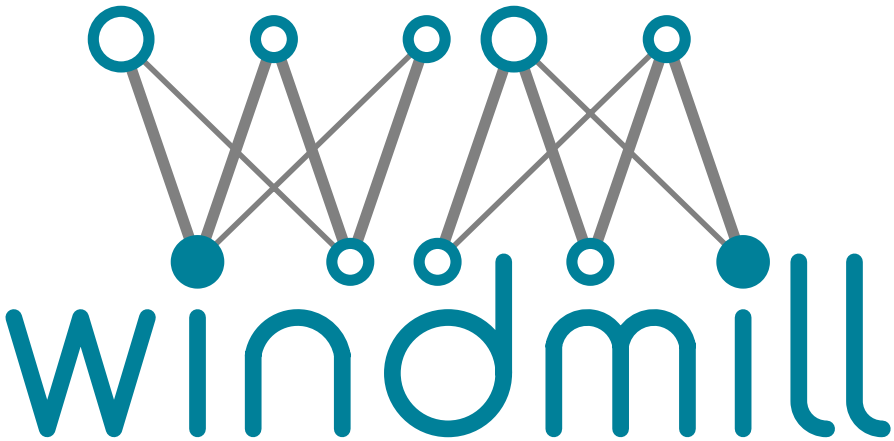
Marie Skłodowska Curie Action

# WINDMILL

Machine Learning for Wireless Communications

# H2020-MSCA-ITN-ETN

# Grant Agreement Number: 813999




## WP5–Data-driven optimisation schemes for radio access management

### D5.2–Intermediate report on machine learning techniques for radio resource management

| | |
|---|---|
| **Contractual Delivery Date:** | Month 32 (August 2021) |
| **Actual Delivery Date:** | August 16, 2021 |
| **Responsible Beneficiary:** | NBLF |
| **Contributing Beneficiaries:** | AAU, CTTC, NBLF, UNIPD, WSE |
| **Dissemination Level:** | Public |
| **Version:** | Final |

PROPRIETARY RIGHTS STATEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813999.

## Document Information

| | |
|---|---|
| **Document ID:** | WP5/D5.2 |
| **Version Date:** | August 16, 2021 |
| **Total Number of Pages:** | 66 |
| **Abstract:** | This document describes the progress made during the last year in addressing several Radio Resource Management (RRM) problems with Machine Learning (ML) techniques. Whereas the previous report D5.1 introduced the main research lines the project would follow, the current report D5.2 provides a more concise account of the most promising methods. First, we describe two applications of Deep Q-Network (DQN) algorithms to wireless systems: slotted random access (RA) in Massive Machine Type Communications (mMTC), and Unmanned Aerial Vehicle (UAV) network routing. Then, we also address the problem of age of information (AoI) optimization with Proximal Policy Optimization (PPO). We also explore if Artificial Intelligence (AI) systems can invent Medium Access Control (MAC) protocols on their own and outperform human-designed protocols. Finally, we introduce a new anomaly detection technique for sensor networks. |
| **Keywords:** | Machine Learning, Reinforcement Learning, Anomaly detection, Random Access, Latency optimization, Routing, Drones, Protocol learning, Lorawan |

## Authors


| **Full name** | **Beneficiary / Organisation** | **e-mail** | **Role** |
|---|---|---|---|
| Muhammad Jadoon | CTTC | muhammad.jadoon@cttc.es | Contributor |
| Chien-Cheng Wu | AAU | ccw@es.aau.dk | Contributor |
| Anay Desphande | UNIPD | anayajit.deshpande@unipd.it | Contributor |
| Sobhi Alfayoumi | WSE | salfayoumi@worldsensing.com | Contributor |
| Mateus Pontes Mota | NBLF | mateus.pontes_mota@nokia.com | Contributor |
| Alvaro Valcarce | NBLF | alvaro.valcarce_rial@nokia-bell-labs.com | Deliverable coordinator |

# Table of Contents

# List of Figures



# List of Acronyms and Abbreviations

**A-BFT** Association-BeamForming Training

**ACK** acknowledgement

**ADDTS** Add Traffic Stream

**AI** Artificial Intelligence

**ANN** Artificial Neural Network

**AoP** age of packet

**AoI** age of information

**AP** Access Point

**AR** Autoregression

**ARIMA** Autoregressive Integrated Moving Average

**ARMA** Autoregressive Moving Average

**ATI** Announcement Transmission Interval

**BEB** binary exponential backoff

**BHI** Beacon Header Interval

**BI** Beacon Interval

**BS** Base Station

**BTI** Beacon Transmission Interval

**BW** bandwidth

**CBAP** Contention-Based Access Period

**CI** confidence interval

**CTDE** Centralized Training and Decentralized Execution

**CPU** Central Processing Unit

**CRC** Cyclic Redundancy Check

**CSMA/CA** Carrier Sense Multiple Access with Collision Avoidance

**CSS** Chirp Spread Spectrum

**DTI** Data Transmission Interval

**DMG** Directional Multi-Gigabit

**DQL** Deep Q Learning

**DQN** Deep Q-Network

**DRL** Deep Reinforcement Learning

**DDPG** Deep Deterministic Policy Gradient

**DCM** Downlink Control Message

**DL** Downlink

**Dec-POMDP** Decentralized Partially Observable Markov Decision Process

**EB** exponential backoff

**EDCA** Enhanced Distributed Channel Access

**eMBB** Enhanced Mobile Broadband

**eNB** eNodeB

**FANET** Flying Ad-Hoc Network

**5G** fifth generation

**6G** sixth generation

**FSK** Frequency-shift keying

**gNB** gNodeB

**IFT** Immediate-first-transmission

**IoT** Internet of things

**KPI** key performance indicator

**LoRaWAN** Long Range Wide Area Network

**LR** learning rate

**LSTM** Long short-term memory

**MA** Moving Average

**MAC** Medium Access Control

**MADDPG** multi-agent deep deterministic policy gradient

**MARL** Multiagent Reinforcement Learning

**MAS** multi-agent systems

**MCS** Modulation and Coding Scheme

**MDP** Markov Decision Process

**ML** Machine Learning

**MLP** Multilayer Perceptron

**mmW** Millimeter Wave

**mMTC** Massive Machine Type Communications

**MRO** Mobility Robustness Optimisation

**MTC** Machine Type Communications

**MU-MIMO** Multi-User Multiple Input, Multiple Output

**ns-3** Network Simulator 3

**nSEB** non-symmetric EB

**OAM** Operations, Administration, and Maintenance

**OFDM** Orthogonal Frequency Division Multiplexing

**PBSS** Personal Basic Service Set

**PCP** Personal Basic Service Set (PBSS) Central Point

**PCP/AP** PBSS Central Point (PCP)/Access Point (AP)

**PDF** Probability Density Function

**PDU** Protocol Data Unit

**PDS** Post Decision State

**PHY** Physical Layer

**PPO** Proximal Policy Optimization

**QoS** Quality of Service

**RA** random access

**RAR** Route Availability Ratio

**ReLU** Rectified Linear Unit

**REO** Route Establishment Overhead

**RL** Reinforcement Learning

**RLC** Radio Link Control

**RLF** Radio Link Failure

**RRC** Radio Resource Control

**RRM** Radio Resource Management

**RSC** Route Sustainability Criterion

**SDN** Software Defined Network

**SF** Spreading Factor

**SON** Self-Organising Networks

**SP** Service Period

**SSW** Sector Sweep

**STA** Station

**SEB** symmetric EB

**sr** SRscheduling request

**sg** SGscheduling grant

**sdu** SDUservice data unit

**TDMA** Time Division Multiple Access

**TSPEC** Traffic Specification

**TTI** Transmission Time Interval

**TB** Transport Block

**TBLER** Transport Block Error Rate

**UAV** Unmanned Aerial Vehicle

**UE** User Equipment

**UKF** Unscented Kalman Filter

**uRLLC** Ultra Reliable and Low Latency Communications

**UCM** Uplink Control Message

**UL** Uplink

**UL-SCH** Uplink Shared Channel

**VBR** Variable Bit Rate

**VR** Virtual Reality

**WSN** Wireless Sensor Network

# 1. Introduction

This Windmill intermediate report is Deliverable 5.2 of Work Package 5 (WP5), which describes our research efforts on the application of ML techniques to RRM problems. The novel techniques described in this report take the form of algorithms to solve various RRM challenges in wireless systems, and are typically executed at layers 2 and above.
WP5 of project Windmill has focused on the following mixture of fundamental and practical problems in wireless networks:

- RA
- Latency optimization
- Data routing in airborne networks
- Protocol learning
- Anomaly detection in sensor networks

Many of these problems can be formulated as sequential decision making tasks, where the solutions entail actions taken in a stepwise manner. Those problems can profit from ML techniques based on Reinforcement Learning (RL), which train models that improve over time through interaction with the network. Other problems, such as anomaly detection in sensor networks, do not require sequential decisions and are more related to Operations, Administration, and Maintenance (OAM), as well as to network monitoring. To address this type of problems, the more classical ML techniques (supervised, unsupervised and semi-supervised learning) are typically used.
Chapter 2 describes our efforts in finding an efficient transmission strategy for slotted RA in mMTC networks. Starting in 5G, new communication use cases have emerged. mMTC is one of them and it brings new requirements on latency and signaling overhead. In chapter 2, we describe how to adapt the DQN algorithm to a multi-user mMTC system to learn a transmission scheme that maximizes bitrate and minimizes delay. We discuss the performance this method can achieve, its limits and the remaining open questions.
Chapter 3 deals with the problem of Age of Information (AoI) minimization, which is relevant for the transmission of time-sensitive information. Mission-critical communications (e.g. telecommand, real-time monitoring, etc) is a good example of systems that could profit from innovations along these lines. We address this problem via Proximal Policy Optimization (PPO), which is a RL method characterized by its high sample efficiency (i.e. less data is needed to train the model). In chapter 3, we present the system model (a 5G network), describe our algorithmic proposal and discuss its results.
Chapter 4 addresses the packet routing challenges of Flying Ad-Hoc Networks (FANETs). These are airborne wireless networks, where each UAV carries a traffic router. The continuous movement of these airborne routers changes the performance of the UAV-to-UAV wireless links and therefore, also of the network topology. There is therefore a need for reconfigurable multihop routing algorithms. In chapter 4 we present two complementary solutions to this problem: A centralized multipath routing scheme, and a Deep Q Learning (DQL)-based algorithm for route selection based on the UAVs position and mobility.

In addition to the use cases introduced by 5G, it is not possible to know what new communication needs will emerge in the next decades. In 2019, few telecom experts would have considered screen-sharing to be an important communication use-case. However, the COVID19 pandemic and the growth of teleworking put this type of traffic at the forefront of communication needs. To avoid having to redesign our communication protocols every time a new use case comes up, in chapter 5 we explore an automated way of *emerging* new protocols. The used technique is Multiagent Reinforcement Learning (MARL) and in this chapter we present early results on this ambitious goal.
Finally, chapter 6 addresses Internet of things (IoT) and Wireless Sensor Networks (WSNs) based on Long Range Wide Area Network (LoRaWAN). The focus is on performance monitoring, which is relevant for large networks and in this chapter we describe the relevant key performance indicators (KPIs) to be monitored, as well as the network data collection mechanism. To facilitate data interpretation, we also present a dashboard for data visualization and introduce the anomaly detection method.

# 2. Deep Reinforcement Learning for Random Access

In future communication networks such a 5G and beyond, mMTC demands novel RA protocols that have a low signaling overhead and that can adapt to sporadic traffic characteristics of Machine Type Communications (MTC) networks. In this report, we present our initial in-progress work towards designing ML-based RA protocols for mMTC. We show the potential of RL for dynamic RA in wireless networks. We use a Deep Reinforcement Learning (DRL) algorithm called as DQN for slotted RA to design a transmission policy that provides a better performance in terms of throughput, delay and fairness as compared to the baseline techniques. We present our initial RL environment for RA and we proposed average age of packet (AoP) as a metric to measure the fairness of the system.
The ALOHA-based RA techniques are particularly well-suited for MTC due to its design flexibility and low signaling overhead. For this reason, we consider collision resolution schemes used for slotted ALOHA as our benchmark schemes in this work and we compare the performance of our proposed solution we these techniques.

## 2.1. Benchmark

The ALOHA protocol was proposed in 1970 by N. Abramson [1] as a medium access control (MAC) procedure in random access (RA) channels, in which a pool of users attempt to transmit packets over a common medium in an uncoordinated fashion. As a variant of ALOHA, Slotted ALOHA operates on discretized time, in that time is divided into slots (whose duration matches a packet length) and transmissions need to be synchronised at slot level [2]. However, these protocols incur packet collisions and make the system unstable if there is no sophisticated procedure applied to resolve them. Commonly used techniques are random backoff schemes such as exponential backoff (EB) for collision resolution in order to avoid low throughput and excessive delays. These techniques use a binary or ternary feedback from the receiver to adjust the transmission probabilities of the users [3, 4]. In a binary feedback system, the receiver informs the transmitters whether the transmission has resulted in collision or not, while in ternary feedback, the receiver informs whether the transmission was success, collision, or idle (no one transmitted on the channel).
In EB, each user backoff exponentially after each consecutive collision. Widely used standards such as Ethernet LAN, IEEE 802.3 or IEEE 802.11 WLAN, have adopted such backoff mechanisms. For instance, the binary exponential backoff (BEB) has also been considered in [5, 6]. However, these backoff techniques show different performances under different assumptions for various system models considered in the literature. For instance, it is common for analysis to assume that the packet queue of each user is full, and this is called as *saturation state* of the system. Moreover, under different assumptions of packet arrival process, the performance of backoff technique is varying. A commonly used stochastic process to model traffic arrivals is Poisson process for which BEB works well. Recently, it has been shown in [7, 8] that BEB may not be a better choice. For these reasons, the performance of EB schemes and which scheme is better, is still an challenging question. Moreover, the assumption of users in *saturation state* is not applicable to the MTC networks and the assumption of Poisson process for packet arrivals is not realistic for these systems. Nonetheless, due to simplicity of this protocol and distributed nature, slotted RA is well-suited for

MTC paradigm for medium access, and for this reason we consider these schemes as our benchmark. However, due to the above-mentioned complexities involved, ML-based tools are required to design transmission policies for slotted RA.
Recently, RL has attracted much attraction in wireless community to design multiple access problems [9–11]. These works consider multiple user and multiple channel systems, and do not provide insights about single channel system and how the transmission probabilities can be learned to improve delay and fairness in slotted RA. In [12], a heterogeneous environment with ALOHA and Time Division Multiple Access (TDMA) is considered to learn both schemes by users. In [9, 13], ALOHA-Q protocol is proposed an RL-based scheme for a single channel system but their technique depends on the frame structure and each user keeps and updates a policy for each time slot in the frame. Moreover, all of the above mentioned works assume users to be saturation state, i.e, users always have a packet in their queue. This assumption is not feasible for MTC networks where the traffic is sporadic and user may or may not have a packet in its buffer. Therefore, to design such a problem, RL algorithms seem to be a valid choice.

### 2.1.1. Progress and Contributions

In this work, we have leveraged DRL to design channel access and transmission strategy for slotted RA for a single channel.

- We design a RL environment for our system model that is based on binary broadcast feedback to all users aiming for reduced signaling. We use previous action and feedback to learn the transmission probability and our simple setting allows us to analyze the learning process of RL for channel access.
- We use the DQN algorithm to learn a single transmission policy by training the algorithm in a centralized way, and this policy is used in a distributed way during the evaluation phase.
- We do not consider users to be in saturation state as opposed to the previously mentioned works, which is a realistic assumption for MTC networks.
- For simplicity, we consider Poisson process for packet arrivals, but our scheme is not dependent on Poisson process.

In the next sections, we present the progress that has been made so far from designing a RL environment for learning the transmission strategy for slotted RA to some results we have achieved compared to some baseline EB schemes.

## 2.2. System Model

We consider a synchronous slotted RA system with $N$ number of users, a single channel that is shared among all the users and one receiver. The time is divided into discrete slots as in slotted ALOHA. The channel is assumed to be error-free, i.e., if a user successfully transmits a packet without colliding with other users, the receiver receives the packet with no error. We assume that each user stores packets in its buffer denoted by $B_n(k)$, and users

can store maximum $\bar{B}$ packets in their buffers. If new packets arrive at the buffer when the buffer is full, i.e., $B_n(k) = \bar{B}$, these packets are discarded and are considered lost.
At every time slot, user receives new packets following Poisson process with average arrival rate $\lambda$. In order to transmit these packets, a user $n$ can take two actions, $A_n(k) \in \{0, 1\}$, where $A_n(k) = 0$ corresponds to the event when user $n$ chooses to wait and not transmit, and $A_n(k) = 1$ corresponds to the event when user $n$ transmits a single packet on the shared channel. We consider that if only one user transmits on the channel, the transmission is successful; and if two or more users transmit in the same time slot, the collision happens. The collided packets are discarded and need to be re-transmitted by all the users until they are successfully received by the receiver.
At the end of each time slot, the receiver sends a *feedback* signal to the users. We assume a broadcast channel where the feedback is sent to all the users. For reduced signaling overhead, we consider a binary feedback signal that specifies whether the collision occurred in time slot $k$ or not, i,e.,

$$F(k) = \begin{cases} 0 & \text{if two or more users transmit} \\ 1 & \text{otherwise.} \end{cases} \tag{2.1}$$

When packets collide, the feedback is 0, and if there was success or no user transmitted on the channel, the feedback is 1. This feedback signal is used to update the buffer $B_n(k)$ of each packet. If user $n$ transmitted successfully, the packet is deleted from it buffer.
After each user takes action and receives feedback, we use this information to create *history* $\mathcal{H}_n(k)$ at each user. We assume that each user can keep a record of its own history. The history tuple contains previous actions, feedback signals and buffer size for up to $T_\text{h}$ past time instants, where we refer to $T_\text{h}$ as the *history length*. We will leverage this history for training the DRL algorithm to design a RA policy. We are interested in a transmission policy that can effectively provide us better performance in terms of delay, throughput and fairness.

**Definition 1.** *We define policy as the access scheme of user n at time slot k as a mapping from user's history to a conditional probability mass function* $\pi_n(\cdot|\mathcal{H}_n(k))$ *over the action space* $\{0, 1\}$.
*Each new action* $A_n(k) \in \{0, 1\}$ *is drawn at random from* $\pi_n(\cdot|\mathcal{H}_n(k))$*, as follows,*

$$\Pr\{A_n(k) = a \mid \mathcal{H}_n(k) = \mathcal{H}_n\} = \pi_n\big(a|\mathcal{H}_n(k)\big). \tag{2.2}$$

In this work, we are interested to design a policy that can be implemented in a distributed manner and where transmission probabilities can be adapted as the traffic arrival rate changes. As mentioned above, the history of each user can be used to design such a scheme using DRL algorithm.

### 2.2.1. Performance Metrics

The main objectives of the access scheme to efficiently use the channel resources by resolving collisions in an effective manner. Moreover, a scheme needs to be fair between users, i.e., each user should receive a fair share of the channel access and not incur unwanted *capture effect*. We evaluate throughput, packet delay and fairness for performance

evaluation of the RA system. For fairness, we propose to use a new metric called as AoP that can provide us better insights of both short-term and long-term fairness. We define the performance as follows:

**Throughput.** Throughput is defined as the total number of successful transmissions during time $K$, where $K$ is the total time over which an experiment is performed. Average throughput is measured by averaging the successful transmission of all the users.

**Packet Delay.** Packet delay is the total time in slots that has elapsed between the generation and arrival of packet in user $n$ buffer until it has been successfully transmitted. Let us consider $G_n(k)$ as success event at time slot $k$, i.e., $G_n(k) = 1$ if a packet has been successfully transmitted and $G_n(k) = 0$ if a packet has not been successfully transmitted. The total number of packets that have been successfully transmitted by user $n$ within $K$ successive time slots, is $\sum_{k=1}^{K} G_n(k)$. The sum of delays is equal to

$$\sum_i D_n(i) = \sum_k 1\{B_n(k) > 0\}, \tag{2.3}$$

where $1\{\cdot\}$ is the indicator function, which evaluates to 1 if the statement in braces is true and 0 otherwise.
We calculate the average packet delay for user $n$ as

$$d_n = \frac{\sum_i D_n(i)}{\sum_{k=1}^{K} G_n(k)}, \tag{2.4}$$

and the average delay for the whole system is

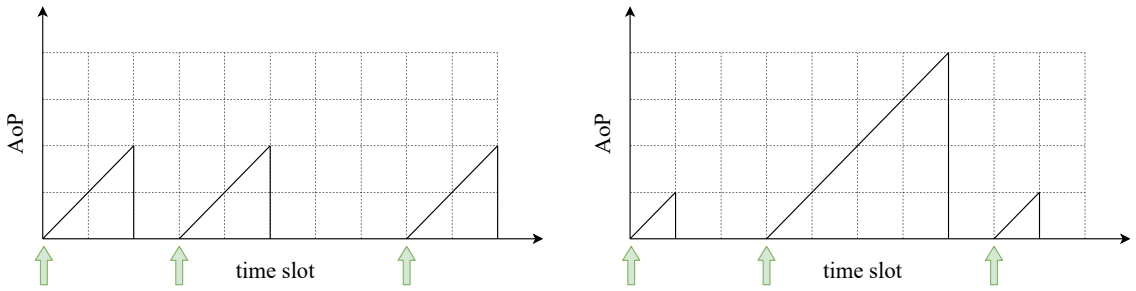
$$\mathcal{D} = \frac{1}{N} \sum_n d_n \tag{2.5}$$

**Age of Packet (AoP).** The AoP can effectively measure if each user has been give a fair-share of the channel or not. Therefore, we propose to use average AoP to measure fairness among users. Although AoP can be seen as cost of delay for a particular user, it provides us insights about *capture effect* and the capture effect causes unfairness among users. Moreover, popular Jain's index [14] to measure the fairness has a limitation that it does not capture the effects of short-term fairness. AoP on other hand can show fairness for both short and long terms.
For a packet that is in each user's buffer, age of packet is increased linearly until it is transmitted successfully and age of packet becomes 0 when it has been transmitted.
Let $w_n(k)$ be a positive integer that represents AoP associated with user $n$ at the beginning of time slot $k$. Assuming that $w_n(1) = 0$, the AoP of user $n$ evolves through time in the following way,

$$w_n(k) = \begin{cases} 0 & \text{if } B_n(k) = 0 \\ w_n(k-1) + 1 & \text{otherwise,} \end{cases}, \tag{2.6}$$

and the average AoP for user $n$ after total time $K$ is calculated as

$$\Delta_n = \frac{1}{K} \sum_k w_n(k). \tag{2.7}$$




(a) Case1 (b) Case 2

Figure 2.1: An example of fairness with AoP. The arrows show packet arrivals

We show with an example how AoP can be an effective tool to measure fairness and also the cost of packet delay incurred due to packets that stay in the queue.

*Example.* Let us consider a system with $N = 3$ users and events within $K = 10$ time slots. Let us assume for the sake of simplicity that only 3 packets arrive and are transmitted successfully within these 10 time slots. Fig. 2.1 shows the packet arrival at each user and the time slots each user takes to successfully transmit the packet. We show two cases and for both cases the average delay is the same, which is 2. It means that on average a user takes 2 times slots to transmit a packet. The average AoP however, for both cases is different, $9/10$ for case 1 and $12/10$ for Case 2. Evidently from Case 2 in Fig. 2.1b, user 2 took 4 time slots to transmit the packet and other two users took only 1 time slot each. Case 2 is an example of unfair channel utilization because during the capture effect when a single user captures the channel, all other users starve and hence the average AoP is high.

## 2.3. RL Environment for Random Access

To learn a transmission policy for slotted RL in a single channel and multiuser system, we consider the RL *environment* as the available physical resources (the channel) as shown in Fig. 2.2.

- Every user interacts with the environment by taking an action and receiving a reward signal. The RL agents learn by interacting with the environment through iterative trial-an-error process.
- We consider that each user keeps track of its past actions and feedback signals; this information is used to learn the access policy and we call is the history of user denoted by $\mathcal{H}_n(k)$.
- The action of each user is whether to transmit or wait based on the output values from the DQN. These output values corresponding to each action are used to calculate the transmit probabilities.
- The environment is partially observable to each user, i.e., each user $n$ is unaware of the history of the other users.

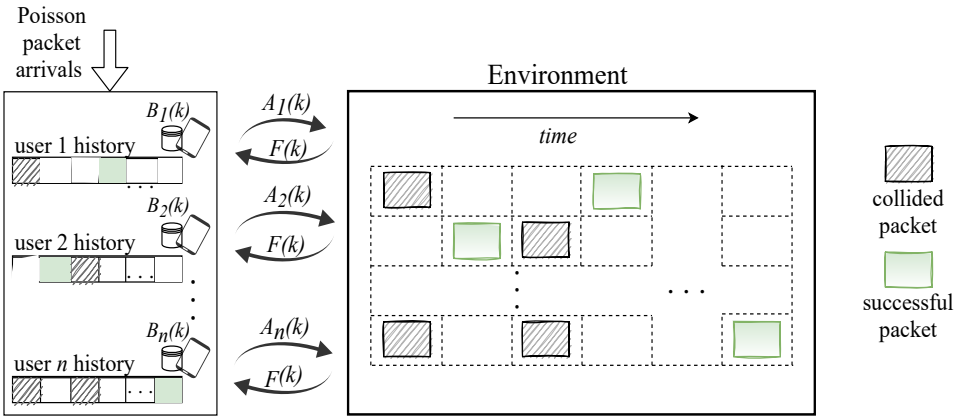



Figure 2.2: Interaction of agents (users) with the environment (channel). Users perform an actions, then receive the feedback signal $F(k)$. Users update their buffers depending on the feedback signal and new packet arrivals to the system.

- During centralized training we assume that each user is able to know whether the transmission on the channel was successful or not. We use this information as reward of the actions. The purpose of each user is to maximize this reward in the long-term.
- This reward signal is used during training process. In evaluation phase, this information is not available to users and each user only relies on its history to choose whether to transmit or wait.

Next, we present how this environment has been used to train the DRL algorithm for channel access policy.

### 2.3.1. DQN Architecture and Training

We use a popular DRL algorithm called as DQN [15] that uses neural network to compute Q-values corresponding to each action. The Q-values are used to calculate the transmit probability of each action. The algorithm uses trial-and-error method by interacting with the environment by taking actions and obtaining the reward signal that shows how 'good' an action is corresponding to each history $\mathcal{H}_n(k)$ tuple at each time slot. Q-values are updated at each time slot and for each possible action in the following way:

$$Q(a,s) \leftarrow Q(a,s) + \alpha\Big[r + \gamma \max_{a} Q(a,s') - Q(a,s)\Big], \tag{2.8}$$

where $0 \leq \alpha \leq 1$ and $0 \leq \gamma \leq 1$ are the learning rate (LR) and the discount factor respectively. The training of the DQN as also shown in Algorithm. 1 involves following steps: At each time slot $k$

1. Each user $n$ obtains the observation $F(k)$ and feeds the history $\mathcal{H}_n(k)$ as the input to the DQN.
2. The output of the DQN are the Q-values $Q(a)$ obtained for possible actions.

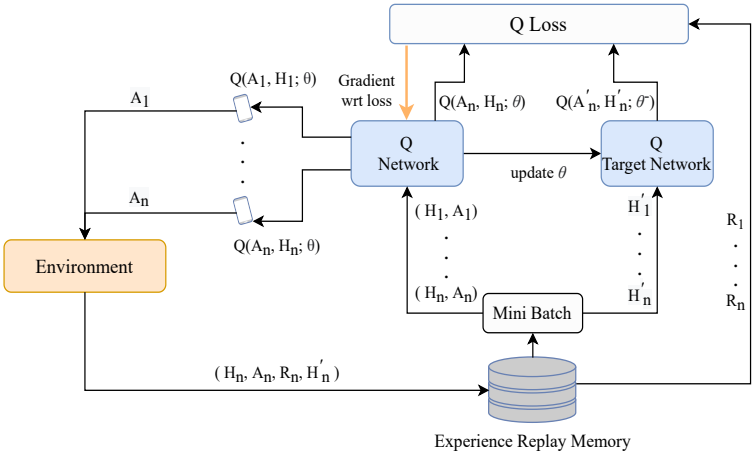



Figure 2.3: DQN training schematic showing policy network, target network and experience replay.

3. Each user *n* draws the action following the policy $\pi$ from the following distribution:

$$\Pr\big(A_n(k) = a\big) = \frac{(1-\epsilon)e^{\beta Q(a)}}{\sum\limits_{a\in\{0,1\}} e^{\beta Q(a)}} + \frac{\epsilon}{2} \qquad (2.9)$$

$$\forall a \in \{0, 1\},$$

for $\epsilon > 0$, and $\beta > 0$ is the temperature.

The values of $\epsilon$ is started from $\epsilon = 1$ and it is gradually decreased to 0. This ensures that agent *explores* the states at the start of training and as the training progresses, the agent becomes more greedy in action selection (exploitation). This well-known phenomenon is exploration-exploitation tradeoff in RL. Similarly, the value of $\beta$ is gradually increased through training and it has similar effect - the higher the value of $\beta$, the more greedy action selection is. The above equation 2.9 provides a balance between *softmax* and *greedy* action selection policies for RL.

For the centralized DQN training, we use the *experience replay* technique by storing the experiences (state, action, reward, next state) for each user in a memory. These experiences are then sampled uniformly in form of mini batches from the memory for training as shown in Fig. 2.3. Moreover, we use double DQN [16] in which two neural networks are used. These two networks are the Q-network with parameters $\theta$ that is used to evaluate and update the policy, and the target network with parameters $\theta^-$ to evaluate target (next state) Q-values. The actual Q-values and the target Q-values from both networks are used to calculate the loss function. The centralized training in this way ensures that we learn a common policy $\pi$ for all the users using the same neural network, which is deployed in a distributed manner in evaluation phase. and users take action without coordinating with each other.

**Algorithm 1:** DQN Training for Slotted RA

**Define** $a \in (0, 1]$, $\gamma \in [0, 1]$, $\epsilon > 0$
**Initialize** $B_n(k) = 0\ \forall n \in \mathcal{N}$; weight update frequency $L$
**for** *time slot* $k = 1, \ldots, K$ **do**
  **for** *user* $n = 1, \ldots, N$ **do**
    Observe $\mathcal{H}_n(k)$ and feed it to Q-network
    Generate the estimate of $Q(a)\ \forall a \in \{0, 1\}$
    Take action $A_n(k)$ according to (2.9)
    Obtain feedback $F(k)$ and reward signal $R(k)$
    Update $B_n(k)$ and obtain new packets to create $\mathcal{H}'_n$
    Observe $\mathcal{H}'_n$ and feed it to both Q-network $Q$ and target Q-network $Q_{\text{target}}$
    Generate estimates from both Q-networks $Q(a)$ and $Q_{\text{target}}(a)\ \forall a \in \{0, 1\}$
  **end**
  Train Q-network with input $\mathcal{H}(k)$ and output Q-values for all users
  **if** $t\%L = 0$ **then**
    $Q_{\text{target}} \leftarrow Q$
  **end**
**end**

Training a DRL algorithm requires a tremendous amount of trial-and-error process. The process involves tuning the parameters and hyper-parameters of the neural network; designing a reward signal and state parameters for the RL environment that suits well to solve the problem. Carefully designing these entities is one of the most important and crucial tasks for training a RL algorithm. Initially, we used buffer state of each user as a reward signal; however, this reward was not sufficient to train users for higher arrival rates. Moreover, the global reward signal instead of local reward helps the DQN during centralized training phase to learn in a competitive way for all the users. We use a simple DQN fully connected feed-forward neural network with two hidden layers with 20 and 10 neurons in each for our initial analysis of the proposed RA scheme.

## 2.4. Results

For performance evaluation of our designed scheme, the DQN is trained for $N = 10$ users. We choose the history size $T_h = 1$ and maximum buffer size $\tilde{B} = 1$. We train DQN for different values arrival rates $\lambda$. It has been observed that for lower arrival rates, user's buffer remain empty most of the times and for higher arrival rates, it remains saturated. Therefore, we use transfer learning to train our model. This is done by choosing a moderate value of lambda and transferring the trained weights to train next lambda value. This way the algorithm can adapt well for different traffic arrival rates.

### 2.4.1. Proposed Transmission Policy

To better understand how the DQN learns the transmit probabilities depending upon the history, we choose history size $T_h = 1$. Therefore, for $F(k) \in \{0, 1\}$, $A_n(k) \in \{0, 1\}$ and $B_n(k) \in \{0, 1\}$, we have total $2^3 = 8$ possible states for each user. However, the focus is only on the states when user $n$ has a packet in the buffer, $B_n(k) = 1$, that is,

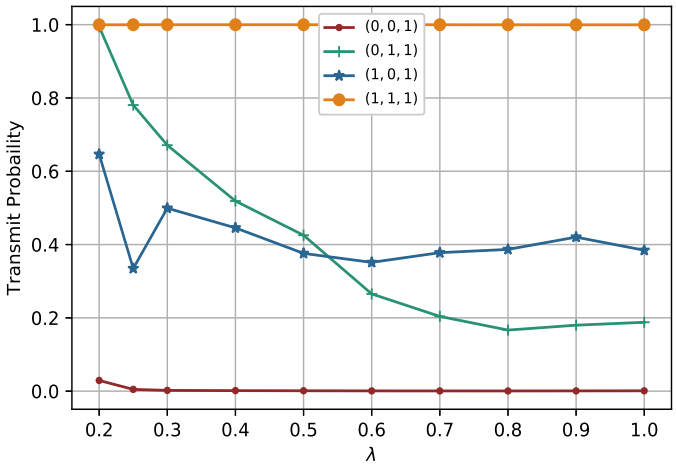
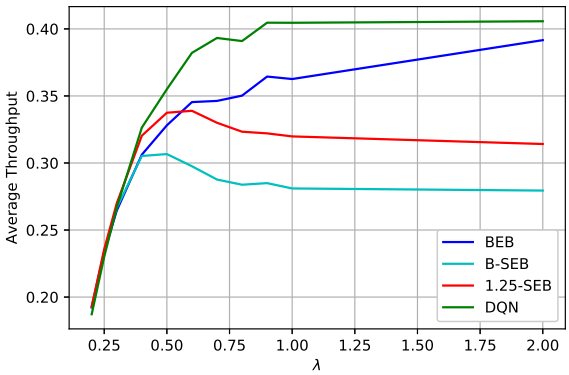



Figure 2.4: DQN policy transition w.r.t the arrival rate $\lambda$ during training.

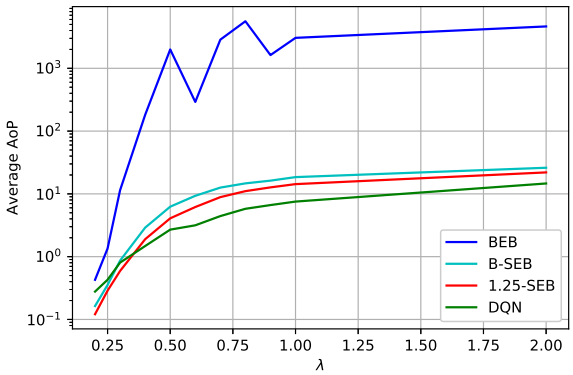
$S_1 = (0,0,1), S_3 = (0,1,1), S_5 = (1,0,1)$ and $S_7 = (1,1,1)$. Naturally, when $B_n(k) = 0$ the users remain silent because there is not packet to transmit. For a small state space, we can now track the policy and if it makes sense as shown in Fig. 2.4. For various arrival rates we show how the transmit probabilities change during training. It is evident that DQN learns to transmit immediately (with $p_n = 0.99$) if success happens, which is like the Immediate-first-transmission (IFT) policy used for retransmission control in slotted ALOHA [3, 4]. The most interesting state is $S_3$, i.e., when the last transmission was successful but user $n$ did not transmit. The transmission probability starts from 1 and decreases gradually as the arrival rate increases. Moreover, the state $S_5$ almost remains consistent around 0.5 transmit probability. These two states, $S_3$ and $S_5$, provide each user more degrees of freedom to achieve better performance compared to baseline schemes and to keep the system stable even if the arrival rate increases.

For results and evaluation, we divide EB techniques into two schemes: non-symmetric EB (nSEB) scheme and symmetric EB (SEB). In nSEB, the transmission probability of users that are involved in the collision is changes and the transmission probability $p_n$ of other users remain unchanged. Once the transmission of user $n$ is successful, the user transmits with $p_n = 1$. On other hand, in SEB, the transmission probability of every user is changed based on collision no-collision event and in both cases the transmissions probabilities are increases and decreased exponentially with a backoff factor $\sigma$. For binary exponential backoff, as the name suggests, $\sigma = 1$. One of the issues with nSEB is that it incurs *capture effect*, where a single user occupies the channel for some time and hence, making other users starve. Apparently, this scheme is unfair and therefore, SEB scheme mitigate this effect by gradually adjusting the transmit probability for all the users.

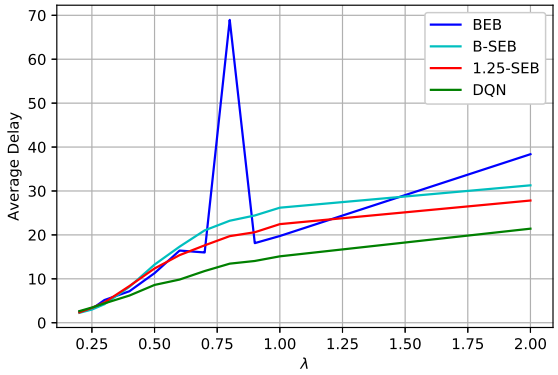
We used the DQN transmission policy for slotted RA in Fig. 2.4 to evaluate average throughput, average AoP and average delay of the system, and compared it to the baseline schemes SEB and nSEB for backoff factors $\sigma = 2$ and $\sigma = 1.25$, as shown in Fig. 2.5. For $\sigma = 2$, the non-symmetric EB is referred to as BEB and for the symmetric EB, it is referred to as binary SEB (B-SEB) in Fig. 2.5. Our results (submitted for publication) show that the proposed DQN-based policy outperforms both baseline techniques in terms of throughput, delay and




(a) Average Throughput




(b) Average AoP (log-scale)




(c) Average Delay

Figure 2.5: Average Throughput, AoP, and delay Comparison of the designed scheme with other baseline schemes for 10 users

AoP. We have seen that nSEB scheme provides better throughput as opposed to SEB but it is unfair as the AoP for few users is very high and very low for others. This shows that the distribution of AoP is uneven among users and hence it is unfair, while for BEB and for the proposed scheme, the AoP is fairly distributed. The proposed policy provides better tradeoff between fairness and throughput compared to both nSEB and SEB scheme.

### 2.4.2. Future Work

In this report, we have presented the potential of DRL for slotted RA environments and how DRL algorithm leverages history to learn a transmission policy. However, we have used small history $t_h = 1$ size and less number of users to provide better insights into the training process of the DQN for analysis. We have been able to observe how the policy changes for different history states, and adapts to different arrival rates. Our next work will focus on using larger history size and exploiting the past information using Long short-term memory (LSTM) networks. We will also consider using the DRL algorithm to optimize AoP. Moreover, we will work on designing a scheme for higher number of users. The scalability to massive settings for mMTC networks is a challenging task, and we will extend our work for massive RA. However, for this purpose, we will consider using large history size, and

possibly more number of parameters in the history to train our model. In massive RA, users will have different sleep cycles and activation times, and sporadic traffic arrivals. A RA scheme that takes into account these characteristics and is adaptive to these patterns is desirable. Moreover, using success as a reward signal may not be sufficient to learn for a system with high number of users. One option is use a reward that is a combination of AoP and success or a combination buffer size, AoP and success. Obviously, it will depend on the updated system model and its parameters.

# 3. Model-free learning for mission-critical latency optimization

## 3.1. Introduction

The traditional metrics such as latency and jitter can not fully characterize the information freshness [17] [18] [19]. Many emerging services or systems such as Autonomous Driving, Industrial Automation, and Tactile Internet require real-time monitoring and low-latency constrained information delivery. The growth of time-sensitive information led to a new data freshness measure named Age of Information (AoI) [19] [20] [21]. AoI measures packet freshness at the destination accounting for the time elapsed since the last update generated by a given source [22]. Consider a cyber-physical system such as an automated factory where many sensors are located at a connected area and they are transmitting time-sensitive information to a remote observer through a wireless network. Each sensor's job is to sample measurements from physical phenomena and transmit them to the monitoring site. Due to the limitation of the wireless bandwidth, the wireless network will challengingly transmit all the fresh data to the monitoring side on time. Hence, a wireless resource scheduler purposed in this chapter will provide some different strategies with the joined consideration of AoI and the other network performance index.

In this work, the designed wireless resource scheduler uses different algorithms to achieve the performance goals of the mission-critical system. The general performance goal is to minimize the AoI of each network node. The AoI minimization problem can be considered as optimizing on average AoI [23]. Motivated by the success of machine learning in solving many of the online large-scale networking problems, we seek a model-free reinforcement learning approach to deal with the correlations that arise in the scheduling problem. Therefore, we transfer the AoI minimization scheduling problem to a multi-arm bandit problem that the minimizing the expected AoI of each node and maintaining the tradeoff between the other performance index. According to this transformation, we consider solving the AoI minimization scheduling problem via optimizing the scheduling priorities of each UE. In this chapter, we propose an algorithm based on Deep Reinforcement Learning (DRL) to solve the formulated wireless resource scheduling problem. DRL is defined by three components (observation, action, and reward). Given a series of network observations, the DRL agent is trained to predict the future actions that maximize the system reward (w.r.t the minimal AoI). To solve this scheduling problem, the DRL agent continuously interacts with the network environment and then tries to find the best scheduling policy based on the reward/cost feeding back from that network environment [24].

The rest of the chapter is organized as follows. Section 3.2 presents a comprehensive review of related work. The problem formulation, system model, network architecture, and the reinforcement learning design are shown in Section 3.3. The proposed learning algorithm is presented in Section 3.4, whereas the numerical simulations are given in Section 3.5. Finally, the conclusions are drawn in Section 3.6.

## 3.2. Background and Related Work

Recently, some papers were tackling the problem of minimizing the AoI of a number of sources that are competing for the available radio resources. [25] considers the problem of many sensors connected wirelessly to a single monitoring node and formulate an optimization problem that minimizes the weighted expected AoI of the sensors at the monitoring node. Moreover, the authors of [26] also consider the sum expected AoI minimization problem when constraints on the packet deadlines are imposed. In [27], sum expected AoI minimization is considered in cognitive shared access.
The scheduling decisions with multiple receivers over a perfect channel are investigated in [28], [29], where the goal is to learn data arrival statistics. Q-learning is used for a generate-at-will model in [28], while policy gradients and DQN methods are used for a queue-based multi-flow AoI-optimal scheduling problem in [29]. In addition, AoI in multi-user networks has been studied in [28]– [30]. The authors shown in [31] that the scheduling problem, where a set of links that share a common channel and the transmitter at each link contains a given number of packets with timestamps from an information source, is NP-hard. Scheduling transmissions to multiple receivers is investigated in [28], focusing on a perfect transmission medium, and the optimal scheduling algorithm is shown to be of threshold type on the AoI. Average AoI has also been studied when status updates are transmitted over unreliable multiple-access channels [32] or multi-cast networks [33]. A source node sending time-sensitive information to many users through unreliable channels is considered in [34], where the problem is formulated as a multi-armed bandit (MAB), and a suboptimal Whittle Index (WI) policy is proposed.
Most prior literature on AoI assumes perfect statistical knowledge of the random processes governing the status update system. However, in most practical systems (e.g., heterogenous UEs with different mission-critical traffic co-located in the same network), the characteristics of the system are not known a priori and must be learned. A limited number of recent works consider the unknown or time-varying characteristics of status update systems, and apply a learning-theoretic approach [35], [29]. To the best of our knowledge, the multi-arm bandit scheduling for a tradeoff between average AoI minimization and throughput limitation is studied for the first time at a multi-user system.

## 3.3. System Model

In this research work, we consider a mission-critical system consists of a 5G network with one base station (gNB), $N$ UEs and UE's controller as described in Fig. 3.1. For instance, a centralized autonomous logistics system in a private park can be demonstrated as a mission-critical system. The centralized controller monitors the state of each UE/vehicle through the 5G wireless network at the remote side. UEs connect to the controller via the base station. The connection between a UE and the controller can be modeled as a virtual link. Since the vehicles move with high speed around the whole park area, to keep the system stable, UEs shall transmit fresh data such as sensor information or velocity to the controller and fetch control signals back to maintain the sysytem operation. All the UE data can be stored into multiple packets and be transmitted to controller individually via the wireless link. For the packet transmission scheduling over a link, a time-slotted system is considered, where

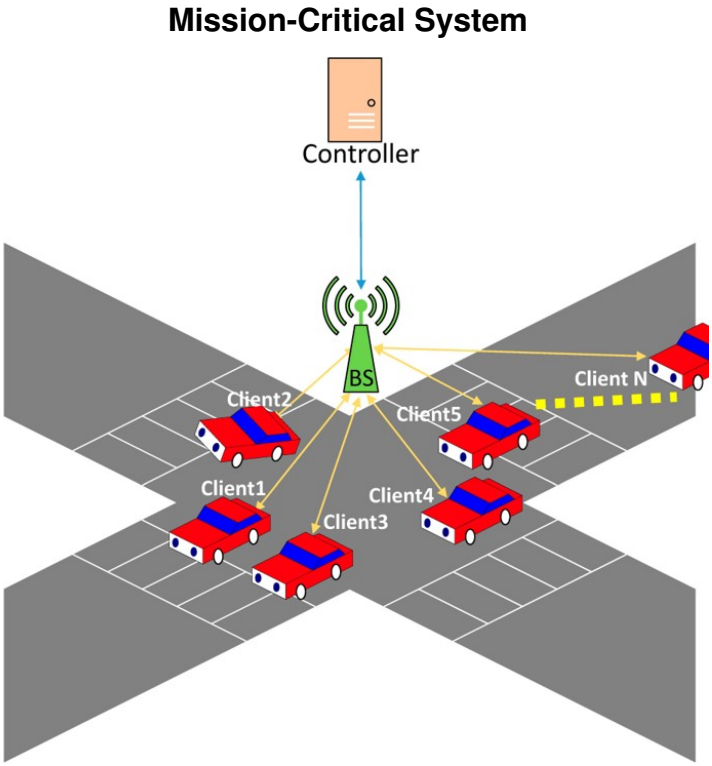



Figure 3.1: An autonomous logistics system at private park

scheduling decisions are made at the beginning of each time slot $t$. Each time slot has a duration of $\tau$ , which is equal to 1 ms or even smaller by using dynamic transmission time interval. Due to the network resource limitation, the network only allows a subset of UEs at a time to send packets to the central controller. The total link bandwidth is denoted as $B$ and $0 < B < N$. Each link was allocated a part (step=1) of channel bandwidth at time slot $t$, $b_n(t)$, and $0 < b_n(t) < B$. We assume that only $m$ UEs ($m \leq N | m, N, B \in \mathbb{N}$) are selected to send its packets at timeslot $t$.

Let $S_n(t)$ be a random variable that indicates the selection of UE $n$ by the gNB at a timeslot t, i.e. $S_n(t) = 1$. if the controller selects UE $n$ at timeslot $t$ , and 0 otherwise. When $S_n(t) = 1$, UE $n$ sends its packets to the controller via the base station. On the other hand, the UE cannot send any packets to the controller. The packet which is successfully received by the controller can be count as an AoI update message. We define all the possible scheduler selections, $\Sigma(S_n(t))_{n=1}^{N}$, as a set $\mathbb{S}$.

In addition, the number of packets generated at time slot $t$ can be denoted by $X_n(t)$, which follows a Poisson arrival process, specifically, M/M/1, with the average packet arrival rate per slot defined by $\alpha$. In M/M/1 systems, arrivals are Markovian (Poisson), which means the packet generation in each UE is triggered by random time intervals. The generated packets are stored at UE transmitters' queues, following the first-come-first-serve (FCFS) policy. The queue length is fixed and given when the system is initialized. The AoI of each UE $n$ is defined as the time elapsed since the generation of the latest packet that has departed the transmitter given by

$$A_n(t) := t - \max_i \{t_G^n(i)) | t_D^n(i)) \leq t\} \tag{3.1}$$

where $t_G^n(i))$ and $t_D^n(i)$ are the generated and departure instants of the $i$th packet of UE $n$. Due to the time-slotted assumption, we have a step increasing of AoI. If no packet generated from UE $n$ is received by the controller in a time slot $t$, the AoI, $A_n(t)$, was increased by 1, and so on. Otherwise if at a time slot $t$, a packet generated from UE $n$ is successfully received by the controller, the AoI will be updated as below:

$$A_n(t+1) = \begin{cases} A_n(t)+1 & if X_n(t)b_n(t)S_n(t) = 0 \\ 1 & if X_n(t)b_n(t)S_n(t) > 1 \end{cases} \tag{3.2}$$

In addition, the network utilization can be presented as below:

$$U_n(t) = \sum_{n=1}^{N} \frac{1}{1+e^{-(b_n(t)-X_n(t))}} S_n(t) \tag{3.3}$$

It is a sigmoid function range from 0 to 1. The function has a maximum value when the allocated bandwidth is just larger than the required bandwidth. And the network utilization $U_n(t)$ is the summation of all client utility value at time $t$. Our goal is to find suitable age-aware scheduling policies for a set of UEs so that their utilization can be maximized with AoI constraints satisfied. We can formulate our goal as the function below: In each time slot t, given the bandwidth requests $(b_n(t))_{n=1}^N$, select a schedule $S(t) := \Sigma(S_n(t))_{n=1}^N$ s.t.

$$S(t) \in \underset{S(t)\in\mathbb{S}}{argmax} \sum_{n=1}^{N} \frac{1}{1+e^{-(b_n(t)-X_n(t))}} S_n(t) - \beta A_n(t) \tag{3.4}$$

where $\beta \in \mathbb{R}$ is a scalar control parameter. However, Problem 3.4 is a non-linear integer programming (NLIP) which is generally complicated to solve [21]. Actually, the optimization variables in 3.4 include sequential decisions. Nowadays, it is shown that DRL has tremendous performance on the long-term sequential decision-making problems without human knowledge [36], [37]. At the same time, to realize the decisions in an automatic and zero-touch manner, DRL-based solutions are necessary. In addition, benefiting from deep neural networks, DRL is capable of handling high-dimensional observation-action spaces. These motivate us to propose policy-based model-free DRL solutions, discussed in the next section.

## 3.4. Proposed Machine Learning Model and Algorithm

In this article, we consider a learning-based approach to find a scheduling policy from network environment observations. Our approach is based on reinforcement learning (RL). In RL, an agent at base station interacts with a network environment to learn a scheduling policy without prior information. If the basestation has prior information such as the incoming traffic from the UEs or the AoI evolution, the basestation scheduler can find the optimal scheduling policy by the conventional algorithms. To deal with the unknown information in the network environment, an RL agent is introduced to gradually learn the scheduling policy from the network environment observations. We define the scheduling policy as $\pi(o_t, \theta_t)$ that returns a schedule $S(t)$ as an action $a_t$ to satisfied $\Sigma(b_n(t))_{n=1}^N \leq B$ . At the time-slotted system, every interaction between the agent and the network environment happends at the beginning of a time slot. In each interation, the agent sample the environment to get an

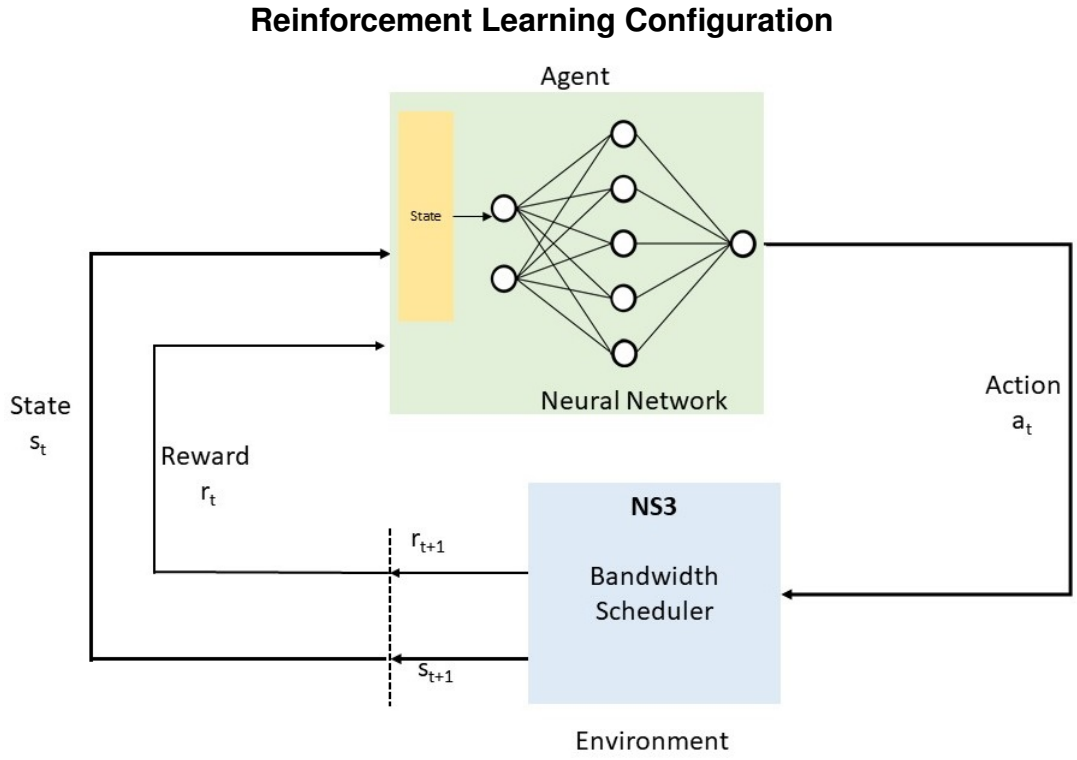



Figure 3.2: A Flowchart of Reinforcement Learning Method

obervation $o_t$ and performs an action $a_t$ based on the scheduling policy. After performing the action, the agent receives a reward $r_t$. Then the agent waits for the next time slot $t+1$ to interate with the network environment. The learning process repeats the interations continously to approximate the optimal scheduling policy which obtains the maximal reward. Hence, the objective of learning is to maximize the expected cumulative reward. The reward is considered to ensure a long-term system regret of the current action. As a result, the optimization scheduling can be defined as

$$J := \max_{\pi_k} \mathbb{E}[\frac{1}{1+e^{-(b_n(t)-X_n(t))}} S_n(t) - \beta A_n(t)] \tag{3.5}$$

$$\text{s.t. } \Sigma(b_n(t))_{n=1}^{N} \leq B, \pi_k = S(t)$$

Our RL agent used the Proximal Policy Optimization (PPO) Algorithm. This PPO algorithm has become one of the most widely used algorithms in RL because it has better sample efficiency [37].
As shown in Fig. 3.2, the learning process is a finite loop (length: $K \in \mathbb{N}$) which is starting from the agent. In the beginning, the agent will fetch an initial environment observation, $o_{t=0} = \Sigma(o_n(t=0))_{n=1}^{N}$, from the network environment. The agent observes a set of metrics from $o_n(t)$ including the buffer status, AoI value of every UE $n$; and the throughput achieved in the last $k$ interactions and feeds these values to the neural network, which will output the next action. The next action is defined by which UE is to be chosen for the next iteration $k+1$ at time $t+1$. The scheduling policy is transformed from the action obtained from the trained neural network. If a UE is selected to transmit packets then the corresponding bandwidth will be reserved for the UE. Next, the base station bandwidth will be allocated to UEs in terms

of a number of radio symbols. After the new scheduling deploys to all the UEs, a reward is then observed and fed back to the agent. The agent uses the reward information to train and improve its neural network model.
Our implementation of PPO algorithm in the scheduling problem is detailed in Algorithm 2. At each iteration $k$, the actor collect a time slot of observations. Then we construct the surrogate loss on these observations and optimiza it with minibatch SGD for $e$ epochs.

**Algorithm 2:** Proximal Policy Optimization Algorithm

**Input:** An initial policy with parameters $\theta$ and initial observation $o$
**for** $k = 0, 1, 2, \cdots$ *until* $k = K$ *or convergence* **do**
  Update age and bandwidth request based on observation
  Take scheduling action using policy $\pi = \pi(\theta)$
  Compute advantage estimate based on the value function
  Optimize surrogate function $\nabla J$ with respect to $\theta$ using $e$ epochs and minibatch size $\mathcal{B}$
  $\theta_k \leftarrow \theta_{k+1}$
**end**

## 3.5. Simulation Results and Discussions

In this section, we provide a simulation to illustrate the performance of the proposed scheme in Section 3.4. To compare the proposed scheme with the prior-art in a realistic cellular network, the simulation is performed in the network simulator (NS3) [38] In addition, we select the round-robin algorithm as baseline 1 and a proportional-fair algorithm as baseline 2 [39]. The simulation parameters in NS3 were listed in the table 3.1.

Table 3.1: Simulation Parameters

| Parameter Name | Value |
|---|---|
| Number of UEs $N$ | 20 |
| Slot Duration $\tau$ | 1 ms |
| Maximum Packet Size (d) | 2048 bytes |
| Numerology | 0 |
| Duplexing | TDD |
| Bandwidth | 20 MHz |
| Transmission Power | 20 dBm |
| Propagation Model | TR 38.901 |

### 3.5.1. Simulation Setup

We revise the NS3 LTE module to implement a 5G environment. The modulation and coding scheme and the resource block allocation are chosen based on the standard [40], [41], [42], [43], [44], and [25]. Users are distributed uniformly in the service area of 80 * 80 meters

but the base station is placed at a fixed location in the service area. The simulation time was chosen to be 900 seconds which ensure enough training samples were collected. For each UE n, the packet arrival rate is randomly set with a random distribution which has a mean frequency range between [50, 300]. Regarding the parameters of actor, we have the batch size $\mathcal{B}$ = 80. Then, we set the step size 0.6 and use a three-layer DNN with hyperbolic tangent (Tanh) activation function, Adam optimizer, and initial learning rate = 0.0003. On the other hand, the neural network structure of critic starts with an action-appended input layer. Then, it connects to a fully connected hidden layer and an output layer with N outputs. The critic also uses the Tanh activation function, Adam optimizer. Additionally, the initial critic learning rate is set to 0.001.

### 3.5.2. Simulation Results

The proposed scheme is assessed mainly in terms of the average AoI and average throughput of UEs. Based on those two metrics, the following scheduling schemes are compared in between:

1. Round-Robin (RR), where all RBs are evenly allocated to each UE;
2. Proportional-Fair (PF), where all RBs that are allocated depend on the known arrival traffic distribution;
3. Proximal Policy Optimization (PPO), where all RBs are allocated by the prediction from our RL agent

We now present our simulation results based on the two baseline algorithms and the Algorithm 2 to perform the reinforcement learning of the scheduler proposed in this study. Fig.3.3 first sketches the average AoI and Fig.3.4 presents the network throughput with the corresponding mean traffic generation frequency. It can be seen that the PPO algorithm outperforms other strategies at the heavy traffic condition due to the use of policy gradient. The RR algorithm has the worst AoI performance and the lowest throughput due to the full fairness for each UE. The PF algorithm has the best AoI and throughput performance at the beginning of heavy traffic because the data generating distribution had been a known parameter.

### 3.5.3. Discussions

If the network transmission capacity is larger than the total traffic generation, there is no network backlog. Thus, any scheduler can achieve the same AoI performance. Therefore, all the algorithms have the same AoI performance at low traffic conditions. On the other hand, when the total network traffic exceeds the network transmission capacity, the individual traffic from each UE starts backlogging. Because of the traffic backlog, the AoI starts increasing after the data generation rate reaching 120 Hz. We can easily find that the RR algorithm has the worst AoI performance and the lowest throughput because the RR algorithm does not consider the individual traffic load. Moreover, the PF algorithm has better AoI and throughput performance at the beginning of the network backlog due to the pre-configured traffic generation distribution. However, in the practical system, the traffic distribution may not be obtained. It can be seen in Fig.3.3 and Fig.3.4 that the PPO algorithm achieves the

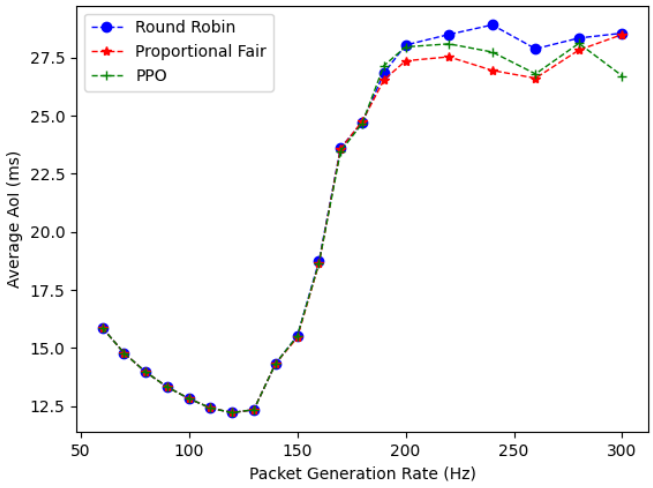



Figure 3.3: Average AoI in different Packet Gereration Rates

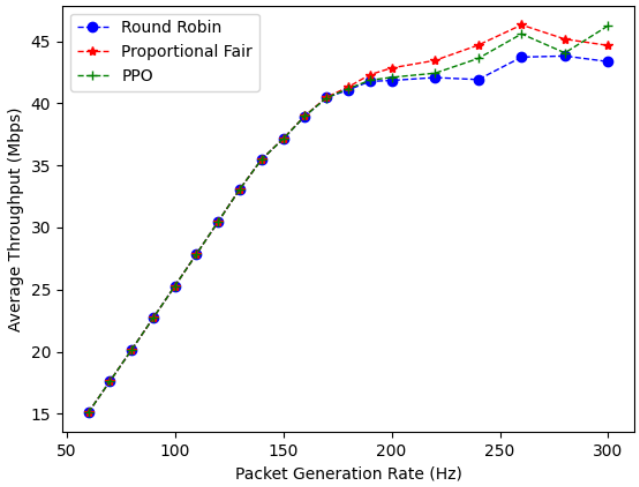



Figure 3.4: Average Throughput in different Packet Gereration Rates

highest throughput while keeps the lowest AoI. Since the PPO algorithm adapts the traffic generation and also the age punishment, the PPO agent can be more flexible to allocate the network bandwidth. And then the PPO agent can handle the random status in advance.

## 3.6. Conclusions

This research has characterized the relationship between AoI and scheduling and design a model-free deep reinforcement learning (DRL) method for optimizing the scheduler. DRL agent guided scheduler to deal with UEs age constraints and unknown wireless environment. The problem formulation and optimization process of AoI provided a theoretical basis for future studies on next-generation network radio resource management. Moreover, the proposed learning framework of centralized estimation and execution could further demonstrate the real network. For our future work, we will consider learning-based scheduling for heterogeneous network architecture.

# 4. Routing in airborne ad-hoc networks using Reinforcement Learning

## 4.1. Introduction

UAV are used in a wide range of applications, from tracking and monitoring animals in remote areas [45] to military applications [46]. In order to effectively accomplish the task, it can be necessary to deploy multiple UAVs, which are expected to coordinate actions in an autonomous fashion or execute direct instructions from a control center, over a certain area. In many scenarios, the UAVs need to exchange a relatively large amount of data with other members of the swarm and/or with the control station to support a given service. For example, distributed area monitoring/patrol applications may require the UAVs to stream high definition video or thermal camera recordings to the control station. Conversely, high bit rate data traffic demands wideband communication technologies (e.g., mmWave) that typically have limited coverage range, so that providing such services over wide areas may require multi-hop data connections, where the UAVs themselves can act as relays for other nodes in the network [47].
On the other hand, the UAVs and the control station also need to exchange light control traffic, which usually has strict latency and reliability constraints, but low bit rate requirements. For example, this control channel can be used by the UAVs to send periodic tracking updates to the control center, which can use these messages to track the UAVs' positions [48–50]. In these scenarios, the UAVs and the control center can use different technologies to carry information and signaling traffic, physically separating the data and control planes. The control traffic, in particular, can be carried by low-rate long-range communication technologies, such as LoRa [51], which can provide direct links between the UAVs and the control center. However, the randomness in the UAVs' movements makes the design of a multihop routing protocol for the data plane a challenging problem. Hence, in the following sections, we propose two different mechanisms to determine sustainable routing schemes while taking into account the position and mobility of the UAVs.

## 4.2. Sustainable and Reliable Multipath Routing

In this section, we design the Sustainable Multipath UAV Routing for Flying Ad-Hoc Networks (SMURF) protocol, a centralized multipath routing protocol for FANETs, which exploits the tracking information available at the control center to estimate the reliability of routes and select the set of routes that guarantees the overall highest reliability. The routing tables can be computed by the control center and propagated to UAVs, following the Software Defined Network (SDN) [52] paradigm.

### 4.2.1. Scenario Definition

We model a FANET as a time-varying graph $G(t) = (V, E(t))$, where $V$ is the set of UAVs in the network and $E(t)$ is the set of existing links at time $t$. Each drone $i$ is characterized by its position $\mathbf{x}_i(t) = (x_i(t), y_i(t), z_i(t))$ in the 3D space. We define the distance $d_{ij}(t) =$

$||\mathbf{x}_i(t) - \mathbf{x}_j(t)||_2$ between UAVs $i$ and $j$ as the Euclidean distance between the two drones. In the following, we consider a link $e_{ij}(t)$ as part of $E(t)$ if the distance between drones $i$ and $j$ is lower than the communication range $R$ (which depends on the communication technology used): $E(t) = \{e_{ij}(t) : d_{ij}(t) \leq R\}$. This simple assumption is justified by the fact that the drones will be in line of sight of each other in most practical scenarios; however, the model can be extended to more complex scenarios and propagation models with negligible effort.
In the following, we omit the time notation for readability; but it is intended that the operations described below need to be repeated at each time step. As the nodes in the network move and update their positions, routes are re-evaluated over time. We assume that the real position $\mathbf{x}_i(t)$ of each UAVs is not known by the control station, which keeps an estimate of its Probability Density Function (PDF) $p(\hat{\mathbf{x}}_i = \mathbf{x})$ instead. We can now define the link existence probability $p_t(e_{ij})$ at time $t$ as:

$$p_t(e_{ij}) = p_R(d_{ij}(t) \leq R) = \int_{\mathcal{B}_R(0)} p(\hat{\mathbf{x}}_i(t) - \hat{\mathbf{x}}_j(t) = \mathbf{x}) d\mathbf{x}, \tag{4.1}$$

where $\mathcal{B}_R(\mathbf{x})$ is the sphere with radius $R$ and center $\mathbf{x}$.
Let $\mathbf{e}$ denote a path i.e. series of adjacent links, from a source ($s$) to a destination ($d$), and $\mathcal{E}_{sd}$ be the set of all such routes. We then define the *optimal route* $\mathbf{e}^*_{sd}$ from $s$ to $d$ at time $t$ as the vector of links that maximize the overall route existence probability:

$$\mathbf{e}^*_{sd} = \arg\max_{\mathbf{e} \in \mathcal{E}_{sd}} \; p_t(\mathbf{e}), \tag{4.2}$$

If all links were independent, as typically assumed in the literature, we would have $P_t(\mathbf{e}) = \prod_{e \in \mathbf{e}} p_t(e)$. Note that loops are always avoided by definition, as a route with a loop always has a lower or equal probability of existence than the same route without the loop.
Taking a further step forward, in this paper we also model the *joint* existence probability of adjacent links, which slightly complicates the expression of $p_t(\mathbf{e})$, as explained later. Once we have found the optimal route $\mathbf{e}^*_{sd}$, we can define its *optimal backup* as the route $\mathbf{b}(\mathbf{e}^*_{sd})$ that maximizes the success probability when the first route fails (an event denoted by $\bar{\mathbf{e}}^*_{sd}$):

$$\mathbf{b}(\mathbf{e}^*_{sd}) = \arg\max_{\mathbf{b} \in \mathcal{E}_{sd} | \bar{\mathbf{e}}^*_{sd}} p_t(\mathbf{b} | \bar{\mathbf{e}}^*_{sd}). \tag{4.3}$$

where $\mathcal{E}_{sd}|\bar{\mathbf{e}}^*$ indicates the set of viable paths from source $s$ to destination $d$, given that the primary path $\mathbf{e}^*_{sd}(t)$ is disrupted. We can generalize the notion of backup route to compute the optimal backup to a set of existing routes, considering the best route if the existing ones all fail. In the following subsections, we report the derivation of the route existence probability, along with the SMURF algorithm to calculate the primary and backup routes.

### 4.2.2. Link existence probability

We now assume that the estimated position distribution for each node is a multivariate Gaussian distribution, $\hat{\mathbf{x}}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. This assumption is justified if the tracking system uses Kalman filtering, as common in the literature [50]. We also assume that the positions of the UAVs are mutually independent. The covariance matrix $\boldsymbol{\Sigma}_i$ is not necessarily diagonal, as we expect a higher error in the direction of movement of UAVs. The PDF of the position for

the node $i$ is given by the multivariate normal distribution:

$$p_i(\hat{\mathbf{x}}) = \frac{1}{2\pi\sqrt{|\mathbf{\Sigma}_i|}} e^{\left(-\frac{1}{2}(\hat{\mathbf{x}}-\boldsymbol{\mu}_i)^H[\mathbf{\Sigma}_i]^{-1}(\hat{\mathbf{x}}-\boldsymbol{\mu}_i)\right)}, \tag{4.4}$$

where $\mathbf{x}^H$ is the Hermitian of vector $\mathbf{x}$. Hence, the link existence probability as expressed in (4.1) for each timestep $t$ is given by:

$$p_t(e_{ij}) = \int_{\mathcal{B}_R(0)} \frac{e^{\left(-\frac{1}{2}(\mathbf{x}-(\boldsymbol{\mu}_{i-j}))^H(\mathbf{\Sigma}_{i-j})^{-1}(\mathbf{x}-\boldsymbol{\mu}_{i-j})\right)}}{2\pi\sqrt{|\mathbf{\Sigma}_{i-j}|}} d\mathbf{x}, \tag{4.5}$$

where $\boldsymbol{\mu}_{i-j} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j$ and $\mathbf{\Sigma}_{i-j} = \mathbf{\Sigma}_i + \mathbf{\Sigma}_j$, as the difference of two independent multivariate Gaussian random variables is itself multivariate Gaussian with those parameters. This integral cannot be solved analytically, but it can be computed efficiently using numerical methods. We can extend this to the conditioned link existence probability for a link that shares a node with one that is known to exist, $p_t(e_{ij}|e_{jk})$:

$$p_t(e_{ij}|e_{jk}) = \int_{\mathbb{R}^3} p_j(\hat{\mathbf{x}}_j) \int_{\mathcal{B}_R(\hat{\mathbf{x}}_j)} \int_{\mathcal{B}_R(\hat{\mathbf{x}}_j)} p_i(\hat{\mathbf{x}}_i) p_k(\hat{\mathbf{x}}_k) d\hat{\mathbf{x}}_i d\hat{\mathbf{x}}_k d\hat{\mathbf{x}}_j \tag{4.6}$$

In the same way, we can derive the probability that link $e_{ij}$ exists if $e_{jk}$ does not, just by integrating $\hat{\mathbf{x}}_k$ outside the region $\mathcal{B}_R(\hat{\mathbf{x}}_j)$. All the computations above are so reduced to the calculation of multivariate Gaussian integrals, which can be performed efficiently with well-known numerical methods [53, 54]. Since the routing algorithm is executed by the control station, which should have sufficient computational power, there are no issues with the limited battery and computational capabilities of the UAVs.

### 4.2.3. Routing Metric Calculation

In order to compute the existence probability of a route, we need to consider all of its links jointly. In the following, we simplify the probability calculation by assuming that links that do not share nodes are independent. This simplification is justified by the fact that each UAVs' movement is assumed to be independent, so that it is reasonable to expect that the mutual dependence of links that do not share nodes is negligible. Considering more faithful approximations is possible, but are left for future work. Therefore, for a path $e = [e_{12}, e_{23}, \dots, e_{n-2,n-1}, e_{n-1,n}]$ we can write:

$$p_t(\mathbf{e}) \simeq p_t(e_{12}) p_t(e_{23}|e_{12}) \dots p_t(e_{n-1,n}|e_{n-2,n-1}). \tag{4.7}$$

Furthermore, the calculations above refer to a single timestep, but the Kalman filter-based location prediction can be extended to the next $T$ timesteps. We can then extend the prediction to a vector:

$$\mathbf{p}_t^{(T)}(e_{ij}) = \{p_t(e_{ij}), p_{t+1}(e_{ij}), \dots, p_{t+T-1}(e_{ij})\}. \tag{4.8}$$

In the same way, we can compute $\mathbf{p}_t^{(T)}(\mathbf{e})$ for any route. Naturally, this is a slight simplification, as the error on the position of the nodes is not independent over time, but cumulates, so there is a positive correlation between the existence probability at one step and the next. However, in the interest of computational simplicity, we make this further simplification.

To compute the routing metric, we could consider any function of the vector $\mathbf{p}_t^{(T)}(\mathbf{e})$, but we take the average over the time horizon for simplicity's sake. Since the route existence probability at any given timestep is computed considering only the dependence on the immediately previous link, we can efficiently build a spanning tree by using the negative logarithm of the link existence probability as a routing metric:

$$W(e_{jk}|e_{ij}) = -\log_{10}\left(\frac{\sum_{\tau=t}^{t+T-1} p_\tau(e_{jk}|e_{ij})}{T}\right). \tag{4.9}$$

In this way, links with a higher existence probability over the whole timeframe are chosen by the routing algorithm.

### 4.2.4. Backup Routes Calculation

By definition, the primary route is the one with the highest probability of existence, but it might still fail in a dynamic scenario. For this reason, we consider a set of backup routes, which can be selected in case the primary one fails. This can significantly increase the reliability of the transmission if the UAVs swarm is dense enough, as there will be multiple viable routes to the destination. In order to calculate the optimal backup, we consider single-link failures and define the conditional path existence probability, given the link is down, as follows:

$$\mathbf{b}_i(\mathbf{e}^*_{sd}) = \underset{\mathbf{b}\in\mathcal{E}_{sd}|\bar{\mathbf{e}}^*_{sd}}{\arg\max}\, P(\mathbf{b}|\bar{e}^*_{i,sd}). \tag{4.10}$$

If the $i$-th link in the primary route does not exist, we can compute the conditional joint position PDF of nodes $i$ and $i+1$, as:

$$p((\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_{i+1}) = (\mathbf{x}, \mathbf{y})|\bar{e}_{i,i+1}) = \begin{cases} \frac{p_i(\hat{\mathbf{x}})p_{i+1}(\hat{\mathbf{y}})}{1-P(e_{i,i+1})}, & \hat{\mathbf{y}} \in \mathcal{B}_R(\hat{\mathbf{x}}); \\ 0, & \hat{\mathbf{y}} \notin \mathcal{B}_R(\hat{\mathbf{x}}). \end{cases} \tag{4.11}$$

We can then adjust other links' existence probabilities with such a conditional PDF and rebuild the spanning tree to find the backup route. After computing the optimal backup $\mathbf{b}_i(\mathbf{e}^*)$ for each link failure, we compare them by considering the probability of the link failing. The optimal backup route is then given by:

$$\tilde{\mathbf{b}}(\mathbf{e}^*_{sd}) = \underset{\mathbf{b}_1,\ldots,\mathbf{b}_{N(\mathbf{e}^*_{sd})-1}}{\arg\max}\; P(\mathbf{b}|\bar{e}^*_{i,sd})(1 - P(e_{i,sd})), \tag{4.12}$$

where $N(\mathbf{e})$ is the number of nodes in route $\mathbf{e}$. We compute successive backups by considering single broken links in the primary route to simplify the calculation, even though the result is slightly suboptimal. The calculation of the backup routes can be extended to longer time horizons in the same way we outlined for the primary route.

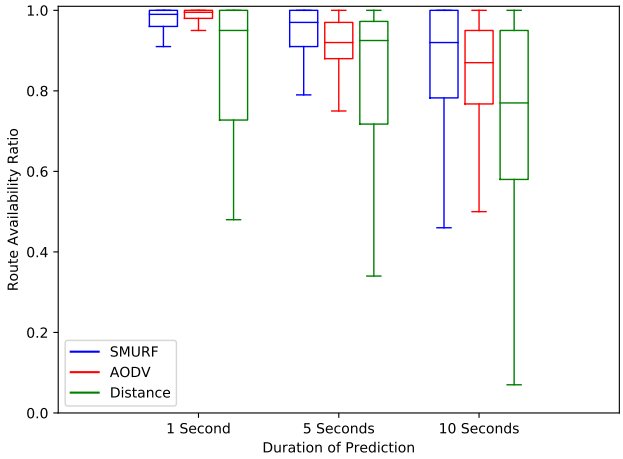



Figure 4.1: Route Availability Ratio for different protocols over different prediction intervals

### 4.2.5. Results

To examine the performance of the protocols, we use three different metrics:

$$\text{Route Availability Ratio} = \frac{\text{Route Availability Time}}{\text{Total Simulation Time}}$$

$$\text{Route Establishment Overhead} = \frac{\text{Number of Routing Packets}}{\text{Number of UAVs} \times \text{Total Simulation Time}}$$

$$\text{Route Sustainability Criterion} = \frac{\text{Route Availability Ratio}}{\text{Route Establishment Overhead}}$$

The Route Availability Ratio (RAR) is used to determine the availability of at least one path between source and destination, the Route Establishment Overhead (REO) is used to determine the overall number of routing packets exchanged per second for any network size, and the Route Sustainability Criterion (RSC) is used to determine the availability of at least one route from source to destination with respect to the number of exchanged routing packets per second. This metric essentially shows the cost incurred to provide a certain route availability, thereby showing the sustainability of the network. Two additional protocols, namely, AODV and Distance-based SDN routing, are considered along with SMURF. The AODV simulation is designed with the assumption that each UAV has perfect information regarding the positions of all other UAVs in the network at the time of network route computation, thereby knowing beforehand the failures that can potentially occur in a route. This is not an entirely realistic assumption, which acts as an upper bound for the real performance of AODV in this scenario. The Distance-based SDN routing, henceforth referred to as Distance, uses the position information obtained by the central controller from the UAVs and determines the shortest path between the source and the destination based on the geographical distance between two UAVs. Figure. 4.1, 4.2 and 4.3 refer to the RAR for the different protocols over different prediction intervals, REO for the different protocols over a prediction interval of 10 seconds and RSC for the different protocols over a prediction interval of 10 seconds respectively. As visible from Figure 4.1, SMURF outperforms both Distance and AODV for 5 seconds and 10 seconds prediction interval. SMURF also has a comparable performance to AODV for 1 second prediction interval and outperforms Distance for 1 second prediction interval. This is because AODV has perfect position information thereby being able to perform

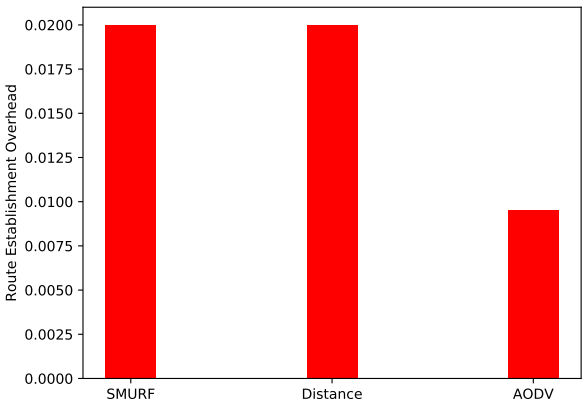



Figure 4.2: Route Establishment Overhead for different protocols over prediction interval of 10 seconds

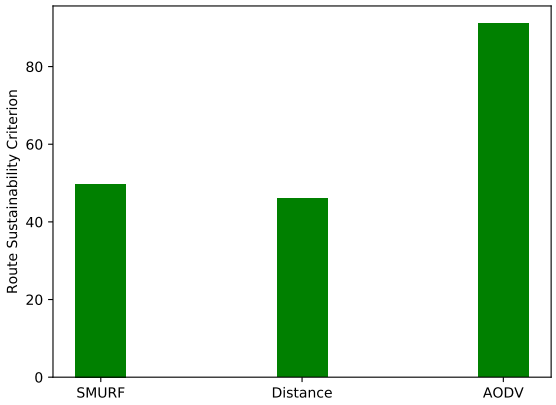



Figure 4.3: Route Sustainability Criterion for different protocols over prediction interval of 10 seconds

well in prediction over a short duration. SMURF outperforming Distance and AODV comes at a cost of high REO as shown in Figure 4.2. As visible, the REO of SMURF and Distance is twice that of AODV. This generally is not a big issue in the system as the SDN control packets attributing to the REO in SMURF and Distance are sent over a long range technology (such as LoRA) thereby not contributing to the overhead in the data plane while the packets sent by AODV interfere with the operations of the data plane. Hence, due to higher REO for SMURF and Distance, the RSC of SMURF and AODV is smaller than that of AODV as visible in Figure 4.3. The reason for high REO and low RSC for SMURF compared to AODV is due to the periodic updates of calculated routes to all the nodes in the network. Hence, to design a more adaptive mechanism for updating the routes to the UAVs, we devised a RL mechanism explained in the next section.

## 4.3. Sustainable Multihop Route Design: A RL approach

SMURF defines a multipath routing scheme for UAVs but requires periodic updates to the UAVs to reconfigure the routes in the network. This causes an increase in REO and decrease in RSC compared to AODV protocol. This motivated the need to adapt the route updates to the UAVs so as to reduce REO while maximising the RAR thereby increasing the RSC of the system. To devise this, we propose an RL mechanism that can adapt and learn the frequency of routing updates.

### 4.3.1. Scenario Definition

The scenario involves a network of $U$ UAVs, which need to communicate through multiple wireless hops. We assume that the successful transmission between drones $i$ and $j$ is represented by a Bernoulli random variable $E_{i,j}(t)$, whose probability is an unknown function of their positions $P(E_{i,j}(t)|\mathbf{x}_i(t), \mathbf{x}_j(t))$. According to the SDN paradigm, a centralized controller sets up the routes between the nodes and propagates them. The problem is to determine the routes that are most stable, considering that sending route update messages has a cost. In the following, we do not consider the load on each link, but limit our analysis to the existence of the links. We formulate the model as a Markov Decision Process (MDP), which is determined by a state space, an action space, and a reward function. We consider the routing for each flow, identified by its source $i$ and destination $j$, as a separate and independent problem.
First, we define a route between a source $i$ and a destination $j$ as a vector $\boldsymbol{\rho}$ of $N_\rho$ relays. The route is composed of the link between $i$ and $\rho_1$, the link between $\rho_1$ and $\rho_2$, and so on, until the last link between $\rho_{N_\rho}$ and $j$ completes the path. As we explained before, the reward function is given by the existence of the route (i.e., the existence of all the links that compose it), minus a constant cost $\theta$ if the controller changes the route:

$$r(\boldsymbol{\rho}(t), \boldsymbol{\rho}(t-1), \mathbf{X}(t)) = E_{i,\rho_1} E_{\rho_{N_\rho},j} \prod_{i=1}^{N_\rho - 1} E_{\rho_i,\rho_i+1} - \theta(1 - \delta(\boldsymbol{\rho}(t), \boldsymbol{\rho}(t-1))), \tag{4.13}$$

where $\mathbf{X}(t)$ is the matrix containing the position of all nodes at time $t$ and the function $\delta(\mathbf{x}, \mathbf{y})$ is equal to 1 if the vector $\mathbf{x}$ is exactly identical to $\mathbf{y}$ and 0 otherwise.
We now define the action space. If we limit the number of relays $N_\rho$ to a maximum $N_{\max}$, we get that the space $A$ of possible actions is equivalent to all possible routes in the network:

$$A = \bigcup_{N_\rho=0}^{N_{\max}} \sigma(\{1, \dots, U\} \setminus \{i, j\}, N_\rho), \tag{4.14}$$

where $\sigma(M, k)$ is the set of non-repeating permutations of length $k$ of the elements of the set $M$. Note that the empty vector $\varnothing$ is a possible action, as it indicates that no relays are used and the route is just the direct link between source and destination. The size of the set of actions grows exponentially with $N_{\max}$, so it might be advisable to reduce it using pruning techniques to limit the number of candidate relays.
We now need to define the state space. At time $t$, the state of UAV $i$ can entirely represented by the output of each Unscented Kalman Filter (UKF), i.e., by the state vector $\mathbf{z}_i(t-1) \in \mathbb{R}^k$,

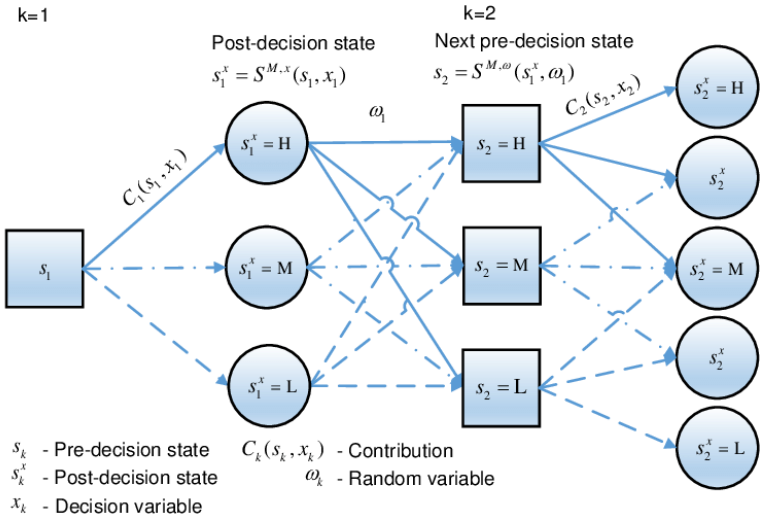



Figure 4.4: Transition diagram with post-decision states.

which can be used to predict $\mathbf{x}_i(t)$. The matrix $\mathbf{Z}$ containing all the UKF outputs then represents the state of the system at time $t$, along with the previously active route $\rho(t-1)$. The state space is then $S = \mathbb{R}^{Uk} \times A$. It contains an infinite number of possible states, so the problem requires the use of DQL. s We remind the reader that the objective of a reinforcement learning approach is to find the policy $\pi^*$ that maximizes the long-term expected reward, defined as:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} \int_S \sum_{a_\tau \in A} r(s_\tau, a_\tau)\pi(a_\tau|s_\tau)P(s_\tau|\pi, s_t)ds_\tau\right] \forall s_t, \tag{4.15}$$

where $\Pi$ is the set of possible functions $\pi : S \mapsto P(A)$, which map each state to a probability distribution over the action space, and $\gamma \in [0, 1)$ is an exponential discount factor.

### 4.3.2. Post-decision states

To simplify learning the policy for the RL problem defined, the use of Post Decision State (PDS) models [55] can be incredibly lucrative. The idea is to define an intermediate state representing the agent immediately after it makes a decision, but before the effects of its action can be evaluated. Instead of taking action $a_t$ in state $s_t$ and transitioning directly to $s_{t+1}$, it goes through an intermediate transition to $s'_t$, which is shown in Fig. 4.4. This transition is deterministic, and can separate the deterministic effects of an action in a state from its consequences in the environment. The transition from the post-decision state to the next pre-decision state is then purely the effect of the environment, as the agent has already made its decision and cannot affect the transition anymore.
In our case, we define the post-decision state space $S'$ as being equal to $S$. Instead of being composed of $\mathbf{Z}(t)$ and $\rho(t-1)$, the post-decision state contains $\mathbf{Z}(t)$ and $\rho(t)$. We can now look at the reward and divide it in two:

$$r(s(t), \rho(t)) = v(s'_t) - \theta(1 - \delta(\rho(t), \rho(t-1))), \tag{4.16}$$

where the PDS value function $v(s'_t)$ is given by:

$$v(s'_t) = E_{i,\rho_1(t)} E_{\rho_{N_{\rho(t)}},j} \prod_{i=1}^{N_{\rho(t)}-1} E_{\rho_i(t),\rho_i+1(t)}. \tag{4.17}$$

Since we know that $s'(t)$ is given by $\mathbf{V}(t)$ and $\rho(t)$, it is a deterministic function of the state and action. We can reformulate the optimal policy formulation given in (4.15) to use PDSs:

$$\pi^* = \arg\max_{\pi\in\Pi} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} \int_S \sum_{a_\tau\in A} \left(v(s'_\tau) - \theta(1-\delta(a_\tau, a_{\tau-1}))\right) \pi(s_\tau, a_\tau) P(s_\tau|\pi, s_t) ds_\tau\right] \forall s_t. \tag{4.18}$$

We can define the reward in terms of the PDS value function $V(s'_t, \pi)$, which is given by:

$$V(s'_t, \pi) = \mathbb{E}[v(s'_t)] + \sum_{a_{t+1}\in A} \int_S P(s_{t+1}|s'_t)\pi(s_{t+1}, a_{t+1}) \left(V(s'_{t+1} - \theta(1-\delta(a_t, a_{t+1}))\right) ds_{t+1}. \tag{4.19}$$

The optimal policy definition then becomes:

$$\pi^* = \arg\max_{\pi\in\Pi} \sum_{a_t\in A} \pi(s_t, a_t) \left(V(s'_t) - \theta(1-\delta(a_t, a_{t-1}))\right) \forall s_t. \tag{4.20}$$

We can then set up learning based on the PDS formulation, simplifying the problem.

## 4.4. Summary

Hence, in this work, we presented the usage of SMURF protocol over a time horizon and devising the route availability with respect to other protocols. We also showed the establishment cost that is incurred while propagating the calculated routes in the system. Additionally, to minimize the cost incurred for route establishment by SMURF, we introduce an adaptive RL mechanism which can learn about the position and mobility of the UAVs in the network and thereby update routes only when there is a need to update instead of a periodic update which was necessary for SMURF. Furthermore, this mechanism is currently being implemented to be simulated over the same UAV network.

# 5. Emerging a MAC protocol with Multi-Agent Reinforcement Learning

In this chapter, we propose a new framework to enable a MAC protocol to be emerged by the network nodes in a multiple access scenario. The comparison with a contention-free baseline shows that our framework achieves a superior performance in terms of goodput and can effectively be used to learn a new protocol.

## 5.1. Introduction

The current 5G networks are designed to support a wide range of services, including Enhanced Mobile Broadband (eMBB), Ultra Reliable and Low Latency Communications (uRLLC) and mMTC, which in turn will support an important growth in the number of applications. This upsurge in novel services and applications is expected to also happen with 6G [56]. This heterogeneity of wireless networks may represent a challenge to protocol design. Therefore, protocols tailored to specific applications may perform better than general-purpose solutions [57].

ML can be used to design protocols, boost the network capacity [58] and reduce the efforts and costs for future standardization [59]. It is possible to view a protocol as the language of a network, since the network nodes have to negotiate how to transmit data by exchanging messages. Then, the idea of emerging a new protocol would be similar to emerging communication between the network nodes. RL is one category of ML that provides the means to reach this goal.

During the last years, research on how to emerge communication in order to achieve collaboration between multiple agents received a growing attention [60–62]. This growth partly relies on recent advances in MARL for cooperative problems [63]. Learning to cooperate by leveraging communication is about teaching agents to either learn existing natural languages or to emerge a fully new communication protocol that would help them collaborate to solve a task.

**Contribution:** Our proposal is to leverage cooperative MARL augmented with communication to allow a fully new MAC protocol to emerge. The idea of learning a given protocol has already been addressed in a previous work [64], but to the best of the authors' knowledge, there is no previous work on studying the emergence of a new MAC protocol (signalling included) with MARL. In the future, this idea may be used to develop application-tailored protocols that could perform better than the human-designed ones.

This chapter is structured as follows. In Section 5.2, we briefly review the literature. In Section 5.3, we give a short background overview of MARL and the algorithm used in this chapter. Section 5.4 describes the system model used and in Section 5.5, we present a new framework allowing the emergence of MAC protocols with MARL. Finally, Section 5.6 illustrates the performance of our algorithm with our numerical results, where we compare the proposed solution with a baseline. The main conclusions are drawn in Section 5.7.

## 5.2. Related Work

Several papers have applied RL to the MAC layer, mostly to solve RRM problems such as scheduling ( [65, 66]) and dynamic spectrum access ( [67, 68]).
In [64], MARL is used to learn a predefined protocol and a new channel access policy. This is done by having a Base Station (BS) which uses a predefined protocol while the UEs are RL agents. The UEs are trained to learn the signaling and how to access the channel without any prior knowledge. This way, they can learn their own channel-access policy, while respecting the target signaling policy. However, in this case the agents only learn to use an already known MAC signaling, rather than developing a new one.
In [57], a framework to design a protocol is proposed by considering the different functions a MAC protocol must perform. An RL agent designs a protocol by selecting which building function to use according to the network conditions. However, in this case, the RL agent still has a prior knowledge due to the use of the predefined protocol functions.
In [69, 70], cooperative MARL is used to emerge a coding scheme by joint learning of communication and cooperation to solve a task with the help of a noisy communication channel. The proposition of both works is to emerge a coding scheme that is tailored to the application. None of these works address the question of learning a new signaling protocol.

## 5.3. Background on MARL

RL is an area of ML that aims to find the best behavior for an agent interacting with a dynamic environment in order to maximize a notion of accumulated reward [71]. The goal of the RL agent is to find the best policy, which is the mapping of the perceived states to the actions to be taken. The action-value function $Q^{\pi}(s_t, a_t)$, also known as Q-function, is the overall expected reward for taking action $a_t$ in state $s_t$ and then following a policy $\pi$.
MARL is an extension of RL for to multi-agent systems (MAS), where multiple agents interact with a system, i.e the environment. In this chapter, we use the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) formulation [72], augmented with communication. A Dec-POMDP for $n$ agents is defined by the global state space $\mathcal{S}$, an action space $\mathcal{A}_1, \ldots, \mathcal{A}_n$, and an observation space $\mathcal{O}_1, \ldots, \mathcal{O}_n$ for each agent. In Dec-POMDP, the agent observation does not fully describe the environment state. All agents share the same reward and the action space of each agent is subdivided into one environment action space and a communication action space. The communication action represents the message sent by an agent and it does not affect the environment directly, but it may be passed to other agents. This formulation is shown in Fig. 5.1, where $o_i$ represents the observation received by the $i^{\text{th}}$ agent, $r$ represents the reward, $a_i$ and $c_i$ represent the environment and communication actions, respectively. In this chapter, the agent internal state $x_i$ may comprise not only the agent's current observation, but also previous observations, actions and received messages.
MARL introduces some new challenges, such as partial observability and non-stationarity [73]. In this chapter, we adopt the multi-agent deep deterministic policy gradient (MADDPG) algorithm [74], an extension of the Deep Deterministic Policy Gradient (DDPG) algorithm [75] to multi-agent problems with Centralized Training and Decentralized Execution (CTDE). It addresses the non-stationarity problem by using a centralized critic. Each agent has an actor

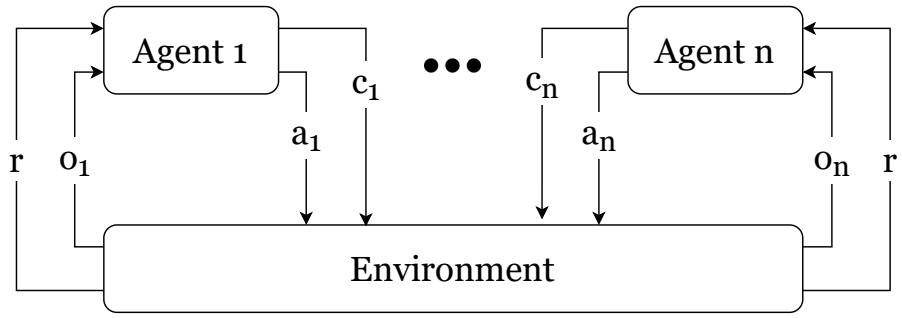



Figure 5.1: Cooperative MA-RL scheme with communication,

network that depends only on its own agent's state in order to learn a decentralized policy $\mu_i$ with parameters $\theta_i$. During the training, each agent has a centralized critic that receives the agent states and actions of all agents in order to learn a joint action value function $Q_i(x, a)$ with parameters $\varphi_i$, where $x = (x_1, x_2, \dots, x_n)$ is a vector containing all the agents' states and $a = (a_1, a_2, \dots, a_n)$ contains the actions taken by all of the agents.
The critic network parameters $\varphi$ are updated by minimizing the loss given by the temporal-difference error

$$L^i \coloneqq \mathbb{E}_{x,a,r,x' \sim \mathcal{D}} \left[ y^i - Q_i(x, a_1, \dots, a_n; \varphi_i) \right] \tag{5.1}$$

where $\mathcal{D}$ denotes the experience replay buffer in which the transition tuples $(x, a, r, x')$ are stored, $Q'$ and $\mu'$ represent the target critic network and the value of the target actor network, with parameters $\theta'$ and $\varphi'$, respectively, and $y^i$ is the temporal-difference target, given by

$$y^i \coloneqq r + \gamma Q'_i(x', a'_1, \dots, a'_n; \varphi'_i)|_{a'_k = \mu'_k(x_k)} \tag{5.2}$$

where $\gamma$ is the discount factor. The actor network parameters $\theta$ are updated using the sampled policy gradient

$$\nabla_{\theta_i} J = \mathbb{E}_{x,a \sim \mathcal{D}} \left[ \nabla_{a_i} Q_i(x, a) \nabla_{\theta_i} \mu_i(x_i) \mid a_i = \mu_i(x_i) \right]. \tag{5.3}$$

The target networks parameters are updated as

$$\varphi'_i \leftarrow \tau \varphi_i + (1 - \tau) \varphi'_i \tag{5.4}$$

$$\theta'_i \leftarrow \tau \theta_i + (1 - \tau) \theta'_i \tag{5.5}$$

where $\tau \in [0, 1]$ is the soft-update parameter. Smaller values of $\tau$ lead to slow target network changes and are generally preferred [75].

## 5.4. System Model

We consider a single cell with a BS serving $L$ UEs operating according to a TDMA scheme, where each UE needs to deliver $P$ SDUs (sdus) to the BS. We assume that each MAC Protocol Data Unit (PDU) contains only one sdu. The network nodes can communicate, i.e. exchange information, using messages through the control channels. In the rest of this chapter, we use the expressions UE and BS to refer to the UE MAC agent and the BS MAC agent, respectively.

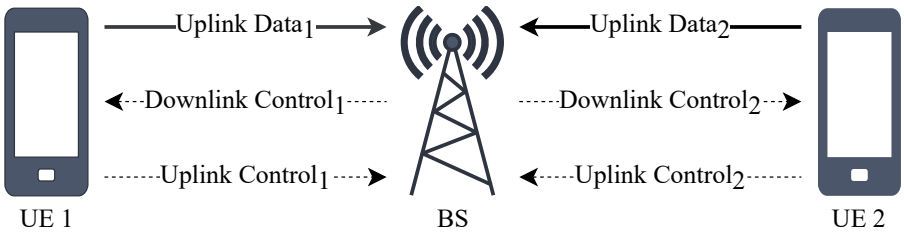



Figure 5.2: System model example with two UEs.

The channel for the uplink data transmission is modeled as a packet erasure channel, where a Transport Block (TB) is incorrectly received with a probability given by a TBLER. The Downlink Control Messages (DCMs) and Uplink Control Messages (UCMs) are transmitted over the Downlink (DL) and Uplink (UL) control channels, which are assumed to be error free and without any contention or collision.
We assume that the sets of possible DL and UL control messages have cardinality $D$ and $U$, respectively. For example, the UCMs in an UL control vocabulary of size $D = 8$ would have a bitlength of $\log_2 D = 3$. This exchange is shown in Fig. 5.2, where dashed lines indicate control information and solid lines indicate user data.
Each UE has a transmission buffer of capacity $B$ MAC sdus that starts empty. At each time step $t$, a new sdu is added to the buffer with probability $p_a$, until a maximum of $P$ sdus have been generated for each UE.
At each time step $t$, the BS can send one control message to each UE and each UE can send one control message to the BS while being able to send data PDUs through the Uplink Shared Channel (UL-SCH). Furthermore, the UEs can also delete a sdu from the buffer at each time step.
The transmission task is considered finished once all sdus are received and all transmission buffers are empty. We define the goodput $G$ (in sdus/TTIs) as the number of MAC sdus received by the BS per unit of time, without considering the retransmissions:

$$G = \frac{N_{\text{RX}}}{N_{\text{TTIs}}} \tag{5.6}$$

where $N_{\text{RX}}$ represents the number of sdus received and $N_{\text{TTIs}}$ is the total time taken to finish the transmission task in Transmission Time Intervals (TTIs). The delivery-rate $\Gamma_{\text{RX}}$ is the percentage of sdus correctly received by the BS:

$$\Gamma = \frac{N_{\text{TX}}}{PL}. \tag{5.7}$$

## 5.5. Emerging a MAC Protocol with MARL

### 5.5.1. MARL Formulation

We can formulate the problem defined above as a MARL cooperative task, where the MAC layers of the network nodes (UEs and BS) are RL agents that need to learn how to communicate with each other to solve an uplink transmission task. In addition, the UE agents need

to learn when to send data through the UL-SCH and when to delete an sdu, in other words, to learn how to correctly manage the buffer.
In order to decide how to act, an agent needs to consider the messages received from the other agents. In addition, the UEs also take into account their buffer status when taking actions, while the BS takes into account the state of the UL-SCH, i.e idle, busy or collision-free reception.
We use the following notations:

- $o_t^{\mathrm{u}}$: Observation received by the $u^{\text{th}}$ UE at time step $t$.
- $o_t^{\mathrm{b}}$: Observation received by the BS at time step $t$.
- $n_t^{\mathrm{u}}$: The UCM sent from the $u^{\text{th}}$ UE at time step $t$.
- $m_t^{\mathrm{u}}$: The DCM sent to the $u^{\text{th}}$ UE at time step $t$.
- $a_t^{\mathrm{u}}$: Environment action of the $u^{\text{th}}$ UE at time step $t$.
- $x_t^{\mathrm{u}}$: Agent internal state of the $u^{\text{th}}$ UE at time step $t$.
- $x_t^{\mathrm{b}}$: Agent internal state of the BS at time step $t$.

The observation $o_t^{\mathrm{u}} \in \{0, \dots, B\}$ is a integer representing the number of sdus in the buffer of the UE $u$ at that time $t$. Similarly, the observation $o_t^{\mathrm{b}}$ received by the BS is a discrete variable with $L+2$ possible states:

$$o_t^{\mathrm{b}} = \begin{cases} 0, & \text{if the UL-SCH is idle} \\ \mathrm{u}, & \text{if the UL-SCH is detected busy with a single PDU from UE u, correctly decoded} \\ L+1, & \text{non-decodable energy in the UL-SCH.} \end{cases} \tag{5.8}$$

where $\mathrm{u} \in \{0, \dots, L\}$. The environment action $a_t^{\mathrm{u}} \in \{0, 1, 2\}$ is interpreted as follows:

$$a_t^{\mathrm{u}} = \begin{cases} 0\text{: do nothing} \\ 1\text{: transmit the oldest sdu in the buffer} \\ 2\text{: delete the oldest sdu in the buffer} \end{cases} \tag{5.9}$$

We assume the episode ends when all the sdus are correctly received by the BS or when a maximum number of steps $t_{\max}$ is reached. The reward given at each time step is:

$$r_t = \begin{cases} +\rho, & \text{if a new sdu was received by the BS} \\ -\rho, & \text{if a UE deleted a sdu that has not been received by the BS} \\ -1, & \text{else,} \end{cases} \tag{5.10}$$

where $\rho$ is a positive integer. This choice of reward is possible by leveraging the CTDE. During the centralized training, a centralized reward system can be used to observe the buffers of the BS and UEs in order to assign the reward.

### 5.5.2. Training Algorithm

The proposed RL solution is based on the MADDPG algorithm [74]. Each entity of the system has its own actor network which outputs the action of an agent given its state. Each agent also has a centralized critic network which outputs the Q-value given the actions and states of all agents. The critic networks are only used during the centralized training.
The actor and critic networks have the same architecture; a fully connected Multilayer Perceptron (MLP) with two hidden layers, of 64 neurons each. The activation function of all hidden layers is the Rectified Linear Unit (ReLU).
The agent state at time step $t$ is a tuple comprising the most recent $k$ observations, actions and received messages:

- UE u: $x_t^{\mathrm{u}} = (o_t^{\mathrm{u}}, \dots, o_{t-k}^{\mathrm{u}}, a_t^{\mathrm{u}}, \dots, a_{t-k}^{\mathrm{u}}, n_t^{\mathrm{u}}, \dots, n_{t-k}^{\mathrm{u}}, m_t^{\mathrm{u}}, \dots, m_{t-k}^{\mathrm{u}})$
- BS: $x_t^{\mathrm{b}} = (o_t^{\mathrm{b}}, \dots, o_{t-k}^{\mathrm{b}}, \mathbf{n}_t, \dots, \mathbf{n}_{t-k}, \mathbf{m}_t, \dots, \mathbf{m}_{t-k})$, with $\mathbf{n}$ and $\mathbf{m}$ containing the messages from all the UEs.

In order to improve training of our MADDPG solution, we make use of parameter sharing [61] for similar network nodes, in this case the UEs. Similarly to the original work [74], we use the Gumbel-softmax [76] trick to soft-approximate the discrete actions to continuous ones. The Gumbel-softmax reparameterization also works to balance exploration and exploitation. The exploration-exploitation trade-off is controlled by the temperature factor $\zeta$.
After training finishes, we have successfully trained a population of $N_{\mathrm{rep}} = 32$ protocols. We then select the protocol that performed best during the last $N_{\mathrm{eval}} = 500$ evaluation episodes. This selection step can be seen as a "survival of the fittest" approach because only one protocol of the population of $N_{\mathrm{rep}}$ is chosen going forward.

## 5.6. Results

### 5.6.1. Simulation Parameters

For simplicity, we assess the performance of a system with one BS and two UEs. The transmission buffer of each user starts empty and the sdu arrival probability $p_{\mathrm{a}}$ is 0.5.
The system is trained for a fixed number of episodes $N_{\mathrm{train}}$. At some points during the training, we evaluate the policy on a total of $N_{\mathrm{eval}}$ evaluation episodes with disabled exploration and disabled learning in order to assess the current performance of the MAC protocol. The evaluation episodes remain the same in order to effectively compare the performance on the same set. At the end of the training procedure, we further evaluate the learned protocol by assessing its performance in $N_{\mathrm{test}}$ episodes with exploration and learning disabled. This whole procedure represents a single training repetition. We evaluate a total of $N_{\mathrm{rep}}$ repetitions, each with a different random seed.
A summary of the main simulation parameters is provided in Table 5.1, while the parameters of the MADDPG and DDPG algorithms are listed in Table 5.2.

### 5.6.2. Baseline Solutions

We compare the proposed solution with a contention-free baseline. We also compare the proposed solution to two simplified approaches where either the communication between

Table 5.1: Simulation Parameters

| **Parameter** | **Symbol** | **Value** |
|---|---|---|
| Number of UEs | $L$ | 2 |
| Size of transmission buffer | $B$ | 5 |
| Number of sdus to transmit | $P$ | $[1, 2]$ |
| sdu arrival probability | $p_a$ | 0.5 |
| Transport block error rate | TBLER | $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$ |
| DCM vocabulary size | $D$ | 3 |
| UCM vocabulary size | $U$ | 2 |
| Max. duration of episode (TTIs) | $t_{max}$ | 24 |
| Reward function parameter | $\rho$ | 3 |
| Number of training episodes | $N_{train}$ | 300k |
| Number of evaluation episodes | $N_{eval}$ | 500 |
| Number of test episodes | $N_{test}$ | 5000 |
| Number of randomized repetitions | $N_{rep}$ | 32 |

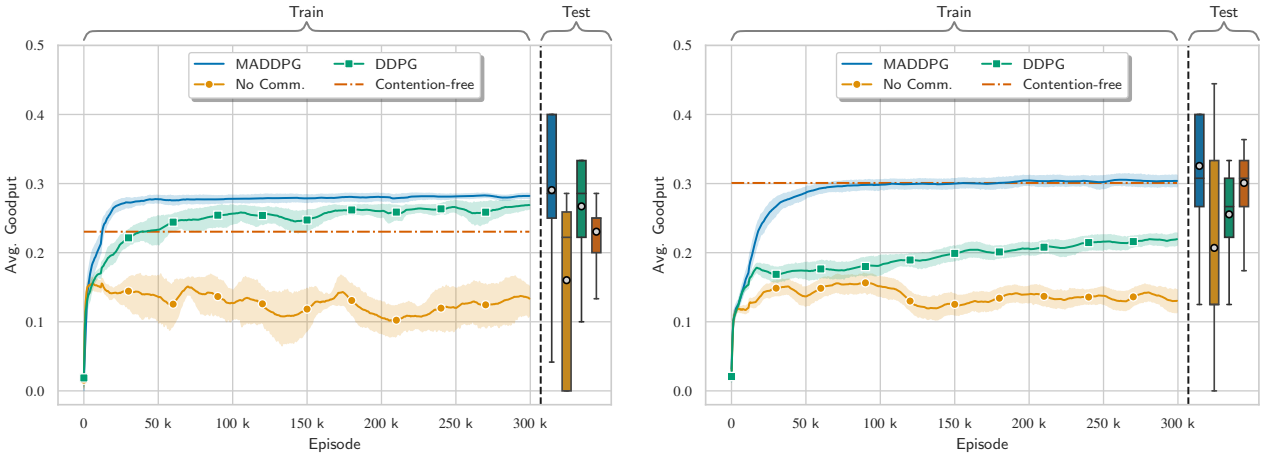
agents is not permitted or the centralized critic is disabled, i.e the DDPG algorithm. The ablation comparison helps to evaluate if communication and the centralized critic are needed to solve this task.
In the contention-free protocol, the UE sends an SR (sr) if its transmission buffer is not empty and it only transmits if it has received a SG (sg). Similarly, it only deletes a TB from the transmission buffer after the reception of an acknowledgement (ACK). At each time step, the BS receives zero or more srs. It then chooses one of the requesters at random to transmit in the next time-step, sending a sg to the selected UE. However, if the UE had made a successful data transmission simultaneously with an sr, the BS will send an ACK to this UE and its sr is ignored.

### 5.6.3. Results

We compare the MADDPG solution with three other solutions, the contention-free baseline, the MADDPG solution without communication, and the DDPG version of the proposed solution, i.e. the proposed solution without the centralized critic. For the RL solutions, the solid lines in Figs. 5.3 and 5.4 show the average performance in the evaluation episodes during the training and the shaded areas represent the 95% confidence interval (CI). The dashed lines show the average performance of the baseline.
In Fig. 5.3, we compare the performance in terms of goodput for the TBLER of $10^{-1}$. Figures 5.3a and 5.3b show the results when the UEs have to transmit one and two sdus, respectively. After evaluating the performance on the $N_{test}$ test episodes, we select the best performing repetitions for each solution in terms of average goodput to compare using boxplots of the test episodes.
By comparing the RL solutions in both cases, the MADDPG has the best performance and the ablation without communication has the worst performance overall. In addition, the MADDPG shows a more stable performance during training, with less variation than both other RL solutions. The ablation without communication has the greatest variation of performance, demonstrated by the CIs and by the boxplots, indicating that communication helps

Table 5.2: Training Algorithm Parameters

| **Parameter** | **Value** |
|---|---|
| Memory length | 3 |
| Replay buffer size | $10^5$ |
| Batch size | 1024 |
| Number of neurons per hidden layer | $\{64, 64\}$ |
| Interval between updating policies | 96 |
| Optimizer algorithm | Adam [77] |
| Learning rate | $10^{-3}$ |
| Discount factor | 0.99 |
| Policy regularizing factor | $10^{-3}$ |
| Gumbel-softmax temperature factor | 1 |
| Target networks soft-update factor | $10^{-3}$ |




(a) Number of SDUs: 1

(b) Number of SDUs: 2

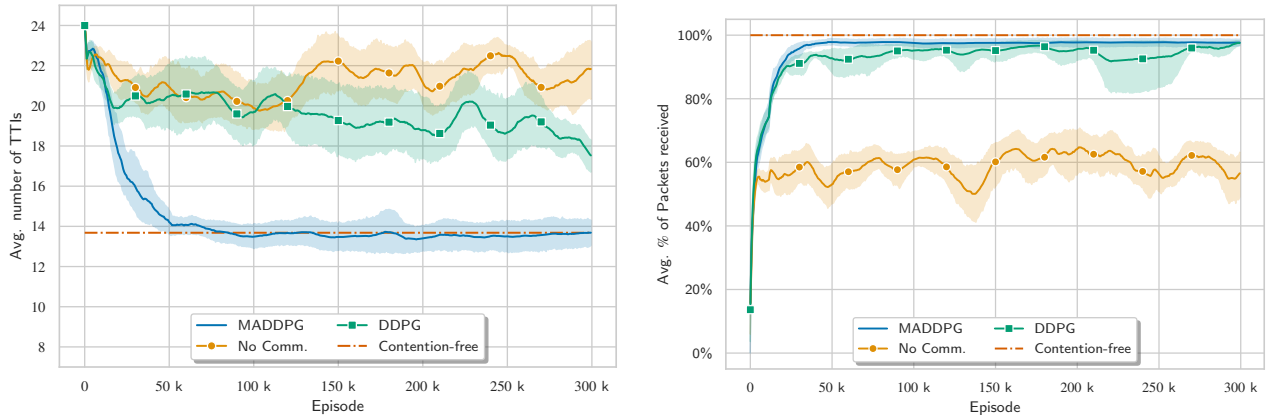
Figure 5.3: Goodput comparison.

achieving a more robust solution.

In Fig. 5.3a, both the MADDPG and DDPG solutions outperform the contention-free baseline, whereas the ablation without communication fails to effectively solve the task in this case. When we move from one sdu to two sdus, in Fig. 5.3b, the DDPG solution does not outperform the baseline. Moreover, the difference in performance between the MADDPG and the baseline is reduced.

To better understand the goodput results of Fig. 5.3b, Fig. 5.4 shows the performance in terms of episode duration, Fig. 5.4a, and of percentage of the total sdus received during the episode as defined in Eq. (5.7), Fig. 5.4b.

As shown in Fig. 5.4b, the DDPG algorithm achieves a high performance in terms of delivery-rate, but it takes more time to solve the task, thus the lower performance in terms of goodput when compared with the MADDPG and the baseline. Comparing the MADDPG with the contention-free solution in Fig. 5.4a, the proposed solution achieves a better goodput by finishing the task in less TTIs. The proposed solution has a delivery-rate lower than the




(a) Episode duration in TTIs

(b) Pct. of Packets received

Figure 5.4: Performance comparison for two SDUs.

contention-free baseline, although it is also very close to 100%.
By applying "survival of the fittest" to pick the best protocol in terms of goodput, the delivery-rate difference becomes even lower than shown in Fig 5.4b. The best protocol produced by the proposed solution has an average delivery-rate on the test episodes of $\Gamma_{\text{MADDPG}} = 99.973\%$ whereas the average of the contention-free baseline is of $\Gamma_{\text{base}} = 99.998\%$

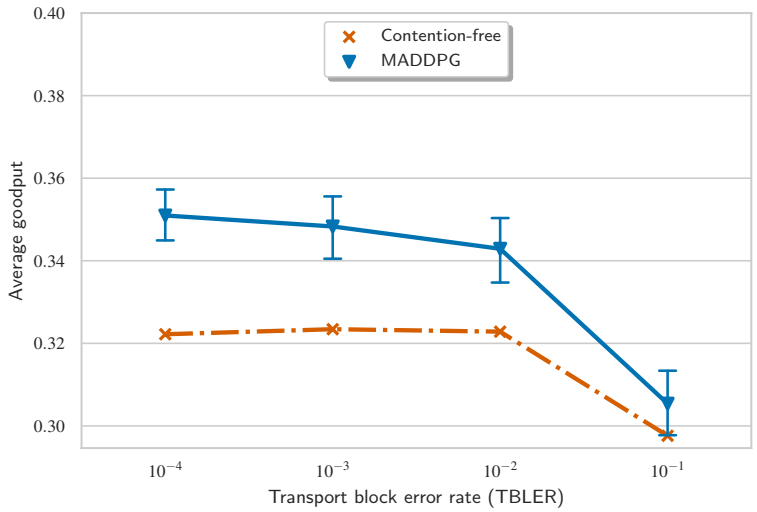



Figure 5.5: Performance in terms of goodput for different TBLERs.

In Fig. 5.5, we compare the proposed MADDPG framework with the contention-free baseline for different TBLERs and with each UE having to transmit two sdus. The performance is evaluated on $N_{\text{test}}$ test episodes by comparing the average goodput achieved. For the MADDPG solution, we also show the 95% CI across randomized repetitions. The proposed solution maintains a better performance than the baseline across the different TBLERs. Moreover, the lowest difference in performance between the baseline and the proposed solution occurs when the TBLER is equal to 0.1, showing that the proposed solution adapts well to lower TBLER regimes.

## 5.7. Conclusions and Perspectives

We have proposed a novel framework based on cooperative MARL augmented with communication, that provides us with the means to emerge a new protocol. In essence, the agents have to learn the signaling policy, representing the control messages they exchange, and the channel-access policy, representing the Physical Layer (PHY) control of the agents. Comparing with two ablations and a baseline, the results show that a solution capable to overcome the challenges in multi-agent systems is needed in order to emerge a protocol. The results also indicate that enabling communication between agents is needed in order to solve a transmission task. In addition, the results illustrate that the proposed solution can produce a protocol tailored to all TBLER regimes that outperforms a more general one.
Concerning future work, we highlight a study on the effect of the different parameters, such as the vocabulary sizes and TBLERs Moreover, we envision a comparison with different MARL algorithms. Finally, the application of this framework to a more complex system model is planned.

# 6. Anomaly detection for effective LoRaWAN network management

## 6.1. Introduction

Low-Power Wide Area Networking (LPWAN) technology offers long-range communication, which enables new types of services. Several solutions exist; LoRaWAN is arguably the most adopted. It promises ubiquitous connectivity in outdoor IoT applications, while keeping network structures, and management, simple. This technology has received a lot of attention in recent months from network operators and solution providers.

### 6.1.1. Overview of LoRaWAN.

LoRa is the physical layer used in LoRaWAN. It features low power operation (around 10 years of battery lifetime), low data rate (27 kbps with spreading factor 7 and 500 kHz channel or 50 kbps with FSK) and long communication range (2-5 km in urban areas and 15 km in suburban areas). It was developed by Cycleo (a French company acquired by Semtech). LoRaWAN networks are organized in a star-of-stars topology, in which gateway nodes relay messages between end-devices and a central network server. End-devices send data to gateways over a single wireless hop and gateways are connected to the network server through a non-LoRaWAN network (e.g. IP over Cellular or Ethernet). Communication is bi-directional, although uplink communication from end devices to the network server is strongly favoured .

LoRaWAN defines three types of devices (Class A, B and C) with different capabilities [78]. Class A devices use pure ALOHA access for the uplink. After sending a frame, a Class A device listens for a response during two downlink receive windows. Each receive window is defined by the duration, an offset time and a data rate. Although offset time can be configured, the recommended value for each receive window is 1 sec and 2 sec, respectively. Downlink transmission is only allowed after a successful uplink transmission. The data rate used in the first downlink window is calculated as a function of the uplink data rate and the receive window offset. In the second window the data rate is fixed to the minimum, 0.3 kbps. Therefore, downlink traffic cannot be transmitted until a successful uplink transmission is decoded by the gateway.

Class A is the class of LoRaWAN devices with the lowest power consumption. Class B devices are designed for applications with additional downlink traffic needs. These devices are synchronized using periodic beacons sent by the gateway to allow the schedule of additional receive windows for downlink traffic without prior successful uplink transmissions. Obviously, a trade-off between downlink traffic and power consumption arises. Finally, Class C devices are always listening to the channel except when they are transmitting. Only class A must be implemented in all end-devices, and the rest of classes must remain compatible with Class A. In turn, Class C devices cannot implement Class B. The three classes can coexist in the same network and devices can switch from one class to another.However, there is not a specific message defined by LoRaWAN to inform the gateway about the class of a device and

this is up to the application. The underlying PHY of the three classes is the same. Communication between end-devices and gateways start with a Join procedure that can occur on multiple frequency channels (e.g. in EU863-870 ISM Band there are 3 channels of 125 kHz that must be supported by all end-devices and 3 additional 125 kHz channels) by implementing pseudo-random channel hopping. Each frame is transmitted with a specific Spreading Factor (SF), defined as SF = log2 (Rc/Rs), where Rs is the symbol rate and Rc is the chip rate. Accordingly, there is a trade-off between SF and communication range. The higher the SF (i.e. the slower the transmission), the longer the communication range. The codes used in the different SFs are orthogonal. This means that multiple frames can be exchanged in the network at the same time, as long as each one is sent with one of the six different SFs (from SF=7 to SF=12). Depending on the SF in use, LoRaWAN data rate ranges from 0.3 kbps to 27 kbps.

The maximum duty-cycle, defined as the maximum percentage of time during which an end-device can occupy a channel, is a key constraint for networks operating in unlicensed bands. Therefore, the selection of the channel must implement pseudo-random channel hopping at each transmission and be compliant with the maximum duty-cycle. For instance, the duty-cycle is 1% in EU 868 for end-devices. The LoRa physical layer uses Chirp Spread Spectrum (CSS) modulation, a spread spectrum technique where the signal is modulated by chirp pulses (frequency varying sinusoidal pulses) hence improving resilience and robustness against interference, Doppler effect and multipath. Packets contain a preamble (typically with 8 symbols), a header (mandatory in explicit mode), the payload (with a maximum size between 51 Bytes and 222 Bytes, depending on the SF) and a Cyclic Redundancy Check (CRC) field (with configurations that provide a coding rate from 4/5 to 4/8). Typical bandwidth (BW) values are 125, 250 and 500 kHz in the HF ISM 868 and 915 MHz band, while they are 7.8, 10.4, 15.6, 20.8, 31.2, 41.7 and 62.5 kHz in the LF 160 and 480 MHz bands. The raw data rate varies according to the SF and the bandwidth, and ranges between 22 bps (BW = 7.8 kHz and SF = 12) to 27 kbps (BW = 500 kHz and SF = 7) [79]. Frequency hopping is exploited at each transmission in order to mitigate external interference [80].

### 6.1.2. Capacity and network size limitations.

In this section we study the LoRaWAN network scale with respect to data rate, duty-cycle regulations, etc.

1. Network size limited by duty-cycle.

   Although the performance of LoRaWAN is determined by PHY/MAC, the duty-cycle regulations in the ISM bands [81], [82] arise as a key limiting factor. If the maximum duty-cycle in a sub-band is denoted by $d$ and the packet transmission time, known as Time On Air, is denoted by $T_{oa}$, each device must be silent in the sub-band for a minimum off-period $T_s = T_{oa}(\frac{1}{d} - 1)$. For instance, the maximum duty-cycle of the EU 868 ISM band is 1% and it results in a maximum transmission time of 36 sec/hour in each sub-band for each end-device. It is known that large SFs allow longer communication range. However, large SFs also increase the time on air and, consequently, the off period duration. This problem is exacerbated by the fact that large SFs are used more often than small SFs.

Although Listen Before Talk is not precluded in LoRaWAN, only ALOHA access is mandatory. Accordingly, the LoRaWAN capacity can be calculated roughly as the superposition of independent ALOHA-based networks (one independent network for each channel and for each SF, since simultaneous transmissions only cause a collision if they both select the same SF and channel; no capture effect is considered).

2. Reliability and Densification drain Network Capacity

In LoRaWAN, reliability is achieved through the acknowledgment of frames in the downlink. For class A end-devices, the acknowledgment can be transmitted in one of the two available receive windows; for class B end-devices it is transmitted in one of the two receive windows or in an additional time-synchronized window; for class C end-devices it can be transmitted at any time . In LoRaWAN the capacity of the network is reduced not only due to transmissions in the downlink, but also due to the off-period time following those transmissions (gateways must be compliant with duty-cycle regulation). Therefore, the design of the network and the applications that run on it must minimize the number of acknowledged frames to avoid the capacity drain. This side-effect calls into question the feasibility of deploying ultra-reliable services over large-scale LoRaWAN networks. At this point of development of the technology, LoRaWAN faces deployment trends that can result in future inefficiencies. Specifically, LoRaWAN networks are being deployed following the cellular network model, that is, network operators provide connectivity as a service. This model is making gateways to become base stations covering large areas. The increase in the number of end-devices running applications from different vendors over the same shared infrastructure poses new challenges to coordinate the applications. In particular, each application has specific constraints in terms of reliability, maximum latency, transmission pattern, etc. The coordination of the diverse requirements over a single shared infrastructure using an ALOHA-based access is one of the main future challenges for the technology. Therefore, a fair spectrum sharing is required beyond the existing duty-cycle regulations. Finally, the unplanned and uncoordinated deployment of LoRaWAN gateways in urban regions, along with the deployment of alternative LPWAN solutions (e.g. SigFox, and NB-IoT), could cause a decrease of the capacity due to collisions and due to the use of larger SFs (to cope with higher interference levels).

### 6.1.3. Problem statement

In WSN, (as mentioned in [83], [84] and [79]), Performance Monitoring is one of the main challenges facing large-scale deployments besides the capacity and cost. So to help the network operator manage and monitor their network in an efficient and effective way and to avoid resource-wasting, this work focuses on building a model for anomaly detection and prediction, that will help us identify the status of the network within the near future so we will be able to handle these issues before they happen. To achieve this goal, we describe a process that includes collecting the metadata from the different network components at different layers and sendint it to a central base station where further treatment and processing, with better resources, are possible. At this central base station, a more effective network monitoring can be achieved. Then the collected metadata from the network will be used to

study the network performance, identify behavior patterns and train different machine learning models to predict network operation with this data. Next, we'll work to define the core parameters and indicators for LoRaWAN implementations so that we can monitor the network's progress. In addition, we plan to test several classification approaches in order to gain further insights into the network's success beyond purely statistical analysis.
Finally, after testing all of the models, the best models will be validated by testing them in the real-time environment to see their functionality in reality. Then in order to close, the final step completes the cycle and corresponds to the ability to behave dynamically in the network depending on the metrics obtained, either automatically (by each node or by a central control tool attached to the sink) or by the Network Manager.

### 6.1.4. Objectives

The main goal of our research project is to build a model able to predict the network status in the near future and detect anomalies before it happens, therefore, the network operator will be able to tackle these problems. To achieve this goal then the following objectives have to be implemented.

1. Identify the key parameters and metrics for LoRaWAN network.
2. Relate the metrics identified to specific network operation scenarios.
3. Design a testbed to collect network data, and develop the needed tools to gather the data from heterogeneous sources (devices, network servers and gateways).
4. Design a dashboard for data visualization.
5. Evaluate some classification methods to get further insights of the performance of the network beyond purely statistical analysis.
6. Identify different methods to predict network operation, and to detect anomalies in network operation.
7. Validate methods of prediction and anomaly detection with the data sets collected from the test-bed.
8. Validate the best methods in real-life environments.

## 6.2. Network management.

### 6.2.1. Operations, Administration, and Maintenance (OAM) Tools.

[85] "OAM" is a general term that refers to a tool set for detecting, isolating, and reporting failures, and for monitoring network performance. For each transmission technology, there are a set of tools used to monitor the performance of the network at different levels. our goal is to be able to monitor the network (system) at different levels, starting from the end-device which is the base of the network, then the gateway, where each gateway has at least one or more end-devices, finally the network (cluster) where each network has at least one or more

gateways. To achieve this, a set of different key performance indicators (KPI) have been defined for each network level. Following are the list of the main network monitoring KPIs:

- The total number of active nodes.
- Number of active gateways.
- Gateway utilization ratio.
- Latency (connectivity with the LoRaWAN network server) server response time (SRT).
- Payload size of the received message.
- Time on Air of the received message.
- Duty Cycle consumption.
- Jitter window of the received messages.
- Distribution of SF in downlink messages.
- Distribution of SF in uplink messages.
- Packet loss per day.
- Distribution of used channel.
- Percentage of CRC Failed.
- Total Messages for UL and DL per day.
- Percentage of nodes per SF and frequency.
- The total network duty cycle/average.
- Overall network coverage map.

### 6.2.2. Data collection.

Generating a dataset is one of the most important steps that should be done at the beginning of the research process. The main research strategy which is being applied here is the design and creation of new IT products development, also called artifacts, and the same approach will be used to generate the data from the network to be able later to use the survey research strategy to study the data in more depth. In the present case, a code has been written to collect the metadata from the network.
Fig.6.1 shows the architecture design proposal for collecting the metadata from the network. The system consists form the following components:

- Messages_logger_1: responsible to retrieve the data from the network server then store all the messages related to the uplink topic inside the database and at the same time, the received message is stored in a queue until it is consumed by the Network monitoring module.

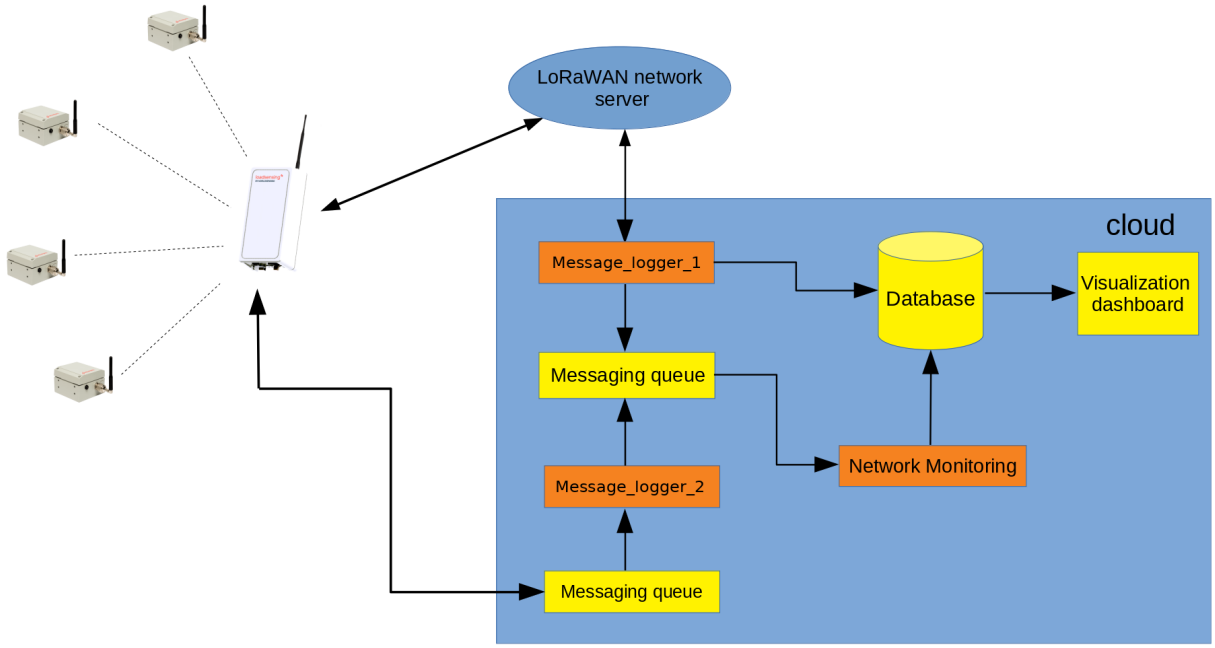



Figure 6.1: Architecture design proposal for collecting the metadata from the network.

- Messages_logger_2: responsible to receive the metadata files that have been sent to the cloud from the gateway then decode the data and store every message in a temporary queue before it will be consumed by Network Monitoring module.
- Network monitoring: the collected metadata from the network will be used to calculate the KPI for the network then store it in the database.

### 6.2.3. Data visualization.

After identify the key parameters and metrics for LoRaWAN and design a testbed to collect the network metadata. Designing a dashboard for data visualization is required and that will provide us with a quick, clear understanding of the information. Grafana is used to read the stored KPIs from the Postgres database and then visualize them. Three main dashboards have been designed one for each data plane level( Network, gateway and end-device). Fig 6.2 and Fig 6.3 shows the dashboard of the Node KPIs.

### 6.2.4. Network status prediction

After defining the KPI for the LoRaWAN network at different levels. We are ready to move on to the next step which is studying the collected data and try to extract useful information about the operation performance of the network then use this knowledge with the KPIs data to train a model that is able to predict the status of the network in the near future. As we mentioned before, building an accurate real-time network status prediction is required in many networking applications like dynamic resource allocation and network management. First of all and before exploring machine learning methods for time series e.g Long Short-Term Memory(LSTM) and recurrent neural network(RNN), it is a good idea to ensure that we have exhausted classical linear time series forecasting methods. Classical time series forecasting


Figure 6.2: Dashboard of the Node KPIs.

methods may be focused on linear relationships, nevertheless, they are sophisticated and perform well on a wide range of problems, assuming that our data is suitably prepared and the method is well configured.

1. Autoregression (AR).
2. Moving Average (MA).
3. Autoregressive Moving Average (ARMA).
4. Autoregressive Integrated Moving Average (ARIMA).
5. Linear regression.

Many predictors from two main different classes, including classic time series and artificial neural networks predictors will be compared. These predictors will be evaluated using real network data. Comparison of accuracy and cost, both in terms of computation complexity and power consumption.


Figure 6.3: Dashboard of the Node KPIs.

# 7. References

[1] N. Abramson, “THE ALOHA SYSTEM: Another alternative for computer communications,” in *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, AFIPS ’70 (Fall), (New York, NY, USA), p. 281–285, Association for Computing Machinery, 1970.

[2] N. Abramson, “The throughput of packet broadcasting channels,” *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 117–128, 1977.

[3] B. Hajek and T. van Loon, “Decentralized dynamic control of a multiaccess broadcast channel,” *IEEE Transactions on Automatic Control*, vol. 27, no. 3, pp. 559–569, 1982.

[4] P. R. Srikanta Kumar and L. Merakos, “Distributed control of broadcast channels with acknowledgement feedback: Stability and performance,” in *The 23rd IEEE Conference on Decision and Control*, pp. 1143–1148, 1984.

[5] B.-J. Kwak, N.-O. Song, and L. Miller, “Performance analysis of exponential backoff,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 343–355, 2005.

[6] J. Goodman, A. G. Greenberg, N. Madras, and P. March, “Stability of binary exponential backoff,” *J. ACM*, vol. 35, p. 579–602, June 1988.

[7] L. Barletta, F. Borgonovo, and I. Filippini, “The throughput and access delay of slotted-aloha with exponential backoff,” *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 451–464, 2018.

[8] L. Barletta, F. Borgonovo, and I. Filippini, "The access delay of aloha with exponential back-off strategies," in *2016 Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pp. 1–6, 2016.

[9] Y. Chu, S. Kosunalp, P. D. Mitchell, D. Grace, and T. Clarke, "Application of reinforcement learning to medium access control for wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 46, pp. 23–32, 2015.

[10] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 310–323, 2019.

[11] C. Zhong, Z. Lu, M. C. Gursoy, and S. Velipasalar, "Actor-critic deep reinforcement learning for dynamic multichannel access," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 599–603, 2018.

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

[13] L. de Alfaro, M. Zhang, and J. J. Garcia-Luna-Aceves, "Approaching fair collision-free channel access with slotted aloha using collaborative policy-based reinforcement learning," in *2020 IFIP Networking Conference (Networking)*, pp. 262–270, 2020.

[14] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *ArXiv*, vol. cs.NI/9809099, 1998.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015.

[16] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," 2015.

[17] R. D. Yates and S. K. Kaul, "The age of information: Real-time status updating by multiple sources," 2017.

[18] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?," in *2012 Proceedings IEEE INFOCOM*, pp. 2731–2735, 2012.

[19] A. Kosta, N. Pappas, and V. Angelakis, *Age of Information: A New Concept, Metric, and Tool*. Now Foundations and Trends, 2017.

[20] S. K. Kaul, R. D. Yates, and M. Gruteser, "Status updates through queues," in *2012 46th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, 2012.

[21] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 350–358, 2011.

[22] J. Doncel and M. Assaad, "Age of information in a decentralized network of parallel queues with routing and packets losses," 2020.

[23] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[24] M. K. Abdel-Aziz, C.-F. Liu, S. Samarakoon, M. Bennis, and W. Saad, "Ultra-reliable low-latency vehicular networks: Taming the age of information tail," *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018.

[25] I. Kadota, A. Sinha, and E. Modiano, "Optimizing age of information in wireless networks with throughput constraints," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 1844–1852, 2018.

[26] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "On the age of information with packet deadlines," *IEEE Transactions on Information Theory*, vol. 64, no. 9, pp. 6419–6428, 2018.

[27] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age of information and throughput in a shared access network with heterogeneous traffic," 2018.

[28] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 561–565, 2017.

[29] H. B. Beytur and E. Uysal, "Age minimization of multiple flows using reinforcement learning," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, pp. 339–343, 2019.

[30] A. M. Bedewy, Y. Sun, S. Kompella, and N. B. Shroff, "Optimal sampling and scheduling for timely status updates in multi-source networks," 2020.

[31] Q. He, D. Yuan, and A. Ephremides, "Optimal link scheduling for age minimization in wireless systems," *IEEE Transactions on Information Theory*, vol. 64, no. 7, pp. 5381–5394, 2018.

[32] R. D. Yates and S. K. Kaul, "Status updates over unreliable multiaccess channels," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 331–335, 2017.

[33] J. Zhong, E. Soljanin, and R. D. Yates, "Status updates through multicast networks," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 463–469, 2017.

[34] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Scheduling policies for minimizing age of information in broadcast wireless networks," 2018.

[35] E. T. Ceran, D. Gündüz, and A. György, "A reinforcement learning approach to age of information in multi-user networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1967–1971, 2018.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.

[37] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” 2018.

[38] “Network simulator 3.”

[39] M. Kaneko, P. Popovski, and J. Dahl, “Proportional fairness in multi-carrier system with multi-slot frames: upper bound and user multiplexing algorithms,” *IEEE Transactions on Wireless Communications*, vol. 7, no. 1, pp. 22–26, 2008.

[40] “3gpp tr 38.913 study on scenarios and requirements for next generation access technologies,” 2017.

[41] “3gpp tr 36.839 evolved universal terrestrial radio access; mobility enhancements in heterogeneous networks,” 2013.

[42] “3gpp ts 24.501 5g; non-access-stratum (nas) protocol for 5g system (5gs); stage 3,” 2020.

[43] “3gpp ts 38.331, "nr - radio resource control (rrc) protocol specification -release 16,” 2020.

[44] “3gpp ts 38.300, nr; overall description; stage-2,” 2019.

[45] L. F. Gonzalez, G. A. Montes, E. Puig, S. Johnson, K. Mengersen, and K. J. Gaston, “Unmanned aerial vehicles (UAVs) and artificial intelligence revolutionizing wildlife monitoring and conservation,” *Sensors*, vol. 16, p. 97, Jan 2016.

[46] M. A. Ma’Sum, M. K. Arrofi, G. Jati, F. Arifin, M. N. Kurniawan, P. Mursanto, and W. Jatmiko, “Simulation of intelligent unmanned aerial vehicle (UAV) for military surveillance,” in *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 161–166, IEEE, Mar 2013.

[47] B. Ji, Y. Li, B. Zhou, C. Li, K. Song, and H. Wen, “Performance analysis of uav relay assisted iot communication network enhanced with energy harvesting,” *IEEE Access*, vol. 7, pp. 38738–38747, 2019.

[48] Y. Wan, K. Namuduri, Y. Zhou, and S. Fu, “A smooth-turn mobility model for airborne networks,” *IEEE Transactions on Vehicular Technology*, vol. 62, pp. 3359–3370, Mar 2013.

[49] J.-D. M. M. Biomo, T. Kunz, and M. St-Hilaire, “An enhanced Gauss-Markov mobility model for simulations of unmanned aerial ad hoc networks,” in *7th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 1–8, IEEE, May 2014.

[50] F. Mason, F. Chiariotti, M. Capuzzo, D. Magrin, A. Zanella, and M. Zorzi, “Combining lorawan and a new 3d motion model for remote uav tracking,” in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 412–417, IEEE, 2020.

[51] M. O. Farooq, "Multi-hop communication protocol for lora with software-defined networking extension," *Internet of Things*, vol. 14, p. 100379, 2021.

[52] Z. Zhao, P. Cumino, A. Souza, D. Rosário, T. Braun, E. Cerqueira, and M. Gerla, "Software-defined unmanned aerial vehicles networking for video dissemination services," *Ad Hoc Networks*, vol. 83, pp. 68–77, Feb 2019.

[53] Z. Drezner, "Computation of the multivariate normal integral," *ACM Transactions on Mathematical Software (TOMS)*, vol. 18, pp. 470–480, Dec 1992.

[54] P. N. Somerville, "Numerical computation of multivariate normal and multivariate-t probabilities over convex regions," *Journal of Computational and Graphical Statistics*, vol. 7, pp. 529–544, Dec 1998.

[55] N. Mastronarde and M. van der Schaar, "Fast reinforcement learning for energy-efficient wireless communication," *IEEE Transactions on Signal Processing*, vol. 59, pp. 6262–6266, Aug. 2011.

[56] K. David and H. Berndt, "6G vision and requirements: Is there any need for beyond 5G?," *IEEE Vehicular Technology Magazine*, vol. 13, no. 3, pp. 72–80, 2018.

[57] H. B. Pasandi and T. Nadeem, "Towards a learning-based framework for self-driving design of networking protocols," *IEEE Access*, vol. 9, pp. 34829–34844, 2021.

[58] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-native air interface," *IEEE Communications Magazine*, vol. 59, no. 5, pp. 76–81, 2021.

[59] S. Han, T. Xie, C.-L. I, L. Chai, Z. Liu, Y. Yuan, and C. Cui, "Artificial-Intelligence-Enabled air interface for 6G: Solutions, challenges, and standardization impacts," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 73–79, 2020.

[60] A. Lazaridou and M. Baroni, "Emergent multi-agent communication in the deep learning era," *arXiv preprint arXiv:2006.02419*, 2020.

[61] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with Deep multi-agent reinforcement learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2145–2153, 2016.

[62] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 2252–2260, 2016.

[63] A. Dafoe, E. Hughes, Y. Bachrach, T. Collins, K. R. McKee, J. Z. Leibo, K. Larson, and T. Graepel, "Open problems in cooperative AI," *arXiv preprint arXiv:2012.08630*, 2020.

[64] A. Valcarce and J. Hoydis, "Towards joint learning of optimal MAC signaling and wireless channel access," *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2021.

[65] F. Al-Tam, N. Correia, and J. Rodriguez, "Learn to schedule (LEASCH): A deep reinforcement learning approach for radio resource scheduling in the 5G MAC layer," *IEEE Access*, vol. 8, pp. 108088–108101, 2020.

[66] F. AL-Tam, A. Mazayev, N. Correia, and J. Rodriguez, "Radio resource scheduling with deep pointer networks and reinforcement learning," in *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pp. 1–6, 2020.

[67] C. Bowyer, D. Greene, T. Ward, M. Menendez, J. Shea, and T. Wong, "Reinforcement learning for mixed cooperative/competitive dynamic spectrum access," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pp. 1–6, 2019.

[68] T. F. Wong, T. Ward, J. M. Shea, M. Menendez, D. Green, and C. Bowyer, "A dynamic spectrum sharing design in the DARPA spectrum collaboration challenge," in *Proceedings of the Government Microcircuit Applications and Critical Technology Conference (GOMACTech)*, 2020.

[69] A. Mostaani, O. Simeone, S. Chatzinotas, and B. Ottersten, "Learning-based physical layer communications for multiagent collaboration," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2019.

[70] J. S. P. Roig and D. Gündüz, "Remote reinforcement learning over a noisy channel," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pp. 1–6, 2020.

[71] C. M. Bishop, *Pattern Recognition and Machine Learning, 5th Edition*. Information Science and Statistics, Springer, 2007.

[72] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 289–353, 2008.

[73] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.

[74] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, pp. 6379–6390, 2017.

[75] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning.," in *ICLR* (Y. Bengio and Y. LeCun, eds.), 2016.

[76] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[77] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[78] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, *"LoRa Specification 1.0" Lora Alliance Standard specification*. International series of monographs on physics, Clarendon Press, 2015.

[79] C. Goursaud and J.-M. Gorce, "Dedicated networks for iot: Phy/mac state of the art and challenges," *EAI endorsed transactions on Internet of Things*, 2015.

[80] T. Watteyne, A. Mehta, and K. Pister, "Reliability through frequency diversity: why channel hopping makes sense," in *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pp. 116–123, 2009.

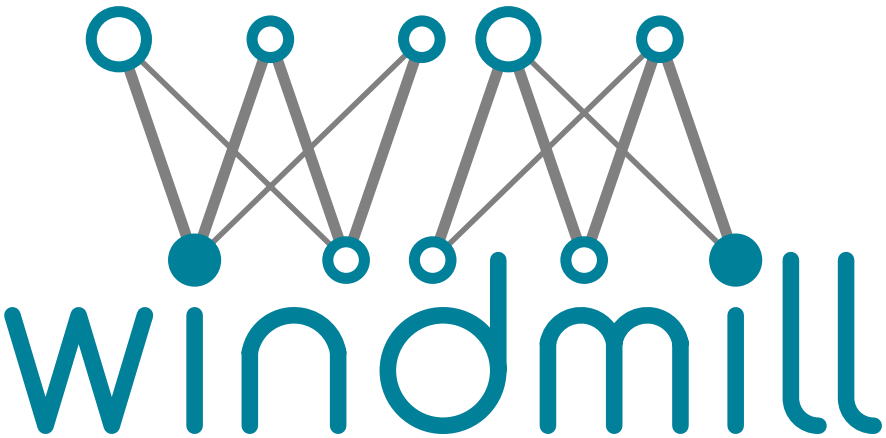
[81] E. C. Committee and Others, *ERC Recommendation 70-03*. International series of monographs on physics, Clarendon Press, Oct. 2016.

[82] F. C. C. et al, *"FCC Part 15–Radio Frequency Devices, Code of Federal Regulation 47 CFR Ch. 1"*. FCC, 10-1-15 Edition.

[83] J. Toussaint, N. El Rachkidy, and A. Guitton, "Performance analysis of the on-the-air activation in lorawan," in *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 1–7, IEEE, 2016.

[84] D. Magrin, M. Capuzzo, and A. Zanella, "A thorough study of lorawan performance under different parameter settings," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 116–127, 2019.

[85] T. Mizrahi, N. Sprecher, E. Bellagamba, and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools." RFC 7276, June 2014.




www.windmill-itn.eu