate above the routing threshold for the object. The selected agent could respond while the other agents continued their current activities.

A.16 User Study Details

The study evaluates how people perceive collaboration with agent teammates under different communication policies in C-WAH, using six CoELA-based agents. We compare three conditions: CREAC, Broadcast, and No-Comm. Neither the condition name nor the underlying policy was revealed to participants. Figure 3 shows the interface.



Figure 3: Screenshot of the user study interface.

Participants. Thirty participants took part, with mean age 26.4 ± 3.9 (range 19–36), 24 male and 6 female, and 19 reporting prior experience interacting with AI agents or robots. All responses were collected anonymously and used only for statistical analysis and research.

Conditions and protocol. The study uses a within-subject design in which every participant plays all three conditions. In Broadcast, generated messages go to all teammates. In No-Comm, inter-agent communication is disabled. In CREAC, messages are routed only to selected teammates. The participant joined the six agents as a seventh teammate, with the same action space as the agents and a text box for sending messages. In the CREAC condition the participant's messages passed through the router like any other message, and the participant saw the agent messages that the condition's policy delivered to them. The order of the three conditions was counterbalanced across participants, with each of the six possible orders used by five participants. Each participant played the same episode under all three conditions, with the start positions and object locations randomized in a controlled way in each session.

Survey procedure. After each collaboration session participants completed a questionnaire with four open-ended questions and four 7-point Likert items. The open-ended questions asked about positive and negative aspects of the collaboration, the effect of agent communication such as questions and reports, how easy it was to understand agent intentions, and suggestions for improvement. The Likert items measured trust, appropriateness, usefulness, and efficiency, where 1 is "strongly disagree", 4 is "neutral", and 7 is "strongly agree".

Full survey instruction.

> **C-WAH User Study Survey**
>
> Hello! Thank you for participating in this survey on user experience with AI agents.
>
> This survey is designed to collect your opinions about the collaboration process that you have just experienced with AI agent teammates. Your responses will serve as important data for improving the performance of AI agents and designing better user experiences.
>
> All responses will be used only for statistical analysis and research purposes. Personal information will be processed anonymously, so please answer honestly and freely. The expected completion time is approximately 1–2 minutes.

Qualitative questions.

1. What were the best aspects and the most inconvenient aspects of collaborating with the agent teammates?
2. How did the communication style of the agent teammates, such as questions and reports, affect the collaboration process? For example, was the amount of communication too much or too little?

3. How easy or difficult was it to understand what actions the agent teammates would take next and what their intentions were?
4. If you have any ideas or suggestions for improving the collaboration ability of the agents, please write them freely.

Quantitative questions.

1. **Trust.** I felt that I could trust the agent teammates.
2. **Appropriateness.** The agent teammates performed appropriate actions that matched my intention.
3. **Usefulness.** The agent teammates were helpful in collaboratively solving the task.
4. **Efficiency.** The agent teammates helped me achieve the goal more efficiently.

Analysis. Table 7 gives the ratings with medians. For example, on usefulness, 21 of 30 participants rate CREAC above Broadcast, 5 tie, and 4 rate it below. Table 25 gives, for each dimension, the Friedman test across the three conditions, with tie correction, and Kendall's $W = \chi_F^2/(N(C-1))$ for $N = 30$ participants and $C = 3$ conditions. CREAC has the highest mean rank on every dimension, and the test is significant on trust, appropriateness and usefulness. Table 26 gives the Holm-corrected pairwise contrasts, which we interpret only on dimensions where the Friedman test is significant: on trust, appropriateness and usefulness, CREAC is rated significantly above both Broadcast and No-Comm. On efficiency the Friedman test is not significant, so we claim no difference on this dimension.

Table 25: Friedman test across CREAC, Broadcast and No-Comm, 30 participants, 2 degrees of freedom, with tie correction, Kendall's W, and mean ranks (CREAC / Broadcast / No-Comm; higher is better).

Dimension	χ_F^2	p	Kendall's W	Mean rank
Trust	7.14	0.028	0.119	2.37 / 1.83 / 1.80
Appropriateness	7.14	0.028	0.119	2.33 / 1.70 / 1.97
Usefulness	12.52	0.002	0.209	2.40 / 1.58 / 2.02
Efficiency	3.94	0.140	0.066	2.23 / 1.78 / 1.98

Table 26: Holm-corrected Wilcoxon signed-rank tests on the two pairwise contrasts per dimension. r: matched-pairs rank-biserial correlation; N': number of non-tied pairs.

Dimension	CREAC vs Broadcast	CREAC vs No-Comm
Trust	$p = 0.043$, $r = 0.48$, $N' = 27$	$p = 0.043$, $r = 0.54$, $N' = 23$
Appropriateness	$p = 0.012$, $r = 0.63$, $N' = 25$	$p = 0.027$, $r = 0.52$, $N' = 23$
Usefulness	$p = 0.002$, $r = 0.75$, $N' = 25$	$p = 0.036$, $r = 0.51$, $N' = 21$
Efficiency	$p = 0.024$, $r = 0.59$, $N' = 23$	$p = 0.083$, $r = 0.42$, $N' = 21$

A.17 Limitations in Full

Cooperative premise. The router reads an emitted message as evidence about the sender's knowledge state, which presumes that agents do not deliberately withhold task-relevant information and ask only about what they lack. Strategic or adversarial withholding is outside the scope of this paper.

One bit per agent and entity. The meta-belief records whether an agent holds valid information about an entity, which does not separate knowing where an object is from holding it or planning to fetch it. A finer meta-belief, with one entry per aspect of an entity's state, is a direct extension, most useful for entities whose state changes continuously, such as a fire front in CREW-Wildfire.

Closed entity schema. The meta-belief indexes a fixed set of goal-related entities derived once per task. Extending it to open-world entity discovery would let the router operate without a prior task structure.

Centralized mediation. All communication passes through a single router. We do not study a decentralized version of the router.

A.18 Compute Resources

All experiments are evaluation-only simulator rollouts; no model training or fine-tuning is performed. Environment simulation, rollout orchestration, logging, and routing code were executed on a local workstation with an AMD Ryzen 7 3800XT CPU, 32GB RAM, and one NVIDIA GeForce RTX 3090 GPU with 24GB memory. LLM inference for gpt-oss-20b was served on a single NVIDIA H100 GPU using vLLM (Kwon et al., 2023); this server is shared by all agent planning calls and the router's extraction calls, and no GPU is dedicated to the router. Qwen3-32B, the agent model of Appendix A.8, was also served on a single NVIDIA H100 GPU. GPT-5, used only in the direct-reasoning comparison, was accessed through the OpenAI API at medium reasoning effort.

A.19 Existing Assets and Licenses

We use existing benchmark, simulator, and model assets only for academic evaluation, and we verified the public release pages and license files. The C-WAH code distributed in the CoELA repository lists Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International in `https://github.com/UMass-Embodied-AGI/CoELA/blob/master/cwah/LICENSE.md`. The VirtualHome API used with C-WAH, on the `wah` branch, lists the same license in `https://github.com/xavierpuigf/virtualhome/blob/wah/LICENSE.md`. The CREW framework used by CREW-Wildfire (Hyun et al., 2025) is installed from `https://github.com/generalroboticslab/CREW`, whose public `LICENSE` file lists Apache License 2.0. The gpt-oss-20b model (Agarwal et al., 2025) is released with license `apache-2.0` on `https://huggingface.co/openai/gpt-oss-20b`, with an accompanying usage policy. The Qwen3-32B model (Team, 2025) is released under Apache License 2.0 on `https://huggingface.co/Qwen/Qwen3-32B`. GPT-5 is accessed through the OpenAI API under OpenAI's terms of use.

A.20 Prompt Templates

Figure 4 gives the prompt of the router's classification and extraction call. It returns the three-way label, the referenced entities with their content and intent, and the teammates the sender addresses; rule 4 derives the label from the evidence tuples, so a tuple with intent `Request` makes the message a QUERY. Figure 5 gives the prompt of the direct-reasoning baseline. Figure 6 gives the prompt that derives the entity list $\mathcal{E}$ from the goal description; it runs once per task on the same model as the extraction call.

```
You are the Observation Module of CREAC.
Convert one observed natural-language message into structured evidence for CREAC's routing
     and table update. Use only what is explicitly stated or strongly implied by the
     message. Do not infer hidden world state.

Input:
- sender: the agent who emitted the message
- agents: the list of all agent names
- task_entities: the list of task-relevant entities
- message: the natural-language message

Output exactly one valid JSON object with the following schema:
{
  "sender": "<sender>",
  "label": "Share" or "Query" or "Others",
  "explicit_targets": ["<agent name>", "..."],
  "evidence": [
    {"entity": "<one entity from task_entities>",
     "content": "Info" or "NoInfo",
     "intent": "Share" or "Request"}
  ]
}

Rules:
1. Evidence tuples
- Each item in "evidence" is one tuple (sender, entity, content, intent).
- Use only entities from task_entities. If none is referenced, return "evidence": [].
- If several entities are referenced, return one item per entity. Remove duplicates.

2. Content label
- "Info" if the sender gives concrete information about the entity, including negative
     facts.
  "The wine is in the kitchen." -> Info.  "The wine is not in the kitchen." -> Info
- "NoInfo" if the sender asks, expresses uncertainty, or states a lack of knowledge.
  "Where is the wine?" -> NoInfo.  "I don't know where the wine is." -> NoInfo

3. Intent label
- "Request" if the message asks for information, asks a teammate to do something, or states
     that the sender lacks knowledge.
- "Share" if the message only passes on information: an observation, the sender's own state
     or what it holds, or the subgoal it is pursuing.
- "Bob, bring the wine from the kitchen." -> content = Info, intent = Request.
- "I don't know where the wine is." -> content = NoInfo, intent = Request.
- Never output content = NoInfo with intent = Share.

4. Message label
- "Others" if "evidence" is empty.
- "Query" if at least one evidence item has intent = Request.
- "Share" otherwise.

5. explicit_targets
- Agents the sender addresses or instructs directly. Exclude the sender.
- Do not include agents that are only mentioned.
- "Alice, where is the wine?" -> ["Alice"].  "Bob knows where the wine is." -> []

Now parse the following message.
sender: {sender}
agents: {agents}
task_entities: {task_entities}
message: {message}
```

Figure 4: Prompt of the router's classification and extraction call.

```
You are the message router of a team of embodied agents that share one goal.
Goal: {goal}
Agents: {agents}
Dialogue so far: {dialogue}
New message from {sender}: {message}

1. Recipients. Decide which agents other than the sender should receive this
message. An agent should receive a report if it does not yet know what the
report says, and a request if it can probably answer or carry it out. Do not
include agents for whom the message is not useful.

2. Query. Decide whether the router itself should ask one agent about one
goal-related entity. Ask only when, judging from the dialogue so far, no agent
is likely to still hold valid information about that entity, or the team is
unlikely to learn it otherwise. Choose the agent most likely to know it, or,
if nobody seems to know it, the agent that has said the least about it
recently. At most one query per step, and none if fewer than 10 steps have
passed since the last query. The query text is fixed:
"[Router Query] {target}, what do you currently know about {entity}?"

Output exactly one JSON object:
{"recipients": ["<agent name>", ...],
 "query": null or {"target": "<agent name>", "entity": "<one goal-related entity>"}}
```

Figure 5: Prompt of the direct-reasoning baseline (Appendix A.6).

```
A team of embodied agents shares the goal below.
List the goal-related entities: the entities whose state the goal specifies, for example
     the objects the goal asks the team to place. Use the names as they appear in the goal.
Output exactly one JSON list of strings.

Goal: {goal}
```

Figure 6: Prompt that derives the entity list $\mathcal{E}$ from the goal description, run once per task.