This paper has been accepted for publication on IEEE Workshop on Wireless Communications and Networking in Extreme Environments, LA, California, June 2022

A Middleware for Digital Twin-Enabled Flying Network Simulations Using UBSim and UB-ANC

Sabarish Krishna Moorthy[1*], Ankush Harindranath[1*], Maxwell McManus[1],
Zhangyu Guan[1], Nicholas Mastronarde[1], Elizabeth Serena Bentley[2] and Michael Medley[2]
[1]Department of Electrical Engineering, University at Buffalo, Buffalo, NY 14260, USA
[2]Air Force Research Laboratory (AFRL), Rome, NY 13440, USA
Email: {sk382, ankushha, memcmanu, guan, nmastron2}@buffalo.edu,
{elizabeth.bentley.3, michael.medley}@us.af.mil

***Abstract*—Data-driven control based on AI/ML techniques has a great potential to enable zero-touch automated modeling, optimization and control of complex wireless systems. However, it is challenging to collect network traces in the real world because of high time and labor cost, weather limitations as well as safety concerns. In this work we attempt to tackle this challenge by designing a multi-fidelity simulator taking wireless Unmanned Aerial Vehicle (UAV) networks into consideration. We design the simulator by interfacing two Unmanned Aerial System (UAS) simulators we have developed in prior years: UBSim and UB-ANC. The former focuses on UAV network optimization and policy training by considering explicitly the network environments such as blockage dynamics, while the latter focuses more on high-fidelity UAV flight control. We first develop a coordination interface referred to as *SimSocket* for signaling exchanges between UBSim and UB-ANC in simulations, and then showcase coordinated simulations based on UBSim and UB-ANC. The new research that can be enabled by the integrated simulator is also discussed for digital twin-based UAS systems.**

***Index Terms*—Unmanned Aerial Vehicle (UAV), Multi-Fidelity Simulation, UBSim, UB-ANC, Data-Driven Control.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have been envisioned as key "*tools*" that can enable a wide set of new applications. Notable examples include flying hotspots in cellular networks [1]–[3], aerial backhaul fabric [4], border surveillance and environmental monitoring [5], precision agriculture [6] as well as battlefield inspection [7]. A primary challenge towards wide adoption of UAVs is that incorporating UAVs can significantly increase the complexity in controlling the underlying wireless networks. Data-driven control based on artificial intelligence (AI) and machine learning (ML) techniques has been shown to have a great potential to enable zero-touch automated modeling, optimization and control of complex wireless systems [8]–[10]. However, the performance of ML (especially deep learning) algorithms highly relies on the availability of a sufficient amount of well-labeled, contextual data for model training, and this will lead to a slow convergence rate when it comes to online applications [11], [12]. Moreover, it is very challenging to collect UAV network traces in the real world due to high time consumption and labor cost, weather limitations, safety concerns, as well as Federal Aviation Administration (FAA) regulations.

Alternatively, different network simulators have been developed that can be used to generate synthetic data for offline ML model training. For example, in [13] the authors develop a simulation platform called Simonstrator for ad hoc unmanned aerial networks. In [14], Bryan Kate et al. develop a simulator called Simbeeotic for simulating and prototyping large-scale Micro Air Vehicle (MAV) swarms at early stages of system design. Simbeeotic has also been integrated with an indoor helicopter testbed for real-world experiments. Emerson marconato et al. develop an ad hoc network simulator called AVENs [15], which is a hybrid aerial network simulator combining network simulator OMNeT++, flight simulator X-plane and architectural model LARISSA. Ahmad Y. Javid et al. develop UAVSim for cyber security analysis in UAV networks based on OMNeT++ and a custom-built UAV simulator [16]. In [17], the authors develop a multi-layer UAV network simulator based on Gazebo, Ardupilot and NS-3. Similarly, Sabur Badiya et al. develop FlyNetSim [18] based on Ardupilot and NS-3.

Different from existing simulators, in this work we focus on simulations of digital twin-enabled data-driven flying networks. A digital twin refers to a virtual replica of the physical world [19]–[21]. Based on the digital twin, agents are allowed to learn the optimal or at least a desirable policy without actually interacting with the physical environments. A primary challenge with a digital twin is that, because of the mismatch between the dynamics of the source domain (i.e., the digital twin) and the target domain (the physical environment), the learned policy may suffer from the so-called sim-to-real gap problem when applied in real networks. To the best of our knowledge, none of the existing simulators have been designed for simulating the dynamic mismatch in digital twin-enabled flying networks.

The main contributions of this work are twofold. First, we develop a multi-fidelity simulator for digital-twin enabled

ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER: (a) Contractor acknowledges Government's support in the publication of this paper. This material is based upon work funded by AFRL, under AFRL Contract No. FA8750-20-1-0501 and FA8750-20-C-1021. (b) Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

Distribution A. Approved for public release: Distribution unlimited AFRL-2022-1933 on 25 April 2022.

[*]The authors contributed equally to this work.

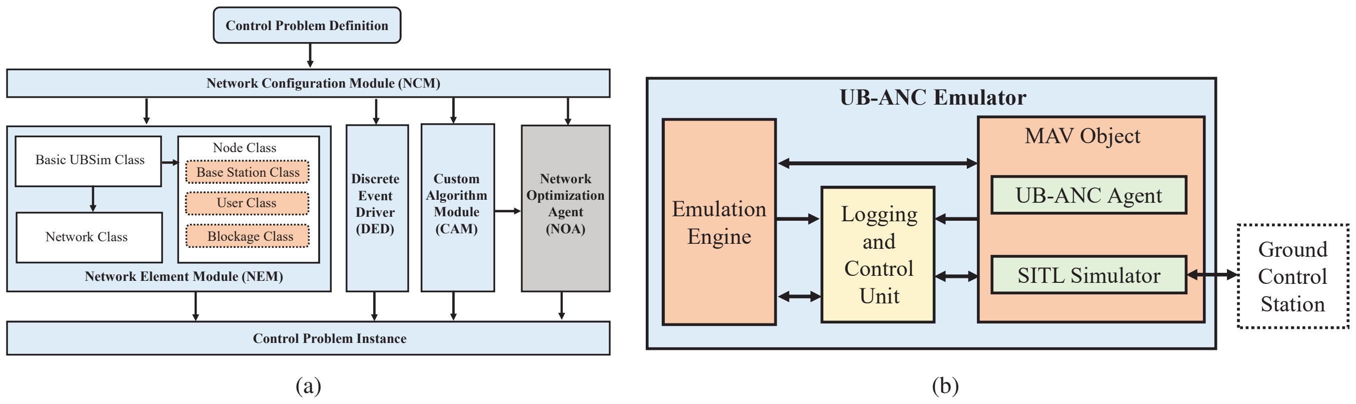


Fig. 1: Architectures of (a) UBSim and (b) UB-ANC.

wireless UAV networks by interfacing UBSim [1] and UB-ANC [22]. A middleware called *SimSocket* has been designed for signalling exchanges between UBSim and UB-ANC. Second, we showcase the multi-fidelity simulation capability of the integrated simulator. The enabled new experiments for digital-twin-based UAV networking are also discussed.

The remainder of the paper is organized as follows. In Sec. II we provide an overview of UBSim and UB-ANC. Then, we introduce the design of SimSocket in Sec. III. We then demonstrate the performance of SimSocket in Sec. IV and discuss the enabled new research in Sec. V. We finally draw the main conclusions in Sec. VI.

II. UBSim and UB-ANC: A Primer

In this section we introduce and discuss the overall architecture of UBSim and UB-ANC before describing the design of SimSocket in Sec. III.

A. UBSim

UBSim is a Python-based event-driven simulator for broadband integrated aerial-ground wireless networks. As shown in Fig. 1(a), UBSim comprises five major modules: *Network Configuration Module (NCM)*, *Network Element Module (NEM)*, *Discrete Event Driver (DED)*, *Custom Algorithm Module (CAM)* and *Network Optimization Agent (NOA)*. Network parameters such as the number of base stations, the number of users, operating frequency, bandwidth and blockage distribution can be configured using this module. The definition of network elements such as users, base station, blockages can be conducted in the network element module. These classes have been designed in a hierarchical manner. At the highest level is a general network element class net_elmt, which defines the basic network element attributes and operations such as registering an element in the network, specifying the parent and children elements of the element. The event-driven discrete simulation is based on open source library SimPy [23]. Finally, custom-designed optimization algorithms such as reinforcement learning algorithms are hosted in the *Custom Algorithm Module*. This module can be accessed by the NOA module for network run-time optimization.

B. UB-ANC

UB-ANC is a simulation framework that can be used to design, implement and test various UAV networking applications in software before deploying them onto real UAVs. UB-ANC has been designed using open-source software components and is therefore easily usable with several off-the-shelf as well as custom-built UAVs. As shown in Fig. 1(b), UB-ANC consists of three main components: UB-ANC agent, logging and control unit, and emulation engine [22], [24], [25]. The emulation engine is the core of the emulator and is responsible for coordinating various tasks and interfaces with each simulated MAV object. Each MAV object contains a UB-ANC agent, which contains the information regarding the target application to be executed on the MAV. A Software-in-the-loop (SITL) simulator is used to simulate the flight controller and can be connected to an open source GUI such as Ardupilot Mission (APM) planner or QGroundControl (QGC) for visualization of the flight path of the emulated MAVs. Readers are referred to [26] for more details of UB-ANC.

III. SimSocket Design

UBSim and UB-ANC focus on different aspects of wireless UAV network simulations. The former primarily focuses on UAV network optimization and policy training considering explicitly the network environments such as blockage dynamics. The latter focuses more on high-fidelity characterization of UAV flight control, packet loss and throughput measurement, among others. Recall in Sec. I that our objective is to enable multi-fidelity simulation of digital twin-enabled wireless networks. There are two challenges to integrate the two simulators. The first challenge is the mismatch in programming languages. UBSim has been developed based on Python, while UB-ANC is a C++ based emulator. Second, there are no available tools for effective signalling exchanges between the two simulators. To address these challenges, next we design a middleware called *SimSocket* for signalling exchanges between UBSim and UB-ANC. As illustrated in Fig. 2, there are three major control interfaces: *initial simulator setup interface,*

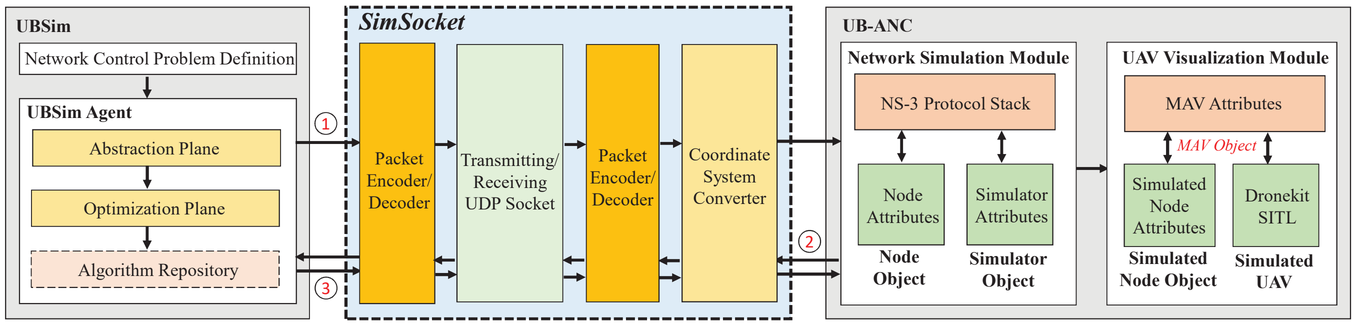


Fig. 2: Coordination interfaces of *SimSocket*: ① initial simulator setup interface, ② network status updating interface, and ③ network optimization and specification interface.

network status updating interface, and *network optimization and specification interface*.

A. Initial Simulator Setup Interface

The job of initial simulator setup is two-fold. First, it provides through UBSim's abstraction plane an interface for the users to define the network control problem. Second, through UBSim's optimization plane, it parses the specified centralized network control problem and decomposes it into a set of sub-problems, for which operational optimization programs will be generated automatically for simulations in UB-ANC. The automatic generation of distributed optimization programs has been based on Wireless Networking Operating System (WNOS) [10].

After initial setup, UBSim shares the network parameters to UB-ANC. This is done via the *initial simulator setup interface* (denoted as ① in Fig. 2). Through this interface UBSim sends two messages to UB-ANC: *node attributes* and *simulator attributes*. The former includes the parameter values of the simulated network, such as the number of nodes, the number of sessions, node coordinates, among others. The latter message includes the values required to run the simulation in UB-ANC such as the total number of packets to be transmitted, packet size, transmission rate, and simulation duration. Before the messages are sent to UB-ANC, they are first encoded by the *packet encoder/decoder* which encodes the packet following predefined format. Then the encoded messages are transmitted via the *transmitting/receiving UDP socket* and further decoded by *packet encoder/decoder* on UB-ANC's side.

B. Network Status Updating Interface

UB-ANC configures the simulations based on the information received from UBSim through the initial simulator setup interface. The simulation results are then sent back to UBSim via the *network status updating interface* (denoted as ② in Fig. 2). The simulation initialization information will also be used to update the network visualization. However, because different coordinate systems have been used by UBSim (Cartesian coordinate) and UB-ANC (Geodetic coordinate), the network topology information received from UBSim cannot be used directly in UB-ANC for simulation visualization.

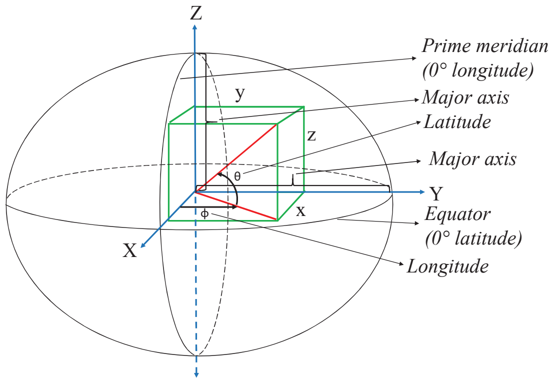


Fig. 3: Depiction of ECEF coordinate system.

To address this challenge, we first convert the node locations from Cartesian coordinates to Earth Centered Earth Fixed coordinates (ECEF) [27] using *Coordinates System Converter*. As shown in Fig. 3, ECEF is a type of Cartesian coordinate system with the origin at the center of mass of the Earth, the Z-axis extending through the true north of the Earth and the X-axis extending out from the equator (0-degree latitude and 0-degree longitude). Then we further convert the ECEF coordinates to Geodetic coordinates using Karl Olsen's closed-form algorithm [28], which considers the shape of the Earth following the World Geodetic System 84 (WGS84) model, a model used in most modern GPS systems. Finally, the resulting Geodetic coordinates can then be forwarded to the QGroundControl for visualization of the UAV flight path [29], [30].

C. Network Optimization and Specification Interface

This module is responsible for solving the network control problem following a three-step process based on the network status information received by UBSim via the network status updating interface (Step ② in Fig. 2). First, the optimizer executes the algorithms loaded during the setup phase and then calculates the optimized network parameters by substituting the run-time values in the algorithms. Then, the optimized

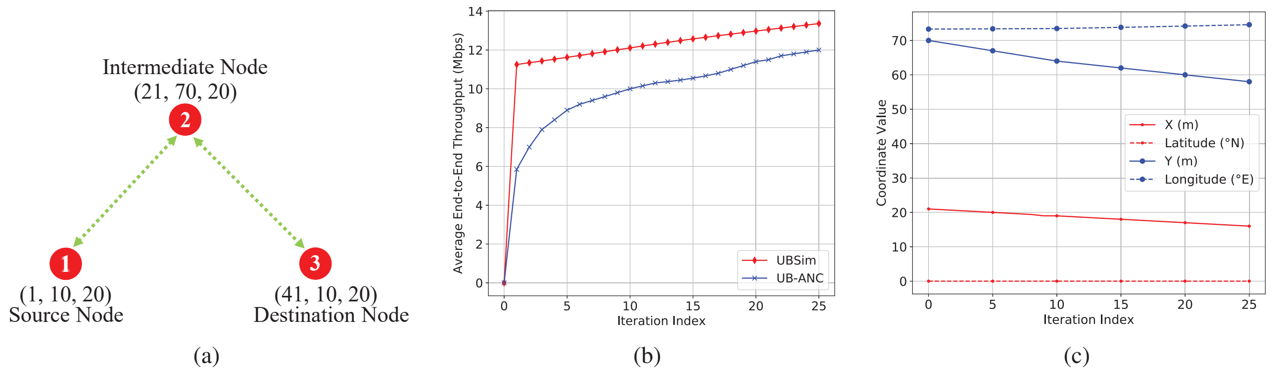


Fig. 4: (a) Topology of simulated network ; (b) End-to-End throughput; and (c) Distance between nodes.

Notation	Attribute Name	Data Type
T	Total Simulation Time	Integer
N	Total Number of Nodes	Integer
K	Total Number of Packets	Integer
R_n	Data Rate for Node n, $n = 0, 1, 2, \cdots, N$	Float
P_n	Transmission Power for Node n	Float
S	Packet Size	Double
x_n	Initial x-coordinate for Node n	Double
y_n	Initial y-coordinate for Node n	Double
z_n	Initial z-coordinate for Node n	Double

TABLE I: Network setup attributes.

network parameters are sent back to UB-ANC via *network optimization and specification interface* (denoted as ③ in Fig. 2) for next-round simulation.

D. SimSocket in Action

We briefly describe *SimSocket* in action to help readers understand better how the signals are exchanged between UBSim and UB-ANC. Table I summarizes the involved network attributes.

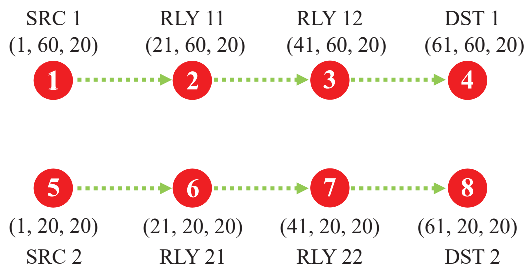
As illustrated in Fig. 2, in Step ① UBSim sends the simulation configuration parameters to UB-ANC. These include the simulation time T, the number of nodes N, the number of packets K. Additionally, for each node n in the network, UBSim sends the data rate R_n, transmission power P_n, and the coordinates of the node x_n, y_n and z_n. These attributes are sent as a comma-separated message for easy decoding at UB-ANC (refer to *packet encoder/decoder* in Fig. 2). For example, the first message will be of form $[T, N, K]$ and the second message will be sent for individual node n which will be in the form of $[R_n, P_n, x_n, y_n, z_n]$. After all the messages are sent, UBSim listens for the response from UB-ANC (refer to *transmitting/receiving UDP socket* in Fig. 2). In the meantime, UBSim will load the corresponding optimization algorithms to solve the network control problem and send the optimized network parameters to UB-ANC periodically during the simulation.

Once UB-ANC receives the necessary information from UBSim through *transmitting/receiving UDP socket*, it decodes the message using *packet encoder/decoder* and performs coordinate conversion using *coordinate system converter*. The decoded messages are then used to update the simulator and node attributes of the NS-3 protocol stack. Then, the network simulation module of UB-ANC conducts simulations and updates the network performance attributes. Example network performance attributes include link capacity, link delay, end-to-end delay, among others. After the current round of simulation, UB-ANC exchanges the simulation results with UBSim, as shown with ② in Fig. 2. The same as in Step ①, the reply messages will also follow the comma-separated format. The network performance results received from UB-ANC is decoded and the values are updated in the optimization algorithm loaded earlier in Step ①. After solving the optimization problem, the *network optimization agent* replies to UB-ANC in Step ③ with the optimized results.

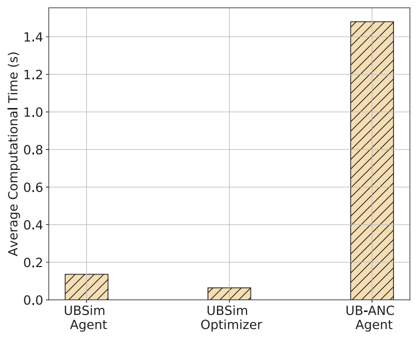
IV. Demonstration and Discussions

In this section we showcase multi-fidelity simulation based on SimSocket, considering ad hoc flying network with IEEE 802.11n as the radio access technique for UB-ANC and the wireless channels experiencing Friis propagation loss in UBSim [1].

In the first demonstration, we consider a three-node network as shown in Fig. 4(a). The intermediate node (node 2) is initially placed far away from source node 1 and destination node 3. The data rates for the nodes are controlled by the *network optimization agent* in UBSim while the actual data transmission is conducted in UB-ANC. From Figs. 4(b) and 4(c), it can be seen that node 2 moves closer to the source and destination nodes in the network and the achievable throughput increases accordingly. It is worth mentioning that Fig. 4(b) plots the average end-to-end throughput for both UBSim and UB-ANC. It can be seen that UB-ANC achieves slightly lower throughput because it takes into account the lost and corrupted packets whereas UBSim assumes capacity-approaching transmissions. This indicates that, although UBSim and UB-ANC have different levels of simulation fidelity, the transmission decisions obtained by one can still be effective for the other. This will allow us to evaluate policy learning algorithms by training them in one domain (e.g., UBSim) and testing in the other (e.g., UB-ANC), and further study the generalizability of the learning algorithms across different domains. Figure 4(c) shows the corresponding trajectories of node 2 in the Cartesian coordinate system in UBSim and in the ECEF coordinate system in UB-ANC.



(a)



(b)

Fig. 5: (a) Example topology for a multi-session, multi-hop network and (b) comparison of the network optimization and simulation time.

In the second experiment, we analyze the computational time of the three main subcomponents of the multi-fidelity simulator, i.e., UBSim Agent, UBSim Optimizer and UB-ANC Agent. We consider a multi-session multi-hop network with four nodes per session. As shown in Fig. 5(a), node 1 (and 5) and node 4 (and 8) are respectively the source (SRC) and destination nodes (DST) for Session 1 (and Session 2). Similarly, node 2 (node 6) and node 3 (node 7) are the relay (RLY) nodes for Session 1 (and Session 2). Figure 5(b) reports the simulation time experienced by each of the three modules in each simulation round. It can be seen that UB-ANC simulator takes 1.48 s per iteration while UBSim simulator and UBSim optimizer take 0.136 s and 0.064 s, respectively. This is not surprising because UB-ANC simulates the network with higher fidelity based on NS-3 and hence higher computational complexity. This will allow us to test policy learning algorithms at different time scales in different domains.

V. Enabled New Research

The proposed multi-fidelity simulator can enable a wide set of new experiments. Examples include adaptive self-configuration, accelerated policy generation, and event prediction and off-policy learning for digital twin-enabled wireless networks.

Domain adaptation. The multi-fidelity simulator can provide a novel framework for testing domain adaptation techniques, leveraging the unique domain dynamics inherent to both UB-ANC and UBSim. Since these simulators provide different levels of fidelity, we can leverage the different observable dynamics between domains, as well as the behavioral differences between simulation and hardware, to evaluate the source-to-target gap of novel domain adaptation methods.

Learning acceleration. The multi-fidelity simulator can be used to accelerate policy convergence by using the UBSim optimizer for transfer learning. First, the training of deep neural networks can be conducted quickly on a high-level implementation of a given network control problem in the UBSim virtual environment. Then, keeping the lower neural network layers unchanged by "freezing" them, the trained model can be passed through SimSocket to re-train the upper network layers in the high-fidelity UB-ANC simulator to improve the accuracy of the learned policy, hence reducing the overall training time of the neural network.

Event prediction and off-policy learning. While performing high-fidelity simulations in UB-ANC, researchers are allowed to use the parallel lower-fidelity simulation instances in UBSim for event prediction and synthetic trajectory generation. This will further allow flexible implementation of both on-policy learning in UB-ANC as well as off-policy learning in UBSim.

VI. Conclusions

In this work we have developed a new multi-fidelity simulator for wireless UAV networks by interfacing UBSim with UB-ANC. We designed a middleware called *SimSocket* for signaling exchanges between the two simulators. We showcased the multi-fidelity simulation capability of the integrated simulator considering flying ad hoc networks. The new research topics that can be enabled by the integrated simulator have also been discussed.

References

[1] S. K. Moorthy and Z. Guan, "FlyTera: Echo State Learning for Joint Access and Flight Control in THz-enabled Drone Networks," in *Proc. of IEEE International Conference on Sensing, Communication and Networking (SECON)*, Como, Italy, June 2020.

[2] H. Shen, Q. Ye, W. Zhuang, W. Shi, G. Bai, and G. Yang, "Drone-Small-Cell-Assisted Resource Slicing for 5G Uplink Radio Access Networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 7, pp. 7071–7086, July 2021.

[3] L. Bertizzolo, T. X. Tran, J. Buczek, B. Balasubramanian, R. Jana, Y. Zhou, and T. Melodia, "Streaming from the Air: Enabling Drone-sourced Video Streaming Applications on 5G Open-RAN Architectures," *IEEE Transactions on Mobile Computing*, accepted for publication.

[4] R. K. Sheshadri, E. Chai, K. Sundaresan, and S. Rangarajan, "SkyHAUL: A Self-Organizing Gigabit Network In The Sky," in *Proc. of International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc)*, Shanghai China, July 2021.

[5] O. Ozkan and M. Kaya, "UAV Routing with Genetic Algorithm Based Matheuristic for Border Security Missions," *International Journal of Optimization and Control: Theories & Applications*, vol. 11, no. 2, p. 128–138, 2021.

[6] N. Delavarpour, C. Koparan, J. Nowatzki, S. Bajwa, and X. Sun, "A Technical Study on UAV Characteristics for Precision Agriculture Applications and Associated Practical Challenges," *MDPI Journal of Remote Sensing*, vol. 13, no. 6, pp. 1–25, 2021.

[7] C. Kerr, R. Jaradat, and N. U. I. Hossain, "Battlefield Mapping by an Unmanned Aerial Vehicle Swarm: Applied Systems Engineering Processes and Architectural Considerations From System of Systems," *IEEE Access*, vol. 8, pp. 20 892–20 903, January 2020.

[8] L. Bertizzolo, S. D'Oro, L. B. L. Ferranti, E. Demirors, Z. Guan, T. Melodia, and S. Pudlewski, "SwarmControl: An Automated Distributed Control Framework for Self-Optimizing Drone Networks," in *Proc. of IEEE International Conference on Computer Communications (INFOCOM)*, Toronto, Canada, July 2020.

[9] L. Bertizzolo, E. Demirors, Z. Guan, and T. Melodia, "CoBeam: Beamforming-based Spectrum Sharing with Zero Cross-Technology Signaling for 5G Wireless Networks," in *Proc. of IEEE INFOCOM*, Toronto, Canada, July 2020.

[10] 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, USA, June 2018.

[11] A. Alkhateeb, "DeepMIMO: A Generic Deep Learning Dataset for Millimeter Wave and Massive MIMO Applications," *arXiv preprint arXiv:1902.06435*, February 2019.

[12] C. She, R. Dong, Z. Gu, Z. Hou, Y. Li, W. Hardjawana, C. Yang, L. Song, and B. Vucetic, "Deep Learning for Ultra-Reliable and Low-Latency Communications in 6G Networks," *IEEE Network*, vol. 34, no. 5, pp. 219–225, September/October 2020.

[13] P. Lieser, J. Zobel, B. Richerzhagen, and R. Steinmetz, "Simulation Platform for Unmanned Aerial Systems in Emergency Ad Hoc Networks," in *Proc. of International Conference on Information Systems for Crisis Response and Management (ISCRAM)*, Valencia, Spain, May 2019.

[14] B. Kate, J. Waterman, K. Dantu, and M. Welsh, "Simbeeotic: A Simulator and Testbed for Micro-Aerial Vehicle Swarm Experiments," in *Proc. of ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Beijing, China, April 2012.

[15] E. A. Marconato, M. Rodrigues, R. de Melo Pires, D. F. Pigatto, L. C. Q. Filho, A. S. R. Pinto, and K. C. Branco, "Avens- A Novel Flying Ad Hoc Network Simulator with Automatic Code Generation for Unmanned Aircraft System," in *Proc. of Hawaii International Conference on System Sciences (HICSS)*, Waikoloa Village, Hawaii, Jan. 2017.

[16] A. Y. Javaid, W. Sun, and M. Alam, "UAVSim: A Simulation Testbed for Unmanned Aerial Vehicle Network Cyber Security Analysis," in *Proc. of IEEE Globecom Workshops (GC Wkshps)*, Atlanta, GA, Dec. 2013.

[17] F. D'Urso, C. Santoro, and F. F. Santoro, "Integrating Heterogeneous Tools for Physical Simulation of Multi-Unmanned Aerial Vehicles," in *Proc. of WOA*, June 2018.

[18] S. Baidya, Z. Shaikh, and M. Levorato, "FlyNetSim: An Open Source Synchronized UAV Network Simulator Based on NS-3 and Ardupilot," in *Proc. of ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, Alicante, Spain, Oct. 2018.

[19] M. McManus, Z. Guan, N. Mastronarde, and S. Zou, "On the Source-to-Target Gap of Robust Double Deep Q-Learning in Digital Twin-Enabled Wireless Networks," in *Proc. of SPIE Conference Big Data IV: Learning, Analytics, and Applications*, Orlando, Florida, April 2022.

[20] J. Sleuters, Y. Li, J. Verriet, M. Velikova, and R. Doornbos, "A Digital Twin Method for Automated Behavior Analysis of Large-Scale Distributed IoT Systems," in *Proc. of Conference on System of Systems Engineering (SoSE)*, Anchorage, AK, USA, May 2019.

[21] Y. Yang, W. Meng, and S. Zhu, "A Digital Twin Simulation Platform for Multi-rotor UAV," in *Proc. of International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, Guangzhou, China, Nov. 2020.

[22] J. Modares, N. Mastronarde, and K. Dantu, "UB-ANC Emulator: An Emulation Framework for Multi-Agent Drone Networks," in *Proc. of IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, San Francisco, CA, Dec. 2016.

[23] "Simpy." [Online]. Available: https://pypi.org/project/simpy/

[24] J. Modares, N. Mastronarde, and K. Dantu, "Simulating Unmanned Aerial Vehicle Swarms with the UB-ANC Emulator," *International Journal of Micro Air Vehicles*, vol. 11, April 2019.

[25] ——, "Realistic Network Simulation in the UB-ANC Aerial Vehicle Network Emulator," in *Proc. of IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Atlanta, GA, May 2017.

[26] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "UB-ANC Planner: Energy Efficient Coverage Path Planning with Multiple Drones," in *Proc. of 2017 IEEE international conference on robotics and automation (ICRA)*, Marina Bay, Singapore, June 2017.

[27] A. Noureldin, T. B. Karamat, and J. Georgy, *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration*. Springer Science & Business Media, 2012.

[28] K. Osen, "Accurate Conversion of Earth-Fixed Earth-Centered Coordinates to Geodetic Coordinates," *PhD Dissertation, Norwegian University of Science and Technology*, Oct. 2019.

[29] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro Air Vehicle link (MAVlink) in a Nutshell: A survey," *IEEE Access*, vol. 7, pp. 87 658–87 680, June 2019.

[30] "Dronekit SITL," https://github.com/dronekit/dronekit-sitl, accessed: 2021-04-10.