Learning to Fly: A Distributed Deep Reinforcement Learning Framework for Software-Defined UAV Network Control

Hai Cheng, Lorenzo Bertizzolo, Salvatore D'Oro, *Member, IEEE*,
John Buczek, Tommaso Melodia, *Fellow, IEEE*, Elizabeth Serena Bentley, *Member, IEEE*

Control and performance optimization of wireless networks of Unmanned Aerial Vehicles (UAVs) require scalable approaches that go beyond architectures based on centralized network controllers. At the same time, the performance of model-based optimization approaches is often limited by the accuracy of the approximations and relaxations necessary to solve the UAV network control problem through convex optimization or similar techniques, and by the accuracy of the channel network models used. To address these challenges, this article introduces a new architectural framework to control and optimize UAV networks based on Deep Reinforcement Learning (DRL). Furthermore, it proposes a virtualized, 'ready-to-fly' emulation environment to generate the extensive wireless data traces necessary to train DRL algorithms, which are notoriously hard to generate and collect on battery-powered UAV networks. The training environment integrates previously developed wireless protocol stacks for UAVs into the CORE/EMANE emulation tool. Our 'ready-to-fly' virtual environment guarantees scalable collection of high-fidelity wireless traces that can be used to train DRL agents. The proposed DRL architecture enables distributed data-driven optimization (with up to 3.7x throughput improvement and 0.2x latency reduction in reported experiments), facilitates network reconfiguration, and provides a scalable solution for large UAV networks.

***Index Terms*—UAV Networks; Non-terrestrial Netoworks, Deep Reinforcement Learning; AI for Wireless Networks; 6G**

I. Introduction

Unmanned Aerial Vehicle (UAV) networks are attracting the interest of the wireless community as a 'tool' to provide flexible and on-demand network infrastructure [1, 2]. There are numerous applications for networked UAVs, including providing airborne emergency infrastructure in disaster scenarios [3], and off-the-grid, on-demand network provisioning in civilian and military scenarios [2, 4–8].

While fielding UAV networks can certainly enable a broad range of new applications, operating a UAV network and controlling (optimizing) its performance (e.g., throughput, latency, power consumption) presents several fundamental challenges when compared to fixed infrastructures.

Challenge (I): Fully wireless access and backhaul. UAV networks are fully wireless (i.e., access and backhaul) and their operations are extremely sensitive to spatially and temporally varying topologies and dynamic RF environments. Basic functionalities such as network formation and point-to-point communications are often impaired by unstable channel conditions and fragile network connectivity typical of infrastructure-less networked systems. This problem is further exacerbated by interference conditions, spectrum availability, and routing operations, which subject multi-hop communications to high and unpredictable delays.

Challenge (II): Centralized control is not practical. Traditional centralized network control approaches, which are typical of fixed-backhaul Radio Access Networks (RANs) applications, are in most cases unfeasible (or unpractical) in fully wireless networked systems. Centralized control approaches typically rely upon a centralized representation of the network (often made possible by collecting network state information over low-latency optical fiber links) to solve a centralized optimization problem and then distribute the solutions to the individual network nodes over the same low-latency wires. While centralized approaches have been applied with some success to fixed wired infrastructure and fixed-backhaul RAN systems [9, 10], they face two fundamental challenges in infrastructure-less wireless networks (i.e., UAV networks, tactical ad hoc networks, mesh, sensor networks, machine-to-machine, Internet-of-things (IoT)), as discussed below.

Challenge (II).i: NP-Hard control problems. Optimizing a distributed wireless network where both access and backhaul operate on the same frequency bands is non-trivial. The centralized formulation of the control problem requires very accurate modeling and, in most cases, is NP-hard because of the non-linear coupling of the many variables involved. The use of heuristics and approximation algorithms necessary to solve the problem often results in sub-optimal solutions. Furthermore, this optimality gap might grow further in highly dynamic and unpredictable infrastructure-less UAV networks, which may make model-based solutions too inaccurate to be of practical relevance.

Challenge (II).ii: Stale information, ineffective control. Information retrieval in multi-hop infrastructures suffers from inherent latency because of the need to relay information over wireless links. The resulting latency may impact the Age of Information (AoI) of the gathered data, which eventually results in stale network state information at the central controller. Consequently, the centralized network state representation might significantly differ from the actual network state. This problem can negatively affect the control process as the solutions computed are optimal with respect to the stale collected information, and thus potentially inefficient at actuation time. The latency also determines how quickly

This work is supported in part by the Air Force Research Laboratory (AFRL) under Contract FA8750-18-C-0122. Distribution A. Approved for public release: Distribution unlimited AFRL-2021-1223 on 20 Apr 2021.

H. Cheng, L. Bertizzolo, S. D'Oro, J. Buczek, and T. Melodia are with the Institute for Wireless Internet of Things, Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail:cheng.hai, bertizzolo.l, s.doro, buczek.j, melodia@northeastern.edu).

E. Bentley is with the Air Force Research Laboratory, Rome, NY 13441 USA (e-mail: elizabeth.bentley.3@us.af.mil).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

solutions computed at the central controller are dispatched to individual wireless nodes. By the time the computed solutions are delivered to the wireless nodes, the network state may have changed, thus making their implementation ineffective.

To address **Challenge (I)** and **Challenge (II)**, in this paper, we propose the two following innovations: (1) We envision Software-defined UAVs equipped with programmable radio front-ends and flight control units (FCU). We leverage the real-time reconfigurability of motion, PHY, and upper-layer operations to implement full-stack cross-layer optimization. Reconfigurability of functionalities at all layers of the protocol stack has already been demonstrated to provide superior performance with respect to systems based on inflexible hardware unable to adapt to dynamic spectrum conditions [1]. (2) To control the operations of the network of UAVs, we propose a two-tier architecture that addresses the challenges of distributed wireless system optimization mentioned above. We propose to address the UAV network control problem via data-driven optimization. We envision a multi-agent Deep Reinforcement Learning (DRL) approach where multiple distributed agents adapt their network parameters cooperatively to optimize a chosen network utility function. With enough data and training time, this approach guarantees optimal decision–making of the agent's networking and motion operation with respect to the defined objective function, without requiring explicit network modeling or a mathematical control problem formulation.

Challenge (III): UAV network data trace generation. How to guarantee sufficient training data to the DRL agents in UAV networks is the third challenge that we aim to solve in this work. UAV networks are battery-powered networks with time-constrained operations. Training a multi-node UAV network through real-world experiments would require excessive manpower and frequent battery charging, ultimately reducing the benefits of data-driven optimization approaches. While simulation environments are cost-effective tools to produce large datasets, they often fail in capturing typical real-time network dynamics or real-world systems (e.g., retransmissions, fragmentation, and buffer delays) that are hard or computationally expensive to model. On the other hand, full-stack emulation tools (e.g., NS-3 [11]) provide a more accurate representation of the real-time operations of a UAV network. Unfortunately, these tools are so far mainly focused on implementing a subset of common wireless standards (e.g, Wi-Fi, LTE, 5G NR), often abstracting a number of low-level details, which make them still unsuitable to accurately represent fully-reconfigurable wireless protocol stacks for UAVs. Given these limitations, how to produce training data that is representative of networked systems with UAVs employing fully-reconfigurable wireless protocols stacks is still an open problem. To address **Challenge (III)**, in this work we present (3) a DRL-based architecture that integrates the Drone networking protocol stack we introduced in [1] with the Common Open Research Emulator (CORE) and the Extendable Mobile Ad-hoc Network Emulator (EMANE) framework. This integration provides real-time emulation capabilities for fully-configurable wireless (and motion) protocol stacks for UAVs. We use this integrated virtual environment to generate extensive data traces with a high degree of realism and to scale up the training process.

The main contributions of this work can be summarized as follows:

- *A Two-tier Architecture for UAV Network Control*: We propose a two-tier architecture consisting of a *Control Framework* and a *DRL Drone Programmable Protocol Stack (DRL DPPS)*. The Network Operator (NO) uses the *Control Framework* to dictate the desired behavior of a distributed UAV network. Our solution automatically generates a set of DRL agents (i.e., a set of policies in the form of Neural Networks (NNs)) that are trained in a virtual environment within the *Control Framework*. Once trained, the NN configurations are tested and automatically distributed to the individual network nodes, where they will be used to control networking and motion parameters in the *DRL Drone Programmable Protocol Stack (DRL DPPS)*. In this way, the individual UAVs distributively implement the NO's objective by optimizing their network performance in real time. By distributing the NN configuration once, and by enforcing the desired network control policy at the edge nodes of the network, this approach does not suffer from stale information retrieval and delayed command typical of centralized control systems. Moreover, the proposed NN-based policies envision full-stack and cross-layer optimization of flight and wireless networking parameters alike, thanks to the use of programmable motion and RF front-ends.
- *A Data-driven Control Approach*: We propose to solve the UAV network control problem via DRL. We envision a multi-agent DRL scenario where each UAV is a different agent, and collectively train complex UAV fielding in a virtual environment for a specific flight mission. Upon training completion, we test and distribute mission-tailored NN configurations to individual UAVs. These use them to compute networking and motion policies to achieve the NO's desired network behavior by adapting to the dynamic network conditions. Compared to model-based optimization, our data-driven approach addresses inaccurate modeling formulation and optimization approximations. Unlike optimization approaches, the DRL agents do not suffer from optimization solver latency and can derive policies with $\mathcal{O}(1)$ complexity.
- *A 'Ready-to-Fly' Virtual Environment*: To collect extensive performance data for battery-powered UAV networks, we develop a highly representative emulation virtual environment. We revisit the Drone Programmable Protocol Stack (DPPS) employed in [1] and integrate it with Deep Reinforcement Learning features and refer to it as DRL DPPS. We integrate the DRL DPPS with the CORE/EMANE emulation tools to obtain a high-fidelity virtual environment that captures the motion, wireless channel, and higher-layer protocol stack interactions alike. We systematically employ our 'ready-to-fly' virtual environment to collect extensive high-fidelity network performance data. Ultimately, this integration effort produces a highly representative emulation environment

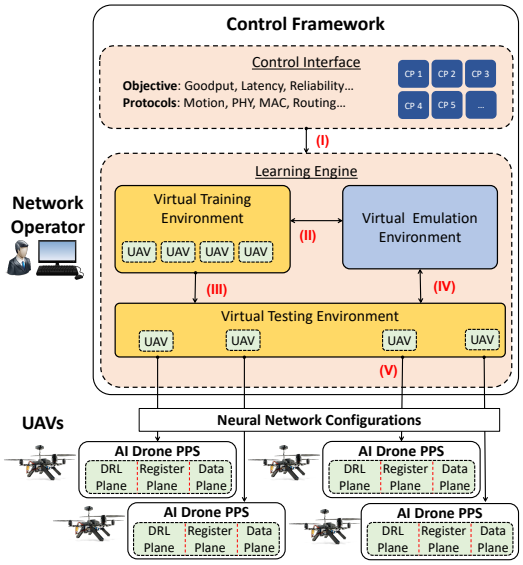

Fig. 1: Two-tier DRL-based architecture.

that allows us to scale up our learning time and to train our DRL agents with a high degree of realism.

Through a series of well-crafted experiments, we prove the *effectiveness* of our control approach in optimizing the desired network performance objectives (up to 3.7x throughput gains and 0.2x latency reduction), network *reconfigurability* to different control problems through re-distribution of NN configurations, and *scalability* to large UAV networks.

The rest of this article is structured as follows: Section II and Section III introduce the Control Framework and the DRL Drone Programmable Protocol Stack, respectively. These are the two building blocks of the proposed architecture. In Section IV, we present the integration of the DRL DPPS with the CORE/EMANE emulation tools that is at the base of our data-driven optimization, while we use Section V to present our UAV network optimization performance assessment. Finally, we review the related work in Section VI and draw the main conclusions in Section VII.

II. The Control Framework

Figure 1 provides a top-level view of the two-tier architecture we propose in this work. This includes the Control Framework and the Deep Reinforcement Learning Drone Programmable Protocol Stack (DRL DPPS). The first interfaces with the Network Operator, whose goal is to define a flight mission and dictate the desired control objective of the UAV network. The Control Framework integrates Deep Reinforcement Learning algorithms in its Learning Engine to train a multi-agent UAV network in an emulated environment (without the burden of planning real flight operations). Once the training terminates, the Control Framework tests and dispatches the objective-specific Neural Network (NN) configurations to the DRL DPPS. The latter implements a complete wireless protocol stack for UAVs inclusive of radio front-end and motion layer. The DRL DPPS's tasks are to support motion and networking functionalities that are vital to the UAV network operations and to optimize precise control parameters (determined by the NO in the first place) to achieve the NO's dictated control objective. This last task is performed by executing the pre-trained NN on board, whose configurations are received from the Control Framework. We see these two architectural components in detail in the following sections.

A. The Control Interface

The Control Framework interfaces with the Network Operator (NO) through the Control Interface. The latter is used to specify the desired flight mission and a UAV network control objective such as "maximize the end-to-end throughput", "maximize network capacity", "minimize the power consumption" (see Step I in Fig. 1). The Interface is also useful to specify what control parameters the NO wants to optimize, and which ones should be kept fixed. These include functionalities at any layer of the protocol stack (if programmable) as well as motion operations alike. Examples of possible control parameters include: "nodes' locations in the 3D space", "transmission power", and "forwarding decisions". When UAVs are equipped with programmable radio front-ends (e.g., Software-defined Radios) Physical Layer parameters such as "transmission power" and "carrier frequency" can be selected as a control parameter. Additionally, the NO can specify node- or layer-specific constraints, such as fixing a UAV's runtime location or limiting the maximum transmission power across all nodes. Ultimately, through the Control Interface, the NO can design a UAV network fielding for a specific mission and dictate a specific network behavior. An example of a possible directive is the following:

> *"The UAV network consists of 6 nodes. Two UAVs are location-constrained and hover close to two sensitive targets (sensing tasks). Two other UAVs are location-constrained and hover close to two base camps (reporting tasks), while the remaining two nodes can hover freely and can operate as relays. All the nodes can reconfigure their transmission power as well as their location in case they are not constrained. The network control objective of the mission is to maximize the aggregate end-to-end throughput of the traffic going from sensing target 1 to base camp 1 and from sensing target 2 to base camp 2."* (A)

These directives can be specified through a few lines of code via the Control Interface as illustrated in Listing 1. The NO specifies the number of UAVs involved in the fielding (`N_uavs`, Line 2), as well as sensing and reporting areas (Lines 8-13). For example, sensing areas could correspond to sensitive targets on the ground to be monitored or recorded. Reporting areas could correspond to locations where to offload the data, such as a base camp. The Control Interface handles these inputs and creates a UAV network configuration where one UAV is assigned to each of these areas. In this case, UAV 1 is instructed to hover on the sensing area 1, while UAV 2, UAV 3, and UAV 4 are assigned to sensing area 2, report area 1, and report area 2, respectively. UAV 5

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

and UAV 6 are free to hover. The NO also specifies which UAV parameters can be controlled (Line 16) and the network control objective, specified as to maximize the aggregate throughput at the report areas (Line 29). This input altogether is handled by the Control Interface by creating UAV objects and assigning control variables to them (Lines 19-26). The constructed network configuration is used in the Learning Engine to instantiate a virtual UAV network and optimize the UAVs' policy making so as to match the NO's desired control objective.

```
1  # Set number of UAVs
2  N_uavs = 6
3
4  # Creating network
5  nwk = Network.create(N_uavs)
6
7  # Set up monitoring areas
8  nwk.add_area("sensing", x_1, y_1)
9  nwk.add_area("sensing", x_2, y_2)
10
11 # Set up reporting areas
12 nwk.add_area("report", x_3, y_3)
13 nwk.add_area("report", x_4, y_4)
14
15 # Set up control variables
16 controls = (LOC, TX_POW)
17
18 # Create UAVs and assign control variables
19 nwk.get_UAV(1).set(location = [x_1, y_1], actions
       = [TX_POW], session = 1, role = source)
20 nwk.get_UAV(2).set(location = [x_2, y_2], actions
       = [TX_POW], session = 2, role = source)
21 nwk.get_UAV(3).set(location = [x_3, y_3], actions
       = [TX_POW], session = 1, role = destination)
22 nwk.get_UAV(4).set(location = [x_4, y_4], actions
       = [TX_POW], session = 2, role = destination)
23 nwk.get_UAV(5).set(actions = [LOC, TX_POW],
24      role = relay)
25 nwk.get_UAV(6).set(actions = [LOC, TX_POW],
26      role = relay)
27
28 # Set up control objective
29 obj = max(sum(nwk.get_Th(role = destination))
```

Listing 1: Control Interface example for quote (A).

B. The Learning Engine

Previous work has focused on tackling this class of problems by first mathematically formulating the underlying network control problem, and then solving it through constraint relaxation, decomposition theory, and convex optimization [1, 12–14]. Indeed, these approaches guarantee scalability, ease of reconfigurability, and operate in a distributed fashion. However, their performance is bound to the accuracy of the employed (motion, channel, switching, etc.) models and to the quality of the performed mathematical relaxations necessary to deem the problem solvable through convex optimization. A key drawback is that the model-based approaches fail to capture network operations such as retransmissions, fragmentation, buffer delays, MAC accesses, failed sensor readings, and all other network architecture dynamics that are hard (or computationally expensive) to model. While the differences and inaccuracies between the model and the actual implementation can be safely neglected for small network instances, the negative effect of such inaccuracies on network performance (or the model complexity, and thus the solver convergence time) might increase when considering more complex and large network instances. For these reasons, in this work, we select a data-driven approach and aim to solve the UAV network control problem through Deep Reinforcement Learning (DRL).

Deep Reinforcement Learning (DRL)-based approaches have progressively gained the attention of the wireless community to address a variety of critical spectrum access challenges, such as handover and power management in cellular networks [15, 16], dynamic spectrum access [17–20], resource allocation/slicing/caching [21–25], modulation/coding scheme selection [26], among others [27]. The driver of this success story lies in the ability of DRL to optimize the performance of a system by solving partially-observable Markov Decision processes (POMDP)-based problems without explicitly providing any details on the model, which is instead learned by observing and exploring the environment. Complex wireless network control problems are no exception. Therefore, DRL provides an effective tool to design control policies that optimize wireless networked systems: (i) that are hard-to-model and hard-to-solve, whose network control problem' representation might excessively differ from the final fielding and whose formulations need to go through several approximations before being deemed solvable; (ii) whose control decisions involve a set of known network actions (e.g., UAV locations, TX power, modulation, coding, medium access, routing, and transport parameters) according to the current wireless environment and optimization objective; (iii) where DRL training can leverage extensive and representative datasets with detailed information on network performance and conditions.

However, collecting extensive and representative performance data is hard, especially in battery-powered UAV networks. Limited flight time, the need to recharge and replace batteries frequently, and flight location regulations pose significant challenges to the collection of experimental data for the DRL training of UAV networks. To overcome this limitation, we developed a Learning Engine architecture that integrates the Drone Programmable Protocol Stack (DPPS) developed in [1] and employed in real UAV networks fielding with the Common Open Research Emulator (CORE) and the Extendable Mobile Ad-hoc Network Emulator (EMANE), a framework that provides detailed radio models and mobile networks scenarios [28]. This integration allows for extensive full-stack data collection in a representative emulated environment that captures the dynamics of real network implementations. This integrated environment is employed in the Learning Engine to emulate the UAV network configured through the Control Interface and to measure its performance for a wide range of configuration parameters. Most importantly, this integrated environment allows us to perform high-fidelity UAV network performance data collection at scale. This architectural system is at the base of the proposed data-driven UAV network optimization.

In a nutshell, the training phase works as follows. The Learning Engine instantiates a virtual UAV network based on the configuration expressed by the NO through the Control Interface in the Virtual Training Environment. As shown in Listing 1, this includes the exact number of UAVs, their

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

flight mission (sensing and report areas), a list of control parameters we can leverage to optimize the performance of the fielding, and the overall network control objective (e.g., 'maximize end-to-end throughput'). In the Virtual Training Environment's UAV network, each UAV is an independent agent employing a DRL Drone Programmable Protocol Stack (DRL DPPS) to carry on motion and wireless stack operations. During the training phase, the DRL DPPS is interfaced with the CORE/EMANE emulation environment which is where motion and physical layer functionalities are executed. As we will see in detail in Section III, the individual agents' DRL DPPS features a Neural Network (NN) for policy decision-making. The NN input and output layers are dimensioned according to the problem inputs and the control variables in play. The NN is used by the agents to perform cross-layer optimization by jointly deriving wireless and motion policies at once. In the Learning Engine, we train the agents' NNs for the specific mission. Thanks to our virtual environment, we perform extensive performance data collection and derive optimal policy making strategies that optimize the desired control objective dictated by the Network Operator (see Step II in Fig. 1). Finally, the performance of the trained NNs are tested in the Virtual Testing Environment (see Step III in Fig. 1), which interacts with the CORE/EMANE emulation tool, similarly to how the Virtual Training Environment does (see Step IV in Fig. 1). The testing phase's main task is to verify the quality of the just-concluded training in terms of control objective optimization before sending the network configurations to the UAVs. We use Section IV to outline the components of this integration effort and explain in detail how our training process operates.

C. Neural Network Configurations

Once the Virtual Testing Environment has trained and tested the NNs, the Control Framework distributes the NN Configurations to the individual network nodes (the agents) over the wireless interface (see V in Fig. 1). Specifically, each UAV receives a different NN Configuration. These configurations are employed at the individual UAVs' DRL Drone Programmable Protocol Stack (DRL DPPS) throughout the mission to optimize the network parameters in real-time. Once dispatched, and differently from the emulated DRL DPPS, the NNs interact with hardware radio and motion front-ends. This way, the agents use the received NN configurations to adapt to the dynamic network conditions and to achieve the overall network behavior defined by the Network Operator. The Neural Network Configurations consist of the DRL NN to be executed by the DRL DPPS and a configuration file. The configuration file provides information about (i) the dimension of the DRL NN; (ii) a mapping between the input neurons of the DRL NN and the UAV network state information; (iii) a mapping between the output neurons of the DRL NN and the UAV's control parameters. Last, the NNs' internal layers are dimensioned according to the state space and the action space, as we will see in detail in Section IV.

An example of NN configuration file for UAV '5' in quote (A) is provided in Listing 2. According to the example reported in quote (A), each NN has 12 inputs, which are mapped to specific UAV network variables: the location of the 6 UAVs and their TX power. The outputs of each NN are instead 15 (the cartesian product between the possible control actions) and are mapped to the control variables of the UAV of interest.For instance, in the case of UAV '5', the 15 NN's output correspond to the combinations of the 5 allowed control actions for the UAV's location (move north, move east, move south, move west, and keep fixed) and the 3 allowed actions for the UAV's TX power (increase TX power, decrease TX power, and keep TX power fixed). The first UAV's 5's NN's output corresponds to a movement of UAV 5 one step north and to an increase of its TX power; the second NN's output corresponds to a movement of UAV 5 one step east and also increases its TX power, and so on. The nodes use this file to feed the DRL NN the correct network state information, and at the same time to interpret the outcomes of the DRL NN and translate them into actions.

Although EMANE makes it possible to specify the three-dimensional location of each UAV, as well as the velocity at which each UAV navigates within the emulated area, for the sake of illustration we assume that all UAVs move at the same velocity and hover at the same altitude.

```
<NN_size>
   <NN_inputs> 12 </NN_inputs>
   <NN_layers> 50, 50 </NN_layers>
   <NN_outputs> 15 </NN_outputs>
</NN_size>

<input_map>
   <in1> UAV1.LOC </in1>
   <in2> UAV1.TX_POW </in2>
   <in3> UAV2.LOC </in3>
   <in4> UAV2.TX_POW </in4>
   ...
</input_map>

<output_map>
   <out1> UAV5.LOC.NORTH, UAV5.TX_POW.UP </out1>
   <out2> UAV5.LOC.NORTH, UAV5.TX_POW.DOWN </out2>
   <out3> UAV5.LOC.NORTH, UAV5.TX_POW.KEEP </out3>
   <out4> UAV5.LOC.EAST, UAV5.TX_POW.UP </out4>
   ...
</output_map>
```

Listing 2: DRL NN configuration file for UAV 5 in quote (A).

Different from central control approaches, this procedure does not suffer from stale information retrieval and control latency. This because the Neural Network Configurations are dispatched once. Thus allowing nodes to optimize performance directly at the edge of the network and in a distributed fashion. The many hours of emulated flight and networking experiments carried on by the Learning Engine help reduce the performance gap between model and implementation typical of model-based approaches. In Section IV-D, we will show how training procedures can be accelerated via environment parallelization.

This approach facilitates reconfigurability of the network and objectives. Specifically, to change the behavior of a network, it is sufficient to define a new control objective. The Control Framework will take care of re-training and distributing new Neural Network Configurations. Moreover, it is worth mentioning that previously trained Neural Network

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

Configurations can be stored locally and fetched on-demand for later uses , which eliminates the problem of re-training the NNs for common and previously solved control problems.

III. The DRL Drone Programmable Protocol Stack

An overview of the DRL Drone Programmable Protocol Stack (DRL DPPS) architecture is reported in Fig. 2. The DRL DPPS is used at individual UAVs to carry our motion and wireless operations at all layers of the protocol stack, as well as in the Control Framework's Learning Engine to train and test the NN policy making for specific mission objectives. In the latter, the Physical layer and Motion operations are performed by the virtualized CORE/EMANE environment, while in the former these operations are implemented through hardware motion and RF front-end. By employing the whole DRL DPPS architecture in the Control Framework's Learning Engine (with the exclusion of the hardware front-ends), we obtain a realistic emulation environment which is key to our high-fidelity performance data collection and effective DRL training. With this premise in mind, in this section, we describe the DRL DPPS design choices and implementation details. Furthermore, the details on how the DRL DPPS is interfaced with CORE/EMANE and employed in the Learning Engine are discussed in Section IV.

To implement the wireless protocol stack of the UAV nodes we start from the Drone Programmable Protocol Stack presented in [1]. This architecture concerns three planes: the *Decision plane*, the *Register plane*, and the *Data plane*. We revisit [1]'s three-plane design by replacing the decision plane with the new *DRL plane*, yet maintaining its architectural functionality; to optimize the networking and motion control parameters at once in a cross-layer fashion. We call this new architecture the DRL Drone Programmable Protocol Stack (DRL DPPS). Here we summarize the three planes' functionalities and the interactions with one another, highlighting the main differences with the implementations in [1].

A. DRL Plane

Upon receiving the Neural Network configurations from the Control Framework, each UAV (agent) stores it in the DRL DPPS *Decision Plane*. The *DRL Plane* is in charge of determining the optimal motion and networking policies in real time, by dynamically adapting to the changing network conditions and making decisions on up-to-date UAV network state information. To derive mission-tailored policy optimization at every given time, the DRL plane executes the NN by feeding its inputs and interpreting its outputs according to the received configuration file. In this work, we exploit the simplicity of a widely adopted DRL variant called Q-learning, which aims at optimizing an estimate (called Q function) of the objective function we are trying to maximize (i.e., the NO's objective). The NN employed by the DRL is a Deep Q-Network (DQN) which uses stochastic gradient descent (SGD) to approximate the Q-function. We cover these design choices in detail in Section IV. What is most important from an architectural standpoint is that the adopted DRL approach approximates a complex optimization function by means of a relatively simple Neural Network that can be 'probed' in real time with $\mathcal{O}(1)$ complexity. This design allows for rapid calculation of new policies which can be implemented at the *Data Plane* with negligible latency which, instead, would be hard to avoid when using traditional mathematical solvers.

B. Data Plane

Similar to [1], the *Data Plane* is responsible for implementing the computed optimal policies by reconfiguring motion control and networking stack parameters. To do so, this plane implements a programmable protocol stack spanning all networking layers with the addition of the motion layer ('Layer 0'.) The protocol stack is fully programmable in software and exposes control APIs to tune key control parameters at all layers of the stack, including MAC, Physical, and Motion. This is possible thanks to the adoption of open-source flight control firmware (we here use Ardupilot[29]) and Software-defined Radios (SDRs) as radio front-end[30]. These are programmable hardware that implements the Physical and MAC layer functionalities in software, and make them controllable in real-time by the OS[31]. In this way, the Data Plane provides the necessary tools to prototype cross-layer control algorithms spanning multiple (potentially all) layers of the protocols stack—plus motion—all at once.

The control interface between the Data Plane and the DRL Plane is dual-purpose and operates as follows: (i) the *DRL Plane*'s NN can retrieve network state information from the *Data Plane* through the *Register Plane* (e.g., UAVs' locations, UAVs' TX power, etc.) to determine the current UAV network state within the DRL's defined state space; (ii) the *Data Plane* can configure the networking and motion parameters of the programmable protocol stack following the optimized policies suggested by the *DRL Planes*' NN (e.g., adopt a new 'UAV location' in space, decrease the node's 'TX power', etc.) The latest optimal policies are stored in the *Register Plane*.

When the DRL DPPS is installed on the UAVs (see Fig. 2), the lower layers of the programmable protocol stack (Physical and Motion) interface with the radio and motion front-ends hardware through the software-defined radio (i.e., USRP hardware driver (UHD)) and the flight controller (i.e., Ardupilot) drivers. These in turn control the hardware through the universal serial bus (USB 3.0) and electronic speed control (ESC) interfaces, as illustrated in Fig. 2. When the DRL DPPS is instantiated in the Learning Engine (which we will describe later in detail using Fig. 3), these two layers do not drive external hardware but instead interface with CORE/EMANE which takes care of emulating the motion and physical layer functionalities. Thanks to this architectural design, the Data Plane guarantees the use of the same programmable protocol stack architecture (and its real-time operations) whether we interface the DRL DPPS with real hardware or the CORE/EMANE emulation environment. This is the architectural innovation at the core of our extensive UAV network data collection.

C. Register Plane

The *Register Plane* acts as a middleware (i) allowing the *DRL Plane* to retrieve fresh network state information from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

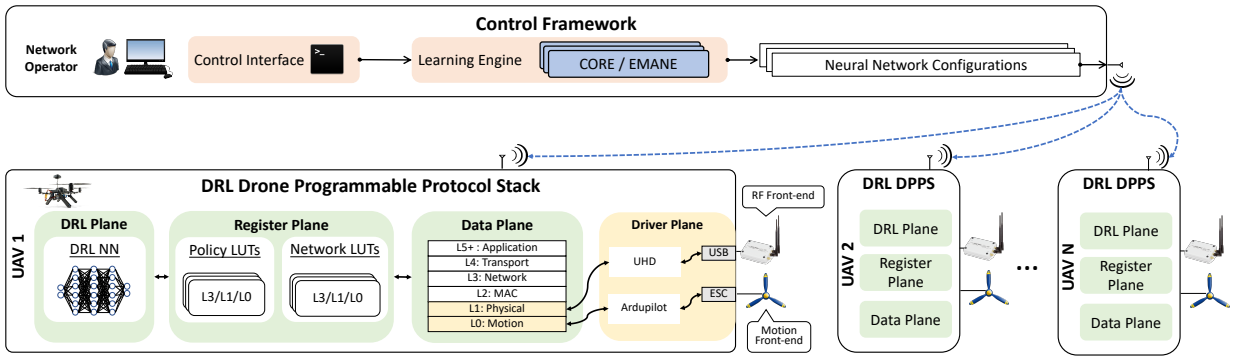

Fig. 2: Three-plane DRL Drone Programmable Protocol Stack architectural design.

the *Data Plane*; and (ii) making the computed optimal policies available to the *Data Plane* through a set of dedicated Look Up Tables (LUTs). Each protocol stack layer has a dedicated Network State LUT in the *Register Plane* where to store all the layer-related network state parameters that are used to represent the DRL state space (e.g., the neighboring UAV locations, the neighboring nodes' TX power, etc.). Conversely, dynamically re-calculated at regular intervals, optimal policies are stored in a similar way in dedicated Policy LUTs, one per control variable of interest (e.g., optimal UAV location, optimal TX power, etc.)

IV. A Ready-to-fly Virtual Environment

Among the main contributions of this work is the integration of the DRL DPPS presented in Section III with the CORE/EMANE emulation tools. The goal of this integration effort is two-fold:

(i) to develop a high-fidelity emulation environment capturing both real-time wireless channel phenomena (e.g., path-loss, delay spread, interference) and networking operations at all layers of the protocol stack (e.g., packetization, segmentation, re-transmissions, traffic bursts, processing delay) which are hard or expensive to model and can only be approximated in simulations;

(ii) to provide researchers with a reconfigurable emulation tool to design different UAV network configurations and topologies and collect high-fidelity UAV network performance data at scale. This effort serves as an effective alternative to the collection of experimental performance data for battery-powered UAV networks that is both time- and resource-expensive. Collecting a high volume of UAV network performance data is the foundation of the development of data-driven optimization.

At the same time, this emulation tool can be employed to test UAV network configurations before experimental implementation. These two operations are performed in the Control Framework's Learning Engine, in the Virtual Training Environment and Virtual Testing Environment, respectively. In the following sections, we provide a detailed description of the development of our ready-to-flight emulation environment.

A. Integrating the DRL DPPS with CORE/EMANE

EMANE is an open-source wireless network emulator developed and maintained by U.S. Naval Research Labs (NRL) and Adjacent Link LLC [28]. EMANE provides a complete and flexible emulation of both physical and MAC layers network modules, as well as providing functionalities to control network topology and node location. Additionally, EMANE embeds an Over-The-Air (OTA) channel emulator component, which allows the configuration of wireless medium propagation characteristics such as fast and slow fading, delay spread, and interference patterns among others. CORE instead provides high-level functionalities like virtualization, network bridge construction, and APIs to instantiate EMANE nodes and conveniently setup Location and Physical layer parameters in EMANE. Importantly, CORE exposes controls to EMANE's parameters (e.g., nodes' location, transmission powers) via a set of APIs configurable in real time.

This way, CORE/EMANE represents a full-fledged and comprehensive emulation environment for designing and performing wireless network experiments with diverse topologies, RF conditions, and mobility patterns. The reader is referred to [28, 32] for further details on the CORE/EMANE emulation environments architecture.

As introduced in Section III, the DRL DPPS implements a full protocol stack for UAVs. This performs the wireless stack operations from the Physical to the Application layer, with the addition of flight control functionalities carried out by the Motion layer. By integrating the DRL DPPS and CORE/EMANE emulator, we obtain a full-stack emulation environment that is representative not only of the channel propagation dynamics, but also of the protocol stack operations internal to individual wireless UAVs. This integration is extremely important as it provides a unique framework where UAV networking functionalities are reconfigurable in real time and their performance can be easily measured. This innovation is systematically leveraged in this work to perform an extensive UAV network performance data collection that is at the base of the proposed data-driven optimization.

The architectural integration between the DRL DPPS and CORE/EMANE is illustrated in Fig. 3. The same architecture is used in the Control Framework's Learning Engine both by the Virtual Training Environment and by the Virtual Testing Environment. The integration between the DRL DPPS and

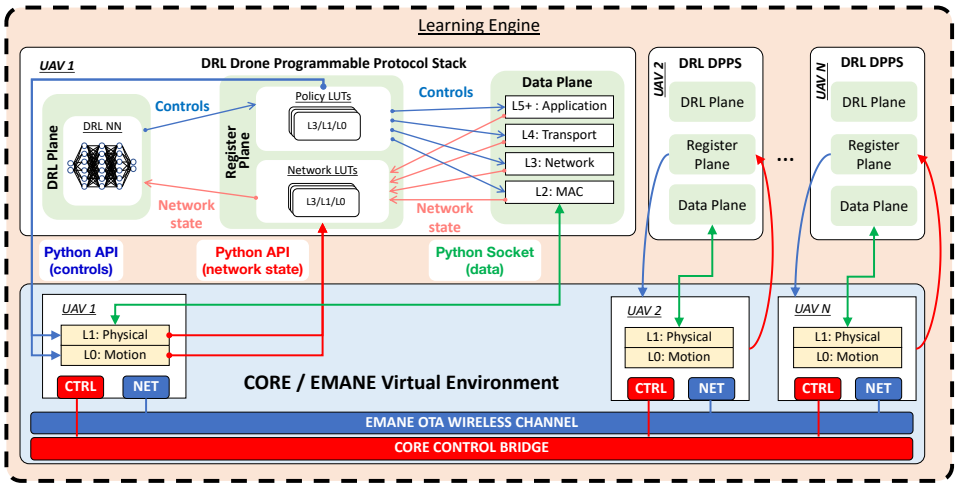

Fig. 3: Learning Engine architecture.

CORE/EMANE implements the following three functionalities:

(i) *Data communication*: The MAC layer (Layer 2) implemented in the DRL DPPS's Data plane communicates with the Physical Layer (Layer 1) emulated in CORE/EMANE. This integration is bidirectional. It guaranteed that the Data Plane's MAC Layer's data is passed down to the Physical Layer module in transmission, and that the CORE/EMANE's Physical Layer's data is passed up to the Data Plane's MAC layer in reception. These two operations are implemented through Python sockets which guarantee in-time data delivery and data integrity and happen simultaneously, in a transceiver fashion.

(ii) *Network State observation*: Similar to the other DRL DPPS's layers, the network state information associated with the Physical and Motion layers emulated in CORE/EMANE is observable by the DRL DPPS's Register Plane. This information includes neighboring UAVs' locations and their transmission powers, for example. Different from the L2+ layers, L0, and L1 layers do not execute in the same binary as the Register Plane, and the information passing is thus implemented through CORE's Python APIs.

(iii) *Motion and Network parameters control*: Similarly, the control actions operated by the DRL DPPS's DRL Plane must be relayed to the Physical and Motion layer emulated in EMANE. We employ CORE's Python APIs between the two environments for this task as well.

From an Operating System (OS) architectural standpoint, we employ Linux containers to instantiate our virtual Training (and Testing) Environment, as well as the individual UAVs residing within the environment. An illustration of the OS architectural organization of our virtual UAV network is provided in Fig. 4. In our implementation, we use a nested container architecture where the Virtual Training Environment is instantiated in a Linux docker container. This, in turn, hosts a set of Linux containers, i.e., the *CORE containers* (one per UAV). Each CORE container operates both the EMANE Motion and Physical layers and the DRL DPPS implementing MAC's and higher layers' functionalities. Once instantiated, nodes communicate with each other at the CORE level through the CORE Control Bridge and their CORE `CTRL` interface [28, 32]. At the EMANE level, nodes interact via the over the air (OTA) wireless channel through the `NET` EMANE interface as reported in Fig. 3. As per the individual UAVs, each CORE container creates one EMANE process and one DRL DPPS process. The two intra-container processes communicate with each other through Python sockets. Specifically, the data-flow streaming between MAC and Physical layer is handled through low-latency UDP sockets, while control and network state information are sent and retrieved through dedicated CORE's Python APIs to EMANE functionalities.

B. Background on Deep Reinforcement Learning

The objective of this section is to provide useful background knowledge on the specific data-driven approach that we use in this work to solve the problem of controlling a distributed UAV network.

DRL is a well-established data-driven approach with foundations in Reinforcement Learning (RL). The latter is a class of machine learning algorithms where an agent iteratively interacts with an environment to learn the optimal control policy that maximizes the desired reward. Different from supervised learning approaches, in RL the agent has no initial knowledge of which actions are more beneficial. Instead, this

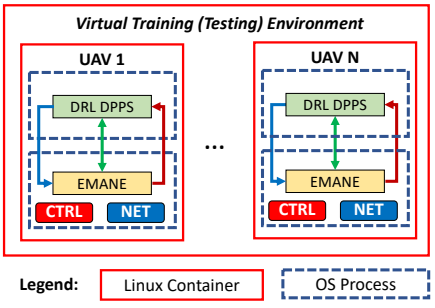

Fig. 4: OS Architecture of the Virtual Training Environment.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

one explores the environment, tries several actions in different environment states, and eventually learns the best policy through experience. Let us here introduce some concepts and RL notations that we will use in the following sections.

- *Agent*: the entity that observes the environment and takes actions accordingly, aiming at maximizing a given reward function;
- *Environment*: the physical (or emulated) world with which the agent interacts;
- *State space* $\mathcal{S}$: representing all of the possible states $s \in \mathcal{S}$ of the environment;
- *Action space* $\mathcal{A}$: all feasible actions $a \in \mathcal{A}$ that can be taken by the agent;
- *Reward* r: a metric that measures the effectiveness and/or success of an action.

In RL, the goal of the agent is to maximize the discounted future reward in (1) by selecting actions according to the observed states from the environment.

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}. \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor used to weigh instantaneous and future rewards. One effective algorithm to maximize the discounted future reward and solve the policy optimization problem is Q-learning, where the agent is trained to compute a policy that results in the selection of the optimal action-value pairs that maximize the so-called Q-values. These represent the expected discounted future reward and are computed by using the following equation

$$Q^*(s, a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a, \pi], \tag{2}$$

where $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is the policy that maps the observed state s_t to selected action a_t. The optimal Q-values are computed by solving the *Bellman's equation*, and can be represented as

$$Q^*(s, a) = \mathbb{E}_{s' \sim \mathcal{S}} \left[r + \gamma \max_{a'} Q^*(s', a') | s, a \right], \tag{3}$$

where s' and a' are the state and action in the next time-step. Although in several cases it is possible to solve (3) directly, the same task becomes challenging when dealing with a large number of states which usually leads to the so-called state explosion. That is, the state space $\mathcal{S}$ is so large that the agent cannot explore all of it, and solving (3) becomes time-consuming. To overcome this issue, a typical approach consists of approximating the Q-values rather than computing them directly. In DRL, this approximation is achieved by using a Deep Neural Network (DNN) (well-known for being a universal function approximator) with weights θ. Specifically, the Q-values in (3) are estimated as

$$Q(s, a; \theta) \approx Q^*(s, a). \tag{4}$$

and the DNN used to approximate the Q-vales Q^* is referred to as the Deep Q-network (DQN) [33].

In this paper, we will focus on the use of double-DQNs for autonomous control of UAV networks. Specifically, we consider this specific class of DQNs as they demonstrated to be effective in stabilizing the training phase and avoid over-estimations of the Q-values in the first stages of the training process. As we will discuss in Section IV-D, this is a critical aspect in data-driven optimization of UAV networks. To optimize its policy making, an agent stores experiences in the form of tuples (s, a, r, s') in a *replay buffer*. The buffer is used during the training phase to randomly extract batches of past experiences and use them to compute the DQN weights θ_M by minimizing the loss function (5) via SGD.

$$L(\theta_M) = \left[r + \gamma \max_{a'} Q(s', a'; \theta_T) \right] - Q(s', a'; \theta_M) \tag{5}$$

In double-DQN, the decision making and the policy optimization are performed by two separate networks to prevent over- (or under-) estimations of Q-values. Specifically, a first DQN, called *main network*, is in charge of learning the Q-values at every iteration by updating the weights θ_M via SGD. A second DQN, called *target network*, is instead used to compute the actions to take, while its weights θ_T are periodically copied from the main network every τ training epochs (*copy period*). Moreover, we consider an ϵ-greedy strategy where the action taken by the DRL agent depends on the $\epsilon \in [0, 1]$ parameter. Specifically, with probability ϵ, the action taken by the agent corresponds to that computed by the DQN. Otherwise, it is randomly drawn from the action space $\mathcal{A}$ with probability $1-\epsilon$.

One of the possible extensions of DRL consists of considering multiple agents cooperating and interacting with each other to maximize a shared reward function. This class of problems is often referred to as multi-agent DRL and involves several agents independently interacting with a shared environment. In multi-agent DRL, each agent has a dedicated double-DQN which is updated according to their own experience of the environment. Although this class of problems has several variations (e.g., competitive agents, partial observability of the environment, heterogeneous rewards), in this work we focus on the case where all agents cooperate to maximize a shared reward in a distributed fashion. The reader is referred to [34] and [35] for a comprehensive survey on multi-agent DRL and its applications, and on cooperative wireless networks of UAVs, respectively.

C. UAV Network Control Problem as Multi-agent DRL

In this work, we model the control problem of a Network Operator (NO) willing to dictate the behavior of a distributed network of UAVs as a multi-agent DRL employing the Q-learning techniques introduced in the previous section. As discussed in the previous sections, DRL provides a framework where a set of agents learn to react to changing environmental conditions by exchanging information, identifying the state of the environment, and adapting their policies to the current environment state. Similarly, the nodes of a distributed UAV network learn to react to the changing network conditions by recognizing the network state, exchanging information with their neighbors, and adapting their wireless networking and motion policies accordingly. As each UAV has full control of its motion and on-board wireless stack operations, we model each UAV as an independent agent in our DRL problem definition.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

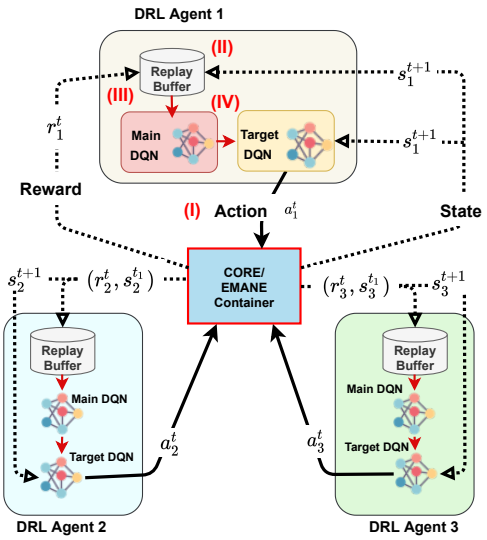

Fig. 5: Multi-agent DRL representation of the UAV network control problem

From a DRL perspective, the Control Framework's Learning Engine constructs the Virtual Training Environment based on the NO's input where:

(i) each UAV is represented as an independent *agent* featuring the DRL DPPS introduced in Section III, and interacting with the CORE/EMANE virtual environment as explained in Section IV;

(ii) the *environment* is represented by the multi-hop UAV network operations where source nodes generate data to be delivered to destination nodes while the rest of the network operates traffic forwarding tasks;

(iii) the agent's *action space* involves the control parameters specified by the NO spanning both networking and mobility domains. It is also worth noticing that in the considered multi-agent scenarios, different agents have different action spaces according to their role in the mission. For example, location-constrained UAVs feature a smaller action space as they can only adapt their networking parameters.

(iv) the *state space* is also discrete and defined as the set of values of the control parameters of the UAVs in the network (e.g., if this is the UAV's location only, the set of all UAV locations);

(v) the *reward* is UAV network-wide and defined according to the network control objective specified by the NO. The agents are collaborative and all work to obtain the highest reward possible.

In our modeling, we do not assume the distributed agents have global knowledge about the environment and the state space. On the contrary, the agents (UAVs) can communicate with their neighbors or other UAVs belonging to the same multi-hop session. Therefore, UAVs can observe and share local state information only. In this way, we model our distributed UAV network control problem as a partially-observable multi-agent DRL problem.

An illustration of the multi-agent DRL problem formulation is reported in Fig. 5. Upon receiving NO's input, the Learning Engine instantiates a virtual representation of the UAV network scenario as described in Section IV where different UAVs interact with each other via the EMANE OTA Wireless Channel. This representation is used for the multi-agent DRL Training. In this phase, the agents iteratively explore the environment, and at each iteration $t \in \mathcal{T}$ they learn to adapt their policies to the environment to maximize the common utility function. At each given training step t, each DRL agent i observes the state $\mathbf{s}_i^t$ and uses the target DQN in its DRL Plane (Step I) to independently take an action $\mathbf{a}_i^t$. These decisions are made following the ϵ-greedy strategy we described in Section IV-B and are based on the observed network state at time t, i.e., $\mathbf{s}_i^t$. It is worth recalling that each action can involve multiple control variables at several layers of the wireless stack and include the UAV's motion capabilities (e.g., change location and increase transmission power).

Upon taking action $\mathbf{a}_i^t$, node i measures the network performance r_i^t and the new network state $\mathbf{s}_i^{t+1}$ (Step II). After each iteration, the tuple $(\mathbf{s}_i^t, \mathbf{a}_i^t, r_i^t, \mathbf{s}_i^{t+1})$ is added to the replay buffer of agent i which is used to train the main DQN (Step III). Then, the weights of the main network are copied to the target DQN every τ iterations (Step IV). Within the DRL DPPS, the actions, observations, and rewards follow the logical flow described in Section III, while agents sharing traffic or operating close-by share policy information at every step t.

In conclusion, by training our multi-agent DRL representation of the UAV network, the agents learn how to adapt their policies to maximize the UAV network performance dictated by the NO in a distributed fashion. The training procedure terminates when the DQN converges and the loss function plateaus to a constant value (or no further improvements are observed). Finally, the trained Neural Network Configurations are evaluated in Virtual Testing Environment to assess their performance and are ready to be dispatched.

We now provide an example of how the NO's directives introduced earlier in the paper (see Section II, quote (A)) are formulated into a multi-agent DRL problem by the Training Engine. Six agents ($i = 1, 2, \ldots, 6$) are instantiated in our virtual *environment* which also implements the fielding location and wireless scenario in EMANE. Each *agent* features a DRL DPPS and interacts with the other agents over EMANE's OTA wireless channel. Following the NO's directive, the agents control each UAV's location in space and Physical layer TX power. We consider a discrete *action space* $\mathcal{A} = \mathcal{L}0 \times \mathcal{L}1$, where $\mathcal{L}1 = \{N, E, S, W, U, D, \_\}$ represents the six possible directions (i.e., the cardinal points, up, and down) each UAV can select as well as a 'stay put' action, and $\mathcal{L}1 = \{-1, \_, 1\}$ represents a step decrease, maintain, and a step increase of the wireless transmission power, respectively. Four of the agents are location-constrained, which means that their NN will not perform location policy optimization (i.e., $\mathcal{A} = \emptyset \times \mathcal{L}1 = \mathcal{L}1$) and the two traffic patterns are specified as flowing from node 1 to node 3 and from node 4 to node 6. Accordingly, the *state space* of the environment is defined as $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_6$, where $\mathcal{S}_i$ represents the location and transmission power of agent i. At each step t of the training, the agents operate as follows: (i) the DRL Plane's NN of each agent i operates policy optimization for the UAV's location in space and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

transmission power by taking *action* $a_i^t = \{\texttt{LOC}_i^t, \texttt{TX\_POW}_i^t\}$ if location unconstrained, $a_i^t = \{\texttt{TX\_POW}_i^t\}$ otherwise; (ii) the DRL DPPS's Register Plane collects observations from the virtual environment in the form of state and reward; (iii) the agent measures the received reward r_i^t and observes the new state s_i^{t+1}; (iv) the experience $(\mathbf{s}_i^t, \mathbf{a}_i^t, r_i^t, \mathbf{s}_i^{t+1})$ is stored in the Replay Buffer and will be used to minimize the loss function in (5) of the main DQN via SGD; (v) the new state s_i^{t+1} is used by the target DQN to compute the next action a_i^{t+1}; (vi) every τ training iterations, the knowledge of the main DQN gets transferred to the target DQN.

D. Scalability and Robustness Against Stochastic Effects

Different from other wireless applications (RAN, Wi-Fi, etc.), measuring the network-wide performance of an infrastructure-less wireless network presents some unique challenges.

I) *Slow start*: Due to the multi-hop nature of UAV networks, the first time instants after instantiating the network might be characterized by the lack of transmission activities in some parts of the network. Indeed, to relay data to the next hop, some nodes must first wait to receive packets from previous hops. However, due to transmission and processing latency introduced by multi-hop relaying, packets generated at source nodes are not immediately available at intermediate nodes. This generates a transient phase where not all nodes of the network are operative at once and performance measurements might fluctuate significantly. Thus, testing the performance of a multi-hop wireless network at regime asks for a measurement period longer than the transient phase;

II) *Slow transitions*: Differently from fixed-backhaul wireless networks and single-hop wireless systems, infrastructure-less wireless networks maintain memory of previous states and network parameters' configurations. For example, the effect of a single control action (e.g., a change in UAV's location and TX power) on a high-level network performance metric (e.g., end-to-end network throughput) is not immediately measurable. Similar to the previous point the traffic needs to propagate through the whole network before the performance can stabilize to the new network configuration. Accordingly, performance measurements need to be long enough to allow the network performance to stabilize to the new configuration.

III) *Stochastic performance metrics*: when measuring high-layer performance metrics such as end-to-end throughput on a multi-hop wireless network, one might notice a stochastic behavior of the measured performance. This is caused by the stochastic nature of the wireless channel, medium access techniques, data availability, and—more generally—lower layer protocol stack operations. This effect is exacerbated by the number of wireless hops the end-to-end data flow traverses. The data distribution of the end-to-end throughput (measured in packets per second) for a 6-UAV network and 50 identical-configuration runs is reported in Fig.6. It reports an average of 40 pkts/s with variance equal to 68 as well as large outliers (in the order of 200 pkts/s). To obtain a representative performance measure of a given network configuration, it is therefore important to perform measures in batch and average out the outliers.

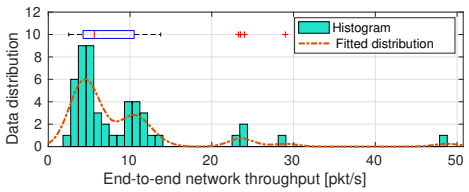

Fig. 6: Performance data distribution for 50 experiments on a 6-UAV network.

These aspects combined pose severe challenges to the collection of extensive and representative performance data which is at the foundation of any data-driven approach. If left unaddressed, these issues result in excessively long data collection periods, even when performed in a virtual training environment.

To overcome the above issues, we have developed a training pipeline that implements parallelization and outlier suppression. To collect the performance metric of a given network configuration, issue I) imposes a measurement time long enough to allow nodes relaying traffic to receive data. Indeed, such measurement time can be evaluated in our virtual environment per individual network fielding. For example, we have measured that the duration of such an interval is 5 seconds for network instances with less than 20 UAVs. As mentioned before, our emulation environment aims at mimicking an actual UAV fielding and, thus, executes in real time. This means that 1 second in the emulated environment corresponds to 1 second in the real world. To address the long training time resulting from this feature, we designed our 'ready-to-fly virtual environment to execute a set of K emulated environments in parallel, each running on a dedicated Linux docker container. By instantiating K independent parallel environments, our agents perform K actions. Therefore, they collect K rewards and K observations, this way generating K times more performance data than single-container executions effectively cutting the exploration time by a factor K. These parallel environments are non-identical and evolve independently over the iterations $\mathcal{T}$, thus representing K independent learning threads for the agents. To tackle issue II), we leveraged the reconfiguration capabilities of our virtual emulation environment and reset the network state upon any action. In this way, we remove any residual memory of previous network configurations, and are able to assess uniquely the performance of the actions taken at a given time instant with no memory of the previous ones.

While parallel executions speed up the training time by a factor K they are not sufficient to address issue III) which causes metric outliers that could result in overestimation (or underestimation) of Q-values and affect the learning performance of the DRL agents. To alleviate this problem, we leverage again parallel virtual executions by instantiating clusters of N_S environments for each of the K containers,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

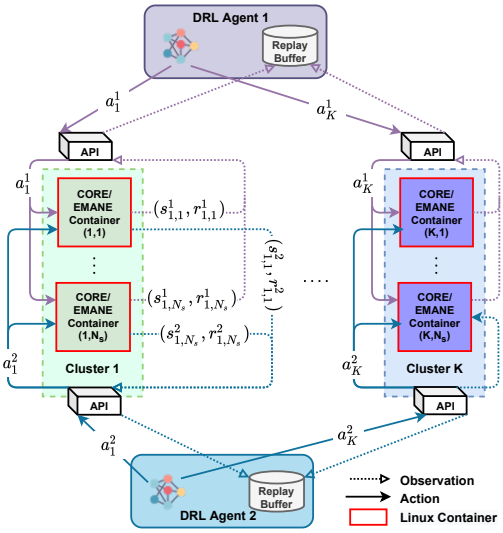

Fig. 7: Architectural design of parallel training.

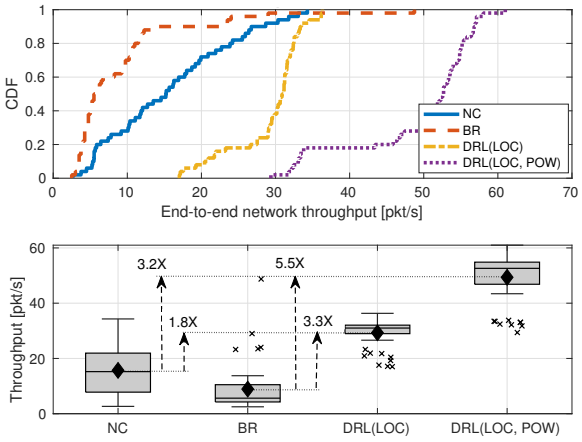

Fig. 8: Average performance results for Max-sum-throughput and different control schemes (100 experiments) under 6 UAVs network.

for a total of $N_S \times K$ parallel environments as shown in Fig. 7. Within each cluster, the instantiated environments are identical copies of each other (i.e., they share the same RF conditions, topology, and network parameters), and are used to compute average performance metrics that mitigate the impact of outliers. As shown in Fig. 7, our framework is designed to combine the rewards coming from each cluster and to remove outliers which stabilizes the training procedure (i.e., the approximation of the Q-values). Architecturally, each environment is instantiated through a Linux docker container. Our 'ready-to-fly virtual environment implementation thus employs $N_S \times K \times |\text{agents}|$ Linux containers executing in parallel. This feature is paramount to ensure scalability with respect to the number of agents, states, and actions, which could severely prolong the training phase. Indeed, our parallel training architecture allows the exploration of K environment instances at once. This results in faster exploration time, and provides a scalable training approach for DRL-based solutions.

In conclusion, the presented 'ready-to-fly' virtual environment architecture allows us to perform an extensive and representative experimental data collection for multi-hop UAV networks without compromising the duration of our training.

V. Evaluation

In this section, we test the performance of our DRL-based optimization on a series of fielding scenarios and control objectives. Here, we report the testing performance of the trained DRL agents as measured in the Control Framework's Virtual Testing Environment. Specifically, we consider the following three control schemes:

- *No Control (NC)*: in this case, all UAVs operate under static network and motion parameters. They use the initial fielding location and wireless networking parameters without optimizing them throughout the experiments. No Control is a non-optimized control scheme and serves as a baseline for our evaluation.
- *Best Response (BR)*: UAVs optimize their motion and wireless networking parameters individually with neither cooperation with other nodes nor considering the cross-layer coupling between protocol stack variables. For example, under this control scheme, UAV location and transmission power are optimized individually without considering the effects of these decisions on the performance of other protocol layers or nodes on the channel. This approach has proved to be limiting in wireless networked systems and serves as a second baseline in our evaluation.
- *Deep-reinforcement Learning (DRL)*: this is our proposed approach. In DRL, each UAV is controlled by an individual DRL agent that uses a pre-trained NN to implement cross-layer motion and networking policy optimization. Furthermore, different agents exchange information with one another in a coordinated yet distributed fashion.

For each UAV network configuration, we evaluate the performance of the aforementioned three control schemes over 100 different initial UAV fielding configurations. Moreover, for each fielding, we perform 10 independent runs, and report the average performance results upon convergence of the agents to a stable operational point.

Experimental Configuration: In all of our experiments, we consider UDP transport layer protocol with 256 Byte long packets, single-path routing scheme, and frequency division multiplexing MAC. At the Physical layer, wireless communications occur in the 2.4 GHz ISM band with operational bandwidth of 1 MHz. From a DRL optimization perspective, we consider optimization intervals t of 5 seconds, Physical layer's TX power step size of 5 dBm, and we constrain the UAVs' relocation range to a $40\,000\,\text{m}^2$ box with step size equal to 20 meters, unless otherwise specified. As RF antennas employed for UAV communications are usually dipole modules mounted on the upper side of the main frame, in this work we only consider UAV mobility on the latitude and longitude planes. This allows us to simplify the emulated signal propagation patterns and avoid modeling frame shadowing, blockage, and reflection effects which incur when wireless UAV fly at different altitudes.

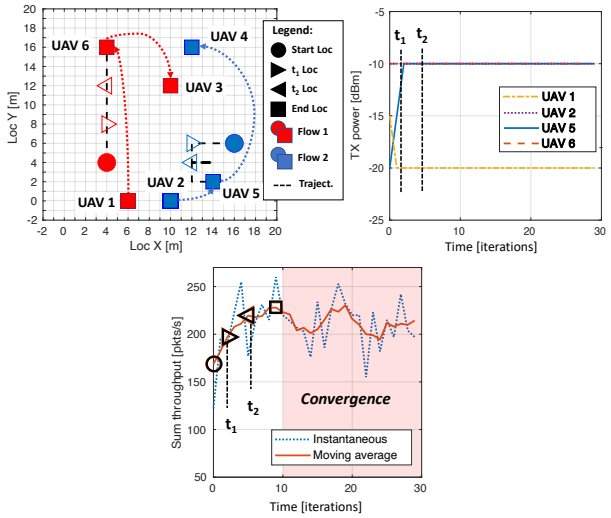

Fig. 9: A single-run experiment for Max-sum-throughput with 6 UAVs

A. *Effectiveness*

We start our analysis by assessing the effectiveness of our DRL-based control scheme in addressing the control problem defined in quote A. First, we assess the performance of the network when the two relays are allowed to control their location only. The average performance results for NC, BR, and DRL when employing only UAV's location control (DRL LOC) are reported in Fig. 8. At the top of the figure, we show the cumulative distribution function (CDF) of the measured end-to-end network throughput, i.e. the objective function specified by the NO, for different control schemes. The box-plots in the bottom part of the figure instead report the average values of those experiments. The proposed DRL-based control outperforms NC and BR, achieving 1.8x and 3.3x better performance.

B. *Flexibility*

Now, we analyze how the optimization performance changes when we increase the degrees of freedom of the DRL agent by extending the control action space. Thanks to the flexibility of our framework, the NO can select among a wide number of UAV control parameters and we can easily carry out this evaluation. We extend the control parameters to include UAVs' Physical layer TX power together with UAVs' location. The performance of this new control problem is reported in Fig. 8. Over 100 testing experiments, the proposed DRL-based control overcomes NC and BR, achieving 3.2x and 5.5x end-to-end network throughput. By extending the range of controls each agent can leverage, we add a degree of freedom to the solution space. This results in a better optimal operational point for the UAV network which reports an increase in performance of 1.7x on the case with location control only.

Figure 9 illustrates the DRL optimization convergence dynamics for a single experiment for the network scenario in Quote (A) processed in Listing 1. The top and bottom-left parts of the figure show the UAVs' location and UAVs' TX power over time, respectively. The bottom-right part of the figure reports the performance metric for this experiment, that is, the sum of the two end-to-end sessions' throughput. From the top part of the figure, we notice how the UAVs' locations are optimized by the DRL agents and go unchanged after the 10-th iteration, while, from the bottom-left part of the figure, we notice that the TX Powers already stabilized at the 4-th iteration. When the UAVs' parameters operational points stabilize, the UAV network can be deemed to have reached convergence and have achieved the NO's desired behavior (apart from small fluctuations at regime due to the stochastic nature of the wireless environment).

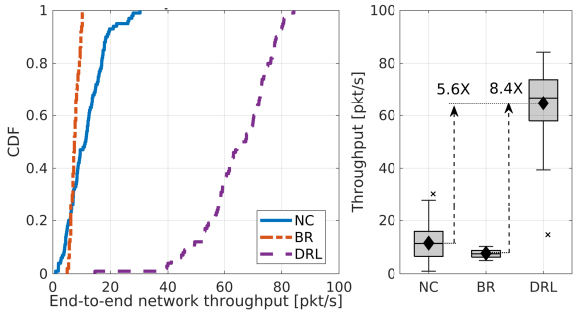

Fig. 10: Average performance results for Max-sum-throughput and different control schemes (100 experiments) under 8 UAVs network.

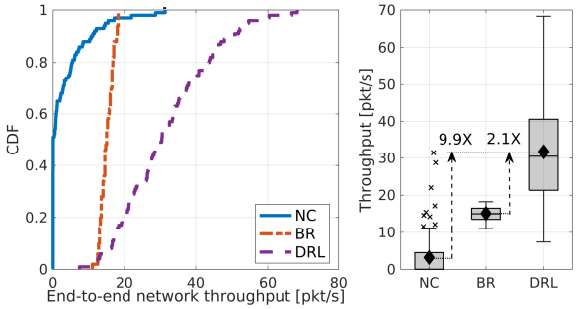

Fig. 11: Average performance results for Max-sum-throughput and different control schemes (100 experiments) under 12 UAVs network.

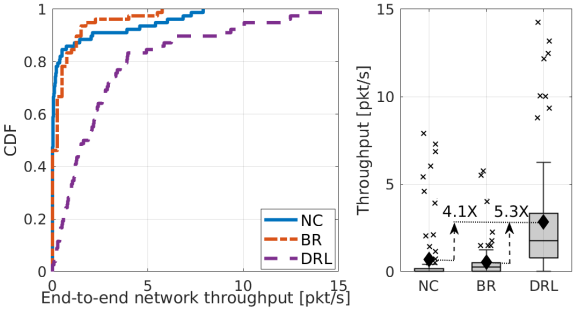

Fig. 12: Average performance results for Max-sum-throughput and different control schemes (100 experiments) under 20 UAVs network.

C. *Scalability*

Here we assess the scalability performance of our data-driven optimization approach on larger-scale scenarios than that considered in Quote (A). In Fig. 10, we report the optimization performance for an 8-UAV network with 2 sensing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

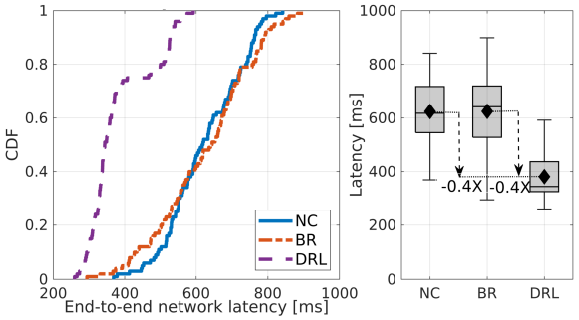

Fig. 13: Average performance results for Min-average-latency and different control schemes (100 experiments) under 6 UAVs network.

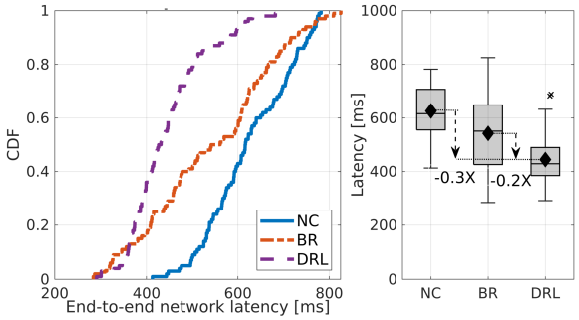

Fig. 14: Average performance results for Min-average-latency and different control schemes (100 experiments) under 8 UAVs network.

areas and 2 report areas (control objective is again to maximize the sum of the two end-to-end sessions' throughput and the controls are again UAVs' locations and TX power). For this mission, 4 UAVs are tied to sensing and reporting tasks and are location-constrained, while the remaining 4 UAVs are free to move and operate as traffic relays to support multi-hop communications. Over 100 testing experiments, DRL outperforms NC and BR by 5.6x and 8.4x.

We assess the performance of our optimization on another test experiment with 12 UAVs, 3 sensing areas, and 3 report areas; keeping control objective and controls unchanged. This time, 6 nodes are location-constrained while the remaining 6 operate as relays. For this experiment we constrain the UAVs' relocation range to a $90\,000\,\mathrm{m}^2$ box. The performance results are reported in Fig. 11. In this test, DRL performs 9.9x and 2.1x better than NC and BR, respectively (measures averaged over 100 experiments).

Last, we test our solution on a complex scenario with 5 sensing areas, 5 report areas, and a total of 20 UAVs. This larger-scale scenario is characterized by strong interference due to the higher number of nodes. Therefore, to achieve the NO's intent, combating interference via efficient network control becomes a paramount task. The results reported in Fig. 12 indicate that, without optimization, the harsh interference conditions impair basic communication procedures. Indeed, under NC and BR control schemes, the network operates at less than a packet per second. Nonetheless, the proposed DRL-based optimization proves its effectiveness even in complex, interference-prone scenarios that are hard to optimize. DRL reports throughput of 3 packets per second with gains of 4.1x and 5.3x over NC and BR, respectively.

In conclusion, DRL improves the performance of the UAV network by an average 3.7x with respect to the second-best performer.

D. Reconfigurability

As discussed in Section II, the presented control architecture facilitates network reconfigurability and can be employed to reconfigure the UAV network to achieve several network control objectives. To achieve different desired network behaviors, the NO only needs to change the desired control objective through the Control Interface (e.g., modify the objective on Line 26 in Listing 1) while the presented Control Framework takes care of generating and dispatching the new Neural Network configurations over the wireless interface. In the following, we demonstrate the programmability features of the presented control approach by measuring the performance of our data-driven optimization for a new control objective. We assess the performance of the 6-UAV and 8-UAV networks presented above for the min-average-latency control objective, which aims at minimizing the average end-to-end latency across the UAV network's sessions (i.e., from sources to destinations). The average testing results for 100 experiments are reported in Fig. 13 and Fig. 14 for the two UAV networks, respectively.

For this new control objective, our DRL-based optimization reports an average end-to-end latency reduction of 0.4x, and 0.2x over the second-best performer for the two UAV networks, respectively.

E. Data-driven Optimization vs Convex Optimization

We conclude our evaluation by presenting the performance comparison between the convex optimization-based control approach presented in [1] and the data-driven control approach presented in this article. Specifically, we compare the performance of the two control solutions on Scenario 6 considered in [1], which involves 8 UAVs, two sensing areas, and two reporting areas. The controls are again UAVs' locations and TX power. For both schemes, we consider the same experimental configuration described at the beginning of this section. As shown in Fig. 15, the proposed data-driven optimization control scheme (DRL) outperforms the convex optimization-based control schemes (SC) in [1], achieving 1.7x better performance. The reason behind this performance gap is that the performance of the model-based optimization proposed in [1] is tightly coupled to the accuracy of the models employed in the problem formulation and to the quality of the approximation and relaxation necessary to solve the UAV network control problem via convex optimization. On complex UAV networks, which require a conspicuous modeling and approximation effort, these effects combined can result in sub-optimal solutions that result in a performance gap between modeling and performance assessment. On the contrary, the performance of the data-driven optimization proposed in this work is driven by the performance data itself. In conclusion, data-driven solutions such as the one proposed in this paper

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

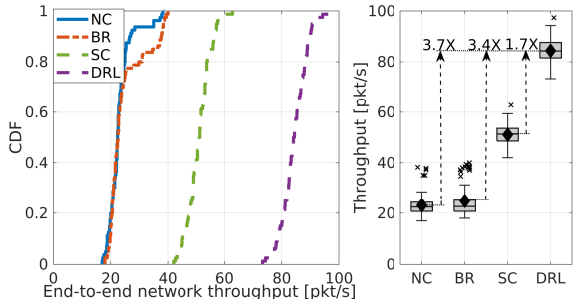

Fig. 15: Average performance results for Max-sum-throughput and different control schemes, included the convex optimization-based control scheme employed in [1] (100 experiments) under 8 UAVs network.

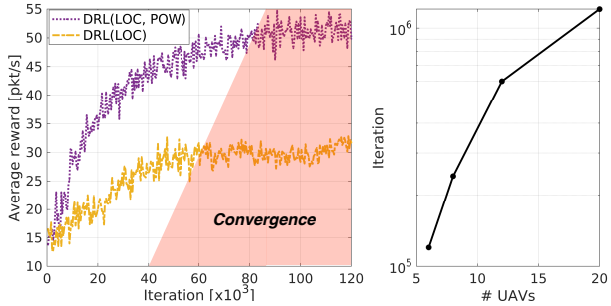

Fig. 16: Left: average reward versus training iterations for 6 UAVs network discussed in Sec. V-A; Right: number of training iterations to train DRL agents for different UAV network size.

have the potential to bridge the gap between UAV network control and network performance optimization.

The complexity of the proposed data-driven control approach is reported in Fig. 16. The left of Fig. 16 illustrates the reward function trend over the training iterations for the two DRL control schemes and the 6 UAV network discussed in Sec. V-A. When the DRL agents can only control the UAV's location, they require a fewer number of training iterations to reach convergence, that is when the reward function cannot be further improved and the system can be deemed 'trained'. The DRL agents require 60k iterations to converge in this case, with respect to the 80k iterations necessary when the agents add the UAV's transmission power to the range of their control parameters. However, it is important to notice that despite requiring a longer number of iterations to reach convergence, a wider range of control parameters corresponds to a higher reward, both at convergence and at every previous iteration. This trend suggests that augmenting the degrees of freedom of the DRL agents always results in improved performance. The right of Fig. 16 reports the number of iteration required to deem the DRL agents 'trained' in the networks with 6, 8, 12, and 20 UAVs presented earlier in this section. As shown in the figure, the required number of iterations to reach convergence grows with the number of UAVs. This is motivated by the fact that the presence of more agents corresponds to a larger solution space to be explored in order to find the optimal network configuration. As mentioned in Section IV-D, each training iteration requires 5 seconds of emulated execution time to allow the agent to gather statistically relevant performance measurements. For these UAV networks, the simulation time required to reach training convergence is approximately 166, 332, 830, and 1660 hours, respectively. Even though such training times might be excessively long for experimental performance data collection, in Section IV-D we explained in detail how to leverage parallelism in our Virtual Training Environment to collect extensive performance data. Thanks to this feature of the proposed Virtual Training Environment, the total training time cost can be efficiently reduced by running multiple containers in parallel.

VI. Related Work

Recent years have seen a surge in interest toward the integration of UAVs and the wireless infrastructure for a variety of applications such as 5G-and-beyond cellular networks [2, 7], millimeter-wave and terahertz networks [5, 6, 36], Wi-Fi [37], and ad-hoc tactical networks [1, 4, 38, 39], to name a few. In this context, AI-based control approaches are on the rise thanks to their effectiveness and applicability to motion-controllable UAV-based wireless nodes [40–51]. For example, [44–47] investigate the placement optimization of multiple UAV-based aerial base stations to maximize the coverage rate of ground users. The latter also optimizes for the energy consumption of the UAVs' recharging and landing operations, while [47] tailors its RL approach to emergency response scenarios. [48] utilized DRL for UAV-based BSs path planning so as to minimize the interference with ground infrastructure and optimize the BS-to-user latency.

Other works, instead, mainly focus on wireless networking operations optimization via machine learning. Machine learning and its flavors are employed to predict the data size of computing tasks for efficient UAV-based MEC [52]; to combat adversarial attacks for cellular-connected UAVs [41]; to optimize content caching on UAV-based BS [42]; or a mix of those [43]. The works in the literature that are the closest to the one presented in this paper are those focusing on the joint optimization of motion and wireless operations. Specifically, [40], [50], and [53] use DRL to optimize the trajectory and the power control for UAV-assisted service networks. [51] proposes a DRL technique to optimize UAVs' trajectory and time resource allocation in UAV wireless-powered IoT networks. The readers are referred to [54] for an extensive survey on AI-based control and optimization for wireless networks.

Different from the above works, which mainly rely on an abstraction of the underlying communication infrastructure and limit their contribution to a mix of analytical results and simulation, we propose a novel two-tier architecture that provides the communication infrastructure and the architectural innovations to design and implement data-driven control and optimization for real UAV network fieldings. With respect to our previous work [1], we extended the Drone Programmable Protocol Stack with data-driven functionalities (DRL DPPS). We integrated the new DRL DPPS into a new full-fledged emulation environment for UAV networks that we systematically employed to train and assess the performance of our DRL-based solution with a high degree of realism.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/OJCOMS.2021.3092690, IEEE Open Journal of the Communications Society

VII. Conclusions

In this article, we presented a novel two-tier architecture to control and optimize UAV networks based on Deep-reinforcement learning (DRL). The presented architecture features a new 'ready-to-fly' virtual environment that integrates fully-reconfigurable wireless protocol stacks for software-defined UAVs with the CORE/EMANE emulation tools. Our 'ready-to-fly' virtual environment allows us to collect extensive high-fidelity UAV network performance data without the burden of carrying out time- and energy-consuming flight experiments thus simplifying the modeling and training procedures for data-driven control solutions. In this work, we showed how our system can be employed to model different UAV network control problems as multi-agent DRL problems, collect extensive performance data, and use the data collected to train a set of DRL agents for mission-specific goals. The proposed DRL architecture implements distributed data-driven optimization (with up to 3.7x throughput gains and 0.2x latency reduction if compared to other approaches), facilitates network reconfigurability, and provides a scalable solution for large UAV networks (up to 20 nodes).

As future work, we will take these research directions:
Experimental Assessment: We intend to assess our DRL-based optimization on real-world UAVs and validate the performance we have tested in our Virtual Testing Environment. Thanks to our 'ready-to-fly' framework design, the trained DRL agents can be easily instantiated on real hardware, a procedure that is straightforward and only involves enabling the hardware drivers in the DRL DPPS. We will carefully analyze the performance of future experimental fieldings to identify discrepancies with the simulation environment [55] and employ techniques of transfer learning to further reduce the emulation-experimental performance gap.
On-line learning: We will also focus on how to optimize the performance of the distributed DRL agents in the case of compromised nodes and outages. Future research threads in this context include hybrid off- and on-line training to investigate the flexibility and adaptability of multi-agent learning in the case of previously unseen network conditions.

References

[1] L. Bertizzolo, S. D'Oro, L. Ferranti, L. Bonati, E. Demirors, Z. Guan, T. Melodia, and S. Pudlewski, "SwarmControl: An automated distributed control framework for self-optimizing drone networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1768–1777.

[2] L. Bertizzolo, T. X. Tran, J. Buczek, B. Balasubramanian, R. Jana, Y. Zhou, and T. Melodia, "Streaming from the Air: Enabling High Data-rate 5G Cellular Links for Drone Streaming Applications," *arXiv preprint arXiv:2101.08681*, 2021.

[3] L. Ferranti, S. D'Oro, L. Bonati, E. Demirors, F. Cuomo, and T. Melodia, "Hiro-net: Self-organized robotic mesh networking for internet sharing in disaster scenarios," in *2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2019, pp. 1–9.

[4] R. K. Sheshadri, E. Chai, K. Sundaresan, and S. Rangarajan, "SkyHaul: An Autonomous Gigabit Network Fabric in the Sky," *arXiv preprint arXiv:2006.11307*, 2020.

[5] L. Bertizzolo, M. Polese, L. Bonati, A. Gosain, M. Zorzi, and T. Melodia, "mmBAC: Location-aided mmWave backhaul management for UAV-based aerial cells," in *Proceedings of the 3rd ACM Workshop on Millimeter-wave Networks and Sensing Systems*, 2019, pp. 7–12.

[6] M. Polese, L. Bertizzolo, L. Bonati, A. Gosain, and T. Melodia, "An Experimental mmWave Channel Model for UAV-to-UAV Communications," in *Proceedings of the 4th ACM Workshop on Millimeter-Wave Networks and Sensing Systems*, 2020, pp. 1–6.

[7] L. Bertizzolo, T. X. Tran, B. Amento, B. Balasubramanian, R. Jana, H. Purdy, Y. Zhou, and T. Melodia, "Live and let live: flying UAVs without affecting terrestrial UEs," in *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, 2020, pp. 21–26.

[8] L. Ferranti, L. Bonati, S. D'Oro, and T. Melodia, "Skycell: A prototyping platform for 5g aerial base stations," in *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2020, pp. 329–334.

[9] M. Abolhasan, J. Lipman, W. Ni, and B. Hagelstein, "Software-defined wireless networking: centralized, distributed, or hybrid?" *IEEE Network*, vol. 29, no. 4, pp. 32–38, 2015.

[10] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "Softran: Software defined radio access network," in *ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 25–30.

[11] S. Baidya, Z. Shaikh, and M. Levorato, "Flynetsim: An open source synchronized uav network simulator based on ns-3 and ardupilot," in *ACM Intl. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2018, pp. 37–45.

[12] Z. Guan, L. Bertizzolo, E. Demirors, and T. Melodia, "WNOS: An optimization-based wireless network operating system," in *Proc. of ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Los Angeles, CA, USA, 2018, pp. 241–250.

[13] Guan, Zhangyu and Bertizzolo, Lorenzo and Demirors, Emrecan and Melodia, Tommaso, "Demo Abstract WNOS: An optimization-based wireless network operating system," in *Proc. of IEEE Conference on Computer Communications (INFOCOM)*, Honolulu, HI, USA, 2018.

[14] L. Bonati, S. D'Oro, L. Bertizzolo, E. Demirors, Z. Guan, S. Basagni, and T. Melodia, "Cellos: Zero-touch softwarized open cellular networks," *Computer Networks*, vol. 180, p. 107380, 2020.

[15] J. Liu, B. Krishnamachari, S. Zhou, and Z. Niu, "Deepnap: Data-driven base station sleeping operations through deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4273–4282, 2018.

[16] Z. Wang, L. Li, Y. Xu, H. Tian, and S. Cui, "Handover control in wireless systems via asynchronous multiuser deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4296–4307, 2018.

[17] Y. Yu, T. Wang, and S. C. Liew, "Deep-reinforcement learning multiple access for heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1277–1290, 2019.

[18] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 257–265, 2018.

[19] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 310–323, 2018.

[20] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing-based approach," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1938–1948, 2018.

[21] M. Feng and S. Mao, "Dealing with limited backhaul capacity in millimeter-wave systems: A deep reinforcement learning approach," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 50–55, 2019.

[22] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2017.

[23] Y. Sun, M. Peng, and S. Mao, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1960–1971, 2018.

[24] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[25] H. Zhang, W. Li, S. Gao, X. Wang, and B. Ye, "Reles: A neural adaptive multipath scheduler based on deep reinforcement learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1648–1656.

[26] L. Zhang, J. Tan, Y.-C. Liang, G. Feng, and D. Niyato, "Deep reinforcement learning-based modulation and coding scheme selection in cognitive heterogeneous networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 6, pp. 3281–3294, 2019.

[27] J. Jagannath, N. Polosky, A. Jagannath, F. Restuccia, and T. Melodia, "Machine learning for wireless communications in the internet of things:

A comprehensive survey," *Ad Hoc Networks*, vol. 93, p. 101913, 2019.
[28] J. Ahrenholz, T. Goff, and B. Adamson, "Integration of the core and emane network emulators," in *IEEE Military Communications Conference (MILCOM)*, 2011, pp. 1870–1875.
[29] Ardupilot, "Autopilotsuite," http://ardupilot.org/, accessed: 2020-2-1.
[30] Ettus Research. Universal Software Radio Peripheral (USRP). https://www.ettus.com/products/.
[31] F. K. Jondral, "Software-defined radio—basics and evolution to cognitive radio," *EURASIP journal on wireless communications and networking*, vol. 2005, no. 3, pp. 1–9, 2005.
[32] B. Company, "Core documentation," https://coreemu.github.io/core/emane.html, accessed: 2020-12-30.
[33] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[34] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.
[35] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 73–82, 2017.
[36] S. K. Moorthy and Z. Guan, "Letera: Stochastic beam control through esn learning in terahertz-band wireless uav networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 1039–1044.
[37] A. Guillen-Perez, R. Sanchez-Iborra, M. Cano, J. C. Sanchez-Aarnoutse, and J. Garcia-Haro, "Wifi networks on drones," in *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*, 2016, pp. 1–8.
[38] Z. Guan, N. Cen, T. Melodia, and S. M. Pudlewski, "Distributed joint power, association and flight control for massive-mimo self-organizing flying drones," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1491–1505, 2020.
[39] Z. Shaikh, S. Baidya, and M. Levorato, "Robust multi-path communications for uavs in the urban iot," in *2018 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*. IEEE, 2018, pp. 1–5.
[40] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory Design and Power Control for Multi-UAV Assisted Wireless Networks: A Machine Learning Approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7957–7969, 2019.
[41] U. Challita, A. Ferdowsi, M. Chen, and W. Saad, "Machine learning for wireless connectivity and security of cellular-connected uavs," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 28–35, 2019.
[42] M. Chen, W. Saad, and C. Yin, "Liquid State Machine Learning for Resource Allocation in a Network of Cache-Enabled LTE-U UAVs," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.
[43] N. Kato, Z. M. Fadlullah, F. Tang, B. Mao, S. Tani, A. Okamura, and J. Liu, "Optimizing Space-Air-Ground Integrated Networks by Artificial Intelligence," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 140–147, 2019.
[44] J. Qiu, J. Lyu, and L. Fu, "Placement optimization of aerial base stations with deep reinforcement learning," in *IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.
[45] V. Saxena, J. Jaldén, and H. Klessig, "Optimal uav base station trajectories using flow-level models for reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1101–1112, 2019.
[46] H. Bayerlein, P. De Kerret, and D. Gesbert, "Trajectory Optimization for Autonomous Flying Base Station via Reinforcement Learning," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.
[47] P. Valente Klaine, J. Nadas, R. Souza, and M. Imran, "Distributed Drone Base Station Positioning for Emergency Cellular Networks Using Reinforcement Learning," *Cognitive Computation*, vol. 10, 10 2018.
[48] U. Challita, W. Saad, and C. Bettstetter, "Interference Management for Cellular-Connected UAVs: A Deep Reinforcement Learning Approach," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2125–2140, 2019.
[49] D. Athukoralage, I. Guvenc, W. Saad, and M. Bennis, "Regret Based Learning for UAV Assisted LTE-U/WiFi Public Safety Networks," in *IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–7.
[50] L. A. binti Burhanuddin, X. Liu, Y. Deng, U. Challita, and A. Zahemszky, "QoE Optimization for Live Video Streaming in UAV-to-UAV Communications via Deep Reinforcement Learning," 2021.
[51] J. Tang, J. Song, J. Ou, J. Luo, X. Zhang, and K. Wong, "Minimum throughput maximization for multi-uav enabled wpcn: A deep reinforcement learning method," *IEEE Access*, vol. 8, pp. 9124–9132, 2020.
[52] G. Wu, Y. Miao, Y. Zhang, and A. Barnawi, "Energy efficient for UAV-enabled mobile edge computing networks: Intelligent task prediction and offloading," *Computer Communications*, vol. 150, pp. 556–562, 2020.
[53] N. Zhao, Z. Liu, and Y. Cheng, "Multi-agent deep reinforcement learning for trajectory design and power allocation in multi-uav networks," *IEEE Access*, vol. 8, pp. 139 670–139 679, 2020.
[54] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1472–1514, 2020.
[55] D. Callegaro, S. Baidya, and M. Levorato, "A Measurement Study on Edge Computing for Autonomous UAVs," in *ACM SIGCOMM 2019 Workshop on Mobile AirGround Edge Computing, Systems, Networks, and Applications*, 2019, p. 29–35.

Hai Cheng is a Ph.D. candidate in Computer Engineering at the Institute for Wireless IoT at Northeastern University. He received his B.Eng degree in 2015 from Xidian University, China, and M.S. in 2018 from ShanghaiTech University, China. His research interests include machine learning and optimization in wireless network systems.

Lorenzo Bertizzolo is a candidate for Ph.D. in Computer Engineering and research assistant at the Institute for the Wireless IoT at Northeastern University. He earned his B.S. and his M.S. with honors in Computer and Communication Networks Engineering from Politecnico di Torino, Italy in 2014 and 2015, respectively. His research focuses on 5G, software-defined networking for wireless, network optimization, and non-terrestrial UAV networks. He is also a collaborator of AT&T Labs Research, working on the integration of Unmanned Aerial System into the next generations' cellular networks.

Salvatore D'Oro [M'17] is a Research Assistant Professor with the Institute for the Wireless Internet of Things (WIoT) at Northeastern University, USA. He received his Ph.D. degree from the University of Catania in 2015. He serves on the Technical Program Committee (TPC) of several international conferences such as IEEE INFOCOM and is Associate Editor of the Elsevier Computer Communications journal. His research interests include optimization, learning, network slicing and their applications to 5G systems and beyond.

John Buczek is currently pursuing a BSMS in Electrical Engineering with concentration in Power Systems at Northeastern University, Boston, MA. He is an undergraduate research assistant at the Institute for the Wireless IoT at Northeastern University. His research interests include Power Electronics, Unmanned Aerial Vehicles and UAV Networks, and the Internet of Things.

Tommaso Melodia [F'18] received the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology in 2007. He is currently the William Lincoln Smith Professor with the Department of Electrical and Computer Engineering, Northeastern University. He is also the Director of the Institute for the Wireless Internet of Things, and the Director of Research for the PAWR Project Office, a public-private partnership that is developing four city-scale platforms for advanced wireless research in the United States. His research focuses on modeling, optimization, and experimental evaluation of wireless networked systems, with applications to 5G networks and Internet of Things, software-defined networking, and body area networks. His research is supported mostly by the U.S. federal agencies, including the National Science Foundation, the Air Force Research Laboratory, the Office of Naval Research, the Army Research Laboratory, and DARPA. He is a Senior Member of the ACM. He is the Editor-in-Chief of Computer Networks, and a former Associate Editor of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON MULTIMEDIA, among others.

Elizabeth Serena Bentley Elizabeth Serena Bentley has a B.S. degree in Electrical Engineering from Cornell University, a M.S. degree in Electrical Engineering from Lehigh University, and a Ph.D. degree in Electrical Engineering from University at Buffalo. She was a National Research Council Post-Doctoral Research Associate at the Air Force Research Laboratory (AFRL) in Rome, NY. Currently, she is employed by the AFRL Information Directorate, performing in-house research and development in the Communication Technology & Systems Branch and managing the Cross-Layer Heterogeneous Autonomous Resilient On-Demand Networks (CHARON) program that focuses on mission-responsive swarm networking. Her research interests are in cross-layer optimization, swarm networking, directional networking, wireless multiple-access communications, and modeling and simulation.