

# DEEP REINFORCEMENT LEARNING-BASED ROBUST ROUTING FOR SCALABLE UAV SWARM COMMUNICATION

*Sopov Ie., Artomova A.*

*Robust routing in mobile ad hoc and UAV swarming networks remains difficult due to rapid topology changes, interference, and intermittent links. We propose DeepCQ+, a scalable routing framework that augments the classical CQ/CQ+ family with a single multi-agent deep reinforcement learning (MADRL) policy trained under the centralized-training, decentralized-execution paradigm. The agent state uses a lightweight top-K neighbor encoding of CQ variables (confidence and hop estimates), while a PPO policy learns when to switch between unicast and adaptive flooding to balance delivery and overhead. To support and benchmark learning, we build an ns-3 (v3.39) slotted evaluation harness with RWP mobility and standard metrics (goodput; normalized MAC overhead). Baseline experiments (unicast only) over  $N \in \{20, 30, 40, 50\}$  nodes and 4/8 flows show that mean goodput rises with density – approximately 0.66 ( $N = 20$ ), 0.63 ( $N = 30$ ), 0.72 ( $N = 40$ ), 0.85 ( $N = 50$ ) – while normalized overhead drops from  $\sim 0.15$  ( $N = 20 - 30$ ) to  $\sim 0.08$  ( $N = 40$ ) and  $\sim 0.062$  ( $N = 50$ ). These results quantify the operating regime where adaptive flooding is most valuable (notably  $N = 30 - 40$ , where variance is highest), and provide a reproducible baseline for learning-based control. The paper details the DeepCQ+ architecture, reward shaping for overhead awareness, and the ns-3/Matlab pipeline for data aggregation and plotting. Collectively, the study establishes a principled path toward routing policies that retain CQ+ robustness while scaling to larger swarms with lower control cost.*

## 1. Introduction

Designing robust and efficient routing algorithms remains one of the most challenging problems in communication and computer networks. This challenge becomes even greater in tactical MANETs, where network conditions are highly dynamic and unpredictable due to node mobility, frequent topology changes, interference, and potential jamming [1]. Such factors substantially reduce reliability. Consequently, many traditional MANET routing protocols require frequent recomputation of end-to-end routes when detecting topology changes, which in turn causes periodic throughput losses while traffic is suspended during recomputation. These losses increase as the rate of topology change grows.

To mitigate these effects, broadcasting (or flooding), where a packet is transmitted to all neighbors, has become a popular approach for maintaining packet delivery performance in highly dynamic networks [2]. Classical MANET routing protocols such as OSPF and OLSR [3], perform reliably under stable conditions

but degrade significantly under rapid mobility. Adaptive extensions also fail to respond quickly enough in highly dynamic scenarios [4].

In this work, we focus on distributed routing algorithms that share only limited information – specifically, two floating-point values – via per-hop acknowledgment (ACK) packets. This approach is efficient since ACKs are already present in lower-layer protocols (e.g., MAC-layer acknowledgments) and require no additional implementation. The seminal work on Q-routing introduced reinforcement learning (RL), specifically Q-learning, to minimize delivery time by enabling each node to learn path qualities (Q-values) locally and update them through ACK exchanges [5]. Building on this, developed CQ-routing, which added confidence values (C-values) to improve adaptability in dynamic networks. To further enhance reliability, the Smart Robust Routing (SRR) algorithm [6] introduced selective broadcasting, enabling nodes to decide when to broadcast based on heuristic rules. This extension, referred to as CQ+ (or Robust Routing for Dynamic Networks, R2DN), effectively balances reliability and overhead. However, CQ+ relies on a single network parameter – the best-path confidence level – to switch between unicast and broadcast. This limited perspective often results in locally optimal decisions that fail to fully account for rapid topology changes or congestion levels. While CQ+ consistently delivers high packet delivery ratios across many scenarios, it incurs a significant broadcast overhead, making it an important but imperfect baseline for more advanced frameworks.

This motivates the question: can emerging deep reinforcement learning frameworks reduce overhead while maintaining, or even improving, CQ-routing performance (particularly goodput)? Routing decisions such as next-hop selection are natural candidates for RL-based techniques. Since Boyan’s original Q-routing protocol [7, 9], a range of RL-inspired methods have been proposed for packet routing and scheduling. However, these approaches typically scale poorly with increasing network size or when system parameters change. To address these limitations, recent studies have explored multi-agent deep reinforcement learning (MADRL) [8], though many of these designs remain constrained by scalability issues, as they require retraining whenever new nodes are added.

To the best of our knowledge, no prior study has simultaneously achieved both scalability and robustness using MADRL in dynamic networks such as MANETs. Our proposed DeepCQ+ overcomes these limitations by embedding MADRL into the CQ+ framework. Specifically, MADRL agents control next-hop flooding decisions while retaining the structural simplicity of CQ+. This design enables a single trained agent to produce scalable, robust routing policies applicable to

any node, regardless of network size. More importantly, the DeepCQ+ methodology allows training under limited scenarios – small networks, selective traffic patterns, or constrained mobility – while remaining effective when deployed in environments outside the training distribution. This property addresses the curse of dimensionality and mitigates catastrophic forgetting, making DeepCQ+ a significant advancement toward reliable routing in highly dynamic networks.

## 2. Resilient Routing Architecture

The SRR algorithm extends the classical CQ-routing by incorporating two key network parameters: the  $H$ -factor and the  $C$ -value [9, 10].

For each node  $i$ , the  $H$ -factor, denoted as  $h(i, j, d)$ , represents an estimate of the minimum number of hops from node  $i$  to destination  $d$  via a potential next-hop  $j$ . To capture network dynamics, each node also maintains a confidence level, or  $C$ -value  $c(i, j, d)$ , which reflects the likelihood that a packet will successfully reach destination  $d$  through  $j$ .

The  $C$ -value increases with successful transmissions (acknowledgements received) and decays with each transmission attempt. Both  $C$ - and  $H$ -values are updated using  $c_{ack}$  and  $h_{ack}$  parameters, which are propagated through ACK packets from the receiving node back to the transmitter. At the next-hop node  $j$ , the ACK-propagated update terms are defined as in (1) and (2):

$$h_{ack} = 1 + h(j, \hat{k}, d) \quad (1)$$

$$c_{ack} = c(j, \hat{k}, d) \quad (2)$$

where  $\hat{k} = \arg \min_k h(j, k, d)(1 - c(j, k, d))$  is the optimal next-hop selected by node  $j$  for reaching destination  $d$ .

Thus, the  $C$ - and  $H$ -values act as exponential moving averages of  $c_{ack}$  and  $h_{ack}$ . In case of a failed transmission (i.e., no ACK received), the  $H$ -value cannot be updated, while the  $C$ -value is degraded by setting  $c_{ack} = 0$ , reflecting path failure. The recursive updates are given in (3) – (4):

$$h_{t+1}(i, j, d) = (1 - \alpha)h_t(i, j, d) + \alpha h_{ack} \quad (3)$$

$$c_{t+1}(i, j, d) = \begin{cases} (1 - \lambda)c_t(i, j, d), & \text{failure} \\ (1 - \lambda)c_t(i, j, d) + \lambda c_{ack}, & \text{other wise} \end{cases} \quad (4)$$

where  $0 \leq \alpha \leq 1$  is the discount factor for new observations (adaptively set as  $\alpha = \max(c_{ack}, 1 - c_t(i, j, d))$ ) and  $\lambda$  is the decay parameter controlling the update rate. When a packet reaches destination  $d$ , both  $c_{ack}$  and  $h_{ack}$  are set to 1, reflecting full confidence at one hop away.

The CQ+ routing protocol consists of three main steps:

1. Reception of ACK packets and update of  $C$ - and  $H$ -values;
2. Reception of data packets, with duplicate detection, queuing, or delivery if the node is the destination;
3. Transmission of data packets via either unicast or broadcast according to the routing policy.

The routing policy selects the next-hop that minimizes the joint metric of path uncertainty and expected hops (5):

$$j^* = \arg \min_j h(i, j, d)(1 - c(i, j, d)) \quad (5)$$

If sufficient confidence exists in the chosen next-hop, the packet is unicasted. Otherwise, the protocol probabilistically triggers broadcast to reduce uncertainty, with probability defined as:

$$P_{BC} = \varepsilon + (1 - c(i, j^*, d))(1 - \varepsilon) \quad (6)$$

where  $\varepsilon$  is a small exploration parameter ensuring a minimum probability of broadcast.

In the enhanced framework, the broadcast/unicast decision of the CQ+ protocol is replaced by a multi-agent deep reinforcement learning (MADRL) model, enabling more robust and adaptive routing in highly dynamic networks.

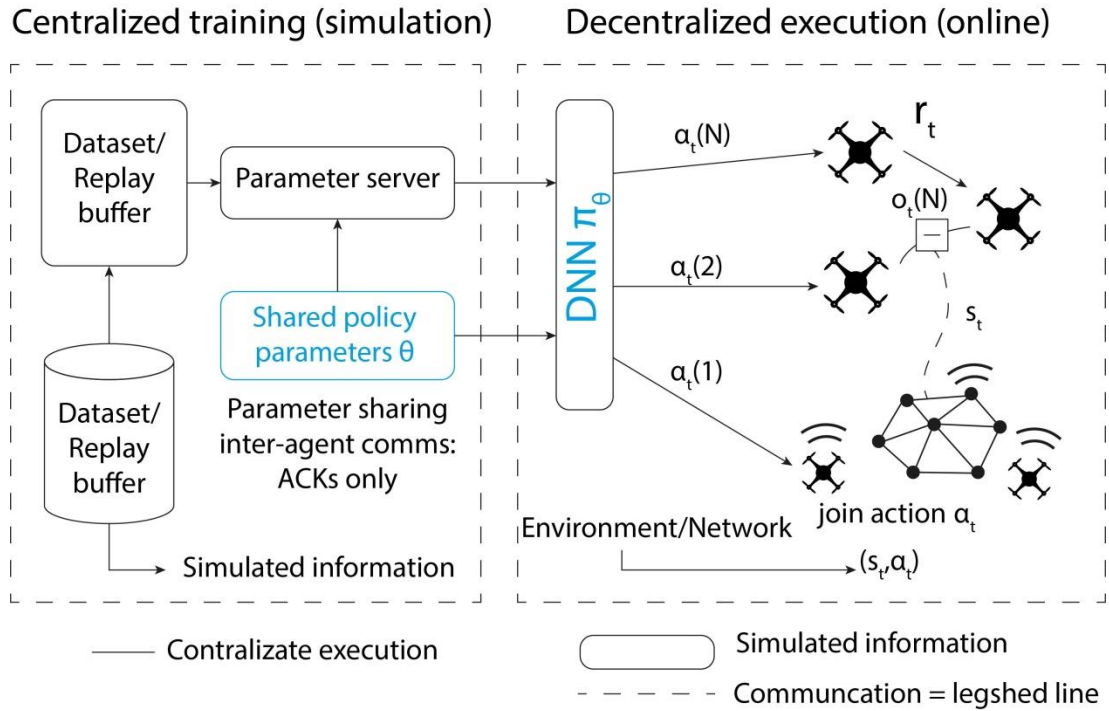
### 3. CQ Routing with Multi-Agent Deep Reinforcement Learning (DeepCQ+)

Our robust routing design follows the decentralized partially observable Markov decision process (Dec-POMDP), a standard framework for cooperative multi-agent decision-making under uncertainty [14]. In this setting, each agent observes only local information and optimizes its behavior while accounting for environmental and inter-agent uncertainty.

**Policy optimization.** We employ policy gradient (PG) methods that optimize the policy directly by ascending the expected discounted return [11]. Specifically, we adopt Proximal Policy Optimization (PPO) [16], which performs multiple epochs of stochastic gradient ascent per update and is known for its stability and ease of implementation.

Centralized training, decentralized execution. Due to partial observability and communication constraints, agents must execute decentralized policies based solely on local observations. Nevertheless, these policies can be trained centrally in simulation, a paradigm widely used in MARL [12].

Parameter sharing. For homogeneous agents, we apply parameter sharing so that all nodes use a single policy  $\pi_\theta$ , improving scalability and convergence [13]. Fig. 1 illustrates the multi-agent routing environment, centralized training with decentralized execution, and shared policy parameters.



**Fig. 1.** Multi-agent routing environment with shared policy parameters

Training is centralized and execution is decentralized. At time  $t$ , each agent  $i$  applies the common policy  $\pi_\theta$  to its local observation  $o_t(i)$  and selects an action  $a_t(i)$ . The environment evolves according to the joint action of all agents, producing the next state  $s_{t+1}$  and per-agent rewards.

#### 4. DeepCQ+ Design Framework and Approach

We consider a homogeneous MANET with variable network size  $N$  (e.g.,  $5 \leq N \leq 50$ ) and multiple unicast flows with randomized source-destination pairs. Nodes move with random velocities/directions; baseline mobility follows Gauss – Markov and random waypoint models, further diversified as discussed in

Section IV-A. Each RL episode spans at least  $T$  time slots (packet-duration slots). The episode terminates once traffic exceeds  $T_{\max}$  and queues drain; no new traffic arrives after  $T_{\max}$ . Goodput is the ratio of successfully delivered packets to injected packets (duplicates at the destination excluded). A single shared policy  $\pi_\theta$  (one DNN for all nodes) is trained on a narrow distribution (single network size/flow, modest dynamics) and tested across wider ranges (larger  $N$ , multiple flows, higher mobility).

We retain only the best  $K$  neighbors (e.g.,  $K = 4$ ) per node based on their  $(C, H)$  values. For notational simplicity, we drop the explicit destination index in  $c_t$  and  $h_t$ . The best- $K$  neighbor vectors are defined as in (7):

$$c_t(i) = [c_t(i, i_1), c_t(i, i_2), \dots, c_t(i, i_K)], \quad h_t(i) = [h_t(i, i_1), h_t(i, i_2), \dots, h_t(i, i_K)] \quad (7)$$

where neighbors  $i, \dots, i_K$  are ordered by the CQ+ metric as in (8),

$$h_t(i, i_1)(1 - c_t(i, i_1)) \leq \dots \leq h_t(i, i_K)(1 - c_t(i, i_K)) \quad (8)$$

so that the most promising next hops appear first.

State/observations. To capture temporal dynamics, we feed a fully connected neural network (FCNN) with first-order differences of observations. The policy input at node  $iii$  is defined in (9):

$$o_t(i) = [c_t(i), h_t(i), \Delta c_t(i), \Delta h_t(i), a_{t-1}(i), p_{t-1}(i)] \quad (9)$$

where  $\Delta c_t(i) = c_t(i) - c_{t-1}(i)$ ,  $\Delta h_t(i) = h_t(i) - h_{t-1}(i)$ ;  $a_{t-1}(i)$  is the previous action at node  $i$ ; and  $p_{t-1}(i)$  indicates whether the current packet arrived via broadcast or unicast.

Tabl. 1 summarizes capabilities across CQ-routing [7], SRR (CQ+) [8], and DeepCQ+ (this work): CQ values, confidence/broadcast, and MADRL control.

Table 1

### Comparative Analysis of CQ-based Protocols vs. DeepCQ+ in Dynamic MANETs

Routing Protocol	Control logic	Scalability	Overhead	Goodput	Notes
CQ-routing [7]	Heuristic (Q-values)	Medium (node-local)	Low (unicast only)	Medium; degrades at high mobility	Baseline RL-inspired
SRR (CQ+) [8]	Heuristic + selective flooding	Medium	High (uncertainty-driven flooding)	High; robust under dynamics	Uses C/H + broadcast policy
DeepCQ+ (this work)	Learned (MADRL)	High (CTDE + sharing)	Medium ( $\downarrow$ vs CQ+)	High ( $\geq$ CQ+)	Adaptive unicast/broadcast

CQ-routing is a heuristic baseline that uses Q-values only; SRR (CQ+) augments CQ with selective flooding, improving robustness (Goodput) at the cost of higher Overhead. DeepCQ+ replaces the broadcast/unicast switch with a learned MADRL policy (CTDE + parameter sharing), preserving high Goodput while reducing Overhead versus SRR and scaling better with network size.

## 5. The DeepCQ+ Reward Function

We first align with CQ+ by defining a stochastic routing policy  $\pi(a|s_t)$  and rewards for broadcast/unicast decisions as in (10):

$$r_t(s_t, a_t) = \begin{cases} 1 - c_t(i, i_1) \tilde{\varepsilon}, & a_t = 1 \text{ (broadcast)}, \\ c_t(i, i_1) \tilde{\varepsilon}, & a_t = 0 \text{ (unicast)}, \end{cases} \quad (10)$$

where  $\tilde{\varepsilon} = 1 - \varepsilon$ . This mirrors CQ+ behavior for small exploration  $\varepsilon$ , providing a consistent baseline.

To explicitly discourage overhead, we define overhead as the ratio of total transmissions  $N_{TX}$  to delivered packets  $N_D$ , normalized by network size  $N$ :  $OH = \frac{1}{N} \cdot \frac{N_{TX}}{N_D}$ . We also distinguish excess transmitted bits per delivered bits and (ii) total (data+ACK) bits per delivered data bit. Incorporating delivery, failure, and ACK-induced replication, the overhead-aware reward is given in (11):

$$r_t = w_1 1_D - w_2 1_Z - w_3 \frac{N_{ack}}{N} \quad (11)$$

where  $1_D$  rewards successful delivery when node  $i$  contributes;  $1_Z$  penalizes transmissions without any received ACKs; and  $N_{ack}$  approximates packet replication/overhead. Weights  $w_1, w_2, w_3$  are tuned to minimize overhead while preserving delivery. This tuned reward defines the DeepCQ+ routing policy, whose algorithms are summarized in Algorithm 1. The proposed DeepCQ+ routing (CTDE with parameter sharing):

Inputs: size range  $N$ ; horizon  $T$ ;  $T_{max}$ ; neighbor cap  $K$  (e.g.,  $K = 4$ ); exploration  $\varepsilon$ ; reward weights  $w_1, w_2, w_3$ ; PPO hyperparameters; initial  $C/H$  tables. Outputs: shared policy  $\pi_\theta$ ; decentralized routing decisions.

*Phase I – Centralized training (simulation)*

1. Initialize policy  $\pi_\theta$  and value estimator; initialize  $C/HC/HC/H$  tables.
2. For each episode: randomize  $N$ , flows, mobility (Sec. IV-A); reset queues and  $C/H$ .

3. For  $t = 1, \dots, T$  until traffic  $> T_{\max}$  and queues are empty:
  - 3.1 For each node  $i$ : build top- $K$  neighbor vectors  $c_t(i), h_t(i)$  as in (7); order neighbors by CQ+ metric as in (8).
  - 3.2 Form observation  $o_t(i)$  with first-order differences and indicators as in (9).
  - 3.3 Sample action  $a_t(i) \sim \pi_\theta(o_t(i))$
  - 3.4 If unicast: select  $j^*$  by (5) and transmit; else broadcast per policy (CQ+ context (6)).
  - 3.5 Process ACKs; compute  $c_{ack}, h_{ack}$  by (1)–(2), update  $C, H$  via (3)–(4).
  - 3.6 Compute reward using (10) or (11); store  $o_t, a_t, r_t, o_{t+1}$ .
4. Run PPO updates (policy gradient) on collected trajectories; repeat episodes until convergence.

*Phase II – Decentralized execution (deployment)*

5. Freeze  $\pi_\theta$ ; deploy the same  $\pi_\theta$  at all nodes (parameter sharing).
6. For each arriving packet at node  $i$ : build top- $K$  per (7) – (8); form  $o_t(i)$  (9); sample  $a_t(i)$ .
7. Forward: unicast via  $j^*$  (5) or broadcast; update  $C, H$  with (1) – (4); suppress duplicates; continue until delivered.
8. Track Goodput and Overhead online (no gradient updates during execution).

We propose DeepCQ+, a multi-agent deep reinforcement learning (MADRL) extension of CQ routing with centralized training, decentralized execution, and shared parameters across nodes. The agent state uses top- $K$  neighbors and lightweight temporal features; the learned policy adaptively switches between unicast and broadcast, while ACK-based updates ensure robustness to rapid topology changes. PPO training provides scalability and stability. Overall, DeepCQ+ achieves higher goodput with lower overhead than SRR (CQ+).

## 6. Experimental Setup and Evaluation

All experiments were carried out in ns-3 v3.39 using the ad-hoc Wi-Fi stack (Yans PHY, Adhoc MAC) and an event-driven slotted simulation runner. The deployment region is a square of side  $L = 300$  m. Terminals (UAV/mobile nodes) follow the Random Waypoint mobility model with pause time drawn from the default ns-3 setting and speed  $v \in [1, 2] \text{ms}^{-1}$ , re-sampled at waypoints. Each episode

comprises  $T = 3000$  time slots of duration 20 ms; traffic injection ceases after  $T_{\max} = 2000$  slots and the simulation continues until all queues drain.

We evaluate network sizes  $N \in \{20, 30, 40, 50\}$  under 4 and 8 simultaneous unicast flows (randomized source–destination pairs per episode). Unless otherwise stated, figures report means across the two traffic intensities with 95% confidence intervals computed from the sample variance across intensities (normal approximation; see Statistical note below).

Two performance indicators are used:

- Goodput: ratio of successfully delivered data packets to injected packets (duplicates at the destination excluded);
- Normalized overhead (OH2): total MAC transmissions (data + ACK + control) divided by the number of delivered data packets and normalized by  $N$  (lower is better).

Aggregated numerical outcomes are reported in Table 2; trends are visualized in Figs. 1–3.

Table 2

**Baseline MANET performance under RWP mobility**

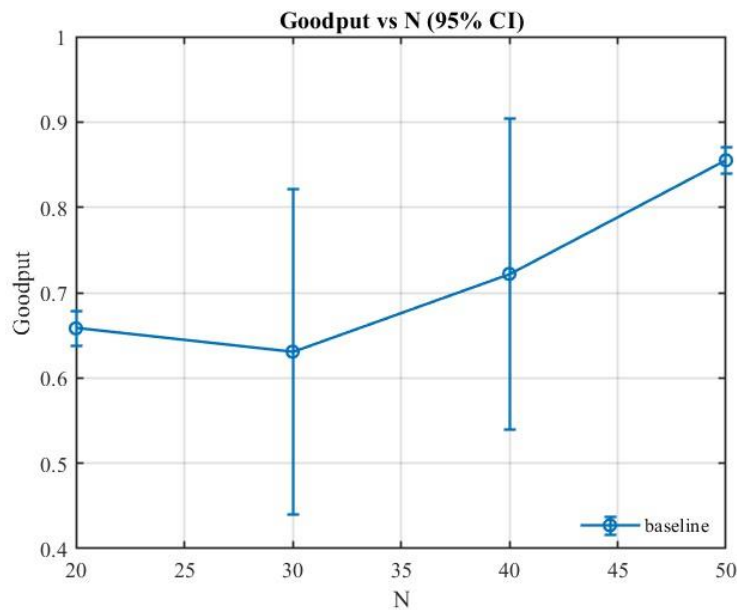
N	tag	N	flows	Injected	Delivered	Goodput	OH1	OH2
1	baseline	20	4	8000	5182	0.64775	0.142667	0.142667
2	baseline	20	8	16000	10697	0.668652	0.144475	0.144475
3	baseline	30	4	8000	5825	0.728125	0.125345	0.125345
4	baseline	30	8	16000	8536	0.533500	0.175832	0.175832
5	baseline	40	4	8000	6517	0.814625	0.0671513	0.0671513
6	baseline	40	8	16000	10067	0.629188	0.0911543	0.0911543
7	baseline	50	4	8000	6904	0.863000	0.0610370	0.0610370
8	baseline	50	8	16000	13555	0.847187	0.0621173	0.0621173

For sparse to moderate densities ( $N = 20 - 30$ ), goodput is 0.53–0.73 with normalized overhead  $\text{OH2} \approx 0.12 - 0.18$ . For denser networks ( $N = 40 - 50$ ), goodput rises to 0.63–0.86 while OH2 drops to 0.06 – 0.09. The spread between 4 and 8 flows explains the error bars shown in the figures.

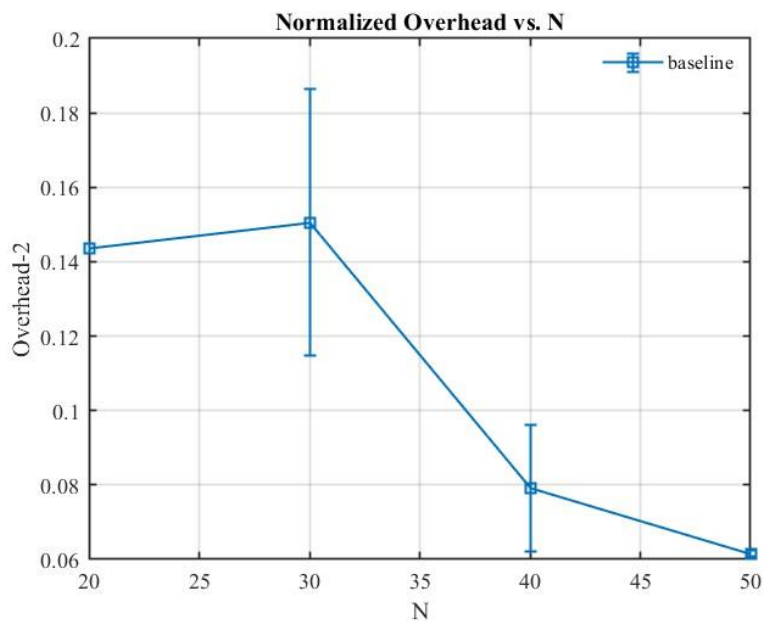
As shown in Fig. 2 (Goodput vs.  $N$ , 95% CI), performance remains essentially flat at low densities and increases markedly with  $N$ : mean goodput  $\approx 0.66$  for  $N = 20$ ,  $\approx 0.63$  for  $N = 30$ ,  $\approx 0.72$  for  $N = 40$ , and  $\approx 0.85$  for  $N = 50$ . The broader confidence bands at  $N = 30 - 40$  arise from sensitivity to load (4 vs. 8 flows): under higher offered load, transient queueing and route changes depress delivery;

adding nodes at  $N = 50$  provides alternative paths and restores end-to-end throughput. These trends are consistent with the raw outcomes in Tab. 2.

Fig. 3 shows a monotonic decrease in OH2 beyond  $N = 30$ : from  $\approx 0.15$  at  $N = 20 - 30$  down to  $\approx 0.08$  at  $N = 40$  and  $\approx 0.062$  at  $N = 50$ . This reduction is attributable to shorter average hop counts and fewer retransmissions as topology becomes richer, which offsets increased contention. The larger variance at  $N = 30$  mirrors the goodput spread and indicates a regime where mobility and flow density interact unfavorably for the baseline policy.

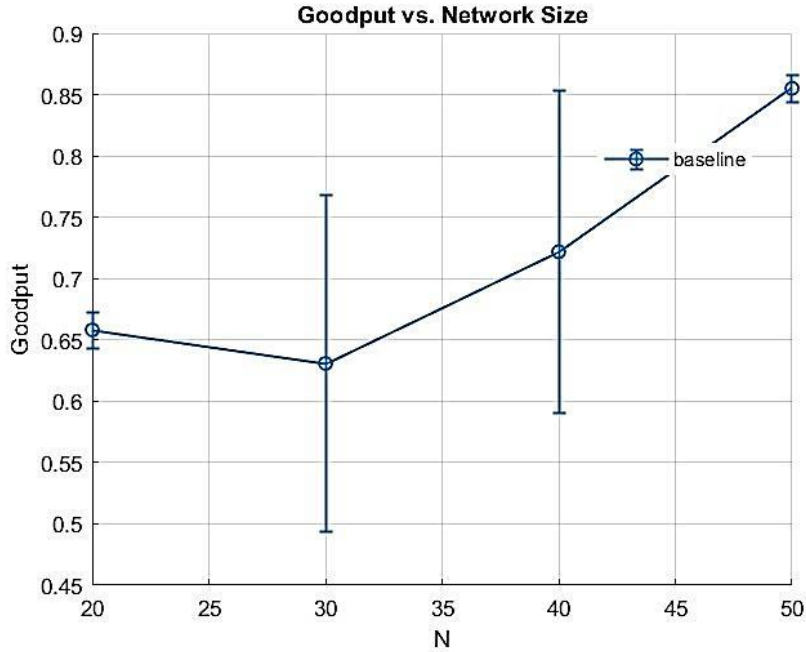


**Fig. 2.** Impact of Node Count on Goodput for Unicast MANET Routing (95% CI)



**Fig. 3.** Normalized Overhead (OH2) vs. Network Size under RWP Mobility

For completeness, Fig. 4 reproduces goodput vs.  $N$  with alternate y-axis scaling for direct comparison against CQ+ and DeepCQ+ in the next section.



**Fig. 4.** Baseline Goodput across  $N$  (y-axis harmonized for comparison)

Even without selective flooding or learned control, increasing node density in this mobility regime simultaneously improves goodput and reduces normalized overhead. These baselines (Table 2, Figs. 1–3) constitute the quantitative reference against which gains of CQ+ and DeepCQ+ will be assessed (e.g., maintaining high goodput already at  $N = 20 - 30$  while avoiding the overhead levels observed in Fig. 2).

Per  $N$ , we aggregate two operating points (4 and 8 flows). The mean is reported across these points; error bars in Figs. 1–3 correspond to 95% confidence intervals computed as  $1.95\sigma/\sqrt{n}$  with  $n = 2$  (normal approximation). With  $n = 2$ , CIs are indicative rather than definitive; in subsequent sections we increase the number of runs per setting when comparing CQ+/DeepCQ+.

## 7. Conclusions

This work introduced DeepCQ+, a scalable and robust routing framework that augments the CQ+/SRR structure with a single shared multi-agent policy learned under the CTDE paradigm. By retaining CQ+'s lightweight per-hop state while delegating the unicast/broadcast switch to a learned controller, DeepCQ+ targets the dual goals that prior MADRL routing attempts have struggled to achieve

simultaneously: scalability across network sizes and robustness under mobility and load variation.

In preparation for the learning-based controller, we built and validated an ns-3 evaluation harness with slotted execution, RWP mobility, and standard metrics. Baseline (unicast-only) experiments show two consistent trends across  $N \in \{20, 30, 40, 50\}$  goodput rises with node density ( $\approx 0.66, 0.63, 0.72, 0.85$ , respectively), and normalized overhead (OH2) falls monotonically beyond  $N = 30$  ( $\approx 0.15 \rightarrow 0.08 \rightarrow 0.062$ ), reflecting shorter routes and fewer retransmissions in denser topologies. These effects persist across 4 and 8 flows, with larger variance at  $N = 30 - 40$  due to load sensitivity. Taken together, the baselines set a quantitative reference against which CQ+ and DeepCQ+ will be assessed in the sequel.

The monotonic OH2 reduction indicates that broadcast need not be frequent in moderately dense swarms; conversely, the variance band at  $N = 30 - 40$  reveals the regime where adaptive flooding is most valuable. DeepCQ+ is expressly designed to exploit these regimes, learning when to broadcast to preserve goodput at low/medium  $N$  while avoiding the overhead spikes seen in the heuristic CQ+ switch.

Current results use RWP mobility, a single physical/MAC stack, fixed packet duration, and only two traffic intensities per  $N$ . Confidence intervals are based on  $n = 2$  operating points and are therefore indicative rather than definitive; subsequent comparisons (CQ+ vs. DeepCQ+) will increase repetitions per setting and diversify mobility (e.g., Gauss–Markov), traffic, and channel models.

We will integrate the learned broadcast/unicast policy into ns-3 and compare against CQ+ under identical scenarios; perform ablations on the top- $K$  neighbor encoding and reward weights; evaluate generalization to untrained  $N$ , flow counts, and mobility/jamming patterns; and report energy and fairness metrics in addition to goodput/overhead. These steps align with the monograph structure and presentation rules (tables/figures and conclusions), ensuring compliance with authoring requirements.

## References

1. Machakaire t. Enhancing manets for military applications: a comprehensive review of innovations, challenges, and research gaps // i-Manager's Journal on Wireless Communication Networks – T. 13. – №. 2. – 2025. DOI: <https://doi.org/10.26634/jwcn.13.2.22063>
2. Ismatov, A., Suh, B. K., Kim, J., Park, Y. B., & Kim, K. I. Adaptive Broadcast Scheme with Fuzzy Logic and Reinforcement Learning Dynamic Membership Functions in Mobile Ad Hoc Networks. *Mathematics*, 13(15), 2367. 2025. DOI: <https://doi.org/10.3390/math13152367>
3. Ostojic, Zivka Slepcevic. Comparative Studies of Link-State Routing Protocols in Wired IP and Ad-Hoc Wireless Network: OSPF, IS-IS and OLSR. Diss. University of Applied Sciences Technikum Wien, 2022.

4. Harib, Mouhcine, Hicham Chaoui, and Suruz Miah. «Evolution of adaptive learning for nonlinear dynamic systems: A systematic survey.» *Intelligence & Robotics* 2.1 (2022): 37-71.
5. Kunzel, Gustavo. «Application of reinforcement learning with Q-learning for the routing in industrial wireless sensors networks». (2021).
6. Darabkh, K. A., Al-Khazaleh, H. F., Al-Zubi, R. T., Alnabelsi, S. H., & Salameh, H. B. Efficient routing protocol for optimal route selection in cognitive radio networks over IoT environment. *Wireless Personal Communications*, 129(1), 209–253. (2023). DOI: <https://doi.org/10.1007/s11277-022-10093-6>
7. Martins, M. S., Sousa, J. M., & Vieira, S. A systematic review on reinforcement learning for industrial combinatorial optimization problems. *Applied Sciences*, 15(3), 1211. (2025). DOI: <https://doi.org/10.3390/app15031211/>
8. Gronauer, S., Diepold, K. Multi-agent deep reinforcement learning: a survey. *Artif Intell Rev* 55, 895–943 (2022). DOI: <https://doi.org/10.1007/s10462-021-09996-w/>
9. R. E. Ali, B. Erman, E. Bas, tug, and B. Cilli, «Hierarchical deep double Q-routing», in ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, 2020, pp. 1–7.
10. Kaviani, S., Ryu, B., Ahmed, E., Larson, K., Le, A., Yahja, A., & Kim, J. H. DeepCQ+: Robust and scalable routing with multi-agent deep reinforcement learning for highly dynamic networks. In MILCOM 2021-2021 IEEE Military Communications Conference (MILCOM) (pp. 31–36). IEEE. (2021, November).
11. Prashanth, L. A., & Fu, M. C.. Risk-sensitive reinforcement learning via policy gradient search. *Foundations and trends® in machine learning*, 15(5), 537–693. (2022)
12. Zhang, K., Yang, Z., & Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, 321–384. (2021).
13. Hu, F., Deng, Y., & Aghvami, A. H. Scalable multi-agent reinforcement learning for dynamic coordinated multipoint clustering. *IEEE Transactions on Communications*, 71(1), 101–114. (2022).