

# DeepCQ+: Robust and Scalable Routing with Multi-Agent Deep Reinforcement Learning for Highly Dynamic Networks

Saeed Kaviani\*, Bo Ryu\*, Ejaz Ahmed\*, Kevin Larson<sup>†</sup>, Anh Le\*, Alex Yahja\*, and Jae H. Kim<sup>†</sup>

\*EpiSys Science, Inc. : {saeed, bo.ryu, ejaz, anhle, alex}@episci.com

<sup>†</sup>Boeing Research and Technology : {kevin.a.larson, jae.h.kim}@boeing.com

*Abstract*—Highly dynamic mobile ad-hoc networks (MANETs) remain as one of the most challenging environments to develop and deploy robust, efficient, and scalable routing protocols. In this paper, we present DeepCQ+ routing protocol which, in a novel manner, integrates emerging multi-agent deep reinforcement learning (MADRL) techniques into existing Q-learning-based routing protocols and their variants, and achieves persistently higher performance across a wide range of topology and mobility configurations. While keeping the overall protocol structure of the Q-learning-based routing protocols, DeepCQ+ replaces statically configured parameterized thresholds and hand-written rules with carefully designed MADRL agents such that no configuration of such parameters is required a priori. Extensive simulation shows that DeepCQ+ yields significantly increased end-to-end throughput with lower overhead and no apparent degradation of end-to-end delays (hop counts) compared to its Q-learning-based counterparts. Qualitatively, and perhaps more significantly, DeepCQ+ maintains remarkably similar performance gains under many scenarios that it was *not* trained for in terms of network sizes, mobility conditions, and traffic dynamics. To the best of our knowledge, this is the first successful application of the MADRL framework for the MANET routing problem that demonstrates a high degree of scalability and robustness even under the environments that are *outside* the trained range of scenarios. This implies that our MARL-based DeepCQ+ design solution significantly improves the performance of Q-learning-based CQ+ baseline approach for comparison and increases its practicality and explainability because the real-world MANET environment will likely vary outside the trained range of MANET scenarios. Additional techniques to further increase the gains in performance and scalability are discussed.

## I. INTRODUCTION

Design of a robust and efficient routing algorithm has been one of the most challenging problems in communication and computer networks. This challenge is compounded in tactical MANETs where the network is highly dynamic and unpredictable with respect to mobility and topology, interference, and possible jamming [1]. These factors significantly reduce the reliability, and therefore, many traditional MANET routing protocols require frequent re-computations of end-to-end routes upon detecting network topology changes, resulting in a periodic loss in throughput due to traffic not being sent during the re-computations. This is exacerbated as the rate of topology change increases. To improve packet delivery performance via rapid and efficient exploration in these highly dynamic networks, broadcasting (i.e. transmission of a packet to all neighbors, also known as flooding) has become a popular

technique [2]–[4]. Traditional MANET routing protocols (e.g. OSPF and OLSR [3], [4]) perform reliably when the network is in a stable state but less effectively in highly dynamic networks. Other traditional adaptive strategies also are not responsive fast enough for dynamic networks [2].

In this paper, we consider a class of distributed routing algorithms that only share limited information (i.e. two single floating value) through per-hop acknowledgment (ACK) packets. This method of cooperation is efficient as ACKs are inherently present in the networking protocols (such as MAC-layer acknowledgement) and do not require any extra implementation in the system. The seminal work [5] proposed Q-routing, which used a reinforcement learning (RL) module (i.e. Q-Learning [6]) to route packets and minimize delivery time. Each node uses  $Q$ -values representing quality of paths which are acquired locally to determine the next hop and shared via ACK messages. Kumar et al. [7] improved Q-routing for dynamic networks with the addition of confidence values (i.e.  $C$ -values) in their  $CQ$  routing protocol. To improve reliability and exploration speed of the  $CQ$ -routing, smart robust routing (SRR) algorithm [8] was proposed to add selective broadcasting actions. SRR utilizes heuristic rules on when to broadcast in order to improve robustness but keep the overall overhead under control. We refer to this technique as  $CQ+$  routing (also known as Robust Routing for Dynamic Networks, or R2DN) as it is an extension of  $CQ$  routing for balancing between reliability and overhead. Although  $CQ+$  routing uses a simple but efficient switching policy to choose between unicast and broadcast, its decisions depend on a single network parameter: best-path confidence level. Consequently, it has a limited snapshot of the entire network that is likely to lead to a locally optimal solution, since it does not fully account for the high rate changes in topology and degree of congestion in likely forwarding paths. Nevertheless, its performance has been consistently producing high delivery ratios across many scenarios used in our study, though at a noticeably high cost of broadcast overhead, and as a result, it serves as a baseline design for our MADRL framework. The question we posed, then, is whether an emerging deep learning framework can help reduce the overhead while maintaining, or exceeding, the performance of  $CQ$ -routing (especially goodput).

Routing decisions such as a next-hop selection are opportune targets for employing RL-based techniques, as it

was originally introduced and initiated by Boyan’s Q-routing protocol [5]. Following the Q-routing approach, a flurry of new techniques and algorithms from the RL community have been developed and applied to packet routing and scheduling [9]–[13]. These techniques are often scale poorly when network sizes increase and system parameters change. To address these shortcomings, new MADRL-based approaches such as [13] and [9] have been proposed, but they suffer from limited scalability due to node-specific policy generated from the training, requiring a re-training every time a new node needs to be introduced to the network.

To the best of our knowledge, no study has been reported on successfully achieving both scalability and robust performance simultaneously using MADRL in dynamic networks such as MANET. Our **DeepCQ+** is both scalable and robust by allowing MADRL to control the next-hop adaptive flooding decisions while maintaining the CQ+ routing protocol “structure”. In this paper, we describe how a single MADRL-based routing agent is designed and trained to produce a robust and scalable routing policy for any node in the network regardless of the network size. More significantly, our **DeepCQ+** design and learning methodology enables network designers to train on a limited range of environment parameters (e.g. small network size, selective traffic flow patterns, and limited variations), and still to be effective even when deployed in scenarios outside the trained range of network size, traffic profiles, and mobility patterns. This is an extremely important and unique benefit of **DeepCQ+** since it does not suffer from the curse of dimensionality due to potentially large network sizes, and catastrophic forgetting even when trained with a wide range of scenarios.

## II. ROBUST ROUTING FRAMEWORK

### A. SRR (CQ+ routing) Protocol

The SRR algorithm [8] uses the network parameters  $C$  and  $H^1$  for the routing decisions primarily introduced in seminal CQ-routing [7]. Each node  $i$  has a  $H$ -factor,  $h(i, j, d)$  (i.e.  $i \rightsquigarrow j \rightsquigarrow d$ ), which represents an estimate of the least number of hops between node  $i$  and destination  $d$  which passes through potential next-hop  $j$ . To monitor the dynamics of the network, each node  $i$  also have a confidence level or  $C$ -value,  $c(i, j, d)$ , that represents the confidence in likelihood the packet will reach its destination  $h(i, j, d)$ . This  $C$ -value is increased with each packet transmission success (receiving the ACK). Every packet transmission, the  $C$ -value is degraded by a decay factor.  $C$ - and  $H$ -factors are updated through the  $c_{\text{ack}}$  and  $h_{\text{ack}}$  which are propagated by the ACK packets from the receiving node (e.g. next-hop) to the transmitting node. These ACK values are computed at the next-hop node  $j$  as

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

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

<sup>1</sup>We have renamed the original  $Q$ -factor in CQ-routing to  $H$ -factor to prevent confusion with the  $Q$  in the  $Q$ -networks and/or  $Q$ -learning in the RL context.

where  $\hat{k}$  is the best next-hop estimate of node  $j$  to destination  $d$  and it is found by  $\hat{k} = \arg \min_k h(j, d, k) (1 - c(j, k, d))$ . In other words,  $C$ - and  $H$ -values are exponential moving average of the  $c_{\text{ack}}$  and  $h_{\text{ack}}$ , respectively. If a transmission fails or there is no ACK to update  $C$ - and  $H$ -levels, then  $H$ -level cannot be updated. However, we degrade the  $C$ -level as in  $c_{\text{ack}} = 0$  to reflect the path failure. The updates of the  $C$ - and  $H$ -levels given by

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

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

where  $0 \leq \alpha \leq 1$  is a discount factor for the new observation with an adaptable value of  $\alpha = \max(c_{\text{ack}}, 1 - c_t(i, j, d))$ .  $\lambda$  is the decay factor for the new observation of  $c_{\text{ack}}$  (or  $1 - \lambda$  is the decay factor for the old observation). If a packet is received at the destination  $d$ , then  $c_{\text{ack}}$ , and  $h_{\text{ack}}$  are set to 1, indicating that we have full confidence that we are 1 hop away from destination. CQ+ routing algorithm consists of the following three steps: (i) reception of ACK and consequently updating  $C$ - and  $H$ -levels; (ii) reception of data packets, detection and dropping of duplicate packets, and queuing if not the destination; and (iii) transmission of data packets either via unicast or broadcast according to the routing policy chosen. Additional detail of the SRR algorithm (or CQ+ routing) is provided in [8].

The CQ+ routing policy chooses the next-hop based on the minimization of the information uncertainty and the expected number of hops. This is given by

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

The next-hop  $j^*$  is considered by the CQ+ routing policy if it unicast the packet to a single neighboring node. However, the CQ+ routing policy triggers broadcast in order to mitigate the next-hop information uncertainty. The CQ+ routing policy is probabilistic and it assigns the probability of broadcast as

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

where a small value  $\epsilon$  is used for the minimum probability of broadcast (defined for exploration purposes). In what follows, we briefly describe the MADRL framework employed for replacing the broadcast/unicast decision of the CQ+ routing protocol with a deep neural network.

## III. MULTI-AGENT DEEP REINFORCEMENT LEARNING FOR CQ ROUTING (DEEPCQ+)

### A. Background

Our robust routing design follows the *decentralized partially-observable Markov decision process (Dec-POMDP)*, a popular framework for decision-making problems in a team of cooperative agents [14]. Dec-POMDP tries to model and optimize the behavior of agents while considering the uncertainties of both the environment and other agents.

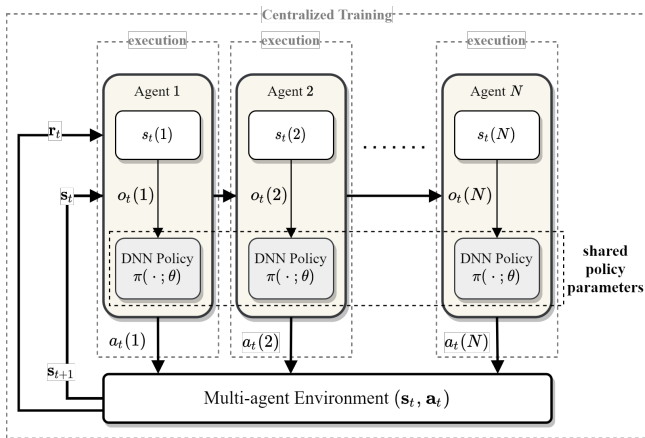


Fig. 1. Multi-agent network routing environment with shared policy parameters between agents. The centralized training and decentralized execution are also shown. Each agent  $i$ , uses the shared policy  $\pi_\theta$  individually to find its own action  $a_i(t)$  based on its own observations  $o_i(t)$ . The multi-agent environment operates based on the joint-actions decided and taken individually and transition to next state  $s_{t+1}$  and the rewards are pulled out based on that.

1) *Policy Optimizations*: Policy gradient (PG) method is another popular choice that is based on the optimization of the policy directly rather than estimating the expected return and it optimizes the policy parameters by descending toward the gradient direction of the expected discounted reward return [15]. Particularly, we deploy a state-of-the-art policy gradient algorithm called Proximal Policy Optimization (PPO) [16], which uses multiple epochs of stochastic gradient ascent to perform each policy updates with ease of implementation.

2) *Centralized Training, Decentralized Execution*: In communication networks, partial observability and communication constraints necessitate the learning of decentralized policies which rely only on local information at each agent. Decentralized policies also avoid the exponentially growing joint action space with the number of agents, and therefore, make them more practical and faster to converge in training. Fortunately, decentralized policies can be learned in a centralized fashion specially in a simulated or controlled environment. The paradigm of *centralized training with decentralised execution* has already attracted a strong attention in the RL community [14].

3) *Parameter Sharing*: A common strategy is to share the policy parameters among agents that are homogeneous [17] (parameter sharing). The multi-agent environment for the routing problem and our proposed framework are summarized in Fig. 1 where the centralized training, decentralized execution, and parameter sharing across agents are illustrated.

## B. DeepCQ+ Design Framework and Approach

We consider a homogeneous MANET with variable network size (e.g.  $5 \leq N \leq 50$ ) with multiple unicast traffic flows with randomized source and destination pairs. The nodes are consistently moving at various random velocities and directions. We employ popular MANET mobility models such as Gauss-Markov model and random way-point. As these

Routing Protocol	C/Q-values	Broadcast	MADRL
CQ-routing [7]	✓	✗	✗
SRR (CQ+) [8]	✓	✓	✗
DeepCQ+ routing (this work)	✓	✓	✓

TABLE I

COMPARISON OF ROBUST ROUTING PROTOCOLS IN DYNAMIC NETWORKS

models are not realistic in general, we enhance these models to provide much more diverse sets of mobility as discussed in Section IV-A. Each realization of a network scenario (also called an *episode* in the context of RL) has at least  $T$  time-slots (each time step is single packet duration time). For simplicity, it is assumed that packet duration is fixed in time (i.e., a slotted system) but data rates can still vary. An episode ends once it exceeds a maximum traffic length,  $T_{max}$  and all nodes complete their transmissions until queues become empty. No new traffic enters the network after time  $T_{max}$ . The goodput is computed as the ratio of the total number of successfully delivered packets to the total packets that entered the network. Duplicate packets at the destination are not included. We train and use the same parameterized policy  $\pi_\theta$  for every node (one policy for all), which is found to be key to achieve scalability and practicality. When the policy function is defined as a Deep Neural Network (DNN), the parameter vector  $\theta$  represents the weights of the neural network. We deliberately choose to train our DNN policy with a single network size, single data flow (single source and destination pair), and small range of dynamic levels, and test for varying network sizes with multiple data flows and a wide range of mobility.

1) *Pre-processing of the policy input features*: We select only the best  $K$  (e.g.  $K = 4$ ) neighbors out of total  $N - 1$  possible neighbors in the network for each node according to their available  $C$  and  $H$  values. For the sake of simplicity, we drop  $i$  and destination  $d$  from  $C$ - and  $H$ -levels and use the next-hop as index (i.e.  $c_t(i, j) := c_t(i, j, d)$ ). Now, in our problem formulation the best  $K$  neighbor  $C$  and  $H$  values are represented by

$$\begin{aligned} \mathbf{c}_t(i) &= [c_t(i, i_1), c_t(i, i_2), \dots, c_t(i, i_K)] \\ \mathbf{h}_t(i) &= [h_t(i, i_1), h_t(i, i_2), \dots, h_t(i, i_K)] \end{aligned} \quad (7)$$

where the neighboring nodes  $i_1, \dots, i_K$  of  $i$  are ordered so that

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

2) *State/Observations*: We use a fully connected neural network (FCNN) to capture temporal behavior of network parameters, and add the change rate between current observations and previous observations as input to the DNN policy. The input features to the DNN policy at node  $i$  are given by

$$\mathbf{o}_t(i) = [\mathbf{c}_t(i), \mathbf{h}_t(i), \Delta \mathbf{c}_t(i), \Delta \mathbf{h}_t(i), a_{t-1}(i), p_{t-1}(i)], \quad (9)$$

where  $\Delta \mathbf{c}_t(i) = \mathbf{c}_t(i) - \mathbf{c}_{t-1}(i)$ ,  $\Delta \mathbf{h}_t(i) = \mathbf{h}_t(i) - \mathbf{h}_{t-1}(i)$ , and  $a_{t-1}(j)$  is the previous action of some node  $j$  from which the current packet is received. The last indicator shows if the current packet is received as a result of another node's broadcast or unicast.

### C. Reward Definition for DeepCQ+ routing

We consider a stochastic routing policy  $\pi(a|s_t)$  that sets the rewards for unicast and broadcast as the probability of those actions as follows:

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

where  $\tilde{\epsilon} = 1 - \epsilon$ . We expect that the performance of this RL problem be close to the CQ+ routing (at least for small values of  $\gamma$ ) as both DeepCQ+ and CQ+ have similar rewards. We refer to the above reward as a DNN-based version of the CQ+ routing. Note that different reward designs will produce different performance objectives, thus providing a higher degree of design flexibility compared to the baseline CQ+ routing.

Note that the above reward specification does not necessarily make the CQ+ routing policy learn to minimize the network overhead. To address this, We first define the overhead as the ratio of the total number of transmissions in the communication network, denoted by  $N_{\text{TX}}$ , to the total number of packets delivered, denoted by  $N_D$  for a specific packet data rate and a window of time. With this, we define the *normalized overhead* by the network size,  $N$ , as  $\text{OH} = \frac{1}{N} \cdot \frac{N_{\text{TX}}}{N_D}$ . Based on this, we introduce two types of overhead: (i) overhead defined as excess transmitted bits per delivered bits (overhead-1); and (ii) and total transmitted bits (both data and ACK) divided by the total delivered data bits (overhead-2). In order to ensure that the learned policy attempts to minimize the overhead, we define a new reward function as follows:

$$r_t = w_1 \mathbb{1}_{\mathcal{D}} - w_2 \mathbb{1}_{\mathcal{Z}} - w_3 \frac{N_{\text{ack}}}{N} \quad (11)$$

where  $\mathbb{1}_{\mathcal{D}}$  is the reward for packet delivery. If a packet is delivered successfully to the destination and node  $i$  has contributed to that delivery then it will be rewarded. Also,  $\mathbb{1}_{\mathcal{Z}}$  is a reward (penalty) indicator which is enabled when we have not received any ACKs from our transmissions. The next term in the reward, i.e.  $N_{\text{ack}}/N$ , is the normalized number of ACKs received as a result of the action taken and therefore closely represents the number of copies of a packet at other nodes and the system overhead. The weights of the reward components, (i.e.  $w_1, w_2, w_3$ ), have been tuned according to the overhead minimization problem. The proposed DeepCQ+ routing technique with tuned reward function is referred to as DeepCQ+ routing policy. The DeepCQ+ routing algorithms are summarized in Algorithm 1.

## IV. EXPERIMENTS AND NUMERICAL RESULTS

### A. Environment Modelling and Training Platforms

We have developed an OpenAI gym [18] environment in Python for training our DeepCQ+ agents. The python-based DeepCQ+ environment has all of the original CQ+ routing protocol features, including a plethora of MANET scenarios consisting of randomly generated dynamic network scenarios, multiple traffic flows, and mobility configurations.

---

### Algorithm 1: The proposed DeepCQ+ routing

---

```

Receive incoming packet at node  $i$ :
if Packet is ACK then
    | Update  $c$  and  $h$  using (3) and (4)
else
    | if packet traversed a loop then
    | | Drop packet, do not return ACK
    | end
    | if packet is already in queue then
    | | Find best next-hop  $i_1$  from (5)
    | | Compute  $c_{\text{ack}}$  and  $h_{\text{ack}}$  from (2) and (1) using  $j^*$ 
    | | Drop packet but return ACK
    | end
    | if packet is not duplicate then
    | | Add packet to the queue
    | else
    | | Do not add packet to the queue
    | end
    end
if ACK Packet is not received then
    | Do not update  $c$  and  $h$ , (i.e. no packet is received)
    end
if Queue is not empty then
    | Form the input to the DNN policy using (9) and (8)
    | DeepCQ+ routing: Choose
    | { Broadcast with probability  $\pi_{\theta}(a = 1|\mathbf{o}_t; \theta)$ 
    |   Unicast with probability  $\pi_{\theta}(a = 0|\mathbf{o}_t; \theta)$ 
    | end

```

---

The environment is interfaced with the *ray*, which is a powerful distributed DRL platform with RLlib library [19] which provides scalable DRL training primitives with many built-in DRL algorithms such as PPO.

Within the MANET research community, there have been various mobility models proposed along with extensive discussions on their impact on the routing protocol performance [20], [21]. At a minimum, a MANET mobility model needs to have enough degree of randomization to cover corner cases and extreme scenarios required for extensive training in MADRL. The random way-point mobility model is often used in the simulation study of MANET, despite some unrealistic movement behaviors, such as exhibiting sudden stops and sharp turns [21]. The Gauss-Markov mobility model, which has more realistic movement behavior and avoids often sudden stops and sharp returns, has less sensitivity of the network performance metrics with respect to the various randomness [21]. To study the performance of MADRL under realistic mobility scenarios, our network scenario region is divided into 5 groups symmetrically between the source and destination nodes. The closer the region is to the center, the faster the nodes move. The central region has double the speed variance compared to variance of the mid-left and mid-right regions. The source and destination regions have half the speed variance compared to the mid-right/left regions. The mobility of the MANET nodes is simulated and shown in Fig. 2.

### B. Numerical Results

Training and testing range of network parameters are summarized in Table II with the hyperparameters used. Our

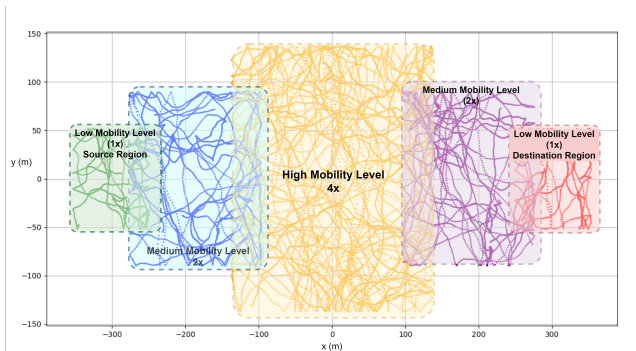


Fig. 2. Mobility regions and network topology. The movements of the nodes are shown for a network of size 30 according to the Gauss-Markov model

TABLE II  
LIST OF CONFIGURATION PARAMETERS AND TRAINING  
HYPERPARAMETERS

Parameters	Value in Training	Value in Test
Network Sizes	12	5 to 50
Learning Rate	0.00005	-
Discount Factor $\gamma$	0.99	-
Episode Length	3000	3000
Region Size Scale	1	2
Dynamic Level Scale	1	5
Policy NN	FCNN(16, 8, 8, 4)	FCNN(16, 8, 8, 4)
Number of Data Flows	1	1 to 4

training is performed over 50 million steps (approximately 15,000 episodes) and tests (where the results are shown in Fig. 3) are performed over network sizes between 10 and 30 with larger dynamic level and region scale randomization compared to the training. Training is performed on single network size (e.g.  $N = 12$ ) scenarios. In our simulation experiments,  $N = 12$  was the smallest size network in which the training results was extended to  $N = 30$ . Fig. 3 also shows the scalability and robustness of our training framework as the performance of the designed routing policies are maintained even with larger network parameters and configurations and networks of as large as 30 (which are not trained for). As shown in the figure, DeepCQ+ routing policy improves the goodput rate (delivery ratio) almost the same as the CQ+ routing but with significantly less overhead (both type 1 and type 2) for DeepCQ+. Indeed, our MADRL framework can adapt to different deployment tuples of goodput rate, broadcast rate, and even delay, which was not available in the CQ+ routing. Our results show that we require 10-25% less overhead while achieves higher (1-5%) delivery ratio (goodput rate), which shows significant improvement over the baseline CQ+ routing algorithm. DeepCQ+ also achieves lower percentage of broadcast transmissions, leading to network resource saving. This overhead saving is attributed only to the policy’s efficient decision on broadcast actions, while the next-hop is still selected based on the same CQ+ routing algorithm as in [5].

## V. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we have presented and successfully demonstrated a novel MADRL-based CQ+ routing protocol which yields a robust, reliable, efficient, and scalable policy for dynamic wireless communication networks, including for scenarios that the algorithm was *not* trained for. It is shown that DeepCQ+ is much more efficient than CQ+ routing techniques, significantly decreasing the required overhead for unit delivery. Moreover, the policy is scalable and uses parameter sharing for all nodes obtained during the training, which allows it to reuse the same trained policy for scenarios with different network configurations. It is noted that the sharing of parameters for all the nodes is not required during execution.

We are currently expanding the action space of DeepCQ+ to include next-hop selection for the unicast mode and other emerging MADRL techniques. Note that the focus of this work was on broadcast or unicast action selection while the next-hop was chosen based on the CQ+ routing. We plan to use recurrent neural network units as policy which is expected to capture higher gains compared to our current approach.

## VI. ACKNOWLEDGEMENT

Research reported in this publication was supported in part by Office of the Naval Research under the contract N00014-19-C-1037. The content is solely the responsibility of the authors and does not necessarily represent the official views of the Office of Naval Research. The authors would like to thank Dr. Santanu Das (ONR Program Manager) for his support and encouragement. Also, we would like to thank the reviewers for their valuable comments to improve the quality of this paper.

## REFERENCES

- [1] G. F. Elmasry, “A comparative review of commercial vs. tactical wireless networks,” *IEEE Communications Magazine*, vol. 48, no. 10, pp. 54–59, 2010.
- [2] C. Danilov, T. R. Henderson, T. Goff, O. Brewer, J. H. Kim, J. Macker, and B. Adamson, “Adaptive routing for tactical communications,” in *MILCOM 2012 - 2012 IEEE Military Communications Conference, 2012*, pp. 1–7.
- [3] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, “Optimized link state routing protocol (OLSR),” 2003.
- [4] J. Moy *et al.*, “OSPF version 2,” 1998.
- [5] J. A. Boyan and M. L. Littman, “Packet routing in dynamically changing networks: A reinforcement learning approach,” *Advances in Neural Information Processing Systems*, 1994.
- [6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [7] S. Kumar and R. Miiikulainen, “Confidence-based Q-routing: An on-line adaptive network routing algorithm,” in *Proceedings of Artificial Neural Networks in Engineering*, 1998.
- [8] M. Johnston, C. Danilov, and K. Larson, “A reinforcement learning approach to adaptive redundancy for routing in tactical networks,” in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, 2018, pp. 267–272.
- [9] R. E. Ali, B. Erman, E. Baştuğ, and B. Cilli, “Hierarchical deep double Q-routing,” in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.
- [10] C. Yu, J. Lan, Z. Guo, and Y. Hu, “DROM: Optimizing the routing in software-defined networks with deep reinforcement learning,” *IEEE Access*, vol. 6, pp. 64 533–64 539, 2018.

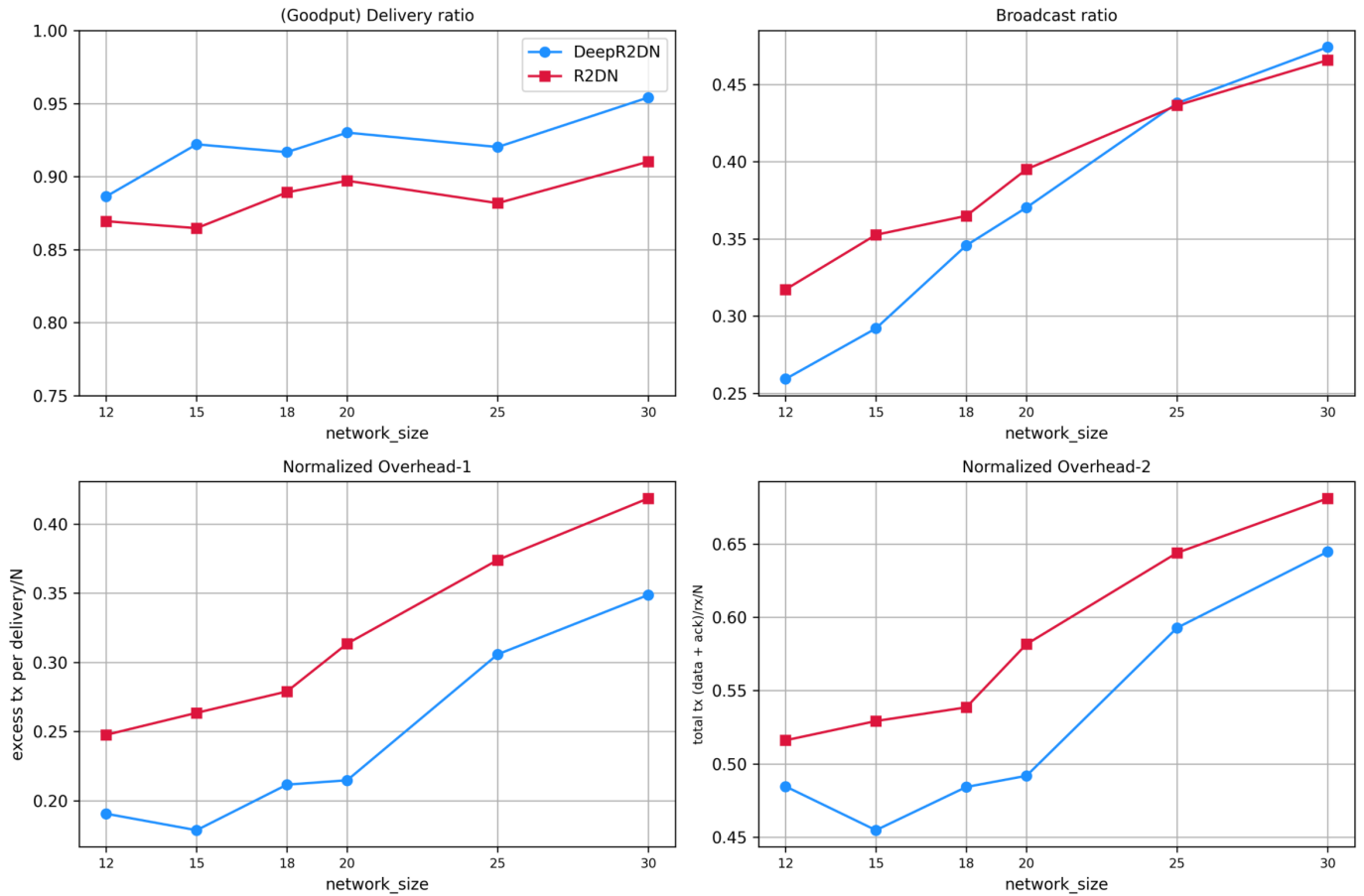


Fig. 3. Comparison of the results of the DeepCQ+ routing trained for a 12-node network only versus CQ+ routing; The results are tested across various network sizes from 12 to 30. Although the DeepCQ+ routing PPO policy is trained on 12-node networks, it scales perfectly for various network sizes. The DeepCQ+ routing achieves significantly lower normalized (divided by network size) overhead types 1 and 2 and higher delivery ratio and lower broadcast rates.

- [11] G. Stampa, M. Arias, D. Sánchez-Charles, V. Muntés-Mulero, and A. Cabellos, “A deep-reinforcement learning approach for software-defined networking routing optimization,” *arXiv preprint arXiv:1709.07080*, 2017.
- [12] H. Ye, G. Y. Li, and B.-H. F. Juang, “Deep reinforcement learning based resource allocation for V2V communications,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.
- [13] X. You, X. Li, Y. Xu, H. Feng, J. Zhao, and H. Yan, “Toward packet routing with fully distributed multiagent deep reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.
- [14] F. A. Oliehoek, C. Amato *et al.*, *A concise introduction to decentralized POMDPs*. Springer, 2016, vol. 1.
- [15] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Advances in neural information processing systems*, 2000, pp. 1057–1063.
- [16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [17] J. K. Terry, N. Grammel, A. Hari, L. Santos, B. Black, and D. Manocha, “Parameter sharing is surprisingly useful for multi-agent deep reinforcement learning,” *arXiv preprint arXiv:2005.13625*, 2020.
- [18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “OpenAI Gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [19] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, “Rllib: Abstractions for distributed reinforcement learning,” in *International Conference on Machine Learning*, 2018, pp. 3053–3062.
- [20] A. K. Gupta, H. Sadawarti, and A. K. Verma, “Performance analysis of MANET routing protocols in different mobility models,” *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 5, no. 6, pp. 73–82, 2013.
- [21] J. Ariyakhajorn, P. Wannawilai, and C. Sathitwiriayong, “A comparative study of random waypoint and gauss-markov mobility models in the performance evaluation of MANET,” in *2006 International Symposium on Communications and Information Technologies*, 2006, pp. 894–899.