EMCoP: Efficient Multi-Task Cooperative Perception with Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Cooperative perception helps connected agents understand their surroundings by exchanging information from different viewpoints. Existing cooperative perception methods often optimize sharing for individual perception tasks. However, agents need to run multiple tasks simultaneously to support safety-critical decisions such as path planning and obstacle avoidance. For instance, combining a surrounding object's position, its expected movement and road layout helps an agent plan a feasible path. A naive extension of state-of-the-art frameworks would be to exchange information for all tasks independently, imposing heavy strain on shared bandwidth. However, tasks can provide overlapping information: e.g., detection and segmentation can both indicate that a region is occupied. If either provides enough information to plan around it, sharing additional features from both may improve their separate predictions without improving the path. Motivated by this insight, we introduce EMCoP, which trains a single reinforcement-learning policy to consider all tasks together when deciding how much information each task should share in the current scene. By evaluating the decision space through its effect on the agent's planned path, the policy learns which combinations keep path errors and predicted collisions low while minimizing communication. EMCoP further reduces communication by identifying recurring situations where recently received information remains sufficient. We evaluate EMCoP in cooperative driving on V2X-Sim. Extensive experiments demonstrate that EMCoP is able to reduce communication volume by up to 85% relative to full sharing, where every agent broadcasts all task information without selection.

1 Introduction

A robotic agent's sensors provide a limited view of its surroundings. Other agents may observe regions beyond that view, see objects hidden behind obstructions, or observe distant objects more clearly. Cooperative perception brings these observations together through messages that contain information derived from each agent's sensor inputs, such as features that describe the surrounding scene (Li et al., 2021). Sharing this information can improve perception, but transmitting all features from every agent requires substantial bandwidth. Selective communication therefore aims to preserve useful information while reducing the volume of the exchanged data (Hu et al., 2022).

Existing methods reduce communication by selecting spatial regions using confidence maps (Hu et al., 2022), selecting features and adapting the amount shared (Mukhopadhyay et al., 2026), or learning which agents exchange information and when (Liu et al., 2020a;b). A key limitation of these approaches is that they optimize sharing for individual perception tasks to preserve task-specific accuracy (Figure 1). However, for navigation, the practical value of shared information lies in how well it supports trajectory planning and obstacle avoidance (Hu et al., 2023; Yu et al., 2024; Liu et al., 2024). These decisions can require information from several tasks together. For example, detection locates another agent but cannot establish whether it will cross the currently planned path; motion prediction adds its expected movement and segmentation the available space, together guiding a path that accounts properly for the other agent's movement, thus preventing collision.

Considering tasks together also creates opportunities to reduce communication. For example, detection and segmentation can both indicate that a region is occupied. If either provides enough information for the planner to avoid it, sending additional features for both may improve their separate predictions without improving the path. Optimizing sharing independently for each task does not account for this overlapping benefit. Moreover, different tasks can require different amounts of

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.

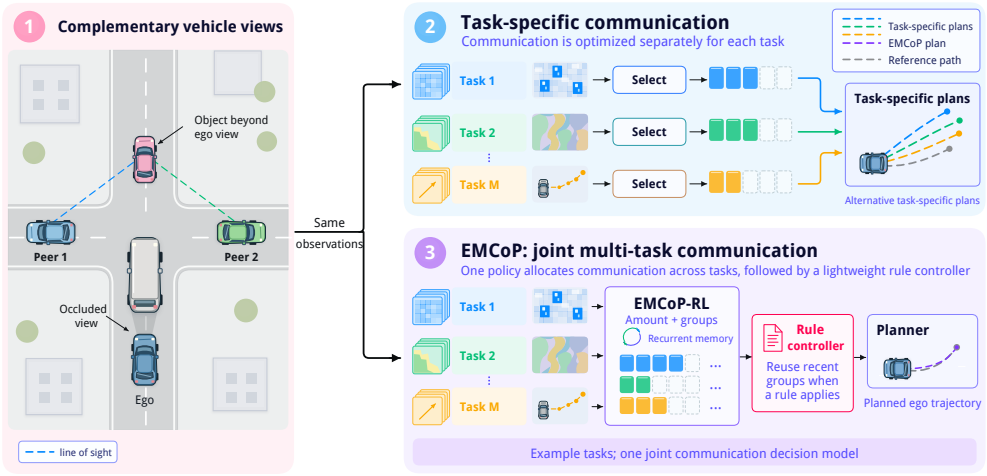


Figure 1: **Multi-task cooperative perception with EMCoP. Left:** Peer vehicles observe information hidden from the ego vehicle. **Upper right:** Task-specific communication methods optimize sharing for an individual task (Liu et al., 2020b;a; Hu et al., 2022; Yang et al., 2023; Mukhopadhyay et al., 2026). **Lower right:** EMCoP jointly chooses what to share across multiple tasks, then uses rules to reuse recent messages. The combined information supports one accurate ego plan.

transmitted data. When their information offers similar planning benefits and sending both adds little value over sending just one, joint optimization can preserve planning quality by choosing information requiring fewer bytes. This raises the question: *how much information should each task share, and which parts to send, to preserve trajectory planning quality with reduced communication?*

We introduce EMCoP, a framework that jointly learns how much information each agent shares for each task and which parts to send, balancing communication cost against the tasks' combined contribution to trajectory planning. The value of a transmission depends on the current scene, confidence in task predictions, transmission cost, and information already available at receivers. For example, as another agent approaches the planned path, movement updates may become more valuable than additional detail about unchanged surroundings. Keeping the same sharing allocation can spend communication on details that add little while missing an important update.

Sharing decisions also affect later plans because receivers retain the information they receive. What is sent now therefore changes what needs to be sent later. To learn these connected decisions, EMCoP trains a single reinforcement-learning policy, EMCoP-RL, using current observations and recent sharing history. RL is well suited to this setting because it learns to balance competing objectives while accounting for how each decision affects later outcomes (Leong et al., 2020; Funk et al., 2021). The policy chooses how much information each agent shares for each task and which parts to send. We evaluate these decisions using a trained trajectory planner that combines the available local and shared task information to predict the agent's future path. Feedback on trajectory error, predicted collisions, and transmitted bytes guides the policy toward sharing choices that minimize the number of bytes transmitted relative to broadcasting information from all tasks without selection, while preserving trajectory planning quality over time. We assess trajectory planning quality using average displacement error (ADE) and predicted collision rate, defined in Section 4.

Our offline comparisons show that some updates selected by EMCoP-RL can be skipped with little change in trajectory quality when receivers already hold recent information. For example, if the area along the planned path remains unchanged and no nearby agent is expected to enter it, another update may add no benefit. EMCoP compares otherwise identical runs with and without selected updates, measuring communication savings and changes in trajectory quality. We derive rules from these comparisons, retaining those supported across additional training scenes to reduce communication with little change in trajectory quality. At runtime, a lightweight controller skips policy-selected updates only when a rule matches and every receiver has a sufficiently recent copy; otherwise, the policy's selection remains unchanged.

We evaluate EMCoP in cooperative driving on V2X-Sim with three tasks: detection, segmentation, and motion prediction (Li et al., 2022). We compare against four adapted baselines: Who2com,

When2com, Where2comm, and CooperTrim. We use two reference settings: *local-only* uses only the planning agent's own observations, while *all-on* combines these with complete task information broadcast by every peer without selection. We report ADE gain retained, the percentage of the reduction in trajectory error achieved by all-on over local-only that a method preserves. Local-only defines 0% retained gain and all-on defines 100%. EMCoP retains 85.65% of this improvement using 14.88% of volume as consumed in all-on communication, and 96.54% using 26.92%.

Our main contributions can be summarized as:

- **Joint multi-task RL.** A single policy jointly chooses each task's sharing rate and content, balancing transmission costs against the tasks' combined effect on trajectory quality.
- **Rule-based update skipping.** Offline trials yield a lightweight controller to skip policy-selected updates using current conditions while ensuring all receivers hold recent copies.
- **Planning-based evaluation.** A common trajectory planner compares communication volume, ADE, and predicted collision rates, with extensive task-combination ablations.

2 Related Work

Communication-efficient cooperative perception. Existing methods improve fusion (Chen et al., 2019; Xu et al., 2022b;a; 2023), learn communication partners and groups (Liu et al., 2020b;a), and distill collaboration graphs (Li et al., 2021). Where2comm and CooperTrim selectively share spatial regions and features (Hu et al., 2022; Mukhopadhyay et al., 2026). Other work addresses delays (Yang et al., 2023; Lei et al., 2022), compression (Hu et al., 2024), reconstruction (Wang et al., 2023a), and heterogeneous tasks (Gao et al., 2025). These approaches leave joint allocation across separate task messages unresolved when planning benefits overlap and byte costs differ.

Cooperative prediction and planning. Prior systems connect cooperation to prediction (Wang et al., 2020; 2025), integrate perception and prediction for planning (Chen & Krähenbühl, 2022; Hu et al., 2023), and enable cooperative driving (Cui et al., 2022; Yu et al., 2024). CoDriving and Select2Drive further guide communication using driving needs (Liu et al., 2024; Huang et al., 2025). However, integrating tasks for planning does not itself determine how much each task should transmit or which information adds value given other tasks' contributions.

Learning decisions and reusing experience. Policy optimization, adaptive penalties, and memory support sequential cost-quality decisions (Schulman et al., 2017; 2016; Stooke et al., 2020; Cho et al., 2014; Hausknecht & Stone, 2015); ReSCo-SAC allocates resources across semantic tasks (Zhang et al., 2026). Experience reuse improves later action selection and execution through feedback and skills (Shinn et al., 2023; Wang et al., 2023b); e.g., FaSTA* reduces planning computation by reusing previously successful tool sequences (Gupta et al., 2026). These methods reuse experience to improve action selection, reducing computation in their respective domains; EMCoP applies it to cooperative perception, skipping unnecessary updates to reduce transmission costs.

EMCoP addresses overlapping planning benefits and unequal byte costs by jointly learning sharing amounts and groups across tasks using their combined planning outcomes. Paired trials yield rules that skip unnecessary updates when existing information remains sufficient for planning.

3 EMCoP: Design and Methodology

We address how agents can minimize communication volume across tasks, measured relative to broadcasting information from all tasks all-on, while preserving trajectory quality. In EMCoP, we train a single RL policy, EMCoP-RL, to jointly choose how much information each agent shares for each task and which parts to send, based on their combined planning benefit and communication cost. We then derive rules from offline comparisons with and without selected updates, allowing a lightweight controller to skip policy-selected updates when recently received information remains sufficient. Figure 2 illustrates the framework.

3.1 Problem setup

Given N cooperating agents performing M tasks, agent 0 plans its trajectory using its own observations and information shared by peers $i \in \{1, \ldots, N-1\}$. At frame t, each peer's information for task $s \in \{1, \ldots, M\}$ is partitioned into G_s selectable groups. A group is a portion of one sender's task information that the policy selects as a whole. For a feature map with C_s channels and spatial size $H_s \times W_s$, a channel group contains several channels over the full map; a spatial group contains

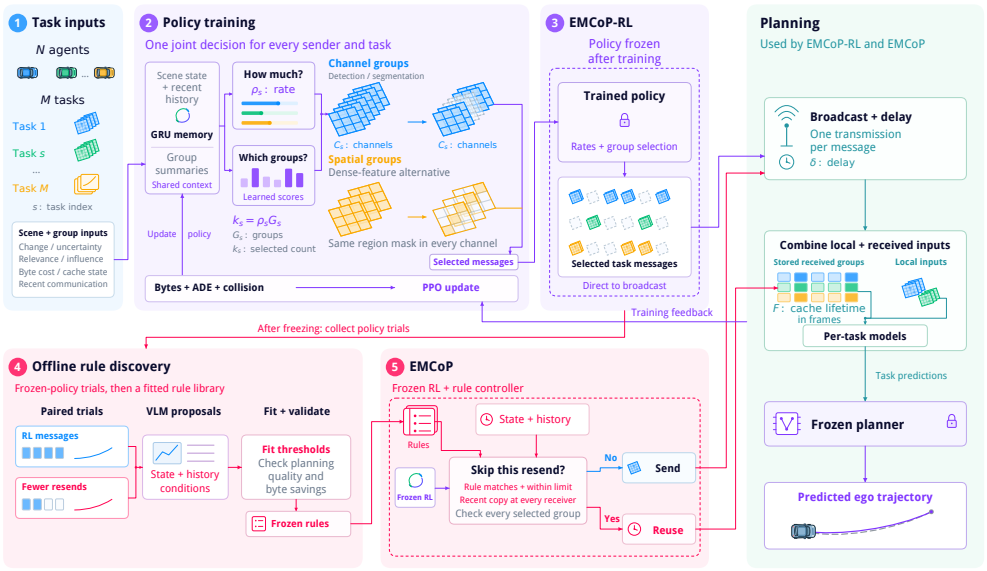


Figure 2: **EMCoP framework. Top:** The policy jointly selects sharing amounts and groups, trained using serialized bytes and planning outcomes. Detection and segmentation use channel groups; motion uses complete histories grouped by region. **Bottom:** Paired omission trials support fitted rules that filter the frozen policy's selections. **Right:** Both variants use the same task-processing and fixed-planner stages. Dense messages use delayed delivery and receiver caches; motion uses history withholding and forecast reuse.

all channels within one region. Detection and segmentation use channel groups. For example, dividing a 64-channel map into 16 equal groups gives four channels per group; selecting one group shares all spatial values in those four channels. For motion prediction, each sender groups together the past observations of objects currently located in the same region. The policy jointly chooses how many groups each sender shares for each task and which groups to send.

Each selected dense message is broadcast once. Receivers combine their local features with delivered peer features. Separate receiver caches retain groups for at most F frames from generation; delayed updates can affect subsequent predictions. Task dependencies are preserved: changing detection information changes the tracks used for motion prediction. We measure communication volume as the total serialized bytes of transmitted task messages, including headers.

For a T-frame training scene, let B_t and B_t^{all} denote transmitted task-message bytes and full-sharing bytes at frame t. We measure trajectory quality with Average Displacement Error (ADE), denoted by E, and collision rate, denoted by C. ADE is the mean distance between predicted and recorded future positions; collision rate is the fraction of predicted trajectories flagged by the collision check.

Let E_L, E_A, and C_A denote the scene's local-only ADE, all-on ADE, and all-on collision rate, with collision rates expressed as fractions. We train the policy π_θ, with learned parameters θ, to minimize the mean communication fraction while targeting 90% retention of the local-only to all-on ADE improvement and a collision rate at most one percentage point above all-on:

$$\min_{\theta} \mathbb{E}_{\pi_\theta}\left[\frac{1}{T}\sum_{t=0}^{T-1}\frac{B_t}{B_t^{\text{all}}}\right] \quad \text{s.t.} \quad E(\pi_\theta) \le E_L - 0.90(E_L - E_A), \quad C(\pi_\theta) \le C_A + 0.01. \tag{1}$$

During training, exceeding either limit increases its corresponding penalty weight.

3.2 Trajectory Planner

To evaluate sharing decisions, we use a separately trained trajectory planner that combines task predictions, their availability and age, ego speed, and a navigation command. The detection-and-segmentation branch forms the base planner, which uses query-based attention to refine and score candidate trajectories and selects the highest-scoring one (Yu et al., 2024). A motion branch adds a bounded correction to this selected trajectory when motion information is available. We train the base planner first, then fit the motion branch while keeping the base planner fixed. Supervision

uses recorded trajectories, collision feedback, and losses encouraging additional task information to improve trajectory quality. All task models and the complete planner remain fixed during communication learning. Appendix B provides the detailed architecture and training losses.

3.3 Rate and group selection

A joint communication controller runs EMCoP-RL, using one recurrent policy for all senders and tasks. Its scene vector o_t summarizes predicted objects, task uncertainty, received information, pending deliveries, recent communication, and driving context. For group g of task s at sender i and frame t, the descriptor vector $c_{t,i,s,g}$ contains measurements of content, change, uncertainty, cost, and freshness. Dense descriptors also estimate how task predictions change when features are removed, using feature-gradient products (Molchanov et al., 2017). We compute the gradient-based descriptors for each scene and cache them for reuse during replay.

At frame t, the scene vector o_t summarizes predicted objects, task uncertainty, received information, pending deliveries, recent communication, and driving context. A gated recurrent unit (GRU), a learned memory module (Cho et al., 2014), combines this vector with the previous context h_{t-1} to produce the updated context $h_t = \mathrm{GRU}(o_t, h_{t-1})$, starting from $h_{-1} = 0$. Task-specific output modules share this context and reuse their weights across senders. For each sender and task, the rate module uses the mean group descriptor to choose how many information groups to send. Each group is a portion of task information, such as feature channels, which the policy selects and sends together. The utility module assigns each group a learned score indicating its priority for transmission. These scores guide which groups are selected within the chosen amount:

$$\begin{gathered}\bar{c}_{t,i,s} = \frac{1}{G_s}\sum_{g=1}^{G_s} c_{t,i,s,g}, \qquad p_{t,i,s} = \mathrm{softmax}(a_s(h_t, \bar{c}_{t,i,s}))\,, \\ \rho_{t,i,s} \sim p_{t,i,s}, \qquad k_{t,i,s} = G_s\rho_{t,i,s}, \qquad u_{t,i,s,g} = f_s(h_t, c_{t,i,s,g}).\end{gathered} \tag{2}$$

Here a_s and f_s are task-specific rate and group-scoring modules. The distribution $p_{t,i,s}$ assigns probabilities to allowed fractions in $\mathcal{R}_s \subseteq [0,1]$. The sampled fraction $\rho_{t,i,s}$ determines how many groups sender i transmits for task s at frame t: $k_{t,i,s} = G_s\rho_{t,i,s}$. Allowed fractions ensure that $k_{t,i,s}$ is an integer. The score $u_{t,i,s,g}$ determines each group's selection probability. During training, the policy samples a rate and then k distinct groups using their scores. Suppressing the time, sender, and task indices, the probability of the ordered group choices is

$$p(g_1, \ldots, g_k \mid \rho, h, c) = \prod_{j=1}^{k} \frac{\exp(u_{g_j})}{\sum_{g \in \mathcal{G}_s \setminus \{g_1, \ldots, g_{j-1}\}} \exp(u_g)}. \tag{3}$$

Here $\mathcal{G}_s = \{1, \ldots, G_s\}$, c collects the sender-task pair's descriptors, and g_j is its jth selected group. Sampling order defines the training likelihood, and the selected set determines the transmitted groups. Evaluation chooses the most likely rate and the highest-scoring groups. Equal fractions can have different byte costs across tasks, so training charges actual serialized messages. Joint planning feedback couples these choices across tasks and time.

3.4 Policy training

We train with proximal policy optimization (PPO) (Schulman et al., 2017). Complete scene replays maintain separate local-only, all-on, and policy-controlled states, including delayed information and detector-derived tracks. Let E_t be the frame's ADE and $s_E > 0$ a fixed distance scale. For collision feedback, we compute a score between 0 and 1 at each predicted future position, increasing as the agent approaches or overlaps an obstacle. Its mean over valid future positions is $\bar{q}_t$. The rewards are

$$r_t^V = -B_t / B_t^{\mathrm{all}}, \qquad r_t^E = -E_t / s_E, \qquad r_t^C = -\bar{q}_t. \tag{4}$$

The mean score provides collision feedback during training, while C is the fraction of trajectories whose score reaches 0.5 at any valid future position. Recorded future trajectories and obstacles provide reward targets after prediction.

A training-only critic uses the scene vector o_t and its own memory to estimate v_t^j, the discounted sum of future rewards for each component $j \in \{V, E, C\}$. Generalized advantage estimation (Schulman et al., 2016) computes

$$\delta_t^j = r_t^j + \gamma v_{t+1}^j - v_t^j, \qquad A_t^j = \delta_t^j + \gamma\lambda A_{t+1}^j. \tag{5}$$

Here δ_t^j is the critic's one-step prediction error, γ discounts future rewards, and λ controls the weighting of later prediction errors. We compute these quantities backward through the T-frame scene, with $v_T^j = A_T^j = 0$. Each advantage A_t^j estimates how much better or worse the sharing decision performed than expected, accounting for later rewards. PPO uses weighted combination,

$$A_t = A_t^V + \mu_E A_t^E + \mu_C A_t^C, \tag{6}$$

where $\mu_E, \mu_C \geq 0$ are weights for trajectory-error and collision-score penalties and are updated after each scene using the quality targets through a Lagrangian-style update (Stooke et al., 2020):

$$\mu_E \leftarrow [\mu_E + \alpha\,(E - [E_L - 0.90(E_L - E_A)])]_+\,, \quad \mu_C \leftarrow [\mu_C + \alpha\,(C - [C_A + 0.01])]_+\,. \tag{7}$$

Here E and C are the measured scene ADE and collision rate, with collision rates expressed as fractions; α is the update step size, and $[x]_+ = \max(0, x)$. Each PPO update uses one scene. We normalize A_t within that scene before the clipped PPO update. Recurrent states are detached between frames, so recurrent gradients cover one step, while rewards from later frames contribute to the advantages. The training-only critic uses the scene vector and its own recurrent memory. Appendix C provides further details on the RL policy, rate and group selection, and training.

3.5 Rules for skipping updates

After freezing EMCoP-RL, paired training-scene trials compare the policy with and without eligible updates omitted for a fixed interval. Both runs start from the same state and continue through the remaining scene, measuring saved bytes and changes in ADE and collision rate, including later tracking and planning effects. We derive conditions on current measurements and recent history that save bytes with small trajectory-error changes and no observed collision increase.

At frame t, the frozen policy selects groups $S_{t,i,s}$ for sender i and task s. A lightweight controller filters these selections using a rule whose stored conditions are all satisfied. Eligible groups $\mathcal{E}_{t,i,s} \subseteq S_{t,i,s}$ must already have a copy at every receiver that was generated at most F_{r} frames earlier. Here F_{r} is the maximum copy age allowed when skipping an update and cannot exceed the cache lifetime F. Groups are ordered freshest first, with ties resolved by smaller feature change, lower estimated influence on task predictions, and lower policy score. The group index breaks any remaining ties consistently. The matching rule with greatest fitted mean byte saving supplies $d_{t,s}$, the omission limit per sender and task; this is zero when no rule is active or required history is unavailable. The omitted and transmitted sets are

$$O_{t,i,s} = \mathrm{First}_{d_{t,s}}(\mathcal{E}_{t,i,s})\,, \qquad S_{t,i,s}^{\mathrm{send}} = S_{t,i,s} \setminus O_{t,i,s}, \tag{8}$$

where First_d returns the first d eligible groups, or all if fewer exist. Rules filter detection and segmentation; motion retains the policy's selection and its dependence on detection. Appendix F details the motivation, trials, rule discovery, fitting, and execution.

4 Experiments

We evaluate EMCoP on V2X-Sim to answer three questions: (1) Does sharing information from multiple tasks together improve trajectory planning? (2) How does EMCoP compare with existing methods in the amount of data transmitted and planning quality? (3) How do the policy components and update-skipping rules contribute to these results?

4.1 Experimental setup

We use vehicle-only V2X-Sim (Li et al., 2022), with up to five vehicles per scene. We train on 79 scenes and evaluate at 850 time steps across ten validation scenes. At each step, the planner predicts the vehicle's path over the next three seconds.

For EMCoP, frozen DiscoNet models (Li et al., 2021) provide detection and segmentation. AB3DMOT (Weng et al., 2020) links detections over time to build object histories used by CMP for motion prediction (Wang et al., 2025). Each method supplies the planner with task predictions based on its local observations and received information. The planner's weights and evaluation procedure remain the same across methods.

Metrics. After predicting a three-second path using the planner, we compare it with the vehicle's recorded path over that interval (Hu et al., 2023; Yu et al., 2024). ADE, denoted by E, is the mean distance between predicted and recorded positions. Collision rate is the percentage of predicted paths that intersect an obstacle at its recorded position at the same time.

Local-only uses the planning vehicle's own observations, while all-on also receives complete task information from all peers (they are the two extreme cases). Their ADE values are $E_L = 2.2368\,\text{m}$ and $E_A = 2.1702\,\text{m}$, respectively. We report retained gain: percentage of error reduction relative to that provided by all-on, that each method yields:

$$G(E) = 100\frac{E_L - E}{E_L - E_A}. \tag{9}$$

Note that matching local-only gives 0% retained gain; matching all-on gives 100%. The extreme collision rates (from local and all on) are 21.88% and 12.94%, respectively. Appendix D.3 provides the metric definitions.

Communication volume is the total transmitted bytes as a percentage of the bytes transmitted under all-on broadcasting, including handshake messages, headers, and selection indices. Each broadcast is counted once, and each directed response is counted separately. Methods with multiple exchanges can therefore exceed 100% volume.

Implementation. Detection and segmentation feature maps have 256 and 512 channels, respectively, on a 32×32 spatial grid. Each map is divided into 16 equal groups of consecutive channels, with 16 channels per detection group and 32 per segmentation group. For motion, a 4×4 grid divides objects into 16 regions based on their current positions. Selecting a region shares the histories of all objects in it, with 10 time steps and 11 values per step. For each of the three tasks, the policy instructs each sender to transmit 0%, 25%, 50%, 75%, or 100% of its 16 groups at each time step; for example, 25% means four groups. The planner predicts 15 positions at 0.2-second intervals.

Communication delays. During training and evaluation, detection and segmentation messages receive reproducible delays sampled uniformly from $[0, 300)$ ms (Lei et al., 2022; Yang et al., 2023). Messages are processed every 200 ms. Receivers can use stored information until it is one second old. For motion prediction, we simulate delay in getting knowhow, by hiding each peer's object observations from the most recent 0.4 s. Between updates, the planner keeps using the earlier prediction of where the object is expected to move. That prediction can be reused for at most one second after it was made. Appendices D and G.2 provide protocol details and delay ablations, respectively.

Update-skipping rules. We fit multiple rules on a subset of training scenes and select the rule library using held-out training scenes. The library is frozen before evaluation. At runtime, the controller checks whether current scene conditions and history match a rule. A matching rule can skip policy-selected updates only when every receiver already holds a copy no more than 0.4 s old.

Baselines. We compare EMCoP with adapted versions of CooperTrim, Where2comm, When2com, and Who2com. Each baseline selects information separately for detection and segmentation, while sharing all motion information. For When2com and Who2com, we adapt their training protocols to jointly train task models and communication selectors.

Settings. We report two settings: low uses a policy which favors lower comm. volume, while high uses a policy that favors higher retained gain. For each setting, we report results with just the policy (EMCoP-RL) and with the update-skipping rule added to that policy (EMCoP).

Table 1: Task sharing combinations. Combining all three tasks improves trajectory planning quality over any single task or pair. Rate is the mean aggregate Mbps across all transmitting agents and tasks shared. Volume, gain, and collision are in %.

Coop. tasks	Rate (Mbps) ↓	Volume ↓	Gain ↑	ADE (m) ↓	Collision ↓
Det	125.84	33.29	50.84	2.2029	13.41
Seg	251.67	66.58	35.89	2.2129	20.35
Motion	**0.45**	**0.12**	2.16	2.2353	20.59
Det + Seg	377.51	99.87	66.54	2.1924	**12.94**
Det + Motion	126.33	33.42	70.91	2.1895	13.06
Seg + Motion	252.12	66.70	47.36	2.2052	20.24
All three	378.00	100.00	**100.00**	**2.1702**	**12.94**

4.2 TASK COMBINATIONS

We test whether sharing information from one or two tasks can match the planning quality of sharing information relating to all three. Relating to each row in Table 1, peers share all information from the tasks in Column 1, which the planning vehicle uses alongside its own detection and segmentation results. Sharing information on all three tasks yields the lowest ADE. Motion information provides little improvement over local-only when shared by itself, but reduces ADE when shared along with detection or segmentation information. These results show that sharing

information from multiple tasks together improves planning over a case where information is shared relating to a single task; this motivates communication decisions that consider tasks together.

4.3 Main results

EMCoP improves path quality-communication volume tradeoff. Table 2 shows that EMCoP achieves higher gain at lower communication volume than most baselines that select information independently for each task. EMCoP (high) uses $\approx 69\%$ less communication volume than CooperTrim, with comparable ADE and lower collision rate. EMCoP's low and high settings achieve higher gain with lower communication volume than Where2comm at 5% & 10% budgets, respectively. Where2comm at 40% achieves similar gain to EMCoP (low), but exceeds all-on communication volume because of multiple communication rounds and replies to individual receivers. Both EMCoP settings achieve higher gain with lower communication volume than When2com and Who2com.

Table 2: Planning results under delayed delivery. EMCoP reaches all-on performance with substantially lower volume. Where2comm labels are configured budgets; volume counts all transmissions.

Method	Volume (%)↓	Gain (%)↑	ADE (m) ↓	Collision (%)↓
Local-only	0.00	0.00	2.2368	21.88
All-on broadcast	100.00	100.00	2.1702	12.94
Where2comm, 5%	19.30	77.81	2.1849	13.76
Where2comm, 10%	37.25	79.73	2.1837	13.88
Where2comm, 40%	144.94	87.51	2.1785	13.18
CooperTrim	88.18	96.80	2.1723	12.94
When2com	222.58	38.20	2.2113	19.65
Who2com	126.32	39.85	2.2102	16.24
EMCoP-RL (low)	20.49	88.19	2.1780	13.18
EMCoP-RL (high)	32.67	**96.93**	**2.1722**	**12.82**
EMCoP (low)	**14.88**	85.65	2.1797	13.18
EMCoP (high)	26.92	96.54	2.1725	**12.82**

Predicted collision rates remain close to all-on. EMCoP-RL (high) and EMCoP (high) have slightly lower predicted collision rates than all-on (12.82% versus 12.94%). Full sharing can include noisy or conflicting features that worsen planning. EMCoP learns to omit these features and achieve a lower collision rate compared to all-on.

Cheaper information can offer greater planning gain. Table 1 shows that sharing detection information alone provides more planning gain than segmentation alone even with about half the communication volume. However, sharing both improves planning gain beyond each by itself. This motivates a joint policy that learns when cheaper task information is sufficient and when additional information from a more expensive task is needed.

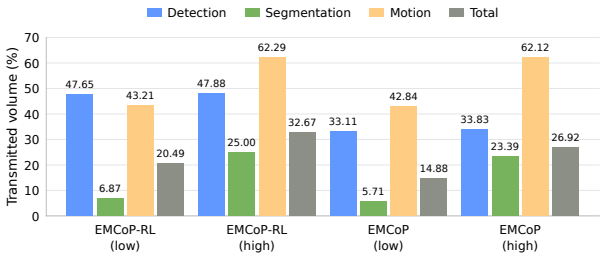


Figure 3: Task volume (%) relative to each task's all-on bytes; total uses all-on bytes across all tasks. Higher-volume settings transmit more segmentation information.

Higher-quality settings allocate more communication to segmentation. Figure 3 shows that EMCoP's high settings share substantially more segmentation information than the low settings, while detection sharing changes little. This might be due to additional information about available driving space helping the planner refine its path beyond what object locations already provide.

4.4 Qualitative trajectories

Figure 4 compares predicted paths from EMCoP, Where2comm, and CooperTrim on one val scene. For the prediction at 5.0 s, EMCoP achieves ADE of 2.092 m, compared with 2.256 m for Where2comm and 2.377 m for CooperTrim. Only EMCoP predicts a collision-free path.

4.5 Policy ablations

Table 3 separates the contributions of update skipping, message representation, learned amounts, and planning feedback. All policy ablations are conducted without the rule controller.

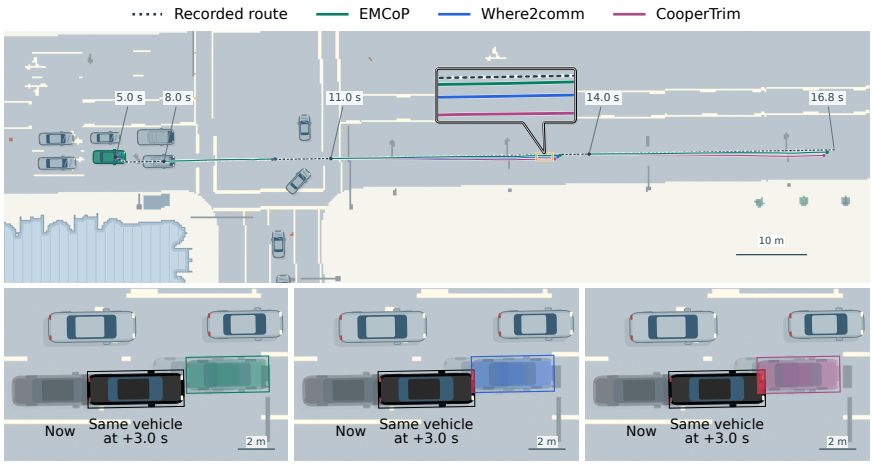


Figure 4: **Qualitative comparison over a sample path from validation set. Top:** The dotted line shows ground-truth (GT) trajectory that methods aim to match. Colored lines show trajectories predicted by each method. EMCoP is able to match the GT trajectory more closely than baselines. **Bottom:** Close-ups compare vehicle positions three seconds after the predictions made at 5.0 s under identical traffic: EMCoP (left), Where2comm (center), and CooperTrim (right). Faint vehicles are current positions; colored vehicles are predicted ego positions; black is another vehicle's future position. Red regions mark collisions. EMCoP predicts a more accurate path without colliding.

Utilizing Recent information reduces update needs. We compare each frozen policy with and without the same rules to test whether recent receiver information can replace selected updates. Skipping these updates reduces communication at both settings with only a small ADE penalty and unchanged collision rates.

Spatial versus channel completeness. EMCoP-RL shares selected feature channels with their full spatial maps. We compare this with separately trained policies that share selected spatial regions with all channels, or send or skip whole messages. Channel selection retains the most planning gain. These results suggest that preserving full spatial information in selected channels benefits planning more than preserving all channels within selected regions.

Table 3: **EMCoP ablations.** Each component of EMCoP, including rules, learned rates, and planning feedback improve the quality-volume trade-off. Volume, gain, and collision are percentages.

Setting	Volume ↓	Gain ↑	ADE (m) ↓	Collision ↓
EMCoP (low)	**14.88**	85.65	2.1797	13.18
EMCoP (high)	26.92	96.54	2.1725	12.82
EMCoP-RL (low)	20.49	88.19	2.1780	13.18
EMCoP-RL (high)	32.67	96.93	2.1722	12.82
Spatial groups	15.53	67.70	2.1917	14.00
Whole messages	44.49	33.76	2.2143	20.12
Random rates + learned groups	49.61	96.69	2.1724	12.78
w/o ADE reward	46.26	92.79	2.1750	**12.71**
w/o collision reward	32.39	**97.26**	**2.1720**	13.18

Learning sharing amounts avoids excess traffic. Recent baselines such as Where2comm select content under configured budgets. We replace the learned sharing amount for each sender and task with a uniformly random choice of 0%, 25%, 50%, 75%, or 100% of its information groups, while retaining the learned selection of which groups to send. Learning these amounts reduces communication at similar planning gains and collision rates.

Planning feedback provides complementary guidance. We retrain with either ADE or collision feedback removed to test their distinct contributions. Without ADE feedback, the policy lowers collision rate but retains lower ADE gain and uses more communication than high-quality EMCoP-RL. Without collision feedback, ADE and volume remain similar, but collision rate increases.

5 Conclusion

The idea behind EMCoP is to coordinate sharing across tasks according to their combined benefit for the planned path, avoiding transmissions that add little beyond information already available. EMCoP learns what each task shares using scene information, recent history, and planning feedback, then applies lightweight rules to skip unnecessary updates. The low-communication setting of EMCoP transmits only 14.88% of the data required to broadcast all task information, while retaining 85.65% of the reduction in path error that full sharing provides over local observations alone. The higher-quality setting retains 96.54% of this reduction using 26.92% of the data.

6 AI USE STATEMENT

For manuscript preparation, we used generative AI tools to improve the clarity and readability of the text and, to a limited extent, to assist with reviewing and understanding relevant literature. We also used a vision-language model (VLM) to propose candidate conditions for the offline rule-discovery procedure described in Section 3.5 and Appendix H. These candidate proposals were subsequently evaluated, fitted, and validated using the experimental procedure described in the paper. The authors reviewed all AI-assisted content and take responsibility for the final content of this work, including its motivation, methodology, experimental results, claims, organization, and text.

7 ETHICS STATEMENT

This work investigates communication-efficient cooperative perception for connected autonomous agents. Our experiments use the publicly available V2X-Sim dataset and pretrained perception models, without involving human participants or private personal data. The findings are based on offline evaluation; deployment in safety-critical autonomous systems would require further real-world testing and safety validation.

8 REPRODUCIBILITY STATEMENT

To support reproducibility, we document the dataset, pretrained models, training protocols, hyper-parameters, and communication and evaluation settings in Section 4.1 and Appendices B, C, and D. Rule fitting and the proposal prompt are detailed in Appendices F and H. All code and trained checkpoints will be released.

REFERENCES

Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17222–17231, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Chen_Learning_From_All_Vehicles_CVPR_2022_paper.html.

Qi Chen, Xu Ma, Sihai Tang, Jingda Guo, Qing Yang, and Song Fu. F-Cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds. In *ACM/IEEE Symposium on Edge Computing*, pp. 88–100, 2019. doi: 10.1145/3318216.3363300. URL https://arxiv.org/abs/1909.06459.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014. URL https://arxiv.org/abs/1406.1078.

Jiaxun Cui, Hang Qiu, Dian Chen, Peter Stone, and Yuke Zhu. Coopernaut: End-to-end driving with cooperative perception for networked vehicles. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17231–17241. IEEE, 2022. doi: 10.1109/cvpr52688.2022.01674. URL http://dx.doi.org/10.1109/CVPR52688.2022.01674.

Niklas Funk, Dominik Baumann, Vincent Berenz, and Sebastian Trimpe. Learning event-triggered control from data through joint optimization. *IFAC Journal of Systems and Control*, 16:100144, 2021. ISSN 2468-6018. doi: 10.1016/j.ifacsc.2021.100144. URL http://dx.doi.org/10.1016/j.ifacsc.2021.100144.

Xiangbo Gao, Runsheng Xu, Jiachen Li, Ziran Wang, Zhiwen Fan, and Zhengzhong Tu. Stamp: Scalable task- and model-agnostic collaborative perception. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 54656–54676, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/87d91d52272c3166315ca20b18519b9e-Paper-Conference.pdf.

Advait Gupta, Rishie Raj, Dang Nguyen, and Tianyi Zhou. FaSTA*: Fast-slow toolpath agent with subroutine mining for efficient multi-turn image editing. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yhhbL9T1QB.

Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *AAAI Fall Symposium Series*, 2015. URL https://arxiv.org/abs/1507.06527.

Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://arxiv.org/abs/2212.10156.

Yue Hu, Shaoheng Fang, Zixing Lei, Yiqi Zhong, and Siheng Chen. Where2comm: Communication-efficient collaborative perception via spatial confidence maps. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2209.12836.

Yue Hu, Juntong Peng, Sifei Liu, Junhao Ge, Si Liu, and Siheng Chen. Communication-efficient collaborative perception via information filling with codebook. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2405.04966.

Jiahao Huang, Jianhang Zhu, Rongpeng Li, Zhifeng Zhao, and Honggang Zhang. Select2drive: Pragmatic communications for real-time collaborative autonomous driving. *arXiv preprint arXiv:2501.12040*, 2025. URL https://arxiv.org/abs/2501.12040.

Zixing Lei, Shunli Ren, Yue Hu, Wenjun Zhang, and Siheng Chen. Latency-aware collaborative perception. In *European Conference on Computer Vision*, 2022. URL https://arxiv.org/abs/2207.08560.

Alex S. Leong, Arunselvan Ramaswamy, Daniel E. Quevedo, Holger Karl, and Ling Shi. Deep reinforcement learning for wireless sensor scheduling in cyber-physical systems, 2020. URL https://arxiv.org/abs/1809.05149.

Yiming Li, Shunli Ren, Pengxiang Wu, Siheng Chen, Chen Feng, and Wenjun Zhang. Learning distilled collaboration graph for multi-agent perception. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2111.00643.

Yiming Li, Dekun Ma, Ziyan An, Zixun Wang, Yiqi Zhong, Siheng Chen, and Chen Feng. V2X-Sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. *IEEE Robotics and Automation Letters*, 2022. URL https://arxiv.org/abs/2202.08449.

Genjia Liu, Yue Hu, Chenxin Xu, Weibo Mao, Junhao Ge, Zhengxiang Huang, Yifan Lu, Yinda Xu, Junkai Xia, Yafei Wang, and Siheng Chen. Towards collaborative autonomous driving: Simulation platform and end-to-end system. *arXiv preprint arXiv:2404.09496*, 2024. URL https://arxiv.org/abs/2404.09496.

Yen-Cheng Liu, Junjiao Tian, Nathaniel Glaser, and Zsolt Kira. When2com: Multi-agent perception via communication graph grouping. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020a. URL https://arxiv.org/abs/2006.00176.

Yen-Cheng Liu, Junjiao Tian, Chih-Yao Ma, Nathan Glaser, Chia-Wen Kuo, and Zsolt Kira. Who2com: Collaborative perception via learnable handshake communication. In *IEEE International Conference on Robotics and Automation*, 2020b. URL https://arxiv.org/abs/2003.09575.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.06440.

Shilpa Mukhopadhyay, Amit Roy-Chowdhury, and Hang Qiu. COOPERTRIM: Adaptive data selection for uncertainty-aware cooperative perception. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.13287.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID lagrangian methods. In *International Conference on Machine Learning*, 2020. URL https://proceedings.mlr.press/v119/stooke20a.html.

Binglu Wang, Lei Zhang, Zhaozhong Wang, Yongqiang Zhao, and Tianfei Zhou. Core: Cooperative reconstruction for multi-agent perception. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8676–8686, 2023a. doi: 10.1109/ICCV51070.2023.00800.

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

Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II*, pp. 605–621, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58535-8. doi: 10.1007/978-3-030-58536-5_36. URL https://doi.org/10.1007/978-3-030-58536-5_36.

Zehao Wang, Yuping Wang, Zhuoyuan Wu, Hengbo Ma, Zhaowei Li, Hang Qiu, and Jiachen Li. CMP: Cooperative motion prediction with multi-agent communication. *IEEE Robotics and Automation Letters*, 2025. URL https://arxiv.org/abs/2403.17916.

Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D multi-object tracking: A baseline and new evaluation metrics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020. URL https://arxiv.org/abs/1907.03961.

Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. V2X-ViT: Vehicle-to-everything cooperative perception with vision transformer. In *European Conference on Computer Vision*, 2022a. URL https://arxiv.org/abs/2203.10638.

Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Li, and Jiaqi Ma. OPV2V: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In *IEEE International Conference on Robotics and Automation*, 2022b. URL https://arxiv.org/abs/2109.07644.

Runsheng Xu, Zhengzhong Tu, Hao Xiang, Wei Shao, Bolei Zhou, and Jiaqi Ma. CoBEVT: Cooperative bird's eye view semantic segmentation with sparse transformers. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 989–1000, 2023. URL https://proceedings.mlr.press/v205/xu23a.html.

Dingkang Yang, Kun Yang, Yuzheng Wang, Jing Liu, Zhi Xu, Rongbin Yin, Peng Zhai, and Lihua Zhang. How2comm: Communication-efficient and collaboration-pragmatic multi-agent perception. In *Advances in Neural Information Processing Systems*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4f31327e046913c7238d5b671f5d820e-Abstract-Conference.html.

Haibao Yu, Wenxian Yang, Jiaru Zhong, Zhenwei Yang, Siqi Fan, Ping Luo, and Zaiqing Nie. End-to-end autonomous driving through V2X cooperation. *arXiv preprint arXiv:2404.00717*, 2024. URL https://arxiv.org/abs/2404.00717.

Ying Zhang, Leibing Yan, and Guang Li. Reinforcement learning-based resource allocation for multi-task multi-codebook semantic communications. *Physical Communication*, 75:102982, 2026. ISSN 1874-4907. doi: https://doi.org/10.1016/j.phycom.2025.102982. URL https://www.sciencedirect.com/science/article/pii/S1874490725003854.

A NATIVE TASK QUALITY

Table 4 shows improved detection AP@0.5, segmentation mIoU, and motion coverage over local outputs under both EMCoP-RL policies before rules. Coverage is the fraction of eligible targets with matched forecasts. Motion minADE and minFDE independently minimize mean and final-position errors across predicted paths per matched target. All-on produces forecasts for more targets, so its motion errors are measured over a different set. Lower motion errors alone therefore do not establish better prediction if fewer targets receive forecasts. Appendix D.3 defines these metrics and coverage-aware error.

Table 4: Task quality under EMCoP-RL's selected groups. Both policies improve detection, segmentation, and motion coverage over local outputs. AP, mIoU, and coverage are percentages; motion errors are meters and conditional on matched targets. Local motion is a task diagnostic.

Setting	AP@0.5 ↑	mIoU ↑	Coverage ↑	minADE ↓	minFDE ↓
Local task outputs	63.32	37.57	7.39	**3.026**	**4.953**
EMCoP-RL (low)	71.19	38.69	18.97	4.044	6.394
EMCoP-RL (high)	73.29	40.66	22.03	4.298	6.880
All-on task outputs	**74.86**	**41.41**	**27.72**	4.619	7.500

B PLANNER ARCHITECTURE AND SUPERVISION

B.1 INPUTS AND TRAJECTORY PREDICTION

The fixed planner in Section 3.2 converts the available task predictions into one future ego path. Figure 5 separates its detection-and-segmentation base from the motion correction. The base receives a detection confidence map, eight segmentation probability maps, and four availability and age maps. A convolutional encoder produces 64 scene tokens. A query conditioned on ego speed and the navigation command attends to these tokens, refines three trajectory candidates, and scores them. The highest-scored candidate gives the base trajectory.

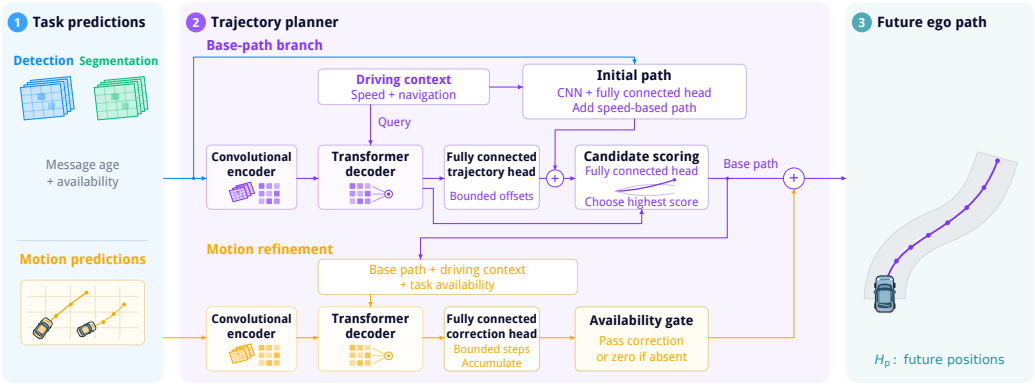


Figure 5: **Fixed trajectory planner.** Detection and segmentation determine a base path; available motion information adds a bounded correction. This structure lets the same planner evaluate different shared task combinations, with its weights fixed during communication learning.

The motion raster contains occupancy at one, two, and three seconds; x and y displacements at one and three seconds; and confidence. Both encoders append two coordinate maps, with x increasing across columns and y across rows, each normalized to $[-1, 1]$. The motion query combines the base path, speed, command, and detection/segmentation availability. Its predicted increments are

bounded by $0.20\tanh(\cdot)$ and accumulated into a path correction, giving

$$\hat{p}_h = \hat{p}_h^{\text{base}} + m_M \sum_{j=1}^{h} 0.20\tanh(z_j), \qquad h = 1, \ldots, H. \tag{10}$$

Here $\hat{p}_h \in \mathbb{R}^2$ is the predicted position, $z_j \in \mathbb{R}^2$ is the motion branch output, m_M indicates available motion input, and $H = 15$. The bound applies to each correction increment. Table 5 gives the dimensions.

Table 5: Planner components. The motion branch augments the base with a smaller attention module; coordinate maps supply explicit spatial positions.

Component	Detection/segmentation base	Motion correction
Task maps	$13 \times 32 \times 32$	$8 \times 32 \times 32$
Encoder input with x/y maps	$15 \times 32 \times 32$	$10 \times 32 \times 32$
Encoded representation	64 tokens, width 256	64 tokens, width 192
Attention decoder	3 layers, 8 heads	2 layers, 6 heads
Feed-forward width	512	384
Attention dropout during fitting	0.1	0.1
Correction increment	$0.15\tanh(\cdot)$	$0.20\tanh(\cdot)$
Output horizon	15 positions at 0.2-second intervals	

B.2 Supervised losses

We fit the base first, then freeze it while training the motion branch. The following losses specify the three-task fitting stage. Let $\mathcal{S}$ contain local-only, each cooperative single task, each pair, and all three tasks. Their unnormalized weights are respectively 0.1, 0.2 per single, 0.6 per pair, and 1.0; normalization gives weights ω_S summing to one. Predictions from two delay schedules are averaged for each condition before applying the losses.

For sample b, condition S, and future step h, let $\hat{p}_{bSh}$ be the prediction, p_{bh} the recorded position, and $v_{bh} \in \{0, 1\}$ its validity. Define $\phi(x) = x^2/2$ for $|x| < 1$ and $\phi(x) = |x| - 1/2$ otherwise, applied separately to x and y coordinates. With $w_h = 1 + (h-1)/(H-1)$ and $\Delta p_h = p_h - p_{h-1}$, the per-sample losses are

$$\ell_{bS}^{\text{traj}} = \frac{\sum_{h=1}^{H} v_{bh} w_h \sum_{a\in\{x,y\}} \phi(\hat{p}_{bSh,a} - p_{bh,a})}{\max(1, \sum_{h=1}^{H} v_{bh} w_h)}, \tag{11}$$

$$\ell_{bS}^{\text{inc}} = \frac{\sum_{h=2}^{H} v_{bh} v_{b,h-1} \sum_{a\in\{x,y\}} \phi(\Delta\hat{p}_{bSh,a} - \Delta p_{bh,a})}{\max(1, \sum_{h=2}^{H} v_{bh} v_{b,h-1})}. \tag{12}$$

For minibatch size n, $L_{\text{traj}} = \sum_S \omega_S n^{-1} \sum_b \ell_{bS}^{\text{traj}}$, and L_{inc} uses the same average. These terms fit recorded positions and their successive changes.

Let $Q_{bS} = \max_{h:v_{bh}=1} q_{bSh}$ be the largest soft collision score along a predicted path, using the metric defined in Appendix D.3. If I_S indexes the $k = \lceil 0.35n \rceil$ largest scores for condition S, collision-tail training uses

$$L_{\text{tail}} = \sum_{S\in\mathcal{S}} \omega_S \frac{1}{k} \sum_{b\in I_S} Q_{bS}. \tag{13}$$

This emphasizes the highest-scoring examples within each task condition. The loss penalizes geometric overlap along predicted paths.

For each of the nine single-to-pair or pair-to-all edges $S \subset S'$, let E_{bS} be the sample ADE and let sg hold a value fixed during gradient calculation. Task-set consistency averages the following terms

over samples and edges:

$$\ell_E(b,S,S') = \frac{[E_{bS'} - \mathrm{sg}(E_{bS}) + 0.001]_+}{0.01}, \tag{14}$$

$$\ell_Q(b,S,S') = \frac{[Q_{bS'} - \mathrm{sg}(Q_{bS}) + 0.0005]_+}{0.005}, \tag{15}$$

$$L_{\mathrm{set}} = \frac{1}{18n} \sum_b \sum_{(S,S')} \big(\ell_E(b,S,S') + \ell_Q(b,S,S')\big). \tag{16}$$

Here $[x]_+ = \max(0,x)$. Positive margins encourage adding task information to improve the path. For aligned all-task predictions (E_b, Q_b) and predictions with motion shuffled between samples $(E_b^{\mathrm{sh}}, Q_b^{\mathrm{sh}})$, the alignment term is

$$L_{\mathrm{align}} = \frac{1}{2n} \sum_b \left(\frac{[E_b - \mathrm{sg}(E_b^{\mathrm{sh}}) + 0.001]_+}{0.01} + \frac{[Q_b - \mathrm{sg}(Q_b^{\mathrm{sh}}) + 0.0005]_+}{0.005} \right). \tag{17}$$

It encourages use of scene-specific motion information. The full three-task fitting loss is

$$L_{\mathrm{planner}} = L_{\mathrm{traj}} + 4L_{\mathrm{tail}} + 0.1L_{\mathrm{inc}} + 4L_{\mathrm{set}} + 2L_{\mathrm{align}}. \tag{18}$$

Training uses batch size 32, initial learning rate 10^{-4}, weight decay 10^{-4}, and gradient-norm clipping at 5.

B.3 Training inputs and replay metadata

Detection and segmentation use pretrained DiscoNet experts. CMP is adapted with detector-derived histories and supervised future targets. Communication training and evaluation build histories from detector predictions and causal associations. All task models and the planner remain fixed during RL, with dropout disabled.

The matched replay supplies ego speed from displacement between the current and next recorded positions divided by 0.2 s, and a coarse navigation command derived from the recorded route endpoint. Future ego positions and obstacle boxes otherwise enter supervised targets, post-prediction rewards, and metrics.

C EMCoP-RL Specification

Figure 6 shows how the policy inputs, memory, and decision branches fit together, along with the training loop. The following sections detail the policy inputs, message contents, architecture, and optimization settings.

C.1 Decisions and inputs

At every 0.2-second frame, EMCoP-RL makes the communication decisions for the active peer senders and the three tasks. The evaluated scene contains an ego vehicle, indexed by zero, and up to four peers. Write A for the number of active vehicles and $P = A - 1$ for the number of peers. The same learned weights process every peer. Detection, segmentation, and motion each have their own rate and group-utility heads, connected through one recurrent context.

The complete actor input at frame t comprises the global observation $o_t \in \mathbb{R}^{26}$, dense group descriptors $c_{t,i,s,g} \in \mathbb{R}^{31}$, motion group descriptors $m_{t,i,g} \in \mathbb{R}^{16}$, and the preceding recurrent state $h_{t-1} \in \mathbb{R}^{128}$. The global vector enters the GRU. A group descriptor is concatenated with the resulting context to score that group; the sender-task mean descriptor enters its rate head. Raw intermediate feature maps provide these descriptors and remain inputs to the perception pipeline.

The input layout contains several distinct kinds of quantity. *Scene variables* describe predicted objects, task uncertainty, and current driving context. *Communication-history variables* describe delivered information, pending deliveries, and recent transmissions. *Identity and geometry encodings* identify senders and groups and locate senders relative to ego. *Control parameters* comprise a fixed quality target and two adaptively updated constraint prices. A constant input supplies a bias. The indices below count all of these quantities while identifying the controls explicitly.

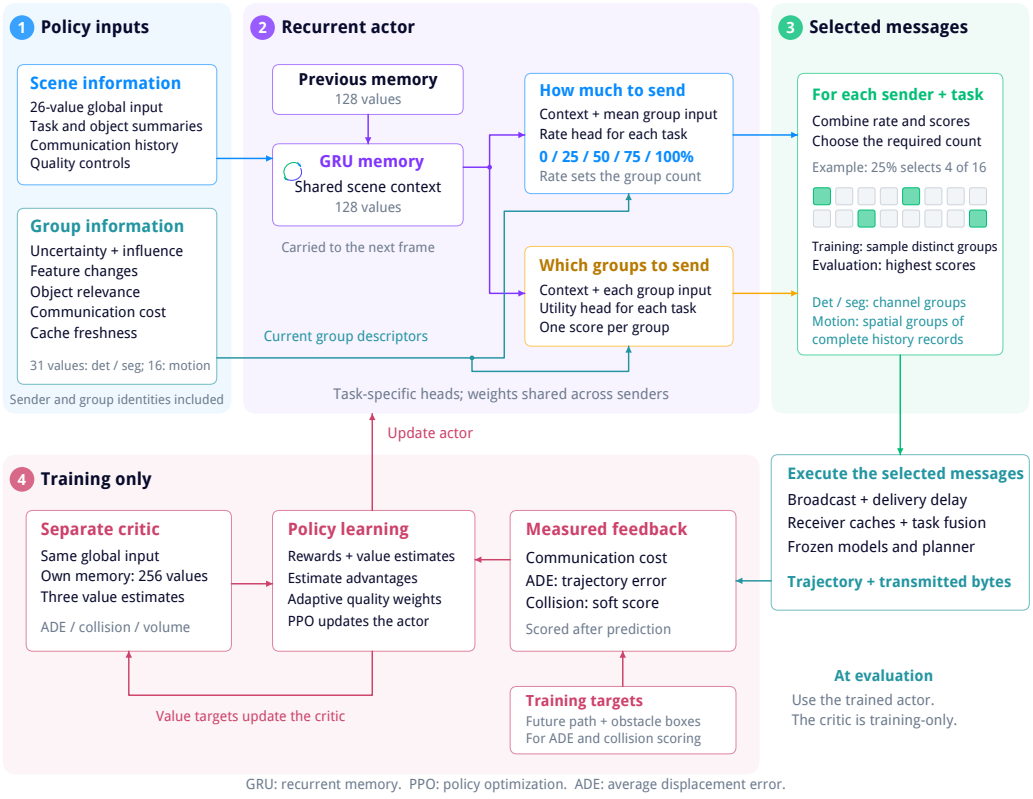


Figure 6: EMCoP-RL architecture and training. The global scene and communication state updates the recurrent memory. Group descriptors enter the decision branches directly: their sender-task mean conditions the rate choice, while each descriptor conditions its group's score. Each sender-task pair then selects the number of groups specified by its rate. Separate task heads share the same recurrent context and reuse their weights across senders. Training samples rates and distinct groups; evaluation chooses the highest-scoring rate and groups. The lower training block combines measured communication cost, trajectory error, and a soft collision score with value estimates from a separate critic to update the policy. Recorded future targets enter feedback calculation after prediction. The critic has its own memory and is used only during training. Group counts and network dimensions describe the evaluated implementation; the illustrated selection is an example.

C.2 Global observation: all 26 entries

Table 6 gives the complete global vector in its implemented order. Let $U_{s,i}$ be the local decoder's mean normalized entropy for task s and vehicle i. Entropy measures how spread out the predicted class probabilities are: a concentrated prediction has low uncertainty, while similar probabilities across classes have high uncertainty. Let $D_{s,i,g,j}$ be static dense descriptor j in Table 7, slots 12–21, numbered locally from one to ten. Global summaries of D pool both dense tasks and all groups of peer senders. Uncertainty summaries of U pool all active vehicles, including ego. Cache summaries pool both dense tasks and all peer groups after their receiver-wise aggregation. These pooling sets are fixed by the input definition. Here pooling means taking the specified mean or maximum over those entries. Throughout the input tables, $\mathrm{clip}(x, a, b)$ limits a value to the interval $[a, b]$. Normalization divides by a stated reference scale so that quantities with different units can enter the network together.

Table 6: Complete global observation. Entries 4–6 are controls and entry 26 is a constant; the remaining entries describe the scene or its communication history. All indices are one-based.

Index	Input	Definition
1	Active-vehicle count	$A/5$.
2	Scene progress	$t/(T-1)$, with $t = 0, \ldots, 84$ and $T = 85$.
3	Previous volume	$v_{t-1} = B_{t-1}/B_{t-1}^{\text{all}}$; initialized to 1.
4	Quality target	Fixed ADE-gain retention target $\eta = 0.90$.
5	Combined constraint price	$(\mu_E + \mu_C)/10$.
6	Repeated price encoding	Same value as entry 5; preserves the existing tensor layout.
7	Detection uncertainty, mean	Mean of $U_{\text{det},i}$.
8	Detection uncertainty, maximum	Maximum of $U_{\text{det},i}$.
9	Segmentation uncertainty, mean	Mean of $U_{\text{seg},i}$.
10	Segmentation uncertainty, maximum	Maximum of $U_{\text{seg},i}$.
11	Object confidence, mean	Mean projected track confidence, $D_{s,i,g,2}$.
12	Object confidence, maximum	Maximum projected track confidence.
13	Object speed	Maximum projected speed descriptor, $D_{s,i,g,4}$.
14	Object approach	Maximum projected approach descriptor, $D_{s,i,g,5}$.
15	Feature support	Mean support descriptor, $D_{s,i,g,7}$.
16	Occlusion proxy	Mean low-support uncertainty descriptor, $D_{s,i,g,8}$.
17	Semantic influence	Maximum semantic-influence descriptor, $D_{s,i,g,9}$.
18	Geometry influence	Maximum geometry-influence descriptor, $D_{s,i,g,10}$.
19	Receiver-cache validity	Mean fraction of receivers with a valid copy of a group.
20	Receiver-cache age	Mean of normalized mean-age descriptors, including their missing-copy value.
21	Ego speed	$\text{clip}(v_{\text{ego}}/20, 0, 2)$, with speed in m/s.
22	Command indicator 0	One when the navigation command equals 0, otherwise zero.
23	Command indicator 1	One when the navigation command equals 1, otherwise zero.
24	Command indicator 2	One when the navigation command equals 2, otherwise zero.
25	Pending-delivery fraction	Pending dense receiver deliveries divided by $6A(A-1)$, clipped to $[0, 1]$.
26	Constant bias	1.0.

Scene progress specifies how far the replay has advanced. Vehicle locations enter through the separate relative-pose descriptors below. Navigation command and speed follow the convention in Appendix B. The pending count counts queued receiver deliveries, including separate receiver copies of a broadcast; its denominator is a fixed normalization for two dense tasks and three delivery intervals. Serialized volume counts the broadcast once.

C.3 Detection and segmentation inputs

Table 7 specifies the dense descriptors. Let $F_{s,i}$ be the current sender feature tensor, with C_s channels, height $H = 32$, and width $W = 32$. For group g, let $\bar{r}_g, \bar{q}_g, \bar{c}_g$ denote the midpoints of its row, column, and channel bounds. Detection uses $C_s = 256$ and segmentation uses $C_s = 512$. Each has 16 contiguous channel groups. The row and column encodings therefore both equal 0.5 in the channel policy, while the channel-center encoding distinguishes the groups.

Table 7: Complete detection and segmentation group descriptor. Each peer supplies 16 rows per task. Entries 4–9 and 26–31 encode cost, identity, or geometry; entry 10 is a control.

Index	Input	Definition
1	Feature energy	Standardized mean absolute activation in the group.
2	Temporal change	Standardized mean absolute feature difference from the preceding frame.

Table 7, continued

Index	Input	Definition
3	Sender uncertainty	$U_{s,i}$; shared by the sender's groups for this task.
4	Relative group cost	Serialized bytes of a singleton group message divided by a full task message, including headers.
5	Sender encoding	$i/(A-1)$ for sender i.
6	Row encoding	$\bar{r}_g/H$.
7	Column encoding	$\bar{q}_g/W$.
8	Detection indicator	1 for detection, 0 for segmentation.
9	Segmentation indicator	1 for segmentation, 0 for detection.
10	Combined constraint price	$(\mu_E + \mu_C)/10$.
11	Previous volume	Same v_{t-1} as global entry 3.
12	Group uncertainty	Spatial decoder entropy projected with the group's activation weights.
13	Object confidence	Projected confidence of locally predicted tracks.
14	Object proximity	Projected $1 - \mathrm{clip}(d/60, 0, 1)$, for sender-relative object distance d in meters.
15	Object speed	Projected $\mathrm{clip}(\|v\|/20, 0, 1)$ from consecutive track centers.
16	Object approach	Projected positive radial-approach proxy toward the sender, scaled by 20 m/s.
17	Object footprint	Projected coarse footprint map from locally predicted track boxes.
18	Feature support	Projected mean absolute activation, normalized by its spatial maximum.
19	Occlusion proxy	Projected product of $(1 - \text{support})$ and spatial entropy.
20	Semantic influence	Group mean of $\|F \odot \nabla_F J_{\mathrm{sem}}\|$, rescaled across the sender's groups.
21	Geometry influence	Group mean of $\|F \odot \nabla_F J_{\mathrm{geo}}\|$, rescaled across the sender's groups.
22	Valid-copy fraction	Number of other active receivers with a valid copy, divided by $A - 1$.
23	Mean copy age	Mean source age over valid receiver copies, divided by 6; 1 when no copy is valid.
24	Maximum copy age	Maximum source age over valid receiver copies, divided by 6; 0 when no copy is valid.
25	Time since selection	Normalized selection-age counter described below; starts at 1.
26	Relative sender x	Sender position in ego's current frame, $x/80$, clipped to $[-1, 1]$.
27	Relative sender y	Corresponding $y/80$, clipped to $[-1, 1]$.
28	Relative sender distance	Distance to ego divided by 80 m, clipped to $[0, 1]$.
29	Relative heading, sine	$\sin(\psi_i - \psi_0)$.
30	Relative heading, cosine	$\cos(\psi_i - \psi_0)$.
31	Channel-center encoding	$2\bar{c}_g/C_s - 1$.

Energy, change, and uncertainty. Energy is the mean of $|F_{s,i}|$ over the group's channels and positions. Temporal change replaces it with $|F_{s,i,t} - F_{s,i,t-1}|$ and is zero on the first frame. For each task, standardization uses a scalar mean and population standard deviation over *all active senders and groups*, including ego: $(x - \bar{x})/\max(\mathrm{std}(x), 10^{-6})$. The sender uncertainty $U_{s,i}$ averages predictive entropy over all spatial positions and, for detection, anchors, then divides by the logarithm of the number of classes. Anchors are reference boxes used by the detector to make box predictions at each location. Group uncertainty retains the spatial entropy map before projection. Detection entropy is averaged over anchors at each position; segmentation entropy is computed over its class probabilities. Bilinear interpolation aligns these maps to the feature resolution.

Spatial descriptor projection. For any spatial descriptor $z(x, y)$, define

$$w_g(x, y) = \frac{1}{|g|} \sum_{c \in g} |F_{c,x,y}|, \qquad z_g = \frac{\sum_{x,y} w_g(x, y) z(x, y)}{\max(\sum_{x,y} w_g(x, y), 10^{-6})}. \tag{19}$$

This supplies entries 12–19. Each channel group's activation pattern determines its descriptor over the shared spatial grid. The spatial-selection variant restricts the projection to each spatial group.

Object maps use the *local-only* detector-derived tracks, keeping their construction independent of the communication action. Track centers are transformed into the sender frame and rasterized on a 32×32 grid over $[-40, 40)$ m in each axis. Confidence, proximity, speed, and approach are averaged when several centers occupy one cell; other cells are zero. Velocity comes from consecutive observed global centers divided by 0.2 s, then rotated into the sender frame. Missing consecutive

observations give zero speed and approach. For local center p and this velocity v, the approach proxy is $\mathrm{clip}(-p^\top v/[20\max(\|p\|, 10^{-6})], 0, 1)$. It uses object velocity expressed in sender axes. The footprint is a coarse axis-aligned patch around each center, with grid radii obtained by rounding up the predicted box dimensions divided by 5. Feature support is mean absolute activation over channels divided by its spatial maximum, with denominator floor 10^{-6}. The occlusion descriptor is the low-support uncertainty proxy defined in the table.

Decoder influence estimates. The two influence entries use first-order feature-gradient products (Molchanov et al., 2017). For detection, J_{sem} is mean maximum foreground probability and J_{geo} is mean squared localization output. For segmentation, J_{sem} is mean maximum non-background probability and J_{geo} is the sum of mean absolute horizontal and vertical differences in class probabilities. Two gradient calculations obtain $\nabla_F J_{\mathrm{sem}}$ and $\nabla_F J_{\mathrm{geo}}$ from a detached copy of the feature tensor. Each group's mean absolute gradient product is rescaled as $(x_g - \min_g x_g)/\max(\max_g x_g - \min_g x_g, 10^{-6})$ within its sender and task. These are first-order removal estimates. Decoder weights stay fixed; the resulting scalar descriptors are detached before policy optimization.

The gradient measures how the chosen decoder statistic changes with its input. Multiplying by the feature value estimates the effect of removing that value. Detaching the resulting descriptor treats it as a fixed input during the policy's gradient update.

For each dense task and frame, computing these entries requires one local decoder forward pass and two input-gradient calculations, with agents processed together. Scene caches reuse the resulting descriptors during repeated replay.

Cache freshness and selection history. Receiver-cache ages use the feature's generation time and exclude the sender's own local copy. Entries 22–24 summarize the other receivers after pending messages have been delivered at the current frame. Valid copies can have source ages up to five frames. The different empty-cache defaults for mean and maximum age are accompanied by the zero valid-copy fraction.

The selection-age counter records outgoing decisions separately. It is initialized to 1. After each action, every counter increases by $1/6$, capped at 1, and selected groups are reset to zero. Thus the next decision sees zero for a group selected at the preceding decision, $1/6$ after one subsequently missed refresh, and so on. Motion uses the same counter rule. Delivery delay can leave a recently selected group's receiver copy old or missing.

C.4 Motion: all 16 group entries

Motion candidates come from the policy-controlled detection and tracking branch, so earlier dense communication decisions can change their contents. Predicted objects between 3 and 60 m from ego are associated across sources. Their current centers define 16 groups on a 4×4 ego-frame grid over $[-40, 40)$ m; positions outside the grid are assigned to its nearest edge group. A group contains every valid history from that sender assigned to it. Table 8 gives the full descriptor. Empty groups remain in the action space with zero object statistics.

Table 8: Complete motion group descriptor. Object quantities use detector-derived histories before the selected peer-history delay is applied.

Index	Input	Definition
1	Grid row	$(r - 1.5)/1.5$, where $r \in \{0, 1, 2, 3\}$.
2	Grid column	$(q - 1.5)/1.5$, where $q \in \{0, 1, 2, 3\}$.
3	Sender encoding	$i/4$.
4	Scene progress	$t/84$.
5	Object count	$\min(n_g/16, 1)$ for the sender's valid records in group g.
6	Sender-object fraction	$n_g/\max(n_i, 1)$, where n_i is the sender's total valid-record count.
7	Mean object distance	Mean current center distance to ego divided by 60 m.
8	Minimum object distance	Minimum current center distance to ego divided by 60 m.
9	Mean object speed	Mean latest history speed divided by 20 m/s, clipped to $[0, 1]$.
10	Full-history validity	Mean validity over all 10 history entries and records in the group.

Table 9: Task data selected by EMCoP-RL. Dense values are FP32; motion histories and source metadata have FP16 payload accounting. FP32 and FP16 use 32 and 16 bits per numerical value.

Task	Full sender representation	One selected group
Detection	$256 \times 32 \times 32$ feature map	$16 \times 32 \times 32$ values
Segmentation	$512 \times 32 \times 32$ feature map	$32 \times 32 \times 32$ values
Motion	Variable number of 10×11 histories and six-field metadata	All records assigned to one of 16 spatial regions, retaining every history field

Table 10: Fields in each observation of a selected motion history. $j = 0, \ldots, 9$ is the history position; j_* is its most recent valid position. Coordinates and sizes come from predicted tracks.

Field	Quantity	Encoding
1–2	Position	Object-relative x, y in meters.
3–4	Velocity	Object-axis v_x, v_y in m/s from adjacent valid observations; zero when a pair is unavailable.
5–6	Box dimensions	First predicted box dimension divided by 10, second divided by 5.
7–8	Heading	Sine and cosine of observation heading relative to reference heading.
9	Validity	One for an observed track state, zero otherwise.
10	History time	$-1 + j/9$ for an observed state; zero for a missing state.
11	Last-observation age	$(9 - j_*)/9$ repeated over the history; zero if no observation exists.

Table 8, continued

Index	Input	Definition
11	Source-to-object distance	Mean norm of the first two metadata fields, clipped to $[0, 2]$ and divided by 2. Those fields already have a 40 m scale.
12	Previous selection	Binary flag indicating whether this sender-group was selected at the preceding frame.
13	Time since selection	Same normalized counter as dense entry 25.
14	Velocity-change proxy	Mean $\|v_j - v_{j-1}\|$ over valid consecutive history pairs, divided by 10 and clipped to $[0, 1]$.
15	History approach proxy	Mean clipped positive radial approach from each record's latest object-relative position and velocity, divided by 20.
16	Recent-history validity	Mean validity over the last four history entries and records.

Motion entry 14 uses velocity differences per history step. Entry 15 applies the radial expression above to the *object-relative history coordinates* defined in Table 10; its reference differs from the sender-relative dense approach map. These entries are numerical history descriptors. For entries 7–11 and 14–16, an empty group receives zero. Group identity, sender identity, progress, and selection history remain defined.

C.5 Shared messages and their contents

The communication action selects task data summarized in Table 9. A dense group is a contiguous block of channels over the full map. A motion group selects complete object histories and their metadata. Its number of records varies with the scene. The detector feature message is counted once; the resulting detections also support the construction of motion histories. The motion predictor produces the planner's $8 \times 32 \times 32$ raster after history selection.

Table 10 spells out the motion fields. Histories contain 10 sampled observations at 0.2-second spacing, expressed relative to the associated object's current reference center and heading. Adjacent valid positions give velocity by division by 0.2 s. Unobserved entries are zero-filled, with a validity field and a last-observation-age field. Six source metadata values accompany each history: source-to-reference $x/40$ and $y/40$ in object axes; sine and cosine of source heading relative to object heading; an ego-source indicator; and mean history validity. Local histories remain available to CMP alongside the chosen peer records.

Table 11: Learned components of EMCoP-RL and its critic. Parameter counts include biases. The inference actor has 280,338 parameters in total.

Component	Architecture and sharing	Parameters
Context and dense heads	One $26 \to 128$ GRU; separate detection and segmentation utility/rate heads, each shared across senders and groups	209,420
Motion heads	Utility and rate heads using the shared context; shared across senders and groups	70,918
Critic	Separate $26 \to 256$ GRU, with three linear scalar value heads	218,883

For a nonempty dense selection of k equal channel groups, serialized bytes are $44 + 2k + 4k(C_s/16)HW$. The archived motion counter assigns $12 + 244n$ bytes to a nonempty selection of n records; Appendix D distinguishes this accounting convention from the model-input dimensions. Empty selections cost zero. A motion rate of 25% selects four spatial regions; its byte fraction depends on their record counts. Averaging rates across senders and frames also produces percentages between the available choices.

Policy input descriptors are computed summaries used by the joint controller. The reported task-message bytes cover the selected representations above under the controller-state and alignment-metadata convention in Appendix D. Sender, task, and group index tensors route descriptor rows to their heads and message masks. Each peer chooses one selection per task; the dense broadcast delivers it to all other active receivers. Destination choice and task-model execution are fixed by this protocol. A zero rate suppresses the corresponding outgoing task message.

C.6 Policy architecture

The actor first updates $h_t = \mathrm{GRU}(o_t, h_{t-1})$ (Cho et al., 2014). For a dense group, the utility head receives the 159-value concatenation $[h_t; c_{t,i,s,g}]$. The corresponding rate head receives $[h_t; \frac{1}{16}\sum_g c_{t,i,s,g}]$. Motion uses the same 128-value context with its 16-value group descriptor, giving 144 input values. Every utility and rate head has two 128-unit fully connected layers with tanh activations and a linear output. Each utility output is a scalar; each rate output contains five logits. Table 11 identifies the shared and separate learned components. Logits are raw numerical scores; exponentiating and normalizing them produces the probabilities used to sample a decision.

For P peers, dense utility logits have $32P$ entries and dense rate logits have shape $2 \times P \times 5$. Motion utility logits have $16P$ entries and motion rate logits have shape $P \times 5$. At the maximum roster, this is 192 group scores and 12 five-way rate decisions. Detection and segmentation candidate rows for ego are removed before scoring. Motion candidates are constructed for peers directly. Numeric sender encodings accompany the rows; there is one set of head weights per task shared across senders.

The rate ladder and ordered without-replacement group distribution are given in Equations 2 and 3. The action comprises the rate and ordered group selection for every peer-task pair. Its log probability is the sum of all rate log probabilities and conditional group log probabilities. The likelihood therefore describes the complete sampled joint action used in PPO. All five rates are available. The configured dense volume cap is 1.0 of all-on, so the selected dense action is retained. Motion uses the same five rates over its 16 spatial groups.

A selected set of k groups has $k!$ possible sampling orders. These orders produce the same message, although their probabilities can differ. PPO uses the probability of the sampled order; the probability of an unordered set would instead sum over its orders. Sequential sampling makes the action likelihood tractable.

During training, a categorical distribution samples each rate, followed by categorical group draws with previously selected groups removed from the support, meaning the remaining eligible choices. A zero rate has an empty group sequence. During evaluation, argmax chooses the rate and repeated argmax over the remaining scores chooses the highest-scoring groups. Equal scores follow the stable candidate order. Separate frames may select the same group again. The rule controller in Section 3.5 subsequently refines these decisions where a retained rule applies; the RL specifications here describe its underlying policy.

C.7 MEMORY AND ROLLOUT

The actor's 128-value recurrent state summarizes preceding *global* observations. Group descriptors enter the heads at the current step, and their aggregate summaries can also influence future recurrent context through o_t. The runtime additionally retains the previous dense feature tensor for change scores, per-group selection ages, pending packets, receiver caches, and detector-derived track histories. Motion entry 16 summarizes four history observations. The rule controller's three preceding segment summaries are separate from the RL observation layout.

At each scene boundary, actor and critic recurrent states are reset to zero, receiver caches and queues are empty, previous feature tensors are absent, previous motion selections are false, selection ages start at 1, and previous volume starts at 1. The actor recurrent vector itself occupies 512 bytes in FP32. The training-only critic vector has 256 values and occupies 1,024 bytes. These sizes describe recurrent vectors; perception features and communication caches are separate runtime storage.

A training rollout follows this order at every frame:

1. Prepare each vehicle's current local dense features. Deliver queued packets and reconstruct the receiver information valid at this frame.
2. Run local-only, all-on, and policy-controlled dense prediction. Update each branch's detector-derived trackers from its own detection outputs and construct the current object-history records.
3. Build the global observation and dense group descriptors from current causal inputs, cache contents, and previous transmissions. Update the actor GRU once and choose the dense rates and groups.
4. Build the motion descriptors from the policy branch's prepared tracks. The motion heads use the same updated GRU context to choose motion rates and spatial groups.
5. Queue the selected dense broadcasts and record their bytes. Current dense predictions were computed before these new transmissions. Apply motion history selection, its configured history delay, and valid prediction reuse, then construct the motion raster.
6. Run the frozen planner on each branch. Compute trajectory and obstacle scores from its output, record the policy rewards and likelihoods, and update previous-feature, previous-volume, and selection-age state.

Thus an outgoing dense selection can change later received features, later detections and associations, and later motion input. Motion selection acts on the records already prepared for the current decision. The motion branch reuses eligible preceding predictions for up to five frames and advances their predicted positions with age, as specified in Appendix D. Each branch maintains its own state throughout the scene.

C.8 REWARDS AND OPTIMIZATION

The three frame rewards in Equation 4 are negative exact communication fraction, negative ADE divided by 10, and negative mean soft collision score. The communication denominator is the same-frame all-on branch's dense and motion bytes. The previous-volume input uses this same ratio. The normalized cost therefore includes the current reference motion record count. This reference and the controller summaries follow the common evaluation-state convention in Appendix D.

Table 12 separates control quantities from observations and learned network weights. The policy's selected rates and groups are its decision variables. The quality target, price update, and optimizer settings govern how those decisions are learned.

For scenes with a positive local-only to all-on ADE improvement, the 90% retention target in Equation 1 gives

$$E_{\lim} = E_L - \eta(E_L - E_A), \qquad C_{\lim} = C_A + 0.01. \tag{20}$$

After collecting the scene, the implementation updates the prices using the scene's measured ADE E and binary trajectory-level collision rate C:

$$\mu_E \leftarrow [\mu_E + 0.1(E - E_{\lim})]_+, \qquad \mu_C \leftarrow [\mu_C + 0.1(C - C_{\lim})]_+. \tag{21}$$

Table 12: Control quantities used by the EMCoP-RL implementation. The adaptive prices are numerical constraint updates.

Quantity	Value or update	Use
Retention target η	Fixed at 0.90	Global entry 4 and scene ADE limit
Collision allowance	Fixed at 0.01	Scene collision-rate limit
Prices μ_E, μ_C	Start at 5 each; nonnegative updates with step 0.1	Separate advantage weights; their sum divided by 10 is an actor input
Rate choices	$0, 0.25, 0.5, 0.75, 1$	Learned categorical decision per peer and task
Maximum dense task rates	1.0 for each task	All rate categories remain available
Dense volume cap	1.0	Allows the full dense selection
Communication-advantage weight	1.0	Weight of the realized byte-cost advantage
Expected-volume coefficient	0.0	The additional expected-rate regularizer has zero weight
Bias input	Constant 1.0	Final global entry

These updated prices weight the ensuing policy update. The recorded actor observations retain the prices present when their actions were collected. The critic's collision reward uses the mean soft score over valid horizon positions, while the price update uses the reported trajectory-level event. Both are defined in Appendix D.3. Reporting across experiments uses the common numerical references in Section 4.1.

For component $j \in \{V, E, C\}$, write V_t^j for its recurrent critic output. Generalized advantage estimation (Schulman et al., 2016) computes

$$\delta_t^j = r_t^j + \gamma V_{t+1}^j - V_t^j, \qquad A_t^j = \delta_t^j + \gamma\lambda A_{t+1}^j, \qquad R_t^j = A_t^j + V_t^j. \tag{22}$$

Here δ_t^j is the one-step prediction error, γ discounts later rewards, and λ controls how strongly later prediction errors enter the current advantage. R_t^j is the target used to train the critic. The scene ends with $V_T^j = A_T^j = 0$. We combine advantages as in Equation 6, then subtract their rollout mean and divide by the population standard deviation with floor 10^{-6} to obtain $\widehat{A}_t$.

If $r_t(\theta) = \exp(\log \pi_\theta(a_t) - \log \pi_{\text{old}}(a_t))$, the minimized actor loss is

$$L_{\text{actor}} = -\frac{1}{T}\sum_t \left[\min\Big(r_t(\theta)\widehat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A}_t\Big) + \beta\mathcal{H}_t\right]. \tag{23}$$

The entropy $\mathcal{H}_t$ sums the rate-category entropy and each conditional without-replacement group entropy over all peer-task pairs. The selected setting uses this full sum to encourage exploration during training. The probability ratio $r_t(\theta)$ compares the updated policy with the policy that collected the action; clipping limits the incentive for a large change in one update. The critic minimizes $\frac{0.5}{T}\sum_t \sum_{j\in\{V,E,C\}}(V_t^j - R_t^j)^2$. These are PPO actor and regression-critic updates (Schulman et al., 2017), with adaptive planning weights following the constrained-optimization interpretation (Stooke et al., 2020).

Each update collects one complete 85-frame scene, sampled with replacement from the 79 training scenes. This gives 85 joint transitions: each frame's action includes all senders and tasks. Five optimization passes reuse this batch with fixed returns, advantages, old likelihoods, and stored pre-action recurrent states. A complete scene retains delivery, tracking, and cache effects while bounding rollout storage. Advantage normalization, PPO clipping, entropy regularization, and gradient clipping control updates. Table 13 gives the settings.

Recurrent states are carried forward during collection and detached between frames. Optimization recomputes one recurrent step from each stored pre-action state, bounding the gradient graph. Advantages incorporate later rewards; GRU gradients cover one recurrent step. The critic estimates three values from the global observation o_t and its own recurrent state., keeping value estimation

Table 13: Optimization and scheduled validation settings of the selected EMCoP-RL training implementation.

Setting	Value
Optimizers	Adam for actor and critic, separate optimizer states
Learning rate	Constant 3×10^{-4} for both
Adam moments and numerical constant	$(0.9, 0.999)$; $\epsilon = 10^{-8}$
Weight decay / AMSGrad	0 / disabled
Gradient norm clipping	10, separately for actor and critic
Discount γ / GAE λ	0.99 / 0.95
PPO ratio clipping ε	0.10
PPO passes / transition batch	5 / one full 85-frame scene
Entropy coefficient β / mode	0.01 / full conditional-entropy sum
Value-loss coefficient	0.5 across the sum of three mean-squared losses
Training scenes / validation scenes	79 / 10, disjoint scene sets
Maximum scheduled updates	500
First scheduled RL validation	Update 50
RL validation interval	Every 5 updates thereafter, complete scenes
RL early-stopping patience	10 validations without a qualifying score improvement
Minimum score improvement	0.001
Arithmetic	FP32; TF32 and cuDNN benchmarking disabled; deterministic algorithms enabled

compact; it receives no detailed group descriptors. One optimizer covers the shared actor and all task heads; another updates the critic.

C.9 Initialization and evaluation

The joint actor, motion heads, and critic start from fresh archived initializations. Perception models, CMP, and the trajectory planner retain their pretrained weights throughout RL. The planner loader sets evaluation mode, and the training entry point checks that its dropout modules remain in that mode. Local decoder gradients used for the influence descriptors change the input-gradient calculation while leaving decoder parameters fixed. The neural policy and critic have the learned components listed in Table 11.

Evaluation uses fixed checkpoint weights and stored constraint prices, with deterministic rate and group choices. It follows the same causal scene replay, with metrics computed from completed planner predictions. Periodic validation archives scene reports and RL checkpoints. The low-communication and higher-quality settings use selected RL checkpoints with the same 90% training target.

The controller's object descriptors and motion records are obtained from causal predicted detections and associations. Task labels, recorded future ego positions, and future obstacle boxes enter supervision, reward calculations, and evaluation after prediction. The supplied route-command convention and the planner's earlier supervised training stage are specified in Appendix B. The deployment forward calculation consists of the frozen task pipeline, descriptor computation, recurrent actor, and message selection; the critic and post-prediction reward calculations serve training and evaluation.

Two forms of stored information have separate roles. Static descriptor caches and lossless validation encoder caches avoid repeated frozen computation; the selected training campaign runs its dense encoders live and reads its precomputed local descriptors. Cache manifests identify scenes, model weights, and source versions, and live-versus-cached checks establish input parity. Receiver communication caches, queues, policy-fused detections, policy tracks, and motion predictions evolve under the current actions and are recomputed through the rollout. Their contents determine which earlier messages can still contribute to planning.

D COMMUNICATION AND EVALUATION PROTOCOL

D.1 DELIVERY, CACHES, AND TASK DEPENDENCIES

Each peer's selected detection or segmentation groups form one broadcast per task and frame. Other active receivers maintain independent copies with source timestamps and poses; the ego sends no dense broadcast in the RL protocol. A copy's source age is the current frame minus its generation frame. Its cache lifetime is five frames, including time spent awaiting delivery. Fusion aligns available peer features with the receiver's local features.

For frame period $\Delta = 200$ ms and delay $d \in [0, 300)$ ms, delivery occurs at

$$t_{\text{arrival}} = t_{\text{send}} + 1 + \lfloor d/\Delta \rfloor. \tag{24}$$

The seeded schedule depends on scene, sender, receiver, and send frame. The order of operations makes even a zero-delay message usable at the next frame. Accounting for delayed evidence is also central to latency-aware cooperative perception (Lei et al., 2022; Xu et al., 2022a; Yang et al., 2023).

AB3DMOT associates detector outputs over time (Weng et al., 2020), using detector confidence threshold 0.01 and a 4 m association gate. Motion selection retains all fields of each chosen object's 10-step history. The delay approximation masks the newest two peer observations, corresponding to 400 ms, while keeping the prepared track's current geometry and source alignment. Forecasts can be reused for five frames, advancing their future positions with age. Each method's delivered detection information determines its subsequent tracks and motion inputs.

D.2 BYTE ACCOUNTING AND UNITS

For a nonempty dense selection of k channel groups, the serializer uses a 44-byte header, a 2-byte index per group, and FP32 feature values:

$$B_s(k) = 44 + 2k + 4k(C_s/16)\,32^2, \qquad k > 0. \tag{25}$$

Here $C_D = 256$ for detection and $C_S = 512$ for segmentation; $B_s(0) = 0$. Thus a segmentation group carries twice as many feature values as a detection group. Shared detection features are counted once, with selected motion records accounted separately.

Motion communication uses a fixed record-allocation convention: a 12-byte sender header and 244 bytes per selected record, including identifiers, allocated history storage, and six FP16 metadata fields. Both the RL communication reward and reported totals use this fixed charge; the CMP input contains the 10-step histories specified above.

With K evaluated frames, communication volume is $100 \sum_t B_t / \sum_t B_t^{\text{all}}$. The denominator is the fixed all-on reference shared by the main tables and delay diagnostics. Mean aggregate rate at 5 Hz is $8 \cdot 5 \sum_t B_t/(10^6 K)$ Mbps, summed across tasks and transmitting agents. Broadcasts count once; each executed directed response counts separately. Consequently, a multi-round method can exceed the single-broadcast all-on reference.

D.3 PLANNING METRICS

ADE averages Euclidean distance between predicted and recorded future positions over valid steps. For predicted center $\hat{p}_h$ and obstacle b with center p_{bh}, heading ψ_{bh}, length l_{bh}, and width w_{bh}, write $(x_{bh}, y_{bh}) = R(-\psi_{bh})(\hat{p}_h - p_{bh})$. The implemented overlap check is

$$|x_{bh}| \le l_{bh}/2 + 2.3, \qquad |y_{bh}| \le w_{bh}/2 + 1.0. \tag{26}$$

It expands each obstacle by fixed ego half-length 2.3 m and half-width 1.0 m in the obstacle's axes. The soft score is

$$q_h = \max_{b\,\text{valid}} \sigma\left(\frac{\min(l_{bh}/2 + 2.3 - |x_{bh}|,\ w_{bh}/2 + 1.0 - |y_{bh}|)}{0.25}\right), \qquad \sigma(z) = \frac{1}{1+e^{-z}}. \tag{27}$$

The score is zero when no obstacle is valid. Its mean over valid steps supplies the RL collision reward; a trajectory is flagged if any valid step has $q_h \ge 0.5$. Collision rate is the fraction of flagged trajectories. Planner tail losses use the maximum score along each trajectory.

All reported methods use the same collision check.

D.4 INDIVIDUAL TASK METRICS

Detection AP@0.5 is bird's-eye-view average precision: the evaluator matches planar box polygons at intersection-over-union 0.5. Segmentation mIoU averages semantic-class IoUs under the valid-pixel mask, excluding the unlabeled class. Native motion evaluation associates forecasts with eligible annotated targets within 4 m only after inference. For target i, mode k, valid future steps $\mathcal{H}_i$, and last valid step h_i^*,

$$\mathrm{minADE}_i = \min_k \frac{1}{|\mathcal{H}_i|} \sum_{h \in \mathcal{H}_i} \|\hat{p}_{ikh} - p_{ih}\|_2, \qquad \mathrm{minFDE}_i = \min_k \|\hat{p}_{ikh_i^*} - p_{ih_i^*}\|_2. \tag{28}$$

Each minimum evaluates an entire predicted mode under its own criterion. The two metrics may select different predicted modes. Planning ADE and collision rate evaluate the planner's single output trajectory.

Coverage is the number of matched forecasts divided by the number of eligible targets. Different sharing policies can cover different targets, making matched-only motion errors insufficient for comparing all targets. The coverage-aware diagnostic assigns a 10 m penalty to each missing forecast:

$$E_{\mathrm{cov}} = \frac{\sum_{i \in \mathrm{matched}} \mathrm{minADE}_i + 10 N_{\mathrm{missing}}}{N_{\mathrm{eligible}}}. \tag{29}$$

The local, low-policy, high-policy, and all-on values are 9.484, 8.870, 8.744, and 8.509 m, respectively.

E BASELINE ADAPTATIONS

The four baselines select detection and segmentation information independently, with motion shared all-on from each method's detections. They use the same frozen trajectory planner and delay protocol. This comparison tests communication allocation under a common planning evaluation while preserving each selector's task-level decisions.

CooperTrim (Mukhopadhyay et al., 2026) uses temporal uncertainty to request feature channels and receive peer responses. Where2comm (Hu et al., 2022) uses confidence maps and spatial requests; its two communication rounds receive 20% and 80% of the configured feature budget. Their required selection and fusion components are fitted to the task pipelines. The configured Where2comm budget controls feature selection, while measured volume includes all executed rounds and directed responses, including service to non-ego receivers.

When2com matches learned query/key descriptions to activate communicating peers and combines their aligned features (Liu et al., 2020a). Who2com uses learned handshakes to choose a peer, combining its aligned features with local features (Liu et al., 2020b). Their adaptations fine-tune the task networks and communication modules. They share the evaluation planner with the other methods. Query, score, selection, and feature exchanges enter their executed-protocol accounting.

F RULE DISCOVERY AND EXECUTION

F.1 PAIRED TRIALS AND TRAINING-SCENE SEPARATION

After freezing the low-communication and higher-quality policies, we compare the original selection with an omission action (d_D, d_S), which caps omitted detection and segmentation groups per peer. Tested caps are $(2,1)$, $(3,1)$, $(4,1)$, $(4,2)$, $(5,2)$, and $(6,3)$. An omitted group must already be available at every receiver, with its oldest copy generated at most two frames earlier. A trial preserves the motion selection while recomputing detections, tracks, and motion predictions affected by changed dense messages.

Collection uses 20 training scenes and both policies, giving 240 alternative trials. Each scene has a predetermined intervention at frame 25 or 50. The replay state includes recurrent memory, random-number state, queues, caches, trackers, and descriptor history. Restoring that state, a trial applies its omission caps for 16 frames and then follows the frozen policy through frame 84. Full-scene

differences in ADE, collision rate, and bytes include the unchanged prefix and all later consequences. Snapshot-versus-full-replay checks establish that the paired comparison starts from the same state.

Four collection scenes provide proposal examples; the other 16 provide numerical fitting. Five scene-level folds keep both policies from the same scene together. Thresholds fitted on four folds are checked on the held-out fold. We select the rule library using complete-sequence evaluations on additional held-out training scenes, then freeze it before evaluation on the validation set.

F.2 Conditions, fitting, and composition

We seek recurring measured conditions under which omissions save bytes with small ADE changes and no observed collision increase. The proposal stage receives all alternatives from its training scenes, including adverse outcomes, with scene measurements and trajectory visualizations. It produces qualitative conditions such as high cache availability or low uncertainty. Numerical fitting determines the thresholds and checks their support. The archived prompt and a representative input/output appear at the end of this appendix.

A condition is a conjunction of one to three finite interval tests on current measurements or averages of the preceding three completed five-frame summaries. Candidate quantities describe uncertainty, detected objects, received copies, queued deliveries, recent volume, and selected information. Conditions cannot use policy or scene identity, future information, or measured trial outcomes. Missing values produce no match. For each proposed low/high condition, fitting tests the 10th, 25th, 50th, 75th, and 90th percentiles, with the opposite interval endpoint set to the fitting minimum or maximum. Subsets of proposed clauses are considered with each omission cap.

Support is checked separately for each policy. A candidate needs at least three matching scenes, positive total savings, and no collision-rate increase in any matching scene; inner-fold fitting uses at least two matches. The mean ADE increase and largest mean after removing one matching scene must each be at most ϵ. The worst scene increase must be at most $\max(0.002, 4\epsilon)$ m, and a condition may cover at most 80% of the combined policy-scene examples.

Greedy composition uses pooled held-out training scene matches and separately checks each policy's error and savings, plus collision changes in every case. If several rules match, the rule with greatest fitted mean byte saving takes priority; fold-specific priorities are fitted without the held-out scene. The fitting objective prioritizes the larger policy saving, then their total, while keeping both nonnegative. The reported setting uses $\epsilon = 0.002$ m. Repeated full-scene replay checks the composed controller, since separately successful omissions can interact through later caches and tracks.

F.3 A fitted rule and a worked example

One retained condition uses recent queued deliveries. Let A be the active-agent count and ℓ_t the queued dense receiver deliveries divided by $6A(A-1)$, clipped to $[0, 1]$. The factor six normalizes two dense tasks over three delivery intervals. Let $\bar{\ell}_t$ average the three preceding completed five-frame summaries. Table 14 gives this concrete example. Its scene-level condition is followed by a separate freshness check for every proposed omission.

Table 14: Example fitted rule. Low recent pending-message load identifies a recurring opportunity to omit updates; each omitted group must still have a recent copy at every receiver. Bounds are rounded here; execution uses full precision.

Field	Value
Condition	$0 \le \bar{\ell}_t \le 0.07986111$
Per-peer omission caps	Up to 6 detection and 3 segmentation groups
Per-group eligibility	Every receiver has a copy with source age at most 2 frames
Held-out training-scene support	15 low-policy scenes; 8 high-policy scenes

Suppose the policy chooses eight detection and four segmentation groups. If this rule matches and five detection and two segmentation groups satisfy the freshness check, those groups are omitted, leaving three detection and two segmentation groups to send. At the next frame, any copy older than

two frames loses eligibility, so the policy-selected group is sent. The rule removes eligible groups from the frozen policy's selection.

F.4 Runtime combination with the policy

At every frame, the policy first selects its groups. Within that selection, eligible groups are ordered by youngest maximum receiver age, then smaller feature change, smaller estimated influence, lower policy score, and finally group index. The index only resolves otherwise exact ties. Influence is the larger of the semantic and geometry estimates. Signed standardized feature change preserves the ordering of raw change magnitudes.

Frames 0–14 accumulate three completed summaries while using the policy unchanged. Rules are checked at frames 15, 20, . . ., 70; a matched cap applies for five frames, with eligibility and ordering recomputed each frame. Frames 75–84 follow the policy while earlier omissions continue to affect received inputs. Insufficient history, missing required measurements, or no rule match leaves the policy selection unchanged. Rules filter detection and segmentation; motion still depends on the resulting detections. The runtime controller performs scalar comparisons and sorts selected groups.

The same frozen library is applied to both operating points. Relative to their underlying policies, the reported ADE increases are 1.70 mm and 0.26 mm, with unchanged collision rates and volume reductions of 5.61 and 5.75 percentage points of all-on bytes. These values are measured with the complete policy-and-rule controller.

G Supporting Experiments

G.1 Planning quality and communication

Figure 7 visualizes the quality-communication tradeoff in Table 2 for Where2comm, CooperTrim, Who2com, When2com, and the two EMCoP operating points. Its left panel shows that most of the all-on error reduction is retained at a smaller task-message cost; its right panel shows nearby predicted-collision rates.

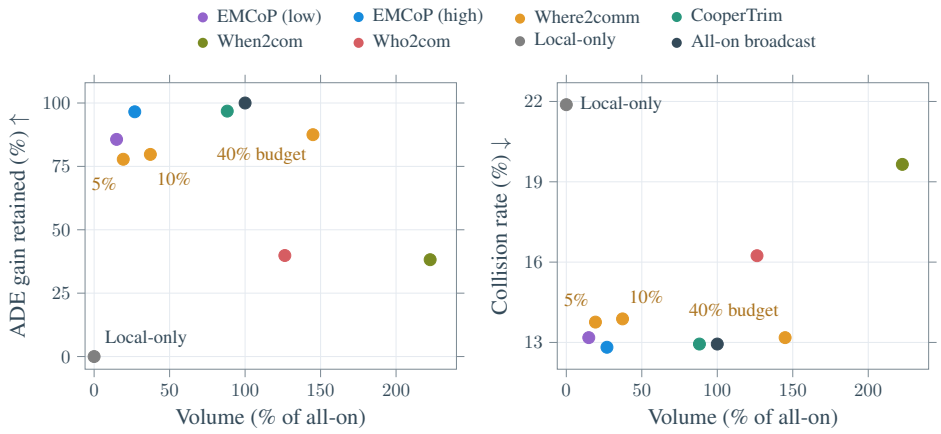


Figure 7: **Planning quality versus communication.** Left: joint sharing and update skipping retain substantial planning gain with fewer task-message bytes. Right: both EMCoP settings have predicted-collision rates close to all-on. Where2comm labels are configured budgets; measured volume includes its complete protocol.

G.2 Communication delay

Figure 8 varies prescribed dense delay from 0 to 300 ms for the low-communication and higher-quality RL policies before rules, retaining the five-frame cache lifetime and fixed main-paper ref-

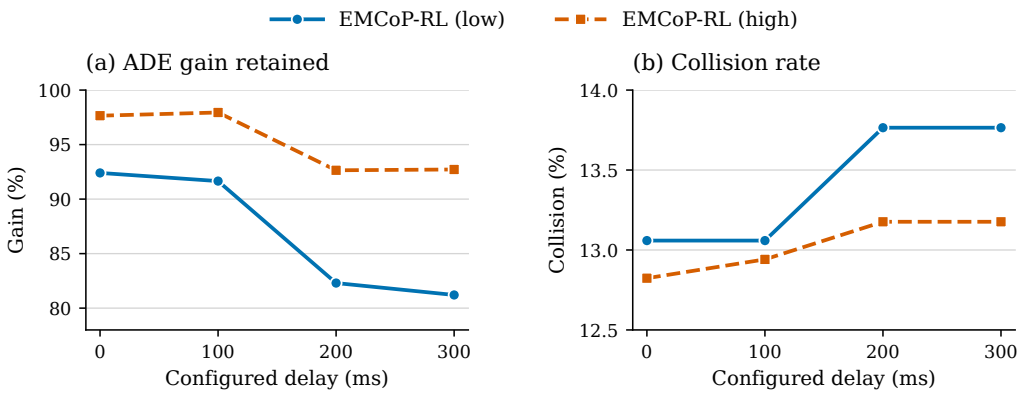


Figure 8: **Sharing remains useful under the evaluated delays.** Before rules, higher-quality EMCoP-RL retains over 92% of all-on ADE gain. Volume ranges are 20.54–21.16% (low) and 32.63–32.67% (high). Dense packets become available at the next runtime frame even at zero configured delay.

erences. The higher-quality setting retains over 92% of the reference planning gain and the low-communication setting over 81%. Both continue to use substantially fewer bytes than full sharing. We observe small non-monotonic changes across delay settings.

H Rule-Proposal Prompt and Examples

The proposal call generates qualitative candidate conditions from training evidence. Numerical fitting and replay then select the retained conditions. The archived call used `gpt-6-astra`, high reasoning effort, a 10,000-token completion limit, and no fallback. The request contains a system message and one user message with a JSON text block followed by a PNG evidence image. Text is serialized with sorted JSON keys.

Exact user-message instruction fields

task. Propose a SHARED library of conditional cross-task communication rules for two frozen recurrent policies.
context. Cooperative driving, V2X-Sim, frozen DiscoNet detection/segmentation and CMP motion, frozen ego planner. Detection and segmentation each contain 16 channel groups covering the complete map. Motion communicates full detector-derived object histories, not annotations. Each peer sender broadcasts once. The RL actor first picks its own rates and groups. There is no further RL or neural training.
intervention. RL first chooses rates and groups. A cache-refresh action can omit only selected groups already valid in EVERY active receiver, whose maximum source age is at most two frames. Order eligible groups by youngest worst-receiver age, lowest SIGNED standardized temporal change, lowest maximum first-order influence, lowest original RL utility, then group ID. Action [d,s] caps omissions per sender for detection and segmentation; it does not force a fixed omission count. Cache eligibility is checked on every frame. Motion selection is never directly reduced. Detection changes recompute policy tracks, CMP and planning. Each measured candidate is applied for 16 frames at its recorded start, then original RL runs through frame 84. Deployment will choose rules every five frames from 15 through 70, apply them through frame 74, and use original RL on frames 75..84.
features. Current causal Arm C statistics and averages of three completed five-frame summaries. past3_ excludes the current frame. weak_ statistics now concern up to the FIRST THREE cache-eligible groups in the above refresh ordering per sender, not arbitrary low-utility groups. Summaries use conservative extrema across senders. Missing means abstain. weak_temporal is the maximum ABSOLUTE standardized temporal score, so low values mean closeness to the normalization mean, NOT necessarily low raw temporal change. The omission ranking itself uses the SIGNED score. Utility entropy is not a safety probability. Local first-order influence is a proxy, not an oracle. Cache

validity alone does not prove that old information remains accurate. All allowed names are exactly those listed below.
delay. Dense messages arrive at the next frame or one frame later (200 ms/frame, maximum random delay 300 ms), and receiver group caches persist for five frames. Motion uses the original two-frame history delay and TTL5. A reduced transmission first changes a receiver after arrival, then influences later tracking, motion and planning.
labels. All ADE/overlap/trajectory labels are offline training evidence only. Deployment receives only allowed_features. Five scene-held-out numerical fits follow, keeping both policies of a scene in the same fold. Separate training scenes then replay repeated rules through complete scenes to test composition. The shown examples are representative evidence, not proof of safety.
collection_schedule. One predetermined start (25 or 50) per training scene; six actions, both parent policies, all 85 frames.
objective. Propose one shared conditional library for both policies. Aim to save 4 to 5 percentage points of all-on-normalized bytes beyond EACH parent. Fit thresholds and action caps locally. Separate numerical checks allow paired mean ADE increases of 0.0005, 0.001, or 0.002 meters and no collision increase in any audit scene. Do not condition on policy ID, seed, scene, future information, or offline metric outcomes.
request. Return at most 24 complementary hypotheses, each with 1..3 low/high clauses and an action. Prefer interactions of cache state, weak-group influence, uncertainty and temporal stability, including cross-task conditions. Explain why apparently good short-horizon reductions can fail later. Qualitative directions only: numerical thresholds will be fitted locally on separate training scenes. Do not use an unobserved obstacle as a predicate. A lack of detected objects does not establish an empty scene. Return one JSON object with proposals and analysis. Do not output executable code.
units. Uncertainty/confidence/validity/occlusion and selected fractions are normalized. Cache age is divided by TTL+1. Object speed and ego speed use /20 m/s normalization. weak_temporal is magnitude of standardized temporal change. previous_volume is last frame's actual serialized bytes / all-on bytes. ADE deltas are meters; overlap deltas are fractions of evaluated trajectory windows, not closed-loop accidents.

Exact control fields and requested output structure


```
{
  "actions": [
    [
      2,
      1
    ],
    [
      3,
      1
    ],
    [
      4,
      1
    ],
    [
      4,
      2
    ],
    [
      5,
      2
    ],
    [
      6,
      3
    ]
  ],
  "data_scope": {
    "audit_scene_count": 8,
    "fit_scene_count": 16,
    "folds": 5,
    "proposer_scenes": [
      10,
      61,
      70,
      80
    ]
  },
```

```
  "output_format": {
    "analysis": "Evidence and delayed-effect failure analysis",
    "proposals": [
      {
        "action": [
          2,
          1
        ],
        "clauses": [
          {
            "direction": "high",
            "feature": "cache_valid"
          },
          {
            "direction": "low",
            "feature": "det_weak_influence"
          }
        ],
        "rationale": "Hypothesis to be tested"
      }
    ]
  },
  "quality_targets": {
    "additional_volume_reduction_pp": 4.0,
    "paired_ade_increase_m": [
      0.0005,
      0.001,
      0.002
    ]
  }
}
```


Allowed condition vocabulary

cache_age, cache_valid, det_selected, det_utility_entropy, det_weak_cache_age, det_weak_cache_valid, det_weak_influence, det_weak_object_proximity, det_weak_temporal, det_weak_uncertainty, detection_uncertainty, ego_speed, motion_rate_entropy, motion_selected, motion_utility_entropy, object_approach, object_confidence, object_speed, occlusion_risk, past3_cache_age, past3_cache_valid, past3_det_selected, past3_det_utility_entropy, past3_det_weak_cache_age, past3_det_weak_cache_valid, past3_det_weak_influence, past3_det_weak_object_proximity, past3_det_weak_temporal, past3_det_weak_uncertainty, past3_detection_uncertainty, past3_ego_speed, past3_motion_rate_entropy, past3_motion_selected, past3_motion_utility_entropy, past3_object_approach, past3_object_confidence, past3_object_speed, past3_occlusion_risk, past3_pending_fraction, past3_previous_volume, past3_seg_selected, past3_seg_utility_entropy, past3_seg_weak_cache_age, past3_seg_weak_cache_valid, past3_seg_weak_influence, past3_seg_weak_object_proximity, past3_seg_weak_temporal, past3_seg_weak_uncertainty, past3_segmentation_uncertainty, pending_fraction, previous_volume, seg_selected, seg_utility_entropy, seg_weak_cache_age, seg_weak_cache_valid, seg_weak_influence, seg_weak_object_proximity, seg_weak_temporal, seg_weak_uncertainty, segmentation_uncertainty.

Scene and policy identifiers label offline evidence and are excluded from runtime rule conditions.

Actual paired-trial input excerpt


```
{
  "action": [
    2,
    1
  ],
  "ade_delta": -3.814697265625e-05,
```

```
  "bytes_saved": 4457508,
  "overlap_delta": 0.0,
  "policy": "armc_seed3_u55",
  "scene": 10,
  "start": 25,
  "state": {
    "cache_valid": 0.6875,
    "det_weak_uncertainty": 0.010869938880205154,
    "past3_pending_fraction": 0.05925925994912784,
    "seg_weak_uncertainty": 0.015033978968858719
  }
}
```


We next show one qualitative proposal produced before threshold and omission-cap fitting; Table 14 reports the deployed fitted rule.

Actual proposal output


```
{
  "action": [
    2,
    1
  ],
  "clauses": [
    {
      "direction": "high",
      "feature": "cache_valid"
    },
    {
      "direction": "low",
      "feature": "det_weak_uncertainty"
    },
    {
      "direction": "low",
      "feature": "seg_weak_uncertainty"
    }
  ],
  "rationale": "Test whether broad cache availability combined with low uncertainty in
     both omission prefixes identifies useful small reductions. "
}
```