This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266643

L-NORM: Learning and Network Orchestration at the Edge for Robot Connectivity and Mobility in Factory Floor Environments

Subhramoy Mohanti*, Debashri Roy*, Mark Eisen†, Dave Cavalcanti†, and Kaushik Chowdhury*
*Institute for the Wireless Internet of Things, Northeastern University and †Intel Corporation
Email: {smohanti, droy}@ece.neu.edu, {mark.eisen, dave.cavalcanti}@intel.com, krc@ece.neu.edu

Abstract—Robotic factory floors will revolutionize the future of manufacturing and the service industry by automating tasks. However, to fully supplement human effort, these robots will need low-latency, reliable connectivity throughout the work zone through links established by wireless access points (APs). This will allow the robot to assuredly respond to programming directives that rely on the real-time relaying of robot-generated sensor data to the Mobile Edge Computing (MEC) server. In this paper, we propose L-NORM, a multi-AP and multi-robot coordination framework, as a multi-tiered solution for such autonomous edge networks. First, multi-robot motion planning through reinforcement learning occurs at the MEC, using as input multi-modal robot sensor data. Second, multi-AP resource orchestration is performed using another reinforcement learning-based method that maps a subset of available APs to each robot toward meeting their sensor data delivery requirements. Furthermore, we suggest diversity combination of uplink channels with the 802.11ax scheduled access mode that will (i) support high reliability of multi-robot uplink sensor packets and (ii) enable multi-AP coordination, for optimized resource utilization. Through extensive simulation studies, we show that the probability of robot deviation to remain within 0.5 m from its optimal path, is 19% more in L-NORM compared to classical 802.11ax based edge network solution, considering ∼1 MB of sensor data per robot.

Index Terms—Edge network, orchestration, robot navigation, reinforcement learning, multi-modal data.

1 Introduction

Industry 4.0 [1], proposes rapid change to technology, industries and processes in the 21st century by fusing multi-tier computing with capabilities from artificial intelligence (AI), networked robotics, large-scale machine-to-machine communication (M2M), and the Internet of things (IoT). This integration results in increased automation, improved communication and self-monitoring, and the use of smart machines that can analyze and diagnose issues without the need for human intervention [2], [3]. However, several challenges must be addressed to realize this vision, in terms of the coexistence of a diverse set of applications with heterogeneous Quality of Service (QoS) requirements, for example in Ultra Reliable Low Latency (URLLC) applications in a dynamic, high user density environment [4]. Our proposed approach called "Learning and Network Orchestration at the Edge for Robot Connectivity and Mobility in Factory Floor Environments" (L-NORM) is a step towards enabling the vision of a smart, adaptive and scalable autonomous control system that can simultaneously enable both low-latency/high-reliability and general purpose applications. L-NORM is designed to operate with the existing state-of-the-art IEEE 802.11ax protocol with added enhancements shown in Figure 1: (1) deployed robot entities that have compatible 802.11ax radios interfaced with different sensory modules, (2) multiple distributed Wi-Fi 6 Access Points (APs), and (3) a controller residing in the MEC server that executes multi-AP coordination, multi-robot navigation and network resource orchestration in a multi-tiered fashion. This joint orchestration is a novel approach as existing resource allocation in classical 802.11ax is independent of the higher layers and does not consider multi-AP coordination and dynamic network resource orchestration.

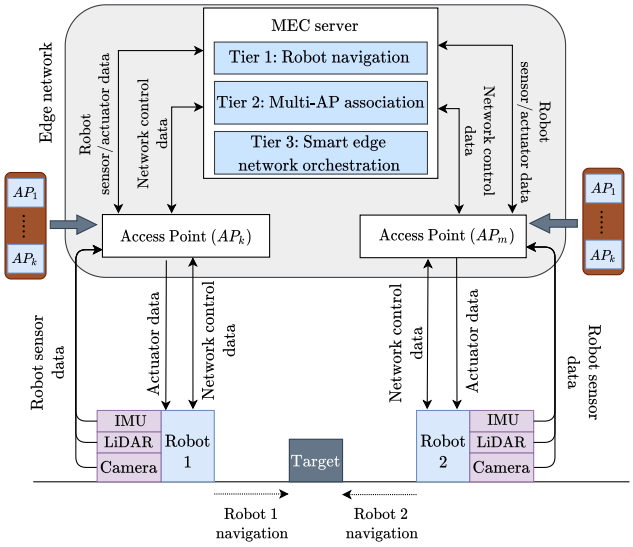


Figure 1. System architecture of L-NORM performing Reinforcement Learning (RL) enabled multi-robot navigation through autonomous edge network orchestration.

1.1 L-NORM motivation and novelty

L-NORM uses robot-generated sensor data as inputs to a navigation control algorithm that runs in the MEC, which then generates and communicates feedback in the form

of linear and angular velocity to the robots to close the loop. Existing methodologies like localized orchestration of robots and cloud-based centralized navigation algorithms [5] are not scalable and not designed to meet the typical time-sensitive URLLC robot application latency threshold ($\Upsilon_{max} \leq 5$ ms) and Packet Error Rate (PER) threshold ($\Omega_{max} \leq 10^{-3}$) [6], [7], as specified in the 3GPP Rel-16 [8]. Furthermore, when computing and wireless infrastructure resources are statically assigned in a dynamic environment, then there is a risk of inefficient operation in high user density scenarios with varying payload sizes. Existing 802.11ax protocol features of Multi-User Uplink (MU-UL) and Orthogonal Frequency Division Multiple Access (OFDMA) for high user density scenarios are leveraged by L-NORM to support our vision of the edge network as an autonomous and resilient control system. L-NORM predicts the state or context of dynamic systems/applications with the help of multi-modal sensor data in real time and learns to adapt its communications and computation resources to meet the QoS of heterogeneous applications. In this paper, we use networked robotics URLLC application as an use-case to validate the performance of L-NORM.

1.2 Conceptual overview of L-NORM operation

In L-NORM, the 802.11ax compatible APs form the edge nodes which serve as the communication bridge between the robots on the factory floor and the robot navigation and network orchestration algorithms running in the multi-tiered compute resources available to the edge network.

Difference with Classical 802.11ax: From a network architecture point of view L-NORM differs significantly compared to classical 802.11ax. In L-NORM the APs are connected through backhaul to an edge-based MEC server for dynamic multi-AP coordination and user association from multi-APs to singular users for enhanced reliability. In classical 802.11ax, the APs form a singular point of contact as gateways to the Internet for the users, that may not always provide the best connectivity in mobile scenarios for URLLC applications, thus resulting in handoffs and lowering the user QoS.

L-NORM Operation: In the L-NORM environment, individual sensors mounted on these robots generate information about the real-time state of the environment. This state information is collected by the APs, and then relayed via backhaul links to the MEC running an RL algorithm for robot navigation. The RL algorithm generates actuator feedback for each of the robots to ensure efficient navigation without any collisions. The APs also forward network control information (signal-to-noise ratio or SNR and individual AP link latency) to the orchestrator in the MEC which decides the best APs for each robot to ensure reliable robot sensor data delivery in the uplink. This method of multi-AP coordination and joint optimization of network resources with robot navigation ensures the correct interpretation of the robot's situational awareness by the robot navigation controller at the MEC server, and generate accurate robot actuator outputs for collision free navigation.

Paper Contributions: The main contributions of L-NORM, designed in a multi-tiered compute solution are as follows:

- We design an RL-based robot navigation process using multi-modal sensor data from the robots, for our *Tier 1* compute solution. For this purpose, we modify and use a deep deterministic policy gradient (DDPG) based actor-critic network which takes environment data from different types of robot sensors and generates the individual actuator feedback to the robots.
- In *Tier 2*, to enable high reliability of delivering robot sensor packets to the edge network, we propose a heuristic network orchestration algorithm. This algorithm utilizes multi-AP coordination by diversity combination of uplink channels from one robot to multiple APs through Maximum Ratio Combining (MRC). This ensures a higher probability of correctly decoding the robot sensor packets at the edge.
- We develop a DDPG-based learning framework that performs learning-based multi-AP coordination in multi-robot and multi-target scenario, as part of our *Tier 3* solution. We deploy this learning algorithm at the MEC server to complement the performance of our heuristic network orchestration algorithm when the network is operating beyond its normal operating capacity.
- We validate the proposed multi-tiered solution in a modular manner using ROS, Gazebo, and Matlab platforms, thereby ensuring the flexibility to adapt L-NORM in different scenarios and with different types of robots. We pledge to open-source the codebases of all three tiers for community use, upon the acceptance of this paper.

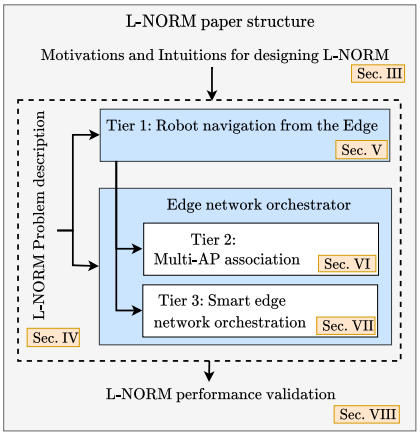


Figure 2. L-NORM components and paper organization.

The rest of the paper is organized as given in Figure 2: In Section 2, we review the relevant literature. Section 3 presents the motivations for designing L-NORM that leads to the overall problem formulation in Section 4. We provide details of the proposed solution in Sections 5, 6 and 7, respectively. We then showcase in Section 8, how these solutions, working in tandem, help us to achieve the overall optimization of Section 4. Finally, we conclude in Section 9.

2 Related Work

We review the most relevant works in the following two areas:

• **Autonomous Robot Navigation:** The authors in [9] present a simulation of Particle Swarm Optimization (PSO) based autonomous robot navigation to reach a destination

Table 1
Table of Notations

Symbol	Description	Symbol	Description
A	AP list	Ω	Packet error rate
α_i	AP 'i' in A	R_j	Robot 'j' in R
$\mathcal{M}$	Total #APs	R	Robot list
$\mathcal{K}$	Total #robots	φ	Robot collisions
Υ	Latency	y	MRC output
t	Time slot	$\mathcal{T}$	Total time to reach target
π_θ	RL policy	r	Reward space
s	State space	$\mathcal{A}$	Action space
ϑ, ψ	RL environments	ψ	RL environment
g	Target distance	l_v	Linear velocity
$\mathcal{L}$	Laser data	c	Camera data
a_v	Angular velocity	Q	Action quality
β	Delivered packets	h	Channel coefficient
w	Uplink weight	Γ	SNR
b	Received signal	x	Uplink payload
p	Transmit power	z	AWGN
δ	Obstacles	η	True position
ζ	Original position	ε	MCS
γ	RTTD	$\mathcal{U}$	MRC list of AP
Λ	Robot deviation	λ	Robot priority list
ρ	Payload duration	Ξ	Uplink transmit time
μ	Obstacle distance	ι	Target location

target in unknown environments. A multi-objective fitness function-based robot path planning was simulated in [10] to generate smooth paths and avoid obstacles. However, these works consider single robots having global knowledge of the environment and static obstacles, which is not scalable for dense robot deployments in cluttered environments. Collision avoidance with both static and dynamic objects was showcased in [11], albeit with a single robot having prior global knowledge of the environment that gets updated as the robot encounters obstacles along its way. The authors in [12] train an RL module locally on a robot for navigation in a rough terrain environment. The work presented in [13] uses RL locally on a robot for single-robot navigation while avoiding collision in an indoor environment. The RL algorithm uses real-time robot sensor data to aid in robot navigation, thus negating the need for robots to have global environment knowledge which needs to be updated every time the environment changes. This approach thus ensures scalability and comes close to our vision of autonomous robot navigation in L-NORM.

• **Wireless Edge Networking:** An extensive survey about emerging technologies for the next generation of wireless networks is presented in [14]. The authors in [15], [16], [17] discuss wireless resource allocation that are control-aware. In [18], the authors discuss the feasibility and potential of providing edge computing services to support a massive number of connected devices requesting a variety of different services such as mobile video streaming, virtual reality (VR), and augmented reality (AR), as well as mission-critical applications. with latency and reliability guarantees. The authors in [19] use the Hungarian algorithm to assign antennas to different users, but not considering different priorities among users, while the validation is limited to a static, indoor testbed environment. A Software Defined Networking (SDN) based wireless network resource orchestrator is proposed by the authors in [20] which optimizes both the networking and computational flows for efficient fog/edge/cloud-based aerial networking, however not considering heterogeneous application QoS requirements or URLLC applications. Similarly, a study of the sensitivity of traffic delay to the location of controllers and the magnitude of inter-controller and controller-node overheads in a multi-controller edge system is carried out in [21]. Mobility prediction and Intelligent handover prediction models are designed and evaluated in [22] and [23] respectively to enable live streaming edge-based applications like mobile gaming, virtual reality, etc. through MEC servers. Using predictive methods for proactive caching to improve user Quality of Experience through the alleviation of backhaul congestion is proposed in [24]. For low-latency applications an RL-based algorithm is proposed in [25] to overcome the limitations of traditional heuristic and evolutionary algorithms.

• **Distributed Reinforcement Learning:** In [26] the authors propose a distributed trustworthy storage architecture with RL in Intelligent Transportation Systems, specifically targeting edge services. The authors in [27] propose a novel hierarchical RL approach for orchestrating the dynamic placement of Virtual Network Functions (VNFs) in Cloud and Edge 5G environment to augment the current state of the art in orchestrators.

All of the above solutions are either static or specifically optimized for fixed use cases. To the best of our knowledge, there are no existing works for designing wireless networks which are context-aware and resilient enough to dynamically adapt to diverse application requirements in potentially unknown environments.

3 Motivations for Designing L-NORM

Before designing L-NORM, we first investigate the limitations in the current state-of-the-art by conducting some preliminary experiments in the domain of robot navigation and edge network orchestration. Specifically, we study the ability of the robots to maneuver around obstacles in an indoor cluttered environment and the capability of the existing IEEE 802.11ax protocol to efficiently deliver robot sensor and actuator data within the time QoS metrics. We use notations summarized in Table 1.

3.1 Current state-of-the-art in robot navigation

• **RL-based Robot Navigation:** Consider an RL based robot navigation scenario with collision avoidance presented in [13], which is most aligned with our assumption of an indoor cluttered factory floor environment. The authors in this work develop a Deep Deterministic Policy Gradient (DDPG) RL agent, that runs locally on each Turtlebot [28] robot and calculates the robot's linear and angular velocity (robot actuator data) for the next time slot based on the observed environment parameters in the current time slot. We use the same model in our preliminary experiment 'as is'. This model assumes that each robot has Odometer and LiDAR rangefinder sensors and the RL agent in each robot has partial observations in the sensing range, with 180° field of view and a range of 12 m.

• **Experiment Setup:** We perform this preliminary experiment in the Robot Operating System (ROS) simulation environment [29], which provides the robot sensor and actuator functions. These functions interface with the Gazebo simulator [30] which creates the indoor cluttered environment for the robot to move around, as shown in Figure 3. The robot navigation RL agent is instantiated in OpenAI [31], which

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266643

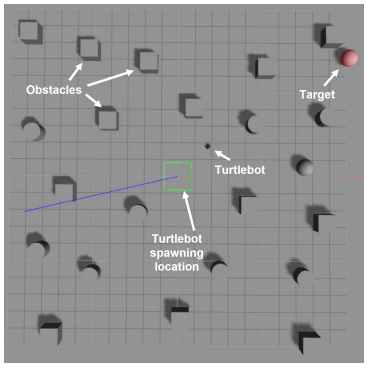


Figure 3. Robot navigation simulation in ROS and Gazebo generated indoor cluttered environment consisting of obstacles, Turtlebot and target destination, for testing autonomous robot navigation.

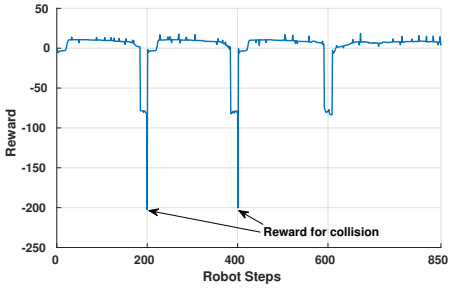


Figure 4. Limitation of the current state-of-the-art [13] robot navigation performance with DDPG in an indoor cluttered environment. Frequent collisions with obstacles can be inferred from the collected negative rewards. Here, the number of collisions, $\varphi = 2$.

interfaces with the ROS functions through Gym wrapper [32]. This simulation environment spans an area of 20 sq. m. with 22 randomly placed obstacles and 10 robots, each navigating to their unique randomly placed target locations. The RL agent running on the robot learns to move the robot from different starting locations to a randomly located target location with the minimum number of steps while not colliding with any obstacles. The DDPG RL agent is trained for over 500 trials with 2000 robot steps per trial.

• **Observation:** We measure the performance of the RL agent in the test phase in the same environment with randomized initial robot and target locations. Since in our experiments, all of the 10 robots faced collisions while navigating to the target locations, we show one robot's result here for ease of understanding. The robot's reward during test is shown in Figure 4. The -200 reward is collected by the robot when it collides with any object, resulting in the condition $\varphi > 0$, where φ is the robot collision metric. We observe that the RL agent performs sub-optimally, resulting in multiple robot collisions.

Motivation 1. *We need to fine-tune the existing DDPG-based robot navigation RL agent to ensure zero collisions ($\varphi = 0$) and also to customize it for multi-robot and multi-target scenario. We explain these modifications in Section 5.*

3.2 Classical 802.11ax for edge-based robot navigation

In this scenario, the robot sensor data is communicated to an edge-based robot navigation algorithm which then generates the movement feedback to the robots. We evaluate whether the state-of-the-art IEEE 802.11ax protocol is effective for enabling time-sensitive multi-robot navigation

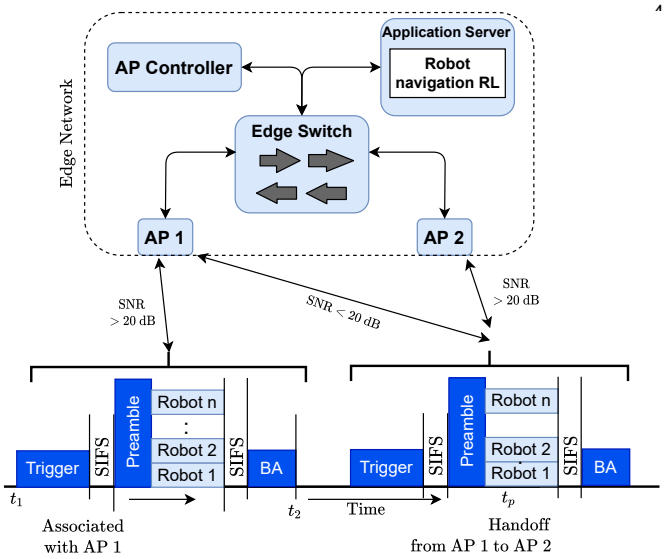


Figure 5. Current state-of-the-art edge network architecture using classical IEEE 802.11ax protocol with singular AP association per robot and network manager-initiated handovers. This figure shows 1 uplink frame at every time t for ease of understanding.

and orchestration. At the time of writing this paper, typical robot navigation simulators do not have wireless network implementations. For this reason, we plug in MATLAB WLAN Toolbox with ROS to perform our network simulation studies in this paper.

• **Experiment Setup:** Our Matlab simulation takes in the timestamped robot location data from the ROS simulation environment. In this simulation, we place APs in uniform grid locations in a 60 sq. m. area and simulate the robot movement in that area by feeding the ROS (Application Server) generated robot location to Matlab's Differential Drive Kinematics (DDK) model [33] to make the robot in Matlab follow the exact path (optimal path) it took in the ROS simulation. The DDK model generates the robot actuator (wheel motor) inputs in terms of linear and angular velocity based on the current robot location and ROS-mandated next waypoint ahead, in order to stay in the optimal path. This actuator data generation is influenced by the wireless network quality since in this scenario the robot motion control directives are generated at the edge based on the accuracy of robot sensor data delivery. The robot network interface is provided by the classical 802.11ax edge network architecture as shown in Figure 5, through the MATLAB WLAN toolbox. For context, the state-of-the-art multi-user uplink management for 802.11ax, which is done through the MU-UL mechanism, is shown in Figure 6. In the robot navigation scenario, we consider a 20 MHz BW which can support a maximum of 9 robots, each with 2 MHz resource unit (RU) allocation, per Physical Layer Protocol Data Unit (PPDU). The rest of the available BW for 802.11ax, we consider being used for general-purpose applications/background traffic. Here, the PPDU containing multi-user data packets is preceded by a Trigger Frame (TF) and followed by a Block Acknowledgement (BA) frame. We use a 5 GHz operating frequency with IEEE indoor channel model B in SISO mode, with the AP transmit power set at 25 mW. The rest of the available BW is reserved for general-purpose Wi-Fi-based applications. For the sake of simplicity, we assume a homogeneous sensor packet arrival rate across all robots at every t =10 ms. This time slot t is divided

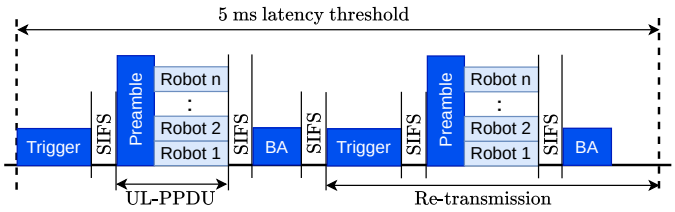


Figure 6. Classical IEEE 802.11ax MU-UL mechanism showcasing the distribution of overhead and data in a typical transmission.

into uplink and downlink thresholds of 5 ms each. We break down the total time $\mathcal{T}$ taken by the robot to reach its target destination into the granularity of t=10 ms time slots, and measure the performance in each of these t time slots. The robot associates with the AP having the highest SNR to transmit the sensor packets to the MEC server. Following the classical Wi-Fi association rules, the robot keeps its current association alive till the SNR degrades below a threshold of 20 dB. After this, the robot initiates the classical steps of AP de-association and re-association to the next available AP with the highest SNR. During this handoff time period, (which typically ranges from 50 ms to ∼100 ms [34] based on the AP vendor), all the data packets generated by the robot are dropped due to the absence of any connected AP to the robot.

• **Simulation Parameters:** Considering the multitude of sensor data originating from the robots, and the fact that the edge-generated actuator feedback is smaller in size (float point data representing linear and angular velocity) compared to sensor data, we monitor the uplink network characteristics between the robot and the AP in this experiment. Following the classical 802.11ax protocol, at every time slot of this simulation study, the least conservative MCS for the uplink is calculated based on the SNR to the associated AP, such that the PER remains within Ω_{max} (10^{-3}). This translates to the uplink payload duration which should be within the uplink latency threshold Υ_{max} (5 ms).

• **Observation:** Since the AP handoff process (re-association with another AP) in classical Wi-Fi is initiated by the client side (robots in our case) and not by the robot navigation RL application on the edge, we observe that the robot sensor data uplink QoS crosses the threshold for both PER and latency multiple times. Due to the dropped sensor packets, the navigation RL agent cannot generate the optimal actuator feedback. Thus, the robots start to deviate from their optimal path, i.e., the path the robot takes when there is no network-related uplink packet error. This increases the probability of $\varphi > 0$, i.e., the robot suffering a collision. In this experiment, the robot's deviation, shown in Figure 7, is ∼3-4 m from the optimal path. Moreover, in high robot density scenarios with classical single-AP deployments, 802.11ax is unable to meet the low latency uplink requirements [35].

Motivation 2. *Even if the edge-based robot navigation RL algorithm is performing optimally, the current state-of-the-art 802.11ax edge network design and architecture, and not the protocol itself, will impact the stringent time-sensitive robot application QoS requirements. This will result in multiple robot collisions in high robot density scenarios and cluttered environments. This motivates the need for an improved multi-AP Wi-Fi based edge network design.*

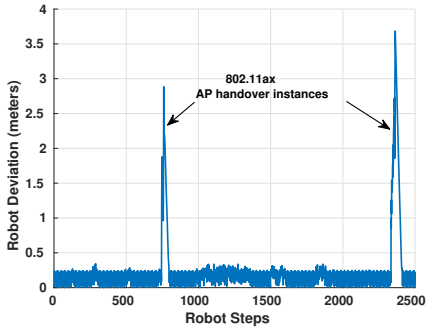


Figure 7. Limitation of current state-of-the-art IEEE 802.11ax [36] based AP association in edge deployment. The robot deviated by 3-4 m from the optimal path during AP handover instances, increasing the probability of robot collisions ($\varphi > 0$).

Goal of L-NORM: Considering the Motivations 1 and 2, our proposed work is a step towards enabling the vision of a smart, adaptive and scalable edge solution that can simultaneously enable both low-latency/high-reliability and general-purpose applications.

- In the context of Motivation 1, we model the edge as an autonomous control system that predicts the state or context of dynamic systems/applications (e.g., robot navigation) in real-time with the help of multi-sensory data.
- In the context of Motivation 2, we design the edge network such that it learns to adapt its communication and computation resources to meet the QoS for heterogeneous applications.

4 L-NORM Problem Description

4.1 Problem statement

We consider $\mathcal{K}$ robots and $\mathcal{M}$ APs in the environment. We define vector R which represents all the robots, hence $R = \{j | 1 \leq j \leq \mathcal{K}\}$. Similarly, the vector to represent APs is $A = \{i | 1 \leq i \leq \mathcal{M}\}$. The latency and PER threshold are parameterized as Υ_{max} and Ω_{max}, respectively. The time taken by the robot R_j to reach its destination is $\mathcal{T}_j$, which has a granularity of t=10 ms time slot, based on the robot sensor packet arrival rate. Accordingly, to reduce the number of collisions, the optimization problem $\mathbb{P}$ is to minimize the deviation Λ_j^t of robot R_j at time slot t from its optimal path, while utilizing the minimum network resources (APs). Λ_j^t is calculated as the Euclidean distance between the robot's location when sensor packets experience network errors and when they are not corrupted at time slot t. We formally define this as:

Problem $\mathbb{P}$ **–** *Given a specific set of robots R, with a minimum number of edge network resources (APs), ensure zero collisions (φ) for each robot and minimal deviation (Λ) of each robot from its optimal path, with minimum network resource ($\mathcal{U}$) utilization, while satisfying the robot sensor application QoS thresholds, viz., PER (Ω) and latency (Υ).*

4.2 Solution approach

From our observations in Section 3, we realize that deviation Λ_j^t depends on the efficiency of the (i)robot navigation algorithm, and (ii) the edge network in its ability to deliver the robot sensor packets. We thus solve $\mathbb{P}$ in a modular fashion, by breaking it down to the sub-problems $\mathbb{P}_0$ and $\mathbb{P}_1$.

$\mathbb{P}_0$ is for efficient robot navigation while avoiding collisions, and $\mathbb{P}_1$ is for efficient edge network orchestration.

- ***Problem*** $\mathbb{P}_0$– *Develop an efficient multi-robot orchestrator, that can ensure each of the robots reaches their intended target location with the minimum number of steps and with no collisions, i.e.,* $\varphi = 0$ *(optimal trajectory), in an indoor cluttered environment. [Details in Section 5]*

After solving Problem $\mathbb{P}_0$, our aim is to make the robot orchestrator perform with the same efficiency from the edge, as it was performing in absence of any wireless network-related packet errors. For this purpose, the edge network resources need to be orchestrated as well, because, the robot orchestrator can generate the correct next-step actions for the robots, only if the robot sensor packets are delivered to it by the edge network in a correct and timely manner. This leads to the second problem,

- ***Problem*** $\mathbb{P}_1$– *Ensure the edge network can deliver each of the robot's sensor data to the edge-based multi-robot orchestrator with the minimum network resources (APs), while satisfying the robot sensor application QoS thresholds, viz., PER* (Ω) *and latency* (Υ)*, for the correct formulation and generation of the next step actions for each robot. This will result in minimum robot deviation* (Λ) *from its optimal trajectory, thus ensuring no collisions, i.e.,* $\varphi = 0$*. [Details in Section 6]*

In the rest of the paper, we show how a joint solution for these two problems enables the realization of a dynamic and adaptive edge-based resource orchestrator.

5 Tier 1: Robot Navigation from the Edge

We first focus on improving the corresponding DDPG RL framework and making it capable of running from the edge, instead of running locally on the robots, since our preliminary experiments in Section 3.1 proved that the "local only" solution in [13] does not scale for zero-collisions in a highly cluttered environment with high user density. Even if we wanted to use a "local-only" approach, the robots needed to share their neighborhood information perceived by their sensors for the RL-agents in the robots to learn optimally. This would have resulted in a robot mesh network type solution which has its own pros and cons like (i) flooding the network with control information, and (ii) design of a better mesh networking protocol (which is out of scope of this paper). Specifically, our target is to make the RL agent more efficient and capable of working with multi-robot sensor information and coordinate each of the robot's actuating functions with the goal of enabling the robots to reach their respective destinations with no collisions.

Motivation for choosing DDPG: DDPG is an actor-critic and model-free algorithm that can operate over continuous action and state spaces. In our multi-robot motion in indoor obstacle-cluttered factory environments, continuous linear and angular velocity (action) is better than discrete values and ensures zero collisions for the robots. Since the focus of our paper is not on proposing a better DRL architecture or algorithm but on the joint orchestration of multi-robot navigation and edge networking nodes, we modified the SoA DDPG model from [13] and then adapted it to fit into our edge-based MEC server, so that it accepts multiple robot sensor data as inputs to generate multi-robot actuator commands as output.

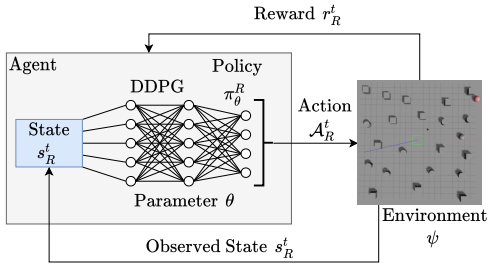


Figure 8. Robot navigation DDPG RL agent architecture. The agent is interacting with the environment based on the policy, robot state, robot action, and the collected reward.

Problem Formulation for $\mathbb{P}_0$ – To reach the goal of zero collisions $\varphi = 0$, we aim to derive an optimal reinforcement learning policy π_θ^R for robot navigation. Therefore, given the total time to reach destination is $\mathcal{T}_j$ for robot R_j, we formulate the robot navigation problem $\mathbb{P}_0$ as:

$$\mathbb{P}_0 : \max_{\pi_\theta^R} \mathbb{E}_{\pi_\theta^R}\Big[\sum_{t\in\mathcal{T}_j}[r_R^t]\Big] \tag{1a}$$

$$\text{s.t.} \quad \mathbb{E}_{\pi_\theta^R}\Big[\frac{1}{\mathcal{T}_j}\sum_{t\in\mathcal{T}_j}[r_R^t]\Big] \geq \mathcal{R}_{\min}^R \tag{1b}$$

$$\sum_{j=1}^{R}\sum_{t=1}^{\mathcal{T}_j}\varphi_j^t = 0 \tag{1c}$$

where r_R^t is the generated reward from robot state s_R^t and action $\mathcal{A}_R^t$ at time slot t for all the robots in the set R. The minimum reward threshold for robot motion environment ψ is defined as $\mathcal{R}_{\min}^R$. In L-NORM, the $\mathcal{R}_{\min}^R$ is kept at -200, which is the reward for collision and when the robot gets to within 0.25 m from any obstacle. We next present our solution approach to this problem by defining our designed DDPG RL agent.

Solution Approach: The overall framework of the designed DDPG RL agent for solving problem $\mathbb{P}_0$ is showcased in Figure 8. The RL agent tunes its policy with the ultimate goal of making the robot reach the target in a minimum number of steps without any collisions. The policy π_θ^R is updated at every time slot t and applied to the actor-critic network, based on the collected reward r_R^t and the observed state s_R^t of the robot, which is generated by the environment ψ after the application of the action $\mathcal{A}_R^t$ on it. We next explain these state, action, and reward spaces, along with the actor-critic network and the environment.

State Space: We define the state space s_R^t as the combination of the multi-modal sensing data (laser ($\mathcal{L}^t$), camera (c^t)), relative target position (g^t) at current time slot t, and velocity (v^{t-1}) from odometer sensor at the previous time slot $t-1$ for all the robots in set R. Overall, s_R^t is formulated as $s_R^t = \{\mathcal{L}^t, c^t, v^{t-1}, g^t\}$, where $\mathcal{L}^t$ is a vector and c^t is a 2D matrix, each of specific dimensions based on user requirement. The v^{t-1} constitute two values corresponding to the linear and angular velocity. We defined v^{t-1} as: $\{l_v^{t-1}, a_v^{t-1}\}$, where l_v^{t-1} and a_v^{t-1} are the linear and angular velocities, respectively, at time slot $t-1$. The g^t is a tuple of dimension 2 corresponding to the 2D coordinates of the target location.

Action Space: The action is designed to allow the DDPG agent to generate the angular and linear velocity of the

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266643

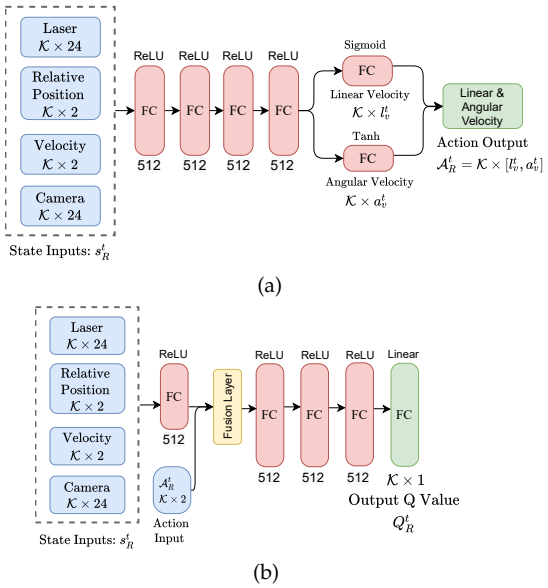


Figure 9. (a) Actor and (b) Critic modeling in DDPG neural network for multi-robot to multi-target navigation with $\mathcal{K}$ robots.

current time slot t. We define the action space $\mathcal{A}_R^t$ for all the robots in R ($|R| = \mathcal{K}$), as the combination of l_v^t and a_v^t, where l_v^t is the linear velocity, and a_v^t is the angular velocity at time slot t, hence: $\mathcal{A}_R^t = \mathcal{K} \times \{l_v^t, a_v^t\}$. Following the principle of reinforcement learning, the action $\mathcal{A}_R^t$ is generated by the designed policy π_θ^R from the state s_R^t. Hence, the action of time slot t is formulated as $A_R^t = \pi_\theta^R(s_R^t)$. We design the policy π_θ^R using the DDPG actor-critic network modeling for solving $\mathbb{P}_0$.

Actor-Critic Network: In an actor-critic model, the actor network learns the optimal policy over time to generate the actions from the state space. Therefore, the optimal policy is determined by the parameters of the trained actor network. To develop the training method for policy π_θ^R which corresponds to solving the problem stated in Equation 1a, we parameterize the actor network as $f_{\theta_R^A}^A$ where $\pi_\theta^R = f_{\theta_R^A}^A$. Next, we formulate the actor network as:

$$\mathcal{A}_R^t = f_{\theta_R^A}^A(s_R^t), \qquad f_{\theta_R^A}^A : \mathbb{R}^{\mathcal{K} \times (|\mathcal{L}^t| + |c^t| + |v^{t-1}| + |g^t|)} \mapsto \mathbb{R}^{\mathcal{K} \times 2}$$

where the action output for all robots R at time slot t, is represented by a matrix $\mathcal{A}_R^t = \mathcal{K} \times \{l_v^t, a_v^t\}$, yielding to an output space of $\mathcal{K} \times 2$. The critic network takes both the state and the generated action from the actor as inputs and determines the quality of the generated action. The critic model is formulated as:

$$\mathrm{Q}_R^t = f_{\theta_R^C}^C(s_R^t, \mathcal{A}_R^t), \qquad f_{\theta_R^C}^C : \mathbb{R}^{\mathcal{K} \times (|\mathcal{L}^t| + |c^t| + |v^{t-1}| + |g^t| + 2)} \mapsto \mathbb{R}^{\mathcal{K} \times 1}$$

where Q_R^t is the Q value generated by the critic at time slot t evaluating the success of the generated action $\mathcal{A}_R^t$, and critic network is parameterized with $f_{\theta_R^C}^C$. Here Q is a function in the DDPG algorithm, which generates the expected rewards for an action taken in a given state. Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state [37]. To facilitate the the operation of L-NORM in a dynamic environment with random number of robots at any given instant, we design the actor critic networks to be adaptable to the variable shape of the input state space. However, the shapes of the intermediate FC layers are fixed and their trained weights are used for generating action output during the test time.

Table 2
Robot reward details

Symbol	Reward Description	Reward Value
$r_R^{g^t}$	Relative distance to target	Euclidean distance (m)
$r_R^{col^t}$	Collision	-200
$r_R^{a_v^t}$	Angular velocity (rad/s)	0 (if $-0.8 < a_v^t < 0.8$) -1 (if $a_v^t \leq -0.8$ or $a_v^t \geq 0.8$)
$r_R^{l_v^t}$	Linear velocity (m/s)	-2 (if $l_v^t \leq 0.2$) 0 (if $l_v^t > 0.2$)
$r_R^{arr^t}$	Reaching target	200
$r_R^{obs^t}$	Distance to obstacle (m), μ^t	0 (if $\mu^t > 0.5$) -200 (if $\mu^t < 0.25$) -80 (if $0.25 \geq \mu^t \leq 0.5$)

Reward Space: We define the reward space as for each robot: $f_{\mathcal{A}}^{\mathcal{R}}(s_R^t, \mathcal{A}_R^t) = -r_R^{g^t} + r_R^{col^t} + r_R^{a_v^t} + r_R^{l_v^t} + r_R^{arr^t} + r_R^{obs^t} = r_R^t$, where $f_{\mathcal{A}}^{\mathcal{R}}(.)$ is the function to generate the rewards over action $\mathcal{A}_R^t$ and state s_R^t at time slot t. The total reward of this RL agent is denoted as $\mathcal{K} \times f_{\mathcal{A}}^{\mathcal{R}}(s_R^t, \mathcal{A}_R^t)$. The individual reward descriptions are given in Table 2.

Environment: Finally the environment is parameterized as $s_R^{t+1}, r_R^{t+1} = \psi(\mathcal{A}_R^t)$, where the environment ψ takes the inputs of the generated action by the actor-critic network at time slot t and generates the next state and rewards for time slot $t+1$.

5.1 Robot navigation RL network hyperparameter tuning

The final version of our modified DDPG robot navigation RL actor-critic model with tuned hyper-parameters for multiple robots is shown in Figures 9(a) and 9(b) respectively. For the sake of simplicity, the input layers in both the actor and critic networks are not shown. To train this RL agent under realistic conditions, we add simulated AWGN noise for IEEE indoor channel model B at 5 GHz frequency to the laser sensor data before feeding this data to the Fully Connected (FC) layers. An extra camera sensor, generating point cloud data is also added on the robots, to improve the RL functionalities. The laser and camera sensor, with a 180° field of view, divides the scanning environment into 24 sectors. This shapes the input dimensions of the camera and laser sensor data to the neural network. The number of FC layers, each with ReLU activation function, is increased from 3 to 4, and the number of neurons per layer is increased from 500 to 512. The neural network is trained using Adam optimizer with a learning rate of 0.0001 and mean squared error loss function. The actor network generates the linear velocity command through a sigmoid function and produces the angular velocity using a hyperbolic tangent function. The critic network uses the state of the robot (s_R^t) as input, which is then passed through an FC layer with 512 nodes. The output of this layer is then merged with the action input $(\mathcal{A}_R^t)$, after which the data is processed by three dense layers. The Q-value is finally generated by a linear activation function.

5.2 Performance validation

The modified RL network is trained in the same ROS simulation environment parameters as in Section 3, albeit with random robots and targets spawning at random locations, at each training iteration.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266643

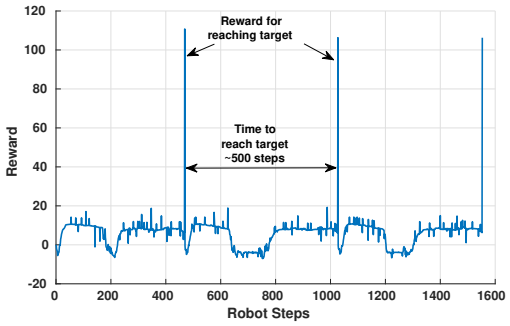


Figure 10. Performance of DDPG with multi-modal sensor data during testing for multi-robot navigation from the edge in an indoor cluttered environment under perfect network conditions, assuming 0 PER and no AP handovers. Rewards >100 signify the robot reached the target destination. The absence of negative rewards proves the robots suffered no collisions ($\varphi = 0$).

• **Computational Complexity:** In terms of the computational complexity of L-NORM robot navigation from the edge MEC server, the policy is trained end-to-end and it maps discrete lidar, velocity, camera information, and relative goal positions from the $\mathcal{K}$ available robots into action command directly instead of using the whole environmental map. The computational complexity of the actor neural network can be denoted as $O(S_{a_{in}}F_{a1} + F_{a1}F_{a2} + F_{a2}F_{a3} + F_{a3}F_{a4} + F_{a4}S_{a_{out}})$, where $S_{a_{in}}$, $S_{a_{out}}$, where $S_{a_{in}}$, $S_{a_{out}}$ denote the dimension of the input layer and output layer for actor neural network, respectively. F_{a1}, F_{a2}, F_{a3}, and F_{a4} represent the dimension of four fully connected layers for actor network, respectively. As shown in Figure 8b, the computational complexity of the critic neural network can be represented by $O(S_{c_{in}}F_{c1} + (F_{c1} + S_{a_{out}})F_{c2} + F_{c2}F_{c3} + F_{c3}F_{c4} + F_{c4}S_{c_{out}})$, where $S_{c_{in}}$ and $S_{c_{out}}$ represent the dimension of the input layer and output layer for critic network, respectively. F_{c1}, F_{c2}, F_{c3}, and F_{c4} denote the dimension of the four fully connected layers for the critic network, respectively.

• **Ideal Wireless Network Scenario:** First, we need to validate the hypothesis that a robot navigation RL network works without collisions under perfect wireless network conditions, where the DDPG RL agent for robot navigation resides in the MEC server and the sensor data from the robots is delivered to the MEC server through 802.11ax protocol. We assume an ideal channel with no packet loss and no AP handovers. From Figure 10, we see that the robots collect the reward of 110 for reaching the target destination multiple times. The absence of the -200 reward for collision proves that the robots trained on this RL network are able to successfully maneuver around obstacles (including other robots), thereby satisfying the condition of zero collisions ($\varphi = 0$), while navigating to the target destination.

• **Realistic Wireless Network Scenario:** Next, we investigate the performance of multi-robot navigation when the robots' sensor data is transmitted through the wireless network to the successfully trained RL agent, which is located at the edge. Here we use the IEEE indoor channel model B for factory environments and AP handovers, which result in robot sensor data packet loss. The experiment configuration from Section 3.B is repeated for this purpose, but with the improved robot navigation RL agent providing the robot locations to the Matlab-based edge network simulation. Figure 11 plots the robot movements directed by the edge

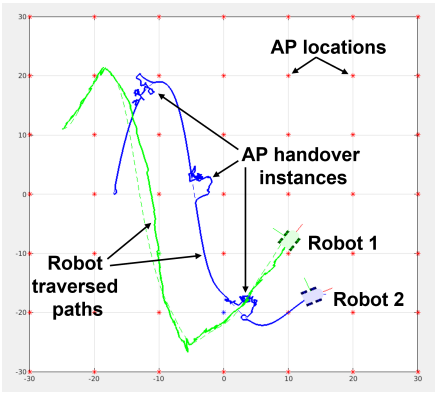


Figure 11. Performance of DDPG with customized hyper-parameter tuning and multi-modal sensor data for multi-robot motion orchestration from the edge with classical 802.11ax in an indoor environment. The robots deviated significantly from their optimal path (dashed line) during AP handover instances.

based RL algorithm under perfect network conditions with no packet drops (dashed line) and in the classical 802.11ax edge network (solid line). The robots diverge significantly from their optimal path (the dashed line) under the classical 802.11ax protocol due to the AP de-associations and re-associations (handovers). During this handover time, all the sensor packets from the robots are dropped by the network, resulting in undesired deviation traced in Figure 11 with a solid line.

The degraded robot navigation performance in a realistic wireless scenario, together with Motivation 2, lead us to develop a dynamic edge network orchestration mechanism based on the existing 802.11ax protocol as part of our Tier 2 solution. This design should not only enable edge-based robot navigation but also can meet the diverse QoS requirements of applications ranging from low-latency/high-reliability to general purpose. We discuss these in detail in the next section.

6 Tier 2: Multi-AP Association from the Edge

The 802.11ax-based edge network experiments in Section 3.2 (Figure 7) and Section 5.2 (Figure 11) prove that from a networking perspective, if the successfully delivered robot uplink data is maximized while maintaining the networked robotics application QoS requirements of PER and latency, then this will have a direct impact on minimizing the probability of robot deviation. Moreover, these experiments show that the re-association feature (handoff from one AP to another AP) in classical Wi-Fi is one of the main bottlenecks for URLLC communications in high user-density scenarios. Given this shortcoming, the next-generation wireless network protocols like IEEE 802.11be (Wi-Fi 7) are considering breakthrough technologies like multi-Access Point (AP) coordination [35], [38], [39] as candidate solutions. Ensuring that each receiver's latency thresholds are satisfied requires timely and correct delivery of packets, which can be ensured by increasing the SNR at the receiver, while carefully associating the users/robots with the best possible APs/edge nodes [40], [41]. By tying this classic networking problem with robot deviation, we have also generalized the problem $\mathbb{P}_1$, because the performance metric of any application can be measured by the amount of successfully delivered packets to the destination while

conforming to the QoS requirements (PER, latency, etc.) In this section, we first formulate the problem $\mathbb{P}_1$ and evaluate the key reasons for the performance limitations of existing state-of-the-art, and then propose enhancements to the 802.11ax protocol to enable multi-AP coordination and joint OFDMA scheduling.

Problem Formulation for $\mathbb{P}_1$ – The objective of problem $\mathbb{P}_1$ is to minimize the deviation of each robot from the optimal path, while maintaining the network QoS requirement of Ω_{max} and Υ_{max}, at every time slot t. This can be achieved if the robot uplink sensor packets are delivered correctly in a timely manner. Specifically, we maximize the uplink packet delivery ratio in the edge network. Assuming, at every time slot t, every robot R_j generates one data packet. The successfully delivered packet being $R_j = \beta_j$, then for $\mathcal{K}$ robots, for a total of $\mathcal{K}$ data packets generated at every t, the problem $\mathbb{P}_1$ can be formulated as:

$$\mathbb{P}_1 : \max \left(\frac{\sum_{j=1}^{\mathcal{K}} \beta_j}{\mathcal{K}} \right) \tag{2a}$$

$$\text{s.t.} \quad \Omega(R_j) \leq \Omega_{max} \tag{2b}$$

$$\Upsilon(R_j) \leq \Upsilon_{max} \tag{2c}$$

where $1 \leq j \leq \mathcal{K}$, $\Omega(.)$ calculates the packet error rate, and $\Upsilon(.)$ calculates the latency.

6.1 Limitations of 802.11 in mobile URLLC applications

In the robot navigation scenario, we assume the Wi-Fi network operates in infrastructure mode, where the AP bridges all data from the users associated to it in the Basic Service Set (BSS) and provides a communication link with the Internet. A handoff occurs when a user moves beyond the radio range of one AP and enters another BSS, and initiates the re-association procedure with the AP in the new BSS. During this process, management and authentication control messages are exchanged between the user and the AP. As a consequence of this action, the AP and the user are unable to exchange data traffic until the re-association process has been completed, and this increases the latency of data communication. An empirical study by the Mishra *et al.* [34] regarding the Wi-Fi handoff impact on latency in mobile applications show that not only are the latencies quite high (50-100 ms) but it also varies significantly for the same configuration of APs and client users.

6.2 Enhancing 802.11ax MU-UL with MRC

Given the limitations of single AP-based infrastructure mode operation in high user density with low latency application scenarios, we consider an edge-based multi-AP coordination strategy for optimal allocation of AP resources to robots/users on the ground. Specifically, we investigate the implementation of Maximum Ratio Combining (MRC) on the uplink, which has traditionally been deployed in cellular networks and has shown potential for improvement in existing Wi-Fi protocols [42], [43].

• **Maximum Ratio Combining (MRC):** Consider a robot R_j transmitting uplink multi-modal sensor (e.g., camera, laser, odometer) data. Let $A = [\alpha_i | 1 \leq i \leq \mathcal{M}]$ be the APs within

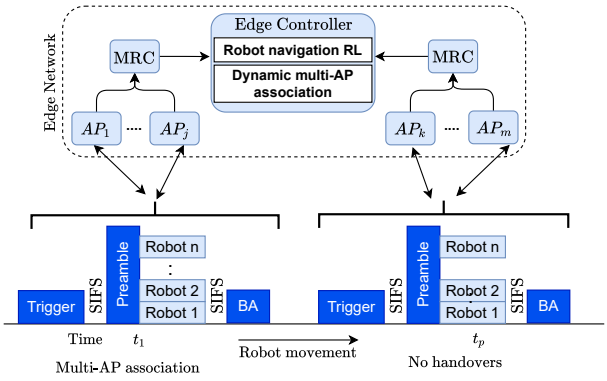


Figure 12. IEEE 802.11ax architecture enabled with MRC for MU-UL. Robots can associate with multiple APs at a time, removing the need for handover. MRC algorithm at the edge processes the data and forwards it to edge-based controller. A single uplink frame per time t is shown here for ease of understanding.

the communication range of the robot R_j. At time slot t, the signal received at each AP α_i, can be represented as:

$$b^t_{\alpha_i} = h^t_{R_j,\alpha_i} \sqrt{p^t_{R_j}} x^t_{R_j} + z^t_{\alpha_i} \tag{3}$$

where h_{R_j,α_i} is the channel coefficient from robot R_j to AP α_i, p_{R_j} is the transmission power of robot R_j, x_{R_j} is the transmitted symbol, and z_{α_i} is the additive white Gaussian noise with variance σ^2 at the AP α_i. MRC assigns weights to each of the AP links from the robot R_j and adds the branches with the best weights at the MEC for maximum SNR as:

$$y^t_{mrc} = \sum_{i=1}^{\mathcal{M}} w^t_i b^t_{\alpha_i} \tag{4}$$

where w is the weight for the Robot-AP uplink, typically estimated from the channel gain conjugate [44]:

$$\begin{aligned} y^t_{mrc} = \sum_{i=1}^{\mathcal{M}} \mathcal{V}_i e^{-\zeta\phi_i} b^t_{\alpha_i} &= \sum_{i=1}^{\mathcal{M}} \mathcal{V}_i e^{-\zeta\phi_i} [\mathcal{V}_i e^{-\zeta\phi_i} x^t_{R_j} + z^t_{\alpha_i}] \\ &= \left(\sum_{i=1}^{\mathcal{M}} \mathcal{V}_i^2 \right) x^t_{R_j} + \sum_{i=1}^{\mathcal{M}} \mathcal{V}_i e^{-\zeta\phi_i} z^t_{\alpha_i} \end{aligned} \tag{5}$$

Here, $\mathcal{V}$ is the amplitude of the signal. The received SNR after MRC thus becomes:

$$\mathbf{\Gamma}^t_{mrc} = \frac{\sum_{i=1}^{\mathcal{M}} \mathcal{V}_i^2 \mathcal{E}_b}{\mathcal{N}_0} \tag{6}$$

where, $\mathcal{E}_b$ is the energy of one bit and $\mathcal{N}_0$ is the noise spectral density.

• **MRC-enabled 802.11ax Architecture:** As shown in Figure 12, our proposed MRC-enabled 802.11ax edge network architecture consists of multiple APs (edge nodes) connected via wired backhaul to a Controller Unit (CU) running the MRC-based AP resource allocation algorithm and the robot navigation RL algorithm. Considering the robot sensors' uplink packet arrival rate at every 10 ms, with MRC, the uplink transmission from a robot will be heard by multiple APs. Since the robots are mobile, the link quality from each robot to the APs will also vary over time. Given these considerations, the controller will assign weights to each robot-AP branch based on the real-time link quality and add the branches with the best weights for maximum SNR and better decoding probability, compared

to a single AP scenario. This increases the chance of better decoding probability, even if some of the links have low SNR compared to others.

• **MRC advantage in URLLC:** Through this method, the robot sensor data can be transmitted with a higher MCS than traditional 802.11ax, since the chance of correct decoding at the edge is now higher. This translates to higher throughput, lower payload duration, and hence more time for retransmissions in case of packet error, thereby increasing the reliability while remaining within the latency threshold.

• **MRC constraints in 802.11ax:** Even though increasing the number of branches increases the probability of successful decoding, it also increases the probability of backhaul traffic congestion at the edge. Also, the APs which are far away from the robot's location will not contribute significantly to the MRC approach, due to its low SNR condition. The controller thus has to choose an optimal subset of APs for applying MRC to each robot in the network, such that the Υ_{max} and Ω_{max} per robot are satisfied. Also, to enable MRC with multiple distributed APs, accurate phase and frequency synchronization is needed between the AP nodes, for correct packet detection and channel estimation, an example of which is shown in [45]. There also exist hardware-based solutions like the clock distribution module, Ettus Octoclock [46] for achieving accurate synchronization between distributed APs and other network nodes, through the backhaul, as done in [47].

6.3 Edge network orchestration: solution to problem $\mathbb{P}_1$

We next address the network resource orchestration for assigning multiple APs for MRC and solve the problem $\mathbb{P}_1$ by proposing a heuristic edge resource (AP) orchestration Algorithm 1. This algorithm decides which APs will be associated with which robots for performing MRC, in order to have minimum deviation of the robots from their optimal path, which ultimately will ensure zero collisions ($\varphi = 0$).

• **Algorithm Initialization:** In the indoor cluttered factory environment, at every time slot t, there are $R = [R_j | 1 \leq j \leq \mathcal{K}]$ robots with their sensor packets for uplink transmission to the MEC server, via the $A = [\alpha_i | 1 \leq i \leq \mathcal{M}]$ APs. Given the overall latency bound $\Upsilon_{total} = 10$ ms, as the time from the origination of the robot sensor packets to the time when the robot actuator feedback reaches back to the robots, each time slot t is designed to have a duration of Υ_{total}. The robot sensor packets are considered to have a periodic and isochronous packet arrival pattern, arriving at every t time slot. In order for the robots to get the correct actuator feedback, it is imperative for the sensor data to be delivered to the MEC server while maintaining the network-mandated uplink QoS metrics ($\Upsilon_{max} \leq 5\,ms$ and $\Omega_{max} \leq 10^{-3}$), and ensure zero robot collisions in the process ($\varphi = 0$).

• **Dynamic Multi-AP Association:** Inferring from the robots' true location $\boldsymbol{\eta}$ and the original location $\boldsymbol{\zeta}$ mandated from the robot path orchestration RL algorithm, our novel approximation algorithm, presented in Algorithm 1, dynamically associates the R robots with the available A APs at every time slot t. The purpose of this association is to maximize the Packet Delivery Ratio (PDR) of the network while minimizing deviation Λ_j^{t-1} for robot R_j observed in the previous time slot $t-1$, as (`(Step 4)`, in Algorithm 1) :

$$\Lambda_j^{t-1} = \sqrt{(\eta_j^{t-1} - \zeta_j^{t-1})^2}, \;\; \forall j \in R \tag{7}$$

Algorithm 1 Dynamic multi-AP association with robots

Inputs: Robot list $\rightarrow R$, AP list $\rightarrow A$,
Robot true location $\rightarrow \boldsymbol{\eta}$, Robot original location $\rightarrow \boldsymbol{\zeta}$,
SNR $\rightarrow \boldsymbol{\Gamma}$, RTTD for APs $\rightarrow \boldsymbol{\gamma}$
while $R > 0$ **do**
  for $j \leftarrow 1$ to $\mathcal{K}$ **do**
    Get robot deviation from optimal path at $t-1$:
    $\Lambda_j^{t-1} \leftarrow \sqrt{(\eta_j^{t-1} - \zeta_j^{t-1})^2}$ compute $\forall j \in \mathcal{K}$
    Compute distance from robot to nearby $\boldsymbol{\delta}$ obstacles at t:
    $\boldsymbol{\mu_{j,q}^t} \leftarrow \sqrt{(\eta_j^t - q)^2}$ compute $\forall q \in \boldsymbol{\delta}$
    Update robot priority list at t: $\boldsymbol{\lambda}^t$
    for $r \leftarrow 1$ to $|\boldsymbol{\lambda}^t|$ **do**
      Find AP with best SNR to robot: $\boldsymbol{\Gamma}_{max}^t \leftarrow [\alpha_i^t, R_r^t]$
      where $1 \leq i \leq \mathcal{M}$ and $1 \leq r \leq \mathcal{K}$
      if $\alpha \geq 1$ **then**
        Find least conservative MCS to meet PER threshold:
        ε_r^t s.t. $\Omega_r^t \leq \Omega_{max}$
        if PER threshold cannot be met: $\Omega_r^t > \Omega_{max}$ **then**
          Select MCS 0 : $\varepsilon_{0,r}^t$
        end if
        Compute payload duration: $\rho_r^t \leftarrow \varepsilon_r^t$
        Update uplink transmit time at this AP if associated with this robot: $\Xi_i^t \leftarrow \rho_r^t$
        Predict RTTD at candidate AP: $\gamma_i^t \leftarrow \Xi_i^t$
        while $\rho_r^t > \Upsilon_{max}$ *or* $\Xi_i^t > \Upsilon_{max}$ **do**
          Find next available AP with highest SNR to robot:
          $\boldsymbol{\Gamma}_{max}^t \leftarrow [\alpha_{i-1}^t, R_r^t]$
          Add this AP to probable AP list for this STA :
          $\mathcal{U}_r^t \leftarrow \alpha_g^t$, with $[g \neq i]$ and $g \in \boldsymbol{A}$
          Perform MRC : $\boldsymbol{\Gamma}_{mrc}^t \leftarrow \frac{\sum_{i=0}^{\mathcal{K}-1} \vartheta_i^2 \mathcal{E}_b}{\mathcal{N}_0}$
          Compute MCS with updated SNR :
          $\varepsilon_{r_{mrc}}^t \leftarrow \boldsymbol{\Gamma}_{mrc}^t$
          for $p \leftarrow 1$ to $\mathcal{U}_r^t$ **do**
            Compute PER, payload duration and RTTD :
            $\Omega_{r_p}^t \leftarrow \varepsilon_{r_{mrc}}^t$
            $\rho_{r_p}^t \leftarrow \Omega_{r_p}^t$
            $\gamma_{r_p}^t \leftarrow \rho_{r_p}^t$
          end for
        end while
        Associate robot with APs in probable AP list :
        $\mathcal{U}_r^t \leftarrow R_r^t$
        for $p \leftarrow 1$ to $\mathcal{U}_r^t$ **do**
          Update RTTD for the AP :
          $\gamma_{r_{mrc}}^t \leftarrow \gamma_{r_p}^t$
        end for
        Remove robot from robot priority list:
        $\boldsymbol{\lambda}^t \leftarrow \boldsymbol{\lambda}^t \setminus R_r^t$
      else
        Discard packet for this robot :
        $R^t \leftarrow R^t \setminus r$ [Failure condition]
      end if
    end for
  end for
  for $j \leftarrow 1$ to R **do**
    if $\eta_j^t = \iota_j$ (robot reaches its target) **then**
      Remove robot from robot list : $R^t \leftarrow R^t \setminus j$
    end if
  end for
  return Robot list, AP RTTD : $\{R, \boldsymbol{\gamma}\}$
end while

where $1 \leq j \leq \mathcal{K}$. Based on Λ_j^{t-1}, the proximity to nearby

obstacles, $\boldsymbol{\mu^t_{j,q}}$, is formulated as (Step 5) :

$$\boldsymbol{\mu^t_{j,q}} = \sqrt{(\eta^t_j - q)^2}, \ \ \forall q \in \boldsymbol{\delta} \tag{8}$$

The robots are added to the priority list $\boldsymbol{\lambda^t} = [R_{\mathcal{K}}, .., R_1, .., R_j]$ based on the proximity values to the obstacles at time slot t and the amount of deviation Λ^{t-1}_j at the previous time slot $t-1$, as (Step 6):

$$\boldsymbol{\lambda^t_j} = \omega_x \Lambda^{t-1}_j + \omega_y \|\boldsymbol{\mu^t_{j,q}}\| \ \ \forall q \in \boldsymbol{\delta} \tag{9}$$

where ω_x and ω_y are the weights for the robot deviation (positive value) and distance to obstacles (negative value), respectively. In this way, the robots with a higher percentage of obstacles having low proximity values and higher deviation values from the mandated path are given higher priority. The SNR from every robot to every AP at time slot t is defined by the matrix $\boldsymbol{\Gamma^t}$ having the dimension $\mathcal{K} \times \mathcal{M}$. Starting with the highest priority robot, the algorithm selects the available AP α^t_i with the highest SNR $\boldsymbol{\Gamma}^t_{max}$ for the robot R^t_r at the time slot t (Step 8), where $1 \leq r \leq |\boldsymbol{\lambda^t}|$. After this, the least conservative MCS ε^t_r which satisfies the network PER threshold $\Omega^t_r \leq \Omega_{max}$ is chosen for that robot R_r. In case the PER threshold cannot be met ($\Omega^t_r > \Omega_{max}$) with the lowest MCS 0, then following the best effort model, MCS 0 ($\varepsilon^t_{0,r}$) is chosen for robot R_r (Step 10 - 13). The payload duration ρ^t_r is computed based on the chosen MCS ε^t_r, following the IEEE 802.11ax MU-UL model in the indoor scenario at 5 GHz frequency (Step 14). Next, the uplink transmission time Ξ^t_i for the candidate AP α_i is calculated to check if robot R_r can be accommodated in one of the 2 MHz RUs in the uplink PPDU frame, considering a 20 MHz BW (Step 15). In case either $\rho^t_r > \Upsilon_{max}$ or $\Xi^t_i > \Upsilon_{max}$, then the candidate AP α_i is designated as 'overloaded', and is removed from the list of available APs till it completes serving sensor data packets from it's currently associated robots.

• **Instantiating MRC:** The algorithm then proceeds to execute MRC for the robot R_r, in order to improve the SNR condition $\boldsymbol{\Gamma}^t_{mrc}$, following Equation 6. The robot R_r will then get the flexibility to choose a higher MCS value, thereby bringing the payload or the uplink frame duration below Υ_{max} in all the candidate APs [$\mathcal{U}^t_r$] which are performing MRC for robot R_r. The algorithm keeps adding the best candidate APs (with the highest SNR to R_r) from the remaining available APs in A, until the latency for R_r, $\Upsilon^t_r < \Upsilon_{max}$ (Step 17 - 25). The status of the selected candidate APs in $\mathcal{U}^t_r$ becomes '*associated* APs' from '*candidate* APs', and their individual uplink frame duration Ξ^t_o, $\forall o \in \mathcal{U}^t_r$ is updated following this association. In case, all available APs in A are exhausted but still the $\Upsilon^t_r > \Upsilon_{max}$, then the failure condition is encountered (Step 32), and the packets for robot R^t_r are discarded.

After this, the algorithm considers the next highest priority robot in $\boldsymbol{\lambda^t_r}$ until all robots in $\boldsymbol{\lambda^t_r}$ are served for the time slot t. The algorithm continues for the next time slots till all robots have reached their respective target locations, thereby satisfying the condition $\eta_j = \iota_j, \forall j \in R$, where η_j is the true position of the robot R_j and ι_j is the target location for robot R_j, respectively.

• **Dynamic Multi-AP Association Algorithm Constraints:** The edge network resource orchestration algorithm, even though is designed to be dynamic and solves the AP handover effects in classical 802.11ax, it still is a *reactive* algorithm which will take time to converge to optimal network utilization in scenarios of reduced AP to robot ratio or in cases of large sensor payload sizes (e.g., real-time 4K video streaming, multi-camera sensors, etc.). This has the potential for reduced network efficiency and thus increased divergence of robots from their optimal paths, leading to collisions.

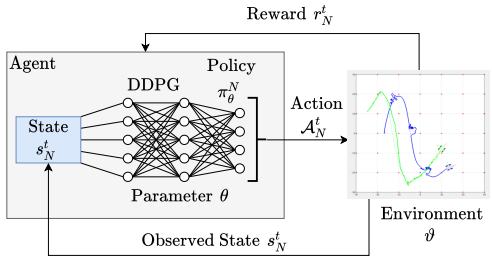


Figure 13. 802.11ax based edge network orchestrator DDPG RL agent architecture. The agent is interacting with the environment based on the policy, observed state, multi-AP association action, and the collected reward.

7 Tier 3: Smart Edge Network Orchestration

Given the constraints with the L-NORM dynamic multi-AP association algorithm, we integrate a network resource orchestrator RL module at the edge as part of our Tier 3 solution. This RL agent will learn the optimal network resource allocation over time, and make the network resource orchestration more *proactive*. The goal of this RL-based edge network resource orchestrator is to improve the edge network performance in such highly constrained scenarios of reduced AP to robot ratio, large sensor uplink payload size, etc.

The Problem: In this case, we denote the optimal RL policy as π^N_θ which will satisfy the Problem $\mathbb{P}_1$ through multi-AP coordination. Similarly, given a total time period $\mathcal{T}$, we formulate a modified version of problem $\mathbb{P}_1$ as $\mathtt{mod}\,\mathbb{P}_1$ which exploits the RL based formulation for the multi-AP coordination problem, stated as:

$$\mathtt{mod}\,\mathbb{P}_1 : \max_{\pi^N_\theta} \mathbb{E}_{\pi^N_\theta}\Big[\sum_{t\in\mathcal{T}}[r^t_N]\Big] \tag{10a}$$

$$\text{s.t. } \mathbb{E}_{\pi^N_\theta}\Big[\frac{1}{\mathcal{T}}\sum_{t\in\mathcal{T}}[r^t_N]\Big] \geq r^N_{\min}, \tag{10b}$$

where r^t_N is the generated reward from state s^t_N and action $\mathcal{A}^t_N$ at time slot t. The $r^N_{\min}$ is defined as the minimum reward threshold defined for the multi-AP coordination environment ϑ. We next present our solution approach to this problem by defining our designed DDPG RL agent for the edge network resource (AP) orchestration.

Solution Approach. The overall framework of the designed DDPG RL agent for solving problem $\mathtt{mod}\,\mathbb{P}_1$ is showcased in Figure 13. The policy of this RL agent is tuned with the ultimate goal of making the robots associate with the APs having the highest SNR at that time slot, which will remain stable for the longest duration of time. We term this policy as 'Look-ahead SNR based reward generation'. Accordingly, the policy π^N_θ is updated at every time slot t, based on the reward r^t_N and observed state s^t_N, which are generated by

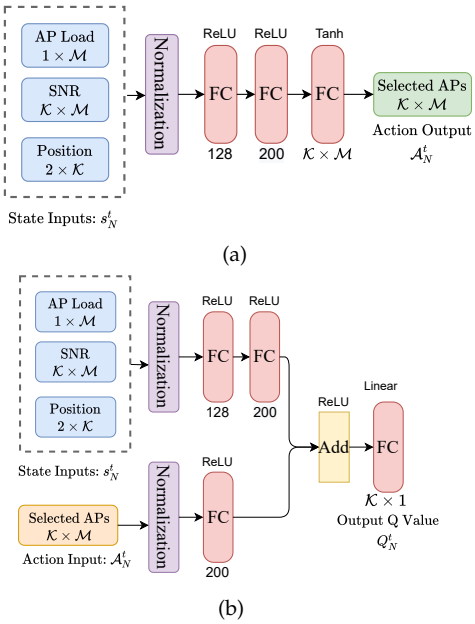


Figure 14. (a) Actor and (b) Critic modeling in DDPG neural network for IEEE 802.11ax based edge network resource orchestration.

the environment ϑ after the application of the action $\mathcal{A}_N^t$ on it. We next explain these state, action, and reward spaces, along with the actor-critic network and the environment.

State Space: The state space of network orchestration s_N^t is defined as the combination of AP load vector L^t, SNR matrix $\mathbf{\Gamma}^t$, and positional information of all robots, P^t at time slot t. The L^t is a vector that contains how many robots are associated with each of the AP, hence, $L^t = \{i | 1 \leq i \leq \mathcal{M}\}$, where $\mathcal{K}$ is the total number of robots and $\mathcal{M}$ is the total number of APs. The SNR matrix $\mathbf{\Gamma}^t$ holds the SNR for each robot to each AP, therefore, $\mathbf{\Gamma}^t = \{(j,i) | 1 \leq j \leq \mathcal{K}, 1 \leq i \leq \mathcal{M}\}$, where $\mathcal{M}$ is the total number of APs. The P^t is defined as: $\{(x_j, y_j) | 1 \leq j \leq \mathcal{K}\}$. Overall, the state-space s_N^t is formulated as:

$$s_N^t = \{L^t, \mathbf{\Gamma}^t, P^t\}$$

Action Space: The action $\mathcal{A}_N^t$ is designed to allow the DDPG agent to generate a multi-AP association matrix AP_N^t at time slot t. The AP_N^t is defined as $\{(i,j)_t | 1 \leq i \leq \mathcal{M}, 1 \leq j \leq \mathcal{K}\}$, where $\mathcal{K}$ and $\mathcal{M}$ are the total number of robots and APs in the environment, respectively. Therefore, the action for network orchestration is defined as $\mathcal{A}_N^t = \{AP_N^t\}$.

On the other hand, the action $\mathcal{A}_N^t$ is generated by the designed policy π_θ^N from the state s_N^t. Hence, the action of time slot t is formulated as: $\mathcal{A}_N^t = \pi_\theta^N(s_N^t)$. So, for edge network resource orchestration the optimal policy π_θ^N is trained using the DDPG actor-critic network modeling for solving $\texttt{mod}\,\mathbb{P}_1$.

Actor-Critic Network: In this case, the optimal policy π_θ^N (corresponding to Equation 10) is determined by the parameters of the trained actor network $f_{\theta_N^A}^A$, where $\pi_\theta^N = f_{\theta_N^A}^A$. Hence, we formulate the actor network as:

$$\mathcal{A}_N^t = f_{\theta_N^A}^A(s_N^t), \qquad f_{\theta_N^A}^A : \mathbb{R}^{|L_N^t| + |\mathbf{\Gamma}_N^t| + |P_N^t|} \mapsto \mathbb{R}^{|\mathcal{A}_N^t|}.$$

Next, the critic network is modeled as:

$$Q_N^t = f_{\theta_N^C}^C(s_N^t, \mathcal{A}_N^t), \quad f_{\theta_N^C}^C : \mathbb{R}^{|L_N^t| + |\mathbf{\Gamma}_N^t| + |P_N^t| + |\mathcal{A}_N^t|} \mapsto \mathbb{R}^1,$$

where Q_N^t is the Q value generated by the critic at time slot t evaluating the success of the generated action $\mathcal{A}_N^t$, and critic network is parameterized with $f_{\theta_N^C}^C$. Similar to Section. 5, these actor and critic networks are adaptable to random number of robots for both training and testing.

Table 3
AP association reward details

Symbol	Reward Description	Reward Value
$r_N^{\Omega^t}$	PER	10^{-4} to 100
$r_N^{\Upsilon^t}$	Latency	-100 to 0
$r_N^{\Lambda^t}$	Robot deviation	Euclidean distance (m)

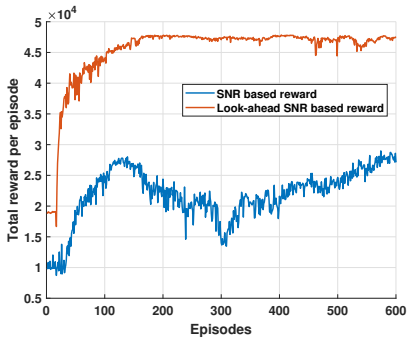


Figure 15. DDPG RL based network orchestrator improvement in reward collection by 1.8% with look-ahead SNR policy compared with only-SNR-based policy.

Reward Space: We define the reward space for each robot as: $r_N^t = f_{\mathcal{C}}^{\mathcal{R}}(s_N^t, \mathcal{A}_N^t) = r_N^{\Omega^t} + r_N^{\Upsilon^t} - r_N^{\Lambda^t}$, where $f_{\mathcal{C}}^{\mathcal{R}}(.)$ is the function to generate the rewards over action $\mathcal{A}_N^t$ and state s_N^t. Here, for every robot's uplink packet transmission, $r_N^{\Omega^t}$ is the reward for PER, and $r_N^{\Upsilon^t}$ is the reward for latency. $r_N^{\Lambda^t}$ is the reward for the robot's deviation. In this case we design $f_{\mathcal{C}}^{\mathcal{R}}(.)$ as a consolidated outcome of $r_N^{\Omega^t}$, $r_N^{\Upsilon^t}$ and $r_N^{\Lambda^t}$. The details of different rewards are given in Table 3. Overall, the total reward of RL agent for AP association is denoted as $\mathcal{K} \times f_{\mathcal{C}}^{\mathcal{R}}(s_R^t, \mathcal{A}_R^t)$.

Environment: Overall, the environment is parameterized as $s_N^{t+1}, r_N^{t+1} = \vartheta(\mathcal{A}_N^t)$, where the AP-allocation environment ϑ takes the inputs of the generated action by the actor-critic network at time slot t and generates the next state and rewards for time slot $t+1$.

7.1 Edge network RL hyperparameter tuning

The final version of our DDPG edge network RL actor-critic model with tuned hyper-parameters for the multi-AP association is shown in Figures 14(a) and 14(b) respectively. For the sake of simplicity, the input layers in both the actor and critic network are not showcased here. The latency information at each AP (AP Load), SNR to each of the robots, and the robots' position values serve as the state inputs s_N^t to this RL network. These values after being normalized are processed through two FC layers with ReLU activation function and having 128 and 200 neurons respectively. The neural network is trained using Adam optimizer with a learning rate of 0.0001 and mean squared error loss function. The action output $\mathcal{A}_N^t$ is generated through an FC layer with Tanh activation function and having $\mathcal{K} \times \mathcal{M}$ neurons. The critic network uses the state data (s_N^t) as input which is then passed through two FC layers with 128 and 200 neurons respectively. The output of this layer is then merged with the action input $(\mathcal{A}_N^t)$ processed through an FC layer with 200 neurons. This merged data serves as the input to a linear activation function which generates the corresponding Q

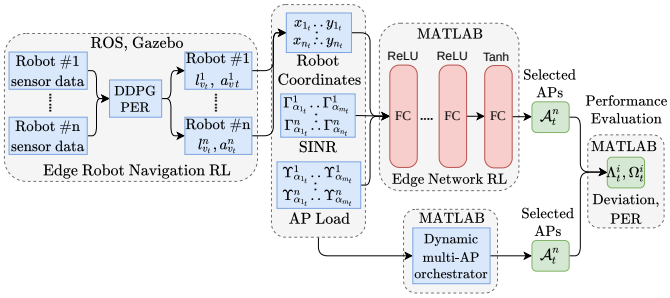


Figure 16. Autonomous edge-based robot and network orchestration data flow model of L-NORM for performance evaluation. The robot locations from ROS simulation were given as input to the Matlab-based edge network orchestrator to observe the robot deviation due to network-related sensor packet errors.

values for the robot association action, having dimension $\mathcal{K} \times 1$.

7.2 Edge network RL agent performance

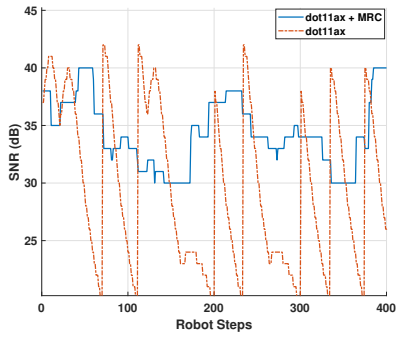
The performance of this RL agent is shown in Figure 15 through the reward plot. At first, the edge network orchestrator RL agent's reward is generated based solely on the SNR values to the AP that the agent decided is best for each robot. From this figure, this reward is plotted (blue line) over time during the training phase. As seen by the lack of stability in the reward collection, it can be surmised that this type of reward generation is not making the RL agent learn the optimal network resource allocations. To improve this performance, the RL agent's policy is updated to make the robots associate with those APs where the SNR remains high and stable for the longest duration of time, during their navigation from source to destination (Look-ahead SNR-based reward). This approach makes the RL agent converge to a very high and stable reward collection, in terms of SNR values, during the training phase, and is showcased in the same figure by the red-colored plot. The collected reward improve by 1.8% compared to the only-SNR-based policy. Motivated by this improvement in performance, we next execute the detailed and overall performance evaluation of L-NORM, comprising all the proposed multi-tiered solutions working in tandem with each other.

8 L-NORM Performance Validation

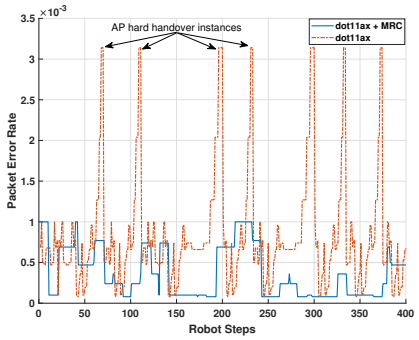
In this section, we extensively evaluate the edge network and robot navigation performance metrics of L-NORM. We implement L-NORM through the integration of ROS/Gazebo robot simulation with Matlab WLAN toolbox [48]. First, we evaluate edge network performance in terms of SNR, MCS, and PER, and then we investigate the robot navigation performance in terms of robot deviation from the optimal path by comparing with three schemes: (a) classical 802.11ax based edge network (denoted as "dot11ax"), (b) L-NORM: 802.11ax edge network enhanced with MRC and dynamic multi-AP allocation (denoted as "dot11ax+MRC"), and (c) L-NORM with added reinforcement learning based edge resource orchestration (denoted as "dot11ax+MRC+RL").

8.1 Simulation setup

We consider a variable number of robots that are randomly deployed in 60 sq. m. indoor area with multiple 802.11ax APs forming the edge nodes connected to an edge-based controller. With each solution (dot11ax, dot11ax+MRC and



(a)



(b)

Figure 17. (a) SNR improvement in multi-AP scenario when using MRC enabled 802.11ax (L-NORM) vs classical 802.11ax on the edge. The robot suffered multiple handovers with classical 802.11ax when the SNR reached 20 dB or lower. (b) PER improvement in the multi-AP scenario when using MRC-enabled 802.11ax (L-NORM) vs classical 802.11ax on the edge. The robot's PER in classical 802.11ax crosses the threshold of 10^{-3} multiple times during AP handover instances.

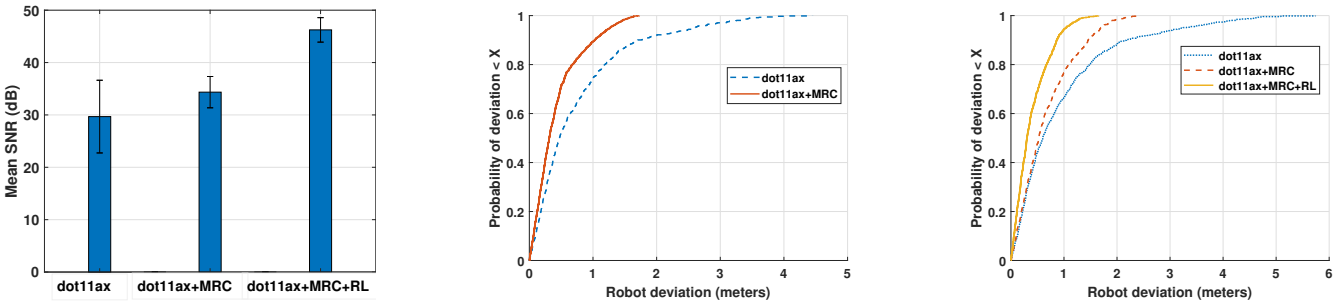
dot11ax+MRC+RL) we run the simulation 300 times with a random number of robots (between 1 and 10), a fixed number of APs (49), and varying packet sizes (1 MB and 10 MB). The simulation data flow is shown in Figure 16. The edge network operates in 5 GHz frequency, with 20 MHz BW in SISO mode and utilizes the IEEE indoor channel model B by initiating the `wlanTGaxChannel` [49] system object in Matlab for 802.11ax. The sensor packet sizes are varied from 1 MB to 10 MB, with 20 robots and 32 APs for this performance evaluation. The AP transmit power is kept at 25 mW. The robots have a maximum linear velocity of 0.8 m/s and a maximum angular velocity of 2 rad/s.

Performance Metrics: For measuring the performance of the edge network we evaluate the *SNR* between the robots and the APs and the *PER* of the robots' uplink sensor data packets. To measure the performance of the robot navigation, we evaluate the *robots' deviation* from the optimal path, i.e., the path the robot could have taken under ideal network conditions (continuous high SNR, no AP handovers, 0 PER).

8.2 L-NORM based edge network

For the edge network performance comparison, for better comprehension and readability, Figures 17(a) and 17(b) show the SNR and PER of the uplink from one random robot in the 'dot11ax' versus the 'dot11ax+MRC' method. In the classical case, the robot suffers from numerous AP handovers, which can be seen by the SNR fluctuation going below 20 dB and rising back up to 40 dB after re-association with the next best AP. This leads to packet loss which can be

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3266643



(a) (b) (c)

Figure 18. (a) Robot SNR performance improvement through the inclusion of MRC and RL in network resource orchestration (L-NORM) for multi-robot navigation. L-NORM ensures more stable SNR compared to classical 802.11ax based edge network solution. (b) Robot performance improvement through the inclusion of MRC with 802.11ax and dynamic edge resource orchestration (L-NORM without network RL) compared to classical 802.11ax base edge network with small payload size. With L-NORM without network RL, the robot deviation to remain within 0.5 m from the optimal path is 19% less than classical 802.11ax. (c) Robot performance improvement through the inclusion of MRC with 802.11ax and edge network resource orchestration RL (L-NORM with network RL) compared to classical 802.11ax base edge network with large payload size. In this case, with network RL enabled L-NORM, the probability of robots to be within 1 m from the optimal path is 28% more than classical 802.11ax.

seen by the high PER values above the threshold of 0.001, during the AP handover instances.

Comparison with state-of-the-art 802.11ax: With L-NORM ('dot11ax+MRC'), due to multi-AP association and dynamic edge resource orchestration, the *SNR fluctuation* remains bounded between 30-40 dB resulting in *controlled PER* which remains within the threshold, as compared to the high fluctuations with 'dot11ax'. Moreover, the RL-enabled edge network resource orchestration in L-NORM ('dot11ax+MRC+RL') improves the performance further in terms of SNR, shown in Figure 18(a). The mean SNR, in this case, is ∼10 dB higher than the MRC-based dynamic resource orchestration approach ('dot11ax+MRC'), and ∼17 dB higher than the classical 802.11ax-based edge network ('dot11ax'). The SNR remains the most stable (2-3 dB deviation) with L-NORM in both the 'dot11ax+MRC' and 'dot11ax+MRC+RL' cases, as compared with the 'dot11ax' case where the SNR deviation is ∼7 dB.

Observation 1. *L-NORM outperforms the 'dot11ax' both in terms of PER and mean SNR. The RL based network orchestrator (Sec. 7) of L-NORM ('dot11ax+MRC+RL') gives ∼10 dB better mean SNR than the proposed dynamic solution (Sec. 6). This leads to higher MCS allocation for the robots, thereby reducing the uplink latency and chances of re-transmission.*

8.3 L-NORM based robot navigation

We next measure the performance of the robots' navigation when it is controlled by the edge-based robot navigation orchestrator under different edge network designs, i.e., 'dot11ax', 'dot11ax+MRC' and 'dot11ax+MRC+RL', with varying sensor payload sizes.

Small Payload Size (∼1 MB/Robot): Figure 18(b) shows the cumulative distribution function (CDF) plot of the robot deviation, with 1 MB sensor payload data per robot, in 'dot11ax' and 'dot11ax+MRC' scenarios.

Observation 2. *With small payload size, the probability of the robots to remain within 1 m from optimal path is 0.89 for L-NORM without network RL ('dot11ax+MRC'), whereas for classical 802.11ax ('dot11ax') it is 0.73. Also, with L-NORM ('dot11ax+MRC'), the probability of the robots to remain within 0.5 m from its optimal path is 19% more than ('dot11ax').*

Large Payload Size (∼10 MB/Robot): We stress test L-NORM by increasing the sensor payload size to 10 MB per robot. The edge network performance with 'dot11ax+MRC' degrades due to the algorithm being reactive, leading to an increase in packet loss at the APs because of congestion and sensor packets overshooting the latency threshold. This is manifested in Figure 18(c) showing the CDF plot of the robot deviation.

Observation 3. *With large payload size, the probability of the robots to remain within 1 m from their optimal path for 'dot11ax+MRC' is 0.78 as compared to 0.66 for 'dot11ax'. However, activating the network RL in L-NORM ('dot11ax+MRC+RL') improves the probability of robot deviation from 0.78 to 0.94.*

When increasing the constraints, i.e., the probability of the robots to remain within 0.5 m from the optimal path, the performance of L-NORM ('dot11ax+MRC') degrades further and is almost the same as 'dot11ax' performance. However, when activating the network orchestrator RL in L-NORM ('dot11ax+MRC+RL'), the performance improves.

Observation 4. *For L-NORM with network RL ('dot11ax+MRC+RL'), the robots' probability to remain within 0.5 m from the optimal path is 0.69 which is an improvement over 'dot11ax' based approach where it is 0.44*

Overall, we can observe that L-NORM (both 'dot11ax+MRC' and 'dot11ax+MRC+RL') provides better results in terms of SNR, PER, and robot deviation while classical 802.11ax ('dot11ax') struggles to keep up with the stringent QoS requirement for time-sensitive robot navigation applications.

8.4 Adaptability of L-NORM

In L-NORM the solutions in Tier 2 and Tier 3, can also be applied to other applications as well. For any application (e.g., live video streaming) or any mix of heterogeneous applications that needs to be served along with robot navigation, L-NORM's Tier-2 solution will assign multi-AP resources for MRC, based on each application QoS performance (PER, latency). In this case, in Algorithm 1, the priority of the packets (Line 6), will be determined by how much each user has experienced packet loss, instead of the robot deviation metric. This will not be an issue for robot navigation, since in the paper we have shown in the preliminary experiments and performance evaluation that

the robot deviation is directly proportional to the sensor packet loss. For the Tier 3 solution, the inputs are AP load, SNR, and user location, based on which the RL agent assigns a set of APs for each user to associate with and the rewards are generated by the resulting PER, latency, and user (robot) deviation from the ideal path. These metrics are common for all applications when measuring performance, except the user deviation. Hence, for other applications also, Tier 3 solution can be applied as is, with the slight modification of the reward space by removing the robot (user) deviation $r_N^{\Lambda^t}$ in Section 7.

8.5 Constraints on testbed experiments

The solutions for L-NORM involve multi-AP coordination and since Wi-Fi is asynchronous, this requires synchronization of the APs at the edge network. Also, L-NORM displaces the data processing from the APs to the edge controller entity. Since the available Commercial-Off-The-Shelf (COTS) 802.11ax devices and drivers are proprietary and they do not support multi-AP coordination, we are unable to perform a testbed experiment-based evaluation of L-NORM.

9 Conclusions

For implementing multi-robot orchestration in an autonomous smart factory, we illustrate the challenges from both robotics and wireless network perspective. We tackle these challenges in a multi-tiered computing solution. We implement autonomous multi-robot navigation in an indoor cluttered environment through reinforcement learning in ROS simulation. We elaborate on the drawbacks of the current state-of-the-art Wi-Fi protocol architecture design to act as an enabler of autonomous multi-robot navigation from the edge network and proposed essential solutions. Our proposed multi-tiered L-NORM solution shows significant performance improvement over the current state-of-the-art 802.11ax protocol. Through extensive simulation studies we show that for small-sized sensor data ($\sim$1 MB) per robot, the probability of robots to remain within 0.5 m from its optimal path is 19% more in L-NORM than the classical 802.11ax. For large-sized sensor data ($\sim$10 MB) per robot, we improve the performance of L-NORM with RL-enabled edge network orchestration and observe that the probability of robots to remain within 0.5 m from its optimal path with RL-enabled L-NORM is 25% more than classical 802.11ax. As a future direction of this work, we envision to improve the L-NORM performance under large robot payload scenarios by incorporating techniques like multi-modal sensor fusion to enable intermittent robot sensor payload transmission without affecting the robot navigation in the process.

10 Acknowledgement

This is a collaboration project with Intel Corporation. We gracefully acknowledge Amit Baxi and Venkat Natarajan, both from Intel Corporation, for their constructive comments to shape this project.

References

[1] C. Bai, P. Dallasega, G. Orzes, and J. Sarkis, "Industry 4.0 technologies assessment: A sustainability perspective," *International journal of production economics*, vol. 229, p. 107776, 2020.

[2] Moore, Mike, "What is Industry 4.0? Everything you need to know." [Online]. Available: https://www.techradar.com/news/what-is-industry-40-everything-you-need-to-know

[3] K. Li, U. Muncuk, M. Y. Naderi, and K. R. Chowdhury, "iSense: Intelligent Object Sensing and Robot Tracking Through Networked Coupled Magnetic Resonant Coils," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6637–6648, 2021.

[4] F. Voigtländer, A. Ramadan, J. Eichinger, C. Lenz, D. Pensky, and A. Knoll, "5g for robotics: Ultra-low latency control of distributed robotic systems," in *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*. IEEE, 2017, pp. 69–72.

[5] F. Gul, W. Rahiman, and S. S. Nazli Alhady, "A comprehensive study for robot navigation techniques," *Cogent Engineering*, vol. 6, no. 1, p. 1632046, 2019.

[6] Cavalcanti, Dave, "Wireless TSN – Definitions, Use Cases Standards Roadmap." [Online]. Available: https://avnu.org/wp-content/uploads/2014/05/Avnu-WirelessTSN-white-paper-V1.0$_Final.pdf$

[7] P. Popovski, Č. Stefanović, J. J. Nielsen, E. De Carvalho, M. Angjelichinoski, K. F. Trillingsgaard, and A.-S. Bana, "Wireless access in ultra-reliable low-latency communication (urllc)," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5783–5801, 2019.

[8] 3GPP Release 16 Specification, "Summary of Rel-16 Work Items (TR21.916)." [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/ SpecificationDetails.aspx?specificationId=3493

[9] E. Krell, A. Sheta, A. P. R. Balasubramanian, and S. A. King, "Collision-free autonomous robot navigation in unknown environments utilizing pso for path planning," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 9, no. 4, pp. 267–282, 2019.

[10] X. Chen and Y. Li, "Smooth path planning of a mobile robot using stochastic particle swarm optimization," in *2006 International conference on mechatronics and automation*. IEEE, 2006, pp. 1722–1727.

[11] D.-L. Almanza-Ojeda, Y. Gomar-Vera, and M.-A. Ibarra-Manzano, "Occupancy map construction for indoor robot navigation," *Robot Control*, 2016.

[12] H. Hu, K. Zhang, A. H. Tan, M. Ruan, C. Agia, and G. Nejat, "A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6569–6576, 2021.

[13] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[14] M. M. Azari, S. Solanki, S. Chatzinotas, O. Kodheli, H. Sallouha, A. Colpaert, J. F. M. Montoya, S. Pollin, A. Haqiqatnejad, A. Mostaani *et al.*, "Evolution of non-terrestrial networks from 5g to 6g: A survey," *arXiv preprint arXiv:2107.06881*, 2021.

[15] B. Demirel, A. Ramaswamy, D. E. Quevedo, and H. Karl, "Deepcas: A deep reinforcement learning algorithm for control-aware scheduling," *IEEE Control Systems Letters*, vol. 2, no. 4, pp. 737–742, 2018.

[16] O. Ayan, M. Vilgelm, and W. Kellerer, "Optimal scheduling for discounted age penalty minimization in multi-loop networked control," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–7.

[17] M. Eisen, M. M. Rashid, K. Gatsis, D. Cavalcanti, N. Himayat, and A. Ribeiro, "Control aware radio resource allocation in low latency wireless control systems," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7878–7890, 2019.

[18] M. S. Elbamby, C. Perfecto, C.-F. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, "Wireless edge computing with latency and reliability guarantees," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1717–1737, 2019.

[19] C. Bocanegra, K. Alemdar, S. Garcia, C. Singhal, and K. R. Chowdhury, "NetBeam: Networked and Distributed 3-D Beamforming for Multi-user Heterogeneous Traffic," in *IEEE DySPAN*, 2019, pp. 1–10.

[20] G. Secinti, A. Trotta, S. Mohanti, M. Di Felice, and K. R. Chowdhury, "Focus: Fog computing in uas software-defined mesh networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2664–2674, 2019.

[21] Q. Qin, K. Poularakis, G. Iosifidis, S. Kompella, and L. Tassiulas, "Sdn controller placement with delay-overhead balancing in

wireless edge networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1446–1459, 2018.
[22] X. Vasilakos, W. Featherstone, N. Uniyal, A. Bravalheri, A. S. Muqaddas, N. Solhjoo, D. Warren, S. Moazzeni, R. Nejabati, and D. Simeonidou, "Towards zero downtime edge application mobility for ultra-low latency 5g streaming," in *2020 IEEE Cloud Summit*. IEEE, 2020, pp. 25–32.
[23] N. Uniyal, A. Bravalheri, X. Vasilakos, R. Nejabati, D. Simeonidou, W. Featherstone, S. Wu, and D. Warren, "Intelligent mobile handover prediction for zero downtime edge application mobility," in *IEEE GLOBECOM*, 2021, pp. 1–6.
[24] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.
[25] Y. Bi, C. C. Meixner, M. Bunyakitanon, X. Vasilakos, R. Nejabati, and D. Simeonidou, "Multi-objective deep reinforcement learning assisted service function chains placement," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4134–4150, 2021.
[26] F. Qiao, J. Wu, J. Li, A. K. Bashir, S. Mumtaz, and U. Tariq, "Trustworthy edge storage orchestration in intelligent transportation systems using reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4443–4456, 2020.
[27] M. Bunyakitanon, X. Vasilakos, R. Nejabati, and D. Simeonidou, "Helicon: Orchestrating low-latent & load-balanced virtual network functions," in *IEEE ICC*, 2022, pp. 353–358.
[28] Open Source Robotics Foundation, Inc., "Turtlebot." [Online]. Available: https://www.turtlebot.com/
[29] Open Robotics, "ROS - Robot Operating System." [Online]. Available: https://www.ros.org/
[30] Gazebo, "Gazebo: Robot simulation made easy." [Online]. Available: http://gazebosim.org/
[31] OpenAI, "Ailgning AI systems with human intent." [Online]. Available: https://openai.com/
[32] Alexander Vandekleut, "TF 2.0 for Reinforcement Learning." [Online]. Available: https://alexandervandekleut.github.io/gym-wrappers/
[33] Mathworks, "Differential Drive Kinematics." [Online]. Available: https://www.mathworks.com/help/robotics/ref/ differentialdrivekinematics.html
[34] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the ieee 802.11 mac layer handoff process," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 93–102, 2003.
[35] T. Adame, M. Carrascosa-Zamacois, and B. Bellalta, "Time-sensitive networking in ieee 802.11 be: On the way to low-latency wifi 7," *Sensors*, vol. 21, no. 15, p. 4954, 2021.
[36] IEEE, "IEEE Std 802.11ax™-2021." [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9442429
[37] Tambet Matiisen, "Demystifying Deep Reinforcement Learning." [Online]. Available: https://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/
[38] M. Yang, B. Li, Z. Yan, and Y. Yan, "Ap coordination and full-duplex enabled multi-band operation for the next generation wlan: Ieee 802.11 be (eht)," in *IEEE Conference on Wireless Communications and Signal Processing*, 2019, pp. 1–7.
[39] A. Garcia-Rodriguez, D. Lopez-Perez, L. Galati-Giordano, and G. Geraci, "Ieee 802.11 be: Wi-fi 7 strikes back," *IEEE Communications Magazine*, vol. 59, no. 4, pp. 102–108, 2021.
[40] M. A. Lema, A. Laya, T. Mahmoodi, M. Cuevas, J. Sachs, J. Markendahl, and M. Dohler, "Business Case and Technology Analysis for 5G Low Latency Applications," *IEEE Access*, vol. 5, pp. 5917–5935, 2017.
[41] P. Schulz and e. al., "Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, 2017.
[42] Y. Kim, G. Kim, and H. Lim, "Cloud-based wi-fi network using immediate ack in uplink data transmissions," *IEEE Access*, vol. 6, pp. 37 045–37 054, 2018.
[43] S. Zhang and D. R. Franklin, "Feasibility study on the implementation of ieee 802.11 on cloud-based radio over fibre architecture," in *IEEE ICC*, 2014, pp. 2891–2896.
[44] R. Janaswamy, *Radiowave propagation and smart antennas for wireless communications*. Springer Science & Business Media, 2001.
[45] J. Zhang, L. Tian, Y. Wang, and M. Liu, "Selection transmitting/maximum ratio combining for timing synchronization of mimo-ofdm systems," *IEEE Transactions on Broadcasting*, vol. 60, no. 4, pp. 626–636, 2014.
[46] Ettus Research, "OCTOCLOCK CDA-2990." [Online]. Available: https://www.ettus.com/all-products/octoclock/
[47] S. Mohanti, C. Bocanegra, S. G. Sanchez, K. Alemdar, and K. Chowdhury, "Sabre: Swarm-based aerial beamforming radios: Experimentation and emulation," *IEEE TWC*, 2022.
[48] Mathworks, "WLAN Toolbox." [Online]. Available: https://www.mathworks.com/help/wlan/
[49] Matlab, "wlanTGaxChannel." [Online]. Available: https://www.mathworks.com/help/wlan/ref/wlantgaxchannel-system-object.html

Subhramoy Mohanti is a PhD candidate at the Department of Electrical and Computer Engineering in Northeastern University. He received the M.S. degree from Northeastern University in 2016. He is the recipient of the IEEE INFOCOM Best Paper Award (2018) and the Northeastern University Graduate Dissertation Research Grant (2015). His current research areas include UAV networking and communication, wireless protocols, networks, scheduling and optimization techniques.

Debashri Roy is currently an Associate Research Scientist at the Department of Electrical and Computer Engineering, Northeastern University. She received her MS (2018) and PhD (2020) degrees in Computer Science from University of Central Florida, USA. She was an experiential AI postdoctoral fellow at Northeastern University (2020-2021). Her research interests are in the areas of AI/ML enabled technologies in wireless communication, multimodal data fusion, network orchestration and nextG networks.

Mark Eisen received the Ph.D in electrical engineering and Masters in Statistics from the University of Pennsylvania in 2019. He is currently working as a research scientist at Intel Labs in Hillsboro, OR. His research interests include machine learning, wireless communications, networked control systems, and statistical optimization. Dr. Eisen was a recipient of the Outstanding Student Presentation at the 2014 Joint Mathematics Meeting, as well as the recipient of the 2016 Penn Outstanding Undergraduate Research Mentor Award. In 2022, Dr. Eisen received the Best Paper Award at the IEEE International Conference on Factory Communication Systems.

Dave Cavalcanti is Principal Engineer at Intel Corporation where he develops next generation wireless connectivity and distributed computing technologies to enable autonomous, time-sensitive systems and applications. He received his PhD in computer science and engineering in 2006 from the University of Cincinnati. He leads Intel Lab's research on Wireless Time-Sensitive Networking (TSN) and industry activities to enable time-critical systems and applications of wireless technologies, including WiFi and beyond 5G systems. He is Senior Member of the IEEE and serves as the chair of the Wireless TSN working group in the Avnu Alliance, an industry group facilitating an ecosystem of interoperable TSN devices and deterministic networking across Ethernet, Wi-Fi and 5G technologies.

Kaushik Chowdhury is a Professor at Northeastern University, Boston, MA. He is presently a co-director of the Platforms for Advanced Wireless Research (PAWR) project office. His current research interests involve systems aspects of networked robotics, machine learning for agile spectrum sensing/access, wireless energy transfer, and large-scale experimental deployment of emerging wireless technologies.