DDACOM: Dual-Decoupled Adaptive Communication for Multi-Agent Reinforcement Learning under Dynamic Networks

Yiqun Wu[1,2,3] **Qi Wang**[1,3*] **Mengxian Li**[1,3] **Yongjun Xu**[1,3]
[1]Institute of Computing Technology, Chinese Academy of Sciences
[2]Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences
wuyiqun23@mails.ucas.ac.cn wangqi08@ict.ac.cn

Abstract

Communication enhances cooperative multi-agent reinforcement learning (MARL) under partial observability. In real networks, however, each agent's available bandwidth varies over time due to changing link quality, interference, and congestion. Most existing MARL communication policies decide whether to transmit based on message importance or novelty, or are trained under fixed bandwidth conditions. As a result, they cannot adapt sending frequency to bandwidth variations, leading to packet loss when bandwidth decreases or missed information when bandwidth recovers. To address this, we propose Dual-Decoupled Adaptive Communication (DDACOM)—a framework that decouples message sending and message generation from the action policy. The message-sending module ranks each message by its deviation from teammates' information and the sender's history, and uses a dynamic token bucket to convert the probe-estimated safe rate into a real-time token budget, enabling the communication frequency to track the safe rate online. The message-generation module is trained independently with receiver-side attention feedback and counterfactual marginal value, encouraging messages that support cooperation. On the evaluated SMAC maps and MPE scenarios, DDACOM achieves leading cooperative performance among the tested baselines. Under dynamic network conditions in ns-3 simulation, DDACOM sustains over 95% safe-rate utilization and packet loss below 0.3% while reducing communication frequency to less than half of full broadcast, confirming its adaptability to runtime network variation.

1 Introduction

Cooperative multi-agent reinforcement learning (MARL) has demonstrated significant success in a range of cooperative decision-making applications, including robot swarms [1], traffic control [2], and unmanned-system coordination [3, 4]. In most realistic scenarios, however, each agent observes only a partial view of the environment and cannot reliably infer the global situation from its own observation alone. Communication alleviates this partial-observability bottleneck by allowing agents to exchange observations, state estimates, or intentions, thereby helping agents coordinate their actions and improve the quality of team-level decisions [5–8].

The benefit of communication is nevertheless constrained by real network conditions. As the number of agents grows, full-broadcast or high-frequency communication rapidly increases the traffic load and introduces substantial bandwidth cost [9]. At the same time, available communication

*Corresponding author.
Code: https://github.com/Linquer/DDACOM.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

resources in real networks are dynamic: link quality, external interference, and congestion can cause bandwidth to decrease or recover over time. Strategies that assume fixed-frequency or full-broadcast communication are therefore difficult to deploy directly in network environments where the available bandwidth can change during task execution.

Prior studies reduce redundant communication through gating, receiver-demand modeling, message-novelty estimation, often under a fixed communication-bandwidth assumption [10–15]. These methods improve communication efficiency, but most learn under static or pre-specified communication conditions and rarely model runtime variation in network resources explicitly. When the deployment-time bandwidth condition differs from the training condition [16], communication decisions can fail in two ways: excessive sending causes congestion and packet loss, whereas overly conservative sending removes information needed for cooperation. Since many gating mechanisms are jointly trained with the action policy, their sending behavior is implicitly coupled to the training-time communication condition. Deployment under a different bandwidth often requires adjusting the gating threshold, or even retraining the main policy under severe distribution shift.

To address this issue, we propose DDACOM (Dual-Decoupled Adaptive Communication), a communication framework that decouples both message sending and message generation from the action policy. To cope with runtime network variation, DDACOM introduces an independent message-sending module. This module evaluates semantic priority by measuring how much a message deviates from teammates' information and from the sender's own historical messages. It then uses a dynamic token bucket to convert the safe rate estimated from probe packets into a real-time token budget [17], and makes the final sending decision through a token-dependent dynamic threshold. During deployment, the module continuously updates the token-recovery rate and sending threshold according to the probe-estimated safe rate, allowing the communication frequency to decrease or increase with network resources. In bandwidth-limited settings, each transmitted message is also more consequential. DDACOM therefore further decouples message generation from the action policy: an independent message policy produces continuous-vector messages and is optimized with a message reward combining receiver-side attention feedback and counterfactual marginal value, improving the cooperative utility of messages under limited communication opportunities.

2 Related work

Early MARL methods typically use full broadcast. CommNet aggregates continuous communication vectors [5], DIAL/RIAL learn protocols through differentiable communication channels [6], and BiCNet coordinates agents through bidirectional recurrent structures [18]. TarMAC [7] and DGN [8] use attention or graph structures for weighted message fusion. These methods generally assume abundant bandwidth and incur broadcast redundancy.

Subsequent work reduces communication through partner selection, message representation, and bandwidth control. ATOC [10] and IC3Net [11] select communicating agents through attention or gating, I2C identifies communication relations through causal inference [12], and MAIC generates targeted incentive messages through decentralized teammate modeling [19]. MAGIC [20] and MASIA [21] improve message representations using graph attention or self-supervised information aggregation. SMS evaluates message contributions through Shapley message values and learns to select communication partners from local information [22]. Its message-selection formulation does not explicitly model physical bandwidth, queueing, or packet loss. For bandwidth-constrained scheduling, SchedNet learns sender selection [23], and ETCNet learns an event-triggered communication policy under a bandwidth constraint [16]. It incorporates bandwidth constraints into learning the triggering policy, whereas DDACOM updates the sending budget online through network probing. VBC reduces communication overhead through variance-based control [24], Message Pruning removes low-value messages under a fixed bandwidth [13], TMC improves robustness to packet loss through temporal smoothing and message caching [14], and CACOM generates receiver-relevant messages under a limited bandwidth through context-aware communication and quantization [15].

LAUREL further studies communication in realistic wireless networks, learning when to communicate from message importance and channel conditions and incorporating network measurements into message content [25]. Its task and communication decisions are jointly learned within a coupled task–wireless environment. DDACOM focuses on adapting communication frequency to runtime

bandwidth changes by coupling the probe-estimated safe rate with semantic message priority, while training an independent message-generation module to improve the utility of transmitted messages.

3 Background

We consider fully cooperative multi-agent tasks under partial observability and model them as a decentralized partially observable Markov decision process (Dec-POMDP), $\mathcal{M} = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, \Omega, O, R, \gamma \rangle$. Here, $\mathcal{N} = \{1, \ldots, n\}$ denotes the agent set, $\mathcal{S}$ the global state space, $\mathcal{A}$ the joint action space, and $\gamma \in [0, 1)$ the discount factor. At each time step t, agent i receives a local observation $o_i^t \in \Omega$ through the observation function $O(s^t, i)$, and selects an action a_i^t according to its policy π_i. The joint action $\boldsymbol{a}^t = (a_1^t, \ldots, a_n^t)$ induces the state transition $s^{t+1} \sim P(\cdot \mid s^t, \boldsymbol{a}^t)$ and produces a team reward $r^t = R(s^t, \boldsymbol{a}^t)$ shared by all agents. The objective is to maximize the expected discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r^t]$. To mitigate environmental non-stationarity while preserving scalability, most MARL algorithms follow centralized training and decentralized execution (CTDE): global information about the team and environment is used to shape learning signals during training, whereas each agent executes using only its own local history.

A purely decentralized policy $\pi_i(a_i \mid o_i)$ can only condition on a single agent's local observation, which may fail to resolve ambiguity caused by unseen teammates, hidden opponents, or delayed consequences. Communication addresses this directly: each agent generates a message $m_i^t = f_{\text{msg}}(o_i^t)$, and a receiving agent i aggregates messages from a subset of teammates $\mathcal{V}_i^t \subseteq \mathcal{N} \setminus \{i\}$ into a context $c_i^t = \text{Agg}\big(\{m_j^t \mid j \in \mathcal{V}_i^t\}\big)$. The policy is then upgraded from $\pi_i(a_i^t \mid o_i^t)$ to $\pi_i(a_i^t \mid o_i^t, c_i^t)$, where c_i^t provides teammate information for action selection.

4 Method

This section presents the design of DDACOM. Figure 1(a) shows decentralized execution for one agent. The message policy generates continuous-vector messages from local encoded features, and the message-sending module (c) selects transmissions under the current bandwidth budget. The message-fusion module (b) aggregates available teammate messages into a context used by the action policy together with local features. During training (d), attention feedback and auxiliary-critic value estimates provide a dedicated reward for independently optimizing the message policy.

4.1 Message generation and fusion

The message-fusion module aggregates teammates' messages into a decision context. We describe it first because its attention weights also provide feedback for training the message policy.

4.1.1 Message-fusion module

At time step t, each agent i encodes its local observation o_i^t into a feature $h_i^t \in \mathbb{R}^{d_h}$ and uses this feature to produce a candidate message $m_i^t \in \mathbb{R}^{d_m}$ (Section 4.1.2). Sending decisions and network delivery precede message fusion and action selection. Let $\hat{m}_{j \to i}^t$ denote the latest message from agent j delivered to receiver i before its current action. The receiver fuses $\hat{M}_{-i}^t = \{\hat{m}_{j \to i}^t\}_{j \neq i}$, reusing cached messages when no new message arrives. In the fusion equations below, m_j denotes the available message from sender j to receiver i; time indices are omitted where clear.

To produce both a fused context for decision making and an explicit feedback signal for the message policy, we use an element-wise multiplicative attention mechanism. Each agent uses its own encoded feature h_i^t as the query, $q_i = W_q h_i$, and projects each incoming message into a key and value, $k_j = W_k m_j$, $v_j = W_v m_j$, with learnable W_q, W_k, W_v. Compared with scaled dot-product attention, which sums over feature dimensions before applying softmax, here the per-dimension match is preserved up to the scoring step:

$$e_{ij} = w_a^\top (q_i \odot k_j), \tag{1}$$

where $\odot$ is element-wise multiplication. The learnable weights w_a determine how each feature dimension contributes to the attention score. The resulting attention weights $\alpha_{ij} = \text{softmax}_j(e_{ij})$ are used both to aggregate messages and to provide receiver-side feedback for message-policy training

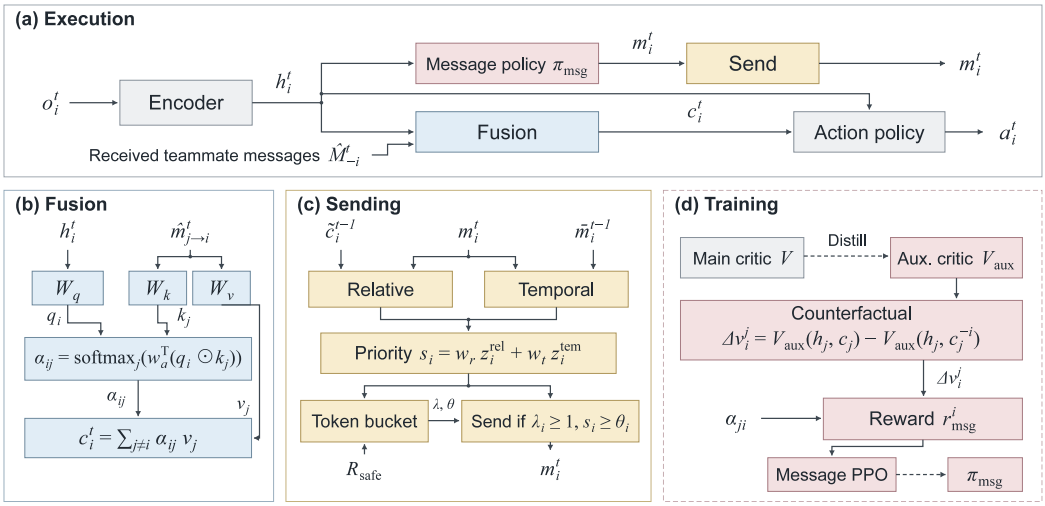


Figure 1: Overall architecture of DDACOM. (a) Decentralized execution for agent i. (b) Attention-based message fusion. (c) Token-bucket-based message sending. (d) Message-policy training with attention and counterfactual feedback from teammates $j \neq i$. Fusion uses the latest delivered or cached messages. The dashed panel is used only during training.

(Section 4.1.2). The fused context is $c_i = \sum_{j \neq i} \alpha_{ij} v_j$, and the action policy conditions on both the local feature and the fused context: $a_i^t \sim \pi_{\text{act}}(\cdot \mid [h_i^t; c_i^t])$.

Thus, α_{ij} measures how much the fusion module weights sender j's message when forming receiver i's context. During training, we combine this attention feedback with an auxiliary-critic estimate of the message's contribution.

4.1.2 Message-generation module

In an end-to-end setup, the message head receives gradient only from the team-reward signal, which must propagate backward through the message fusion and the action policy before reaching it. Sparse team rewards can make credit assignment to individual messages difficult.

We therefore implement message generation as a separately optimized policy module π_{msg}. Its input is the local feature $h_i^t \in \mathbb{R}^{d_h}$ encoded from agent i's observation, and its output is a continuous candidate message $m_i^t \in \mathbb{R}^{d_m}$, sampled as $m_i^t \sim \pi_{\text{msg}}(\cdot \mid h_i^t)$. We treat the message vector as the action of this policy and optimize its parameters with a separate PPO objective using the dedicated message reward r_{msg}. The action policy and main critic are trained using the team reward. The message reward is computed from receiver-side feedback for transmitted messages.

The central challenge is how to evaluate the contribution of an individual message to team cooperation [26]. If the attention weight α_{ij} is used directly as the reward, agents may learn to generate messages that attract attention but provide little decision value. We therefore propose Influence-Aware Counterfactual Credit Assignment, which uses counterfactual marginal value to calibrate attention feedback. The resulting reward encourages messages that both attract receiver attention and improve cooperative decision making.

(1) Auxiliary critic. To estimate the marginal value of a message, we introduce an auxiliary critic $V_{\text{aux}}([h_i; c_i])$ that outputs a scalar value estimate from the receiver feature and its fused context. To align this auxiliary critic with the main critic, we distill from the main-critic value $V(x_i^t)$:

$$\mathcal{L}_{\text{aux}} = \mathbb{E}_{i,t}\left[\left(V_{\text{aux}}([h_i^t; c_i^t]) - V(x_i^t)\right)^2\right], \tag{2}$$

where x_i^t is the input to the main critic, such as centralized observations or the global state under centralized training. During auxiliary-critic training, gradients are stopped for both the input features and the distillation target, and only the parameters of V_{aux} are updated. This prevents the auxiliary branch from perturbing the optimization of the main actor-critic.

(2) Counterfactual marginal value. We estimate the marginal contribution of message m_j to receiver i by comparing the auxiliary-critic values with and without that message:

$$\Delta v_j^i = V_{\text{aux}}([h_i; c_i]) - V_{\text{aux}}([h_i; c_i^{-j}]), \tag{3}$$

where c_i^{-j} is obtained by removing m_j, recomputing and renormalizing attention over the remaining messages, and aggregating them. Both evaluations keep h_i and the auxiliary-critic parameters fixed; only the fused context changes. A positive (negative) Δv_j^i means that the message raises (lowers) the auxiliary-critic estimate. We use this auxiliary-critic estimate as a training signal for the message policy, together with the receiver's attention feedback.

(3) Hybrid reward. The message reward combines two terms: (i) attention above the uniform baseline $1/(N-1)$; and (ii) the estimated counterfactual value. The two terms jointly determine the magnitude and sign of the reward. We define

$$r_{\text{msg}}^j = \frac{1}{N-1} \sum_{i \neq j} \left(\alpha_{ij} - \frac{1}{N-1} + \lambda_v \tanh(\Delta v_j^i) \right), \tag{4}$$

where N is the number of agents, λ_v controls the weight of the marginal-value term, and $\tanh$ bounds the value contribution to a stable range. Averaging over receivers combines their feedback into a single reward for sender j's message. Optimizing π_{msg} with this reward yields messages that combine receiver-side attention relevance and cooperative value. Although computing Δv_j^i for all (i, j) pairs is $\mathcal{O}(N^2)$ per step, V_{aux} is a small MLP and is only evaluated during centralized training, so deployment cost is unaffected.

4.2 Message-sending module

Section 4.1.2 addresses the cooperative value of message content. Under dynamically constrained networks, however, not every generated message can be transmitted. DDACOM therefore introduces a message-sending module that decides whether to send a message. The module consists of message-priority estimation and dynamic token-bucket scheduling [17]. The former estimates semantic importance, while the latter represents the available network bandwidth and constrains the final sending decision.

4.2.1 Message-priority estimation

We define the transmission priority of a message based on its deviation from teammates' information and from the sender's own historical messages. DDACOM evaluates two complementary quantities: relative deviation, which measures the difference from teammates' information, and temporal deviation, which measures change from the sender's own history. Their departures from the sender's historical pattern determine the priority.

(1) Relative deviation. For agent i, we construct a teammate-message summary from the previous time step, $\tilde{c}_i^{t-1} = \sum_{j \neq i} \alpha_{ij}^{t-1} \hat{m}_{j \to i}^{t-1}$, where α_{ij}^{t-1} is the attention weight produced by the message-fusion mechanism at the previous time step. The summary $\tilde{c}_i^{t-1}$ aggregates raw messages and can therefore be compared directly with m_i^t, whereas the action context c_i aggregates projected values. The relative deviation is then

$$u_i^{\text{rel},t} = \frac{1}{d_m} \left\| m_i^t - \tilde{c}_i^{t-1} \right\|_2^2. \tag{5}$$

Using the previous-step summary makes this score available before the current sending decision.

(2) Temporal deviation. Even when a message differs from teammates' information, it may still contribute little new information if it is highly similar to the sender's recent messages. We maintain an exponential moving average of historical messages, $\bar{m}_i^t = \rho \bar{m}_i^{t-1} + (1-\rho) m_i^t$, and use $\bar{m}_i^{t-1}$ as the historical reference:

$$u_i^{\text{tem},t} = \frac{1}{d_m} \left\| m_i^t - \bar{m}_i^{t-1} \right\|_2^2. \tag{6}$$

Both deviations use sender-local information: the candidate message, cached teammate messages, and the sender's own message history.

Because agents can differ in message scale and observation distribution, directly comparing raw deviations can be biased. We normalize each deviation using the sender's own historical distribution. For any deviation u, DDACOM maintains its running mean μ and variance ν, and computes $z = \left|(u-\mu)/\sqrt{\nu+\epsilon}\right|$, where ϵ is a numerical-stability constant. The absolute value captures unusually large and unusually small deviations from the sender's historical pattern. Larger scores increase transmission priority, allowing departures in either direction to influence message selection within the available token budget. This places the normalized scores $z_i^{\text{rel},t}$ and $z_i^{\text{tem},t}$ on a common scale before combining them into a single transmission priority.

(3) Combined priority score. The final priority score is $s_i^t = w_r z_i^{\text{rel},t} + w_t z_i^{\text{tem},t}$, with $w_r + w_t = 1$. DDACOM uses a short warm-up window at the beginning of each episode in which all agents send messages, allowing the running statistics to stabilize before priority-based scheduling is applied. These transmissions are included in traffic and budget accounting.

4.2.2 Token-bucket-based sending decision

Message priority captures semantic importance, but real networks impose physical bandwidth constraints. The probe-estimated safe rate sets the token-recovery budget, message priority adjusts recovery within this budget, and the resulting token count determines the sending threshold.

Each agent i maintains a token pool λ_i. Sending one message consumes one token. At each time step, tokens are replenished with rate r_i^t and clipped to $[0, \lambda_{\max}]$. Let λ_i^t denote the number of available tokens before the sending decision at time t. Unlike a fixed-rate token bucket, DDACOM defines the token-recovery rate as the sum of a network base rate and a semantic adjustment:

$$r_i^t = r_{\text{base}}^t + r_{\text{adjust}}^t \tanh\big(s_i^t/\tau\big)\,, \tag{7}$$

where r_{base}^t is the base token-recovery rate, r_{adjust}^t is the token-recovery margin reserved for semantic adjustment, and τ is a smoothing coefficient. The network-safe rate is determined according to each scenario's packet-loss and mean-delay requirements. Before task execution, we probe the network at different sending rates and select the highest tested per-agent rate that meets these requirements. During execution, measurements from task messages, supplemented by auxiliary probe packets when needed, update the estimate as network conditions change. Reusing existing transport-layer feedback, such as congestion-window changes, round-trip time, loss indications, and ECN feedback, could support these updates with less dedicated probe traffic.

For a per-agent safe rate R_{safe} in bit/s, the token budget per decision step is $R_{\text{safe}}\Delta t/(8S)$, where S is the message-packet size in bytes and Δt is the decision interval. The split between r_{base}^t and r_{adjust}^t can be adjusted as task requirements change while preserving the total budget. A larger adjustment share makes token recovery more responsive to message priority, whereas a larger base share gives agents more uniform recovery rates.

The token update is

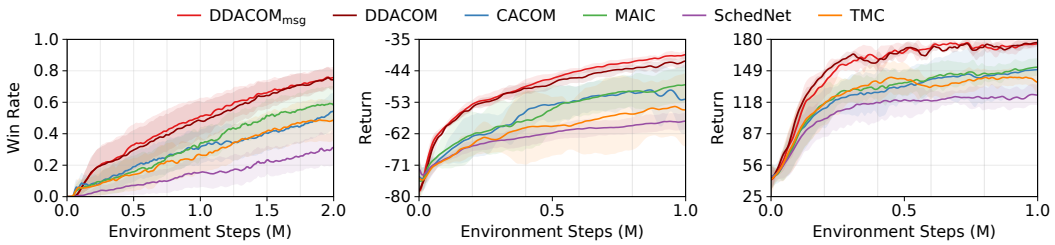
$$\lambda_i^t = \text{clip}\big(\lambda_i^{t-1} + r_i^t - \text{Send}_i^{t-1}, 0, \lambda_{\max}\big)\,. \tag{8}$$

Before the sending decision at time t, the pool is replenished according to the current recovery rate and decremented according to the previous sending decision. This design reflects the physical bandwidth constraint while allowing unused tokens to accumulate for later decisions and giving high-priority messages a limited advantage in token recovery.

The threshold should rise as tokens become scarce, so that limited resources are reserved for high-priority messages. We therefore couple the threshold to the token count via $\theta_i^t = \theta_0 \exp(-(\lambda_i^t - 1)/\lambda_{\text{scale}})$, where θ_0 is a base threshold scalar and λ_{scale} controls its sensitivity to the token count. The final sending decision requires both sufficient tokens for one transmission and a priority score at or above the current token-dependent threshold:

$$\text{Send}_i^t = \mathbf{1}[\lambda_i^t \geq 1] \cdot \mathbf{1}[s_i^t \geq \theta_i^t]. \tag{9}$$

$\lambda_i^t \geq 1$ prevents token-budget violation, while $s_i^t \geq \theta_i^t$ reserves limited resources for semantically important messages. If no new message arrives before action selection, receivers use the latest delivered message; late arrivals update the cache for subsequent decisions.



(a) SMAC: average win rate over 8 maps (b) MPE: Cooperative Nav. (c) MPE: Predator-Prey

Figure 2: Cooperative performance across SMAC and MPE. (a) Average win rate across eight SMAC maps. (b)–(c) Returns on the two MPE scenarios.

5 Experiments

We evaluate DDACOM on the StarCraft II Multi-Agent Challenge (SMAC) [27] and the Multi-Agent Particle Environment (MPE) [28] to assess cooperative performance and communication efficiency. Network simulations further assess safe-rate tracking and task performance under bandwidth constraints. This section first presents the experimental setup (§5.1), then reports the main SMAC and MPE results (§5.2), dynamic network adaptability (§5.3), and ablation studies (§5.4).

5.1 Experimental setup

Environments. SMAC experiments use eight maps covering different unit compositions, team sizes, and coordination patterns. MPE experiments use Cooperative Navigation and Predator-Prey, two representative scenarios in which communication is beneficial.

Network simulation. We use ns-3 [29] to replay the sending module's transmission schedules and measure packet loss and safe-rate utilization. For task-level bandwidth evaluations, we develop a network simulator that integrates with communication-based MARL training and evaluation. At each decision step, it models links, queues, and packet transmission; delivery and arrival times determine the messages available for action selection. The resulting actions change the task state and subsequent messages, forming a closed loop between network transmission and agent decisions. We use this simulator for both fixed- and dynamic-bandwidth task evaluations.

Baselines and methods. We compare DDACOM with four representative communication-based MARL baselines: CACOM [15], MAIC [19], SchedNet [23], and TMC [14]. DDACOM$_{\text{msg}}$ combines the proposed message-generation module with full broadcast. DDACOM additionally uses the adaptive message-sending module to select which messages are transmitted. All baselines use the original hyperparameters from their open-source implementations; each method is run with five seeds. Tables 1 and 2 report mean $\pm$ standard deviation across seeds. During DDACOM training, the upper bound of token-recovery rate is randomly resampled at the end of each episode, exposing the message-sending module to multiple bandwidth budgets. Implementation details are in Appendix C.

Evaluation metrics. For cooperative performance, SMAC reports win rate and MPE reports return. For communication efficiency, we focus on whether a method can fully utilize the available safe rate while staying within it, and report six network-simulation metrics: **theoretical capacity** R_{theory} (Kbps) [30], **safe rate** R_{safe} (Kbps), **safe-use rate** (actual sending rate divided by the safe rate), **window-over rate** (the fraction of 500 ms sliding windows that temporarily exceed the safe rate), **total-over rate** (whether the total transmission over the whole run exceeds the safe rate), and **packet loss**. R_{theory} is computed from the channel model, while R_{safe} is estimated by the network probing module using probe-packet loss and latency. The two over-rate metrics respectively capture short-term burst violations and whole-run violations.

5.2 Cooperative performance and bandwidth robustness

Figure 2 reports the average SMAC win rate and the returns in the two MPE scenarios. Per-map SMAC win-rate curves are provided in Appendix A.

Table 1: SMAC win rates (%) under different aggregate bandwidths.

Aggregate bandwidth	Safe budget (%)	DDACOM	MAIC	TMC	CACOM	SchedNet
Unlimited	100.0	$\mathbf{84.2 \pm 3.7}$	74.8 ± 2.3	69.0 ± 2.1	71.6 ± 1.9	57.6 ± 3.3
1.5 Mbps	98.9	$\mathbf{85.1 \pm 0.7}$	75.1 ± 1.6	69.5 ± 3.1	69.2 ± 3.0	55.1 ± 2.1
1.0 Mbps	72.0	$\mathbf{83.8 \pm 2.5}$	72.4 ± 3.1	63.2 ± 4.9	70.6 ± 3.1	55.2 ± 3.6
0.5 Mbps	21.5	$\mathbf{79.7 \pm 2.2}$	62.9 ± 5.2	35.8 ± 5.0	58.1 ± 9.2	44.9 ± 7.4

Table 2: DDACOM under OOD bandwidth and weakened-communication baselines on SMAC.

Bandwidth condition	Average send rate	Win rate (%)
Training distribution	46%	82.3 ± 3.6
OOD low bandwidth	24%	80.1 ± 4.5
Random scheduling	24%	75.0 ± 6.3
No communication	0%	67.5 ± 5.1
Noisy messages	100%	65.8 ± 7.6

On the eight evaluated SMAC maps, DDACOM and $\text{DDACOM}_{\text{msg}}$ achieve the highest average win rates among the tested baselines. The lead is most pronounced on `corridor`, a super-hard map demanding tight long-range coordination, where most baselines fail to converge stably. The advantage is not uniform across maps: on `3s5z`, for instance, some baselines remain competitive. On MPE, both variants achieve high returns in Cooperative Navigation and Predator-Prey. The small gap between DDACOM and $\text{DDACOM}_{\text{msg}}$ shows that the message-sending module reduces communication frequency while maintaining similar task performance.

Win-rate degradation under different bandwidths. Using our closed-loop simulator, we evaluate trained policies under fixed aggregate bottleneck bandwidths shared by all agents (Table 1). Each map retains its actual number of agents and includes three interfering flows of 0.1 Mbps each. Unlimited denotes full broadcast, with one message per agent every 10 ms. Probing converts each bandwidth into a safe per-agent sending budget. Relative to full broadcast, these budgets average 98.9%, 72.0%, and 21.5% at 1.5, 1.0, and 0.5 Mbps, respectively, weighted by each map's agent count. These budgets specify the available sending rates, while each method's policy determines actual transmissions; calibration details appear in Appendix B.

DDACOM achieves the highest win rate among the evaluated baselines at every tested bandwidth. At 0.5 Mbps, its win rate decreases from 84.2% to 79.7%, a relative drop of 5.34%, compared with 15.91%, 48.12%, 18.85%, and 22.05% for MAIC, TMC, CACOM, and SchedNet, respectively.

To further assess bandwidth robustness and the role of communication, we evaluate four conditions: an out-of-distribution (OOD) bandwidth regime, in which the test budget is sampled from a strictly lower range than at training ($r_{\text{base}} \sim \mathcal{U}(0.1, 0.5)$ versus $\mathcal{U}(0.5, 1.5)$); random scheduling at a matched sending rate; no communication; and noisy messages (Gaussian samples matched to the empirical message statistics). Table 2 shows that DDACOM still reaches 80.1% under the OOD bandwidth, whereas random scheduling at the same $\sim 24\%$ sending rate drops to 75.0%, and removing messages or replacing them with noise drops further to 67.5% and 65.8%. These results show that message content and sending decisions both affect cooperative performance, and that the trained model handles lower bandwidth budgets without retraining.

5.3 Dynamic network adaptability

Safe-rate tracking. We first evaluate how the sending module tracks changing network capacity in ns-3. Probe packets estimate the safe rate from packet loss and latency, and this rate determines the token-bucket recovery budget. The controller generates a sending schedule, which is then replayed in ns-3 to measure packet loss and safe-rate utilization. We consider three scenarios in which the agent population, the interference intensity, or both change within an episode while the learned policy remains fixed. Static results and the time-resolved rate-tracking curve appear in Appendix B.

Table 3: Safe-rate tracking under dynamic network conditions.

Scenario	R_{theory} (Kbps/agent)	R_{safe} (Kbps/agent)	Safe-use	Window-over	Total-over	Packet loss
Varying agents	54.75	52.24	95.5%	0.00%	0.00%	0.00%
Varying interference	38.99	38.49	97.9%	0.00%	0.00%	0.29%
Varying both	35.16	34.04	97.2%	1.36%	0.00%	0.00%

Table 4: SMAC win rates under ideal full communication and dynamic bandwidth.

Method	Ideal full communication (%)	Dynamic bandwidth (%)	Relative drop (%)
DDACOM	$\mathbf{77.7 \pm 3.2}$	$\mathbf{73.9 \pm 5.9}$	$\mathbf{4.89}$
Dynamic-SchedNet	63.2 ± 4.9	54.1 ± 6.9	14.40

Across the three dynamic scenarios in Table 3, safe-rate utilization exceeds 95%, with packet loss at or below 0.29%. The 0% total-over rate means that transmissions remain within the accumulated safe budget over the whole run. Unused tokens can accumulate and support brief sending bursts, so the transmitted traffic may exceed the safe budget in individual 500 ms sliding windows. The largest window-over rate is 1.36%, observed when both agent population and interference vary, and packet loss is 0% in that scenario. These results show that the module uses the available bandwidth efficiently while keeping overall traffic within budget.

Task performance under dynamic bandwidth. We next evaluate whether DDACOM maintains cooperative performance when bandwidth and interference change within an episode, using the closed-loop simulator described in Section 5.1. We choose SchedNet [23] for this comparison because its top-k scheduler allows the sending rate to vary by changing the number k of senders per step. DDACOM adapts its sending budget through online probing. In our Dynamic-SchedNet variant, we set k_t to DDACOM's number of transmission attempts at step t and select the k_t agents with the highest learned priorities. Both methods retain their original backbones and training procedures, with all learned parameters fixed during evaluation.

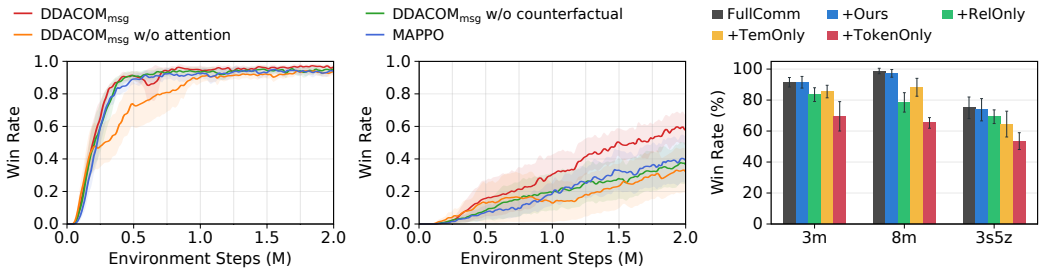
We evaluate the two methods on `3s_vs_4z`, `5m_vs_6m`, `8m`, and `8m_vs_9m` with five seeds. Every 5–7 decision steps, we resample the aggregate bottleneck bandwidth from $\{0.5, 1.0, 1.5\}$ Mbps, the number of interfering flows from $\{1, 2, 3\}$, and each flow's rate from $\mathcal{U}(0.08, 0.12)$ Mbps, with at least one parameter changing. Receivers use messages delivered before action selection and otherwise retain their latest cached messages. Ideal full communication delivers every current message before action selection. Table 4 reports mean win rates over the four maps, with standard deviations across seeds. Relative to their respective ideal full-communication means, DDACOM and Dynamic-SchedNet show win-rate drops of 4.89% and 14.40%, respectively. Network-stress tests appear in Appendix B.1.

5.4 Ablation studies

We conduct ablations along the two decoupling dimensions of DDACOM. First, we test whether both attention feedback and counterfactual marginal value are necessary in the message-generation reward. Second, we attach the message-sending module to an independent full-broadcast backbone to evaluate its effectiveness as a standalone component.

Ablation of the message-generation module. We remove key components of the influence-aware credit assignment and compare four variants: (i) DDACOM$_{\text{msg}}$, the complete message-generation method; (ii) DDACOM$_{\text{msg}}$ w/o counterfactual, which keeps only the attention-feedback term; (iii) DDACOM$_{\text{msg}}$ w/o attention, which keeps only the counterfactual marginal-value term; and (iv) MAPPO, a no-communication lower bound.

Figure 3(a) reports the average win rate on three lower-complexity maps. The variants differ little and remain close to MAPPO, indicating that the main MARL policy is already sufficient in these settings and that the marginal benefit of communication is less visible here. Figure 3(b) reports results on three more coordination-intensive maps. DDACOM$_{\text{msg}}$ achieves higher win rates than the other variants, while the two single-signal variants can fall below MAPPO. Combining attention feedback



(a) Lower-complexity maps (b) Coordination-intensive maps (c) Message-sending ablation

Figure 3: Ablations of message generation and message sending.

and counterfactual marginal value therefore improves performance over either signal alone, with the benefit most evident on maps requiring stronger coordination.

Ablation of the message-sending module. To isolate the effect of the message-sending module from message generation, we attach the message-sending module to a MAPPO-based full-broadcast backbone (**FullComm**) that contains none of DDACOM's message-generation components. We compare five variants: (i) **FullComm**, full broadcast without the message-sending module; (ii) **+Ours**, with the complete message-sending module; (iii) **+RelOnly**, only relative deviation; (iv) **+TemOnly**, only temporal deviation; and (v) **+TokenOnly**, only the token-bucket constraint, with semantic-priority estimation removed.

Figure 3(c) shows that FullComm + Ours reaches win rates close to FullComm, confirming that the message-sending module can be attached to a full-broadcast backbone with little performance loss. +RelOnly and +TemOnly both degrade and their ordering varies across maps, indicating that the two deviations provide complementary information for message selection. +TokenOnly degrades on all maps, showing that the token-budget constraint alone cannot identify cooperation-critical messages and that semantic-priority estimation is essential. The two components together are also what drives DDACOM's adaptation to dynamic bandwidth: the token bucket converts the runtime safe rate into a per-step budget, while the priority signal decides which messages are worth that budget.

6 Conclusion

We presented DDACOM, a communication framework for cooperative MARL under dynamic networks. DDACOM decouples message sending and message generation from the main action policy. The message-sending module combines semantic message priority with a dynamic token bucket, adapting the communication frequency to real-time safe bandwidth; the message-generation module trains an independent message policy with receiver-side attention feedback and counterfactual marginal value, improving the cooperative utility of transmitted messages. Experiments on SMAC and MPE and network simulations in ns-3 show that this design preserves cooperative performance under tightened bandwidth, sustains over 95% safe-rate utilization, and keeps packet loss below 0.3% under dynamic and out-of-distribution network conditions. Bootstrapping the safe rate currently relies on dedicated probe packets, whose bandwidth cost is small but not negligible; reusing the congestion-control [31–33] signals already exposed by the TCP/IP stack—round-trip time, ACK-based loss indication, and ECN feedback—could provide comparable information without injecting probe traffic and would integrate the controller more naturally with existing transport protocols, a direction we leave to future work.

Acknowledgments and Disclosures

This work was supported by the Beijing Natural Science Foundation under Grant No. 4262018 and the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDB0680301. The authors declare no competing interests.

References

[1] Yang Lv, Jinlong Lei, and Peng Yi. A local information aggregation-based multiagent reinforcement learning for robot swarm dynamic task allocation. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

[2] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE transactions on intelligent transportation systems*, 21 (3):1086–1095, 2019.

[3] Pengyan Dong, Jiahong Liu, Hang Tao, Rukhsana Ruby, Muwei Jian, and Hanjiang Luo. An optimized scheduling scheme for uav-usv cooperative search via multi-agent reinforcement learning approach. In *2024 20th International Conference on Mobility, Sensing and Networking (MSN)*, pages 172–179. IEEE Computer Society, 2024.

[4] Jiajun Chai, Zijie Zhao, Yuanheng Zhu, and Dongbin Zhao. A survey of cooperative multi-agent reinforcement learning for multi-task scenarios. *Artificial Intelligence Science and Engineering*, 1(2):98–121, 2025.

[5] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29, 2016.

[6] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

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

[8] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*, 2018.

[9] Marwa Chafii, Salmane Naoumi, Reda Alami, Ebtesam Almazrouei, Mehdi Bennis, and Merouane Debbah. Emergent communication in multi-agent reinforcement learning for future wireless networks. *IEEE Internet of Things Magazine*, 6(4):18–24, 2023.

[10] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31, 2018.

[11] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

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

[13] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni. Learning agent communication under limited bandwidth by message pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5142–5149, 2020.

[14] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Succinct and robust multi-agent communication with temporal message control. *Advances in neural information processing systems*, 33:17271–17282, 2020.

[15] Xinran Li and Jun Zhang. Context-aware communication for multi-agent reinforcement learning. *arXiv preprint arXiv:2312.15600*, 2023.

[16] Guangzheng Hu, Yuanheng Zhu, Dongbin Zhao, Mengchen Zhao, and Jianye Hao. Event-triggered multi-agent reinforcement learning with communication under limited-bandwidth constraint. *arXiv preprint arXiv:2010.04978*, 2020.

[17] Tatsuya Fukui, Yuki Sakaue, Katsuya Minami, and Tomohiro Taniguchi. Delay-based shaper with dynamic token bucket algorithm for deterministic networks. *IEEE Access*, 10:114424–114433, 2022.

[18] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[19] Lei Yuan, Jianhao Wang, Fuxiang Zhang, Chenghe Wang, Zongzhang Zhang, Yang Yu, and Chongjie Zhang. Multi-agent incentive communication via decentralized teammate modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9466–9474, 2022.

[20] Yaru Niu, Rohan R Paleja, and Matthew C Gombolay. Multi-agent graph-attention communication and teaming. In *AAMAS*, pages 964–973, 2021.

[21] Cong Guan, Feng Chen, Lei Yuan, Chenghe Wang, Hao Yin, Zongzhang Zhang, and Yang Yu. Efficient multi-agent communication via self-supervised information aggregation. *Advances in Neural Information Processing Systems*, 35:1020–1033, 2022.

[22] Di Xue, Lei Yuan, Zongzhang Zhang, and Yang Yu. Efficient multi-agent communication via shapley message value. In *IJCAI*, pages 578–584, 2022.

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

[24] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. *Advances in neural information processing systems*, 32, 2019.

[25] Diyi Hu, Chi Zhang, Viktor Prasanna, and Bhaskar Krishnamachari. Learning practical communication strategies in cooperative multi-agent reinforcement learning. In *Asian Conference on Machine Learning*, pages 467–482. PMLR, 2023.

[26] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[27] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019.

[28] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[29] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[30] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. *ACM SIGCOMM Computer Communication Review*, 32(4):295–308, 2002.

[31] Ali Ghaffari. Congestion control mechanisms in wireless sensor networks: A survey. *Journal of network and computer applications*, 52:101–115, 2015.

[32] Huiling Jiang, Qing Li, Yong Jiang, Gengbiao Shen, Richard Sinnott, Chen Tian, and Mingwei Xu. When machine learning meets congestion control: A survey and comparison. *Computer Networks*, 192:108033, 2021.

[33] Rasool Al-Saadi, Grenville Armitage, Jason But, and Philip Branch. A survey of delay-based and hybrid tcp congestion control algorithms. *IEEE Communications Surveys & Tutorials*, 21 (4):3609–3638, 2019.

A Experimental details and per-map results

The appendix contains the per-map SMAC training win-rate curves, the network-simulation protocol and rate-tracking results, additional closed-loop task evaluations, and the key implementation details of the message-generation and message-sending modules.

Figure 4 reports the eight SMAC maps used in the main per-map comparison.

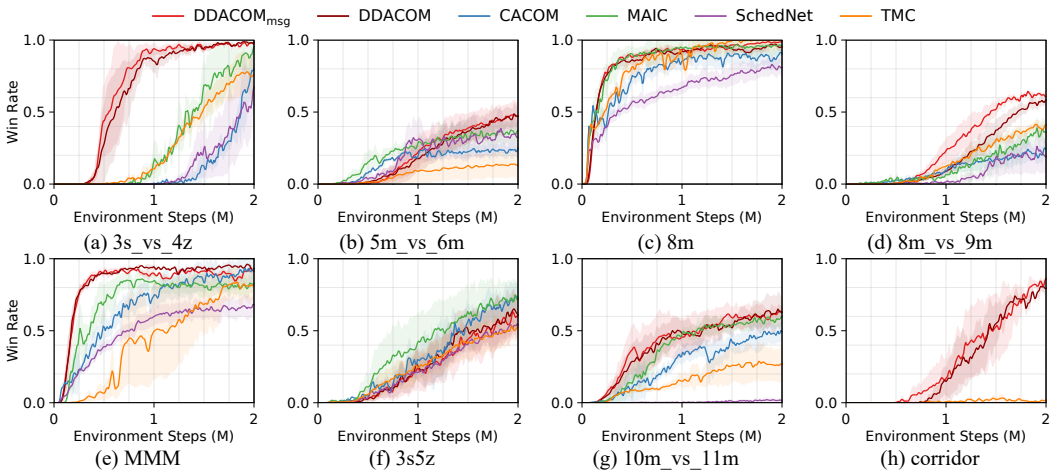


Figure 4: Per-map training win rate across eight SMAC maps.

Figure 5 reports the main-critic and auxiliary-critic training losses averaged across the eight SMAC maps over 2 million environment steps.

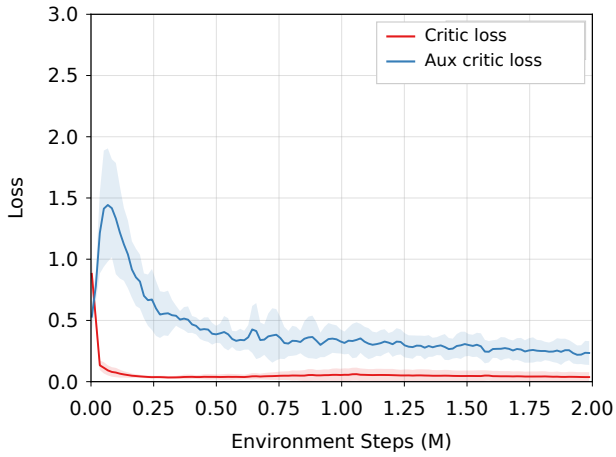


Figure 5: Main-critic and auxiliary-critic training losses averaged across eight SMAC maps.

B Network simulation protocol and results

For Table 1, a closed-loop network simulator models links, queues, and packet transmission at each decision step, so message delivery directly affects policy inputs. We calibrate each map using its actual number of agents, aggregate bottleneck bandwidths of 1.5, 1.0, and 0.5 Mbps, and three interfering flows of 0.1 Mbps each. Under full broadcast, each agent sends one message at every decision step, so the per-agent full-broadcast rate is

$$R_{\text{full}} = \frac{8S}{\Delta t},$$

where the message-packet size is $S = 128\,\mathrm{B}$ and the decision interval is $\Delta t = 10$ ms. This gives $R_{\text{full}} = 102.4$ Kbps. For each bandwidth condition, probing searches for the maximum per-agent network-safe rate R_{safe} with packet loss at most 1% and mean delay at most 10 ms. The safe budget for each map is

$$\frac{\min(R_{\text{safe}}, R_{\text{full}})}{R_{\text{full}}} \times 100\%.$$

Weighting these budgets by the number of agents on each of the eight maps gives 98.9%, 72.0%, and 21.5% at 1.5, 1.0, and 0.5 Mbps, respectively. These percentages describe the available budget; actual sending depends on each method's communication policy.

The ns-3 rate-tracking experiments use the same pipeline for static and dynamic settings. First, we construct the ns-3 bottleneck-link topology and set the number of active agents and the interference intensity. Second, probe packets estimate the safe sending rate from loss and latency, and this rate is converted into the token-bucket recovery rate used by the sending controller. Third, the controller runs and records each agent's sending decisions. Finally, the resulting sending schedule is replayed in ns-3 to measure end-to-end network behavior. Theoretical rate and safe rate are reported as per-agent rates. For dynamic scenarios, we first compute the time-averaged aggregate rate over active agents and then divide it by the average number of active agents, obtaining the average per-active-agent rate.

We report three scheduler-centered metrics. **Safe-use** is the achieved sending rate divided by R_{safe}, reported as a percentage. **Window-over** is the fraction of 500 ms sliding windows in which the transmitted packets exceed the safe budget of that window, measuring short-term burst violations. **Total-over** is the fraction by which the total number of transmitted packets exceeds the total safe packet budget over the whole run, measuring overall budget violation. Packet loss is measured during the ns-3 replay stage and reported as a percentage.

The static-bandwidth evaluation contains two scenarios. Scenario A fixes interference intensity and varies the number of agents; Scenario B fixes the number of agents and varies the interference intensity. The dynamic scenario changes the number of agents, the interference intensity, or both within a single replay episode using piecewise-constant schedules. Static results show that the sending controller keeps the total transmission within the safe budget and avoids end-to-end packet loss across different loads and interference levels.

The low-pressure rows in Tables 5 and 6 have lower safe-use rates because the probed safe bandwidth is already higher than the demand of full communication. In this regime, each agent can already send one message at every decision step without reaching the safe-rate limit, so the actual traffic is capped by the full-broadcast demand rather than by R_{safe}. Consequently, the ratio between actual sending rate and R_{safe} appears low even though there is no communication shortage. As the number of agents or interference intensity increases, R_{safe} approaches or falls below the full-broadcast demand, and the controller uses nearly all available safe bandwidth while keeping packet loss near zero.

Figure 6 visualizes rate tracking in the dynamic setting where both agent population and interference intensity change. The safe rate varies substantially over time as the network condition changes, while the actual sending rate follows the safe-rate envelope from below. This time-resolved result complements Table 3: the controller adapts its token recovery and sending threshold to the probed safe rate, uses most of the available bandwidth, and avoids sustained budget violation.

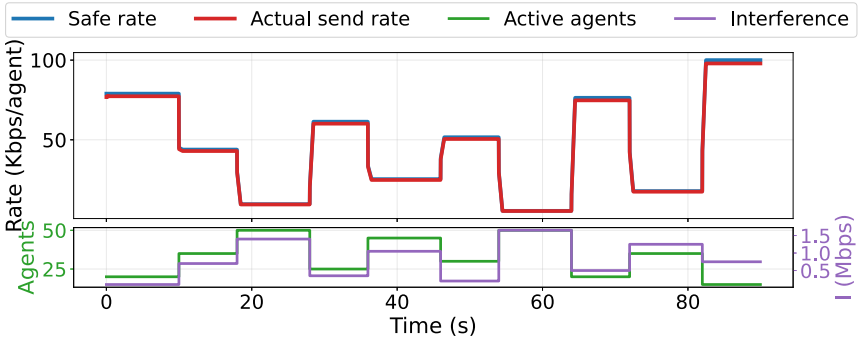


Figure 6: Tracking the probe-estimated safe rate under dynamic network changes.

The main network-simulation parameters include bottleneck bandwidth, link delay, queue size, message-packet size, decision interval, probe constraints, and sliding-window length. The detailed configuration is summarized below.

Network-simulation configuration

Parameter	Value
Bottleneck bandwidth	2.0 Mbps for network simulation; 1.5/1.0/0.5 Mbps for the fixed-bandwidth comparison in Table 1
Table calibration interference	Three flows of 0.1 Mbps each
Bottleneck / access delay	5 ms / 1 ms
Bottleneck queue	50 packets
Packet size S	128 B
Decision interval Δt	10 ms
Probe constraints	loss $\leq 1\%$, average delay ≤ 10 ms
Probe simulation	6 s, 8 binary-search iterations
Evaluation window	500 ms
Table 1 calibration	Actual map-specific agent count, $N \in \{3, 5, 6, 8, 10\}$
Static Scenario A	$I = 0.10$ Mbps, $N \in \{5, 10, 20, 30, 50\}$
Static Scenario B	$N = 10$, $I \in \{0.10, 0.50, 0.90, 1.30, 1.70\}$ Mbps
Dynamic scenarios	piecewise-constant changes in N, I, or both

Table 5: Static scenario A: varying agent population N under fixed interference.

N	R_{theory} (Kbps/agent)	R_{safe} (Kbps/agent)	Safe-use	Window-over	Total-over	Packet loss
5	380.00	315.44	31.4%	0.00%	0.00%	0.00%
10	190.00	157.79	62.8%	0.00%	0.00%	0.00%
20	95.00	78.97	94.7%	5.64%	0.00%	0.00%
30	63.33	52.70	96.3%	8.04%	0.00%	0.00%
50	38.00	31.51	95.8%	3.57%	0.00%	0.00%

Table 6: Static scenario B: varying interference intensity I under fixed agent population.

I (Mbps)	R_{theory} (Kbps/agent)	R_{safe} (Kbps/agent)	Safe-use	Window-over	Total-over	Packet loss
0.10	190.00	157.79	62.8%	0.00%	0.00%	0.00%
0.50	150.00	147.75	67.1%	0.00%	0.00%	0.00%
0.90	110.00	102.00	85.1%	0.00%	0.00%	0.00%
1.30	70.00	64.00	94.5%	1.87%	0.00%	0.00%
1.70	30.00	24.63	97.3%	0.32%	0.00%	0.00%

B.1 Closed-loop task–network evaluation

Network delivery, timeliness, and task performance. We conduct a closed-loop stress test at 0.5 Mbps on four maps with five paired seeds, scaling the base token-recovery rate by 1.0, 1.5, 2.0, and 2.5. Ideal and Network use the same policies and sending controller but differ in message delivery: Ideal delivers selected messages immediately, whereas Network models contention, queueing, and link-layer retransmission. Final packet delivery ratio (PDR) measures eventual delivery. Current-step unavailability includes messages lost or arriving after the 10 ms decision interval; late messages update the receiver cache after arrival. Delay statistics summarize the transmission delays of successfully delivered packets using the mean and the 95th and 99th percentiles.

Table 7 jointly reports network and task metrics. At $1.0\times$, all packets are eventually delivered, but 15.90% miss the current decision interval, and the Network win rate is 2.10 percentage points below Ideal. As load increases, PDR falls to 61.19% and current-step unavailability reaches 43.11%. Across the four loads, network transmission reduces the mean win rate by 2.10–4.70 percentage points.

Table 7: Network delivery, timeliness, and closed-loop task performance at 0.5 Mbps. Sending rates are relative to full broadcast; win-rate differences are in percentage points (pp).

(a) Network metrics

Token multiplier	Send rate (%)	Final PDR (%)	Current-step unavailability (%)	Mean delay (ms)	P95 delay (ms)	P99 delay (ms)
1.0×	11.63±4.35	100±0.00	15.90±8.64	7.24±1.57	18.23±6.58	24.15±11.60
1.5×	17.47±6.55	85.30±17.68	24.99±16.75	7.12±1.56	17.67±5.83	25.03±11.92
2.0×	23.29±8.72	68.69±34.43	39.21±29.93	7.17±1.31	17.67±6.67	27.09±9.92
2.5×	29.11±10.91	61.19±27.59	43.11±23.12	6.96±2.91	18.02±13.22	27.23±14.05

(b) Task performance

Token multiplier	Ideal win rate (%)	Network win rate (%)	Difference (pp)
1.0×	71.2±3.4	69.1±4.7	2.10
1.5×	74.9±2.8	70.2±5.2	4.70
2.0×	74.7±5.2	70.1±3.2	4.60
2.5×	75.3±4.4	71.3±4.4	4.00

C Overall pseudocode and implementation details

Algorithm 1 summarizes the overall training and execution flow of DDACOM. The action policy, message-fusion module, auxiliary critic, and message policy are trained during centralized training, while the token-bucket sending controller uses the probed safe rate at execution time to adapt communication frequency without retraining the action policy.

Algorithm 1: DDACOM training and execution

Input: cooperative environment, message size S, warm-up length K, token limit $\lambda_{\max}$, running-statistics decay, PPO hyperparameters.

Initialize: action policy π_{act}, main critic V, message policy π_{msg}, message-fusion module, auxiliary critic V_{aux}, receiver caches, message-history statistics, and token pools $\{\lambda_i\}_{i=1}^{N}$.

1. For each training episode, sample or probe a communication budget and convert the safe rate into token recovery parameters r_{base} and r_{adjust}.
2. At each time step t, encode each local observation into h_i^t and generate a candidate message $m_i^t \sim \pi_{\mathrm{msg}}(\cdot \mid h_i^t)$.
3. Compute priority s_i^t using the previous teammate-message summary $\tilde{c}_i^{t-1}$ and the sender's message history. During the first K warm-up steps, transmit all candidates; afterwards, update λ_i^t and θ_i^t, and send m_i^t only if $\lambda_i^t \geq 1$ and $s_i^t \geq \theta_i^t$.
4. Pass selected messages through the network and update receiver caches upon delivery. Only messages arriving before action selection can refresh the current context; otherwise, retain the latest delivered message $\hat{m}_{j\to i}^t$. Late arrivals update the cache for later decisions.
5. Fuse the available messages to obtain c_i^t, and cache $\tilde{c}_i^t = \sum_{j\neq i} \alpha_{ij}^t \hat{m}_{j\to i}^t$ for the next step. Sample $a_i^t \sim \pi_{\mathrm{act}}(\cdot \mid [h_i^t; c_i^t])$.
6. Store trajectories, attention weights, fused contexts, sending decisions, rewards, and critic inputs in the replay buffer.
7. Update π_{act} and V with the main PPO objective. Train V_{aux} by distilling the main-critic value while stopping gradients through its inputs and target.
8. Estimate each transmitted message's counterfactual value by removing it and recomputing normalized attention over the remaining messages. Combine this value with attention feedback to form r_{msg}, and update π_{msg} with message PPO. DDACOM$_{\mathrm{msg}}$ applies this reward to every generated message because all candidates are transmitted under full broadcast.
9. During deployment, keep the learned policies fixed. The network probe updates the safe rate online, and the token-bucket controller adapts the sending frequency through r_i^t, λ_i^t, and θ_i^t.

Table 8 summarizes the communication architecture and message-learning hyperparameters used by DDACOM and DDACOM$_{\text{msg}}$. It lists the dimensions of the message generator, fusion module, and auxiliary critic, together with the optimizer settings for each training objective.

Table 8: Communication architecture and message-learning hyperparameters.

Component	Setting
Message dimension	32
Message generator	two-hidden-layer MLP with hidden sizes $64 \to 128 \to 128$ and 32-dimensional message output
Message-fusion	query projection $64 \to 64$; shared key/value projections $32 \to 64$; element-wise query-key interaction scored by a linear layer $64 \to 1$; softmax attention over other agents' messages
Auxiliary critic	two-hidden-layer MLP, input $2H \to 128 \to 128 \to 1$, trained by value distillation from the main critic
Main PPO	actor/critic learning rates 5×10^{-4}; clipping coefficient 0.2; entropy coefficient 0.01; value-loss coefficient 1; gradient norm 10; discount factor 0.99
Message PPO	actor learning rate 10^{-4}; PPO epochs 5; clipping coefficient 0.2; entropy coefficient 0.01; gradient norm 10
Auxiliary critic optimizer	Adam with learning rate 2×10^{-4}, $\epsilon = 10^{-8}$; 10 recent steps are sampled for distillation
Attention-value mixing	attention weight coefficient 1.0; normalized marginal-value coefficient 0.2

Table 9 reports the sending-control parameters. The sender computes relative deviation between the current message and the previously effective teammate information, and temporal deviation between the current message and the exponentially smoothed message history. Both deviations are normalized by running statistics, combined into a sending score, and coupled to the budget state through a dynamic threshold. The per-step token-recovery budget is $r_{\text{base}} + r_{\text{adjust}} = R_{\text{safe}}\Delta t/(8S)$, with the per-agent safe rate in bit/s and the message-packet size in bytes. It is split with a fixed ratio $r_{\text{base}} : r_{\text{adjust}} = 8{:}2$, so that most of the budget reflects the network base rate while a small margin is reserved for semantic adjustment.

Table 9: Message-sending module parameters.

Parameter	Value / rule
Relative and temporal weights	0.7 and 0.3
History/statistics decay	0.5
Deviation normalization	$z = \lvert(u-\mu)/\sqrt{\nu+\epsilon}\rvert$, with $\epsilon = 10^{-8}$
Sending score	$s = 0.7z_{\text{rel}} + 0.3z_{\text{tem}}$
Base token recovery	resampled at episode reset from $r_{\text{base}} \sim \mathcal{U}(0.5, 1.5)$
Low-bandwidth OOD evaluation	$r_{\text{base}} \sim \mathcal{U}(0.1, 0.5)$ for Table 2
Dynamic threshold	$\theta = \theta_0 \exp(-(\lambda-1)/\lambda_{\text{scale}})$, with $\theta_0 = 0.5$ and $\lambda_{\text{scale}} = 2.0$
Sending rule	send if $\lambda \geq 1$ and $s \geq \theta$; all messages are sent during the first 5 warm-up steps

D Compute resources and inference cost

All experiments are conducted on a single workstation with an NVIDIA GeForce RTX 4090 GPU and an Intel Core i9-13900H CPU. Training and evaluation use the GPU; the message-sending controller can be deployed on either device.

The latency-critical component at deployment time is the per-step sending decision. Table 10 reports its per-step inference time over batched runs on both devices.

Two observations follow. First, both devices run well below the $\Delta t = 10$ ms decision interval, so the sending controller is not a runtime bottleneck. Second, the CPU implementation is roughly $7\times$ faster than the GPU one in our setting. This is because the decision pipeline only performs element-wise potential computations, running-statistics updates, and a token-bucket update—none

Table 10: Per-step inference time of the message-sending decision (ms).

Device	Avg	Median	Min	Max
RTX 4090 (GPU)	0.816	0.787	0.686	1.385
i9-13900H (CPU)	0.115	0.109	0.104	0.375

of which involves large matrix multiplications. On the GPU side the per-step cost is dominated by kernel-launch and host–device transfer overhead. These measurements indicate low CPU overhead for the sending controller.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state DDACOM's two contributions—a token-bucket-based message-sending module that tracks the probe-estimated safe rate and a decoupled message-generation module—and the quantitative claims (5.34% drop under $4.7\times$ bandwidth reduction, $> 95\%$ safe-rate utilization, $< 0.3\%$ packet loss) match the results in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the main limitation in the Conclusion section. Specifically, the message-sending module currently relies on dedicated probe packets to estimate the safe rate, which bandwidth cost is small but not negligible; we discuss replacing probes with TCP/IP congestion-control signals (RTT, ACK-based loss indication, ECN) as a natural next step.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results or theorems. The contributions are algorithmic (decoupled message-sending and message-generation modules) and empirical, evaluated on SMAC, MPE, and ns-3 simulations.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5.1 specifies the environments, baselines, seeds, evaluation protocol, and metrics. Appendix C provides the architecture, hyperparameters, sending-control parameters, and overall pseudocode (Algorithm 1). The DDACOM source code is publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The DDACOM source code is publicly available, with installation instructions in the repository. Experimental settings and implementation details are provided in Section 5.1 and Appendix C.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 reports the SMAC maps, MPE scenarios, baselines, seeds, and training/evaluation protocol. Appendix C lists the PPO and message-PPO hyperparameters, the auxiliary-critic optimizer, the attention/value mixing coefficients, and the sending-control parameters. The network-simulation configuration is summarized in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All results are reported over 5 random seeds with 1000 evaluation episodes per seed per setting. Training curves show mean with shaded variance bands; Tables 1 and 2 report mean $\pm$ std across seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All experiments use a workstation with an NVIDIA RTX 4090 GPU and an Intel Core i9-13900H CPU. Per-step latency of the message-sending decision and a discussion of GPU vs. CPU deployment are reported in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and confirm that the research conducted in this paper conforms to it. The work is a simulation-based study of cooperative multi-agent communication under dynamic network conditions and does not involve human subjects, private data, or intentionally harmful applications.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: DDACOM targets bandwidth-efficient multi-agent coordination, which can benefit UAV/robot swarms and traffic-control deployment under dynamic networks. Potential negative impacts include misuse in large-scale autonomous-coordination scenarios and interference with other wireless traffic if probe-based safe-rate estimation is mis-configured; we partially mitigate the latter by reporting window-over and total-over rates and by flagging the probing dependence in the Limitations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk models such as pre-trained language models or image generators. The work focuses on reinforcement-learning policies and a communication-scheduling controller for simulated multi-agent cooperation, so the safeguard requirements for high-risk data or model releases are not directly applicable.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring

that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original works for the SMAC environment, the MPE environment, the ns-3 network simulator, and the four communication-MARL baselines (CACOM, MAIC, SchedNet, TMC). All of these assets are publicly available through their original open-source repositories, and our use complies with their licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The DDACOM source code is publicly released under the MIT license, with installation instructions provided in the repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects. All experiments are conducted in simulated cooperative multi-agent environments and an ns-3 network simulator.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects, crowdsourcing, or user studies. Therefore, Institutional Review Board approval or equivalent human-subjects review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as an important, original, or non-standard component. DDACOM is a reinforcement-learning-based multi-agent communication framework consisting of element-wise multiplicative attention for message fusion, an auxiliary-critic-based counterfactual reward for message generation, and a dynamic-token-bucket controller for message sending. Any use of LLMs for writing, editing, or formatting would not affect the core methodology, scientific claims, or experimental results.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.