Application-Level Data Rate Adaptation in Wi-Fi Networks Using Deep Reinforcement Learning

Ibrahim Sammour, Gérard Chalhoub

Application-Level Data Rate Adaptation in Wi-Fi Networks Using Deep Reinforcement Learning

Ibrahim Sammour, Gerard Chalhoub

Application-Level Data Rate Adaptation in Wi-Fi Networks Using Deep Reinforcement Learning

Ibrahim Sammour
LIMOS-CNRS
University of Clermont-Auvergne
Clermont-Ferrand, France
0000-0002-7673-310X

Gerard Chalhoub
LIMOS-CNRS
University of Clermont-Auvergne
Clermont-Ferrand, France
0000-0003-1687-598X

***Abstract*—Wireless technologies are used in almost every application domain. They are easy to deploy and some of them offer high data rates. Modern applications require more bandwidth to cope with the growing quality of network content (i.e. multimedia). The increasing demand for network capacity in terms of bandwidth and the growing number of users are causing densification in the deployed networks. Most wireless technologies, such as Wi-Fi, suffer from deterioration of Quality of Experience (QoE) in dense deployments. To overcome this problem, one of the techniques that can be used is data rate adaptation depending on the state of the network. In this paper, we propose a Deep Reinforcement Learning (DRL) approach for decentralized application-level data rate adaptation in dense Wi-Fi networks. We present the training procedure of the DRL model using NS-3 simulator and TensorFlow. The model is then evaluated in dense scenarios and compared to an existing approach from the literature. Results show that using DRL can help to better cope with the current capacity of the wireless network.**

I. Introduction

Over the past few decades, wireless technologies have enabled a wide variety of applications that have a significant impact on the way we live. Many modern applications rely on these technologies. They eased the way humans and objects interact and removed limitations of cable-based networks. These revolutionary technologies are being deployed almost everywhere from airplanes and space rockets [1] to laptops and mobile phones, going all the way down to tiny implantable medical devices [2].

Wireless Fidelty (Wi-Fi) is one of the popular wireless technologies. Wi-Fi devices are included in almost every smartphone and laptop. The availability of Wi-Fi access points has become a necessity in any public, business, or commercial place. Hence, Wi-Fi faces an increasing amount of challenges in terms of the Quality of Service (QoS). Over the years, Wi-Fi standards have introduced advanced features to improve the data rates, coverage, and the multi-user functionality. For example, Multi-User Multiple Input Multiple Output (MU-MIMO) [3], Orthogonal Frequency Division Multiple Access (OFDMA) [4], and higher order modulation and coding schemes (MCS) such as the 1024-QAM in Wi-Fi 6. These features made it possible to deploy the technology with many simultaneously connected devices in dense networks such as stadiums [7], airports [8], or public transportation.

This research was funded by the French government IDEX-ISITE initiative 16-IDEX-0001 (CAP 20-25).

The increase in the number of Wi-Fi clients causes an increase in the offered load per access point. Indeed, Wi-Fi adopts the CSMA/CA random access protocol to manage the medium access. However, the contention increases in dense networks resulting in longer backoff durations and degradation in the overall performance of the network.

One way to deal with this overload is to adapt the offered load to the capacity of the network. For example, network content can be delivered in different forms such as 360p up to 4K videos. Usually, this adaptation is based on the Quality of Service (QoS) metrics visible to the application layer such as throughput, packet loss, delay or jitter.

Machine learning is playing an increasing role in recent years in improving Wi-Fi performance. Reinforcement Learning (RL), a branch of machine learning capable of solving optimization problems in complex environments, is frequently used in Wi-Fi [10]. In RL, we train an agent to learn about the environment (through measurements/sensors), take actions (optimize control parameters), and receive rewards to evaluate the actions. DRL offers a better generalization by introducing deep learning in RL. DRL introduces significant improvements over traditional optimization methods for wireless networks such as threshold based techniques or supervised learning [11]. DRL has the ability to handle more complex scenarios and generalize its knowledge to situations it has never encountered before [12].

In this paper, we tackle the problem of application-level data rate adaptation in dense Wi-Fi networks. We propose a realistic decentralized DRL mechanism for dense networks that is able to cope with high congestion scenarios by adapting the data rate at the application level based on goodput and packet loss metrics. The rest of the paper is organized as follows: In Section II, we briefly talk about previous DRL approaches in Wi-Fi networks. In Section III, we formulate our problem and present the structure of our proposed approach. In Section IV, we present the training scenario and discuss the validation results of our DRL model compared to an existing approach from the literature. We conclude the paper and talk about future work in Section V.

The main novelty of this paper is the use of a DRL approach

on the application layer for a generic application by adapting the offered load depending on the current conditions of the wireless network.

II. Deep Reinforcement Learning in Wi-Fi networks

In this section, we provide a quick overview of the recent applications of DRL in Wi-Fi networks.

Recent surveys, such as [13], described the various Wi-Fi areas where ML is applied, mainly RL and Deep Learning (DL), DRL has not been extensively studied. However, the papers working on a similar problem as ours are dedicated to certain applications such as video streaming where only a few ones use RL as in [14].

Authors in [15] used Deep Q-Learning (DQL) to enhance the performance of CSMA/CA in dense Wireless Local Area Networks (WLANs) by observing the backoff values, performing actions to increase or decrease them, and learning from a reward calculated based on the probability of collisions. Results showed enhancement in terms of throughput, channel access delay, and fairness compared to other mechanisms in the literature. To overcome the performance degradation due to the random selection of Resource Units (RUs) in Wi-Fi 6 OFDMA, authors in [16] proposed a DRL mechanism based on energy detection and acknowledgments for fair resource distribution. This mechanism led to higher average throughput and lower average latency. Double Deep Q-Learning was used for rate adaptation on the physical layer of Wi-Fi networks in [17]. The state representation included metrics such as Modulation and Coding Schemes (MCS) and Received Signal Strength Indicator (RSSI). The agent relied on the goodput to calculate the reward and selects a new triplet (MCS, channel width, number of spatial streams) predefined profiles. The agent was deployed on Intel 802.11ac Network Interface Cards (NICs). It outperformed the Intel and Linux default rate adaptation algorithms by more than 200%. Authors in [18] used DQL based algorithm to check if clients actually benefit from participating in Multi-User MIMO (MU-MIMO). The metrics used in the study were Channel State Information (CSI) and SNR. The experimental results show that using DRL to solve the pre-screening problem improves the system throughput by almost 40%. In [19], authors improve the data rate obtained by Wi-Fi clients using a client-access point association scheme based on DQL. The proposed method takes into account the application demands of the user and link capacity. Results showed improvements over standard signal strength based association in terms of throughput and ensuring application requirements. Authors in [20] proposed a video quality adaptation algorithm called End-to-End MAC (E2E-MAC). The algorithm combines throughput measurements with the number of re-transmissions on the MAC layer to improve the QoE of users.

III. Proposed Approach

In this section we address the network performance degradation in dense Wi-Fi networks. The problem is framed as a Markov Decision Process (MDP). A DRL approach is proposed for adapting the application profiles based on network conditions.

A. Problem Formulation

Modern applications are more and more demanding in terms of network needs. Thus, more bandwidth is required by each Wi-Fi node in order to transmit/receive data while maintaining the satisfaction of the users. Studies on Wi-Fi performance have shown the significant decrease of the overall network throughput as the network size increases [21].

In an attempt to reproduce this behavior, we evaluated the performance of Wi-Fi using fixed application-level data rates for all nodes. We performed simulations using NS-3 by varying the number of nodes from 1 to 100. Simulation parameters are presented in table I. Each node is transmitting using the same application-level data rate toward the same access point. Figure 1 depicts the aggregated goodput obtained in the network averaged over 50 runs with the standard deviation. During simulation, the network becomes saturated for a certain number of nodes depending on the application-level data rate. When more nodes are introduced to the network, the aggregated goodput drops from the saturation value and keeps decreasing. This is mainly due to the probabilistic behavior of the CSMA/CA algorithm used by Wi-Fi which does not guarantee access to the medium especially in high offered load scenarios [33]. Optimizations have been studied to enhance the performance of CSMA/CA [23] but this can only be done by updating the specifications of the 802.11 standard. Hence, our proposal that is situated on the application level can benefit any device without requiring modifications on the standard.

Figure 1 also shows that the maximum goodput increases when the application-level datarate is increased. Higher application-level data rates reach their saturation goodput earlier. For example, when using a data rate of 3.5 Mb/s, we reach the saturation goodput with 25 nodes and it decreases afterwards. However, when using a data rate of 0.7 Mb/s, we reach the saturation goodput with 75 nodes. Also note that for the same number of nodes, using a higher application profile gives higher goodput. Except for when the number of nodes is high, the goodputs values obtained by different application profiles are close.

The deterioration of the network performance is caused mainly by 2 reasons. First, (i) the increase in the Offered load: when each node increases the amount of data, the overall offered load becomes greater than the reception capacity of the access point. This reduces the goodput of each node and decreases the QoE. Second, (ii) the use of a random access protocol (CSMA/CA) in the Medium Access Control (MAC) layer: after transmitting a data packet, a node waits for an acknowledgment from the receiver, if no acknowledgement is received, the node backs off and waits for a random number of time slots before accessing the channel again. At each new attempt to retransmit the packet, the probability of choosing a longer backoff duration is increased. This mechanism causes

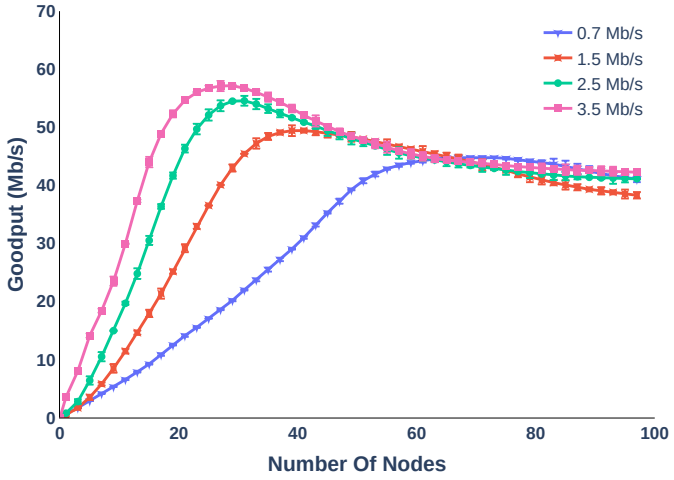

Fig. 1: The goodput of nodes transmitting using a single application profile.

significant access delays and collisions especially in dense networks [22].

In what follows, we propose a mechanism that reduces the offered load per node in an attempt maintain the goodput level for a higher number of nodes. For example, under high contention conditions, a node preserves the satisfaction of the user by selecting a lower application-level data rate.

B. Deep Reinforcement Learning Model

The idea behind RL is to employ a self-learning agent that can interact with the environment through a series of actions. The agent is rewarded after each action it makes, moving it closer to its goal. The agent improves its decision-making through training, which helps it learn the best possible action for each state it may encounter in a real environment [24]. However, the number of different network conditions an agent may encounter is large. Indeed, this large space of values can be discretized. In order to provide more accuracy and better coverage of a real environment, we decided to use DRL. In DRL, the decision-making process of an agent is represented by a neural network. A neural network takes the network conditions as an input and outputs the corresponding suitable application-level data rate.

The adaptation from one application-level data rate to another is a challenging task due to the varying nature of wireless networks and the large space of network conditions that a Wi-Fi node can encounter. To tackle this issue, multiple techniques have been introduced in the literature based on the knowledge of physical layer metrics such as the signal to noise ratio (SNR) in [25]. The knowledge of these metrics gives an immediate estimate of channel conditions [26]. However, applications may be deployed in various types of devices that do not provide access to the physical layer metrics. For that reason, we will only use certain metrics that can be available on the application level for any device, namely, goodput which is the amount of correctly received data, and packet loss which is the ratio of lost packets due to collisions and buffer overflow. Note that other metrics can also be used such as the Age of Information (AoI) as done in [27] which provides feedback about the time spent to access the channel and send the data..

1) Model Architecture: The optimization problem can be framed as a MDP consisting of (i) an agent exploring the environment, (ii) a set of states *S*, (iii) a set of actions *A*, (iv) a reward function *R*, (v) a transition probability *T*, and (vi) a reward discount factor $\gamma \in [0,1]$.

The *agent* is installed on the application layer of the Wi-Fi node. The agent sends data toward an access point over an interval of time and observes the *state* s $\in$ *S* of the environment. The state is expressed as the goodput (g) and the packet loss (pl) of the transmissions in the previous interval $[g, pl]$. The agent then takes an *action* a $\in$ *A* by selecting data rate based on a policy (π). The policy is a neural network which provides a set of probabilities, each of which corresponds to an action. Based on the policy, the agent samples an action and decides whether to keep using the current data rate or to select a new one. The reward r is then calculated as the difference between the overall goodput and packet loss aiming at enhancing the overall performance of the network. Note that our reward function expresses the QoE. As we chose a generic application, we reduced the QoE to these metrics, note that for other specific applications such video streaming, other metrics can be added to the reward function such as buffer size, access delay, or current quality of image. Also note that the fairness issue and energy consumption are out of the scope of our approach. As a first step, we only focus on two results, namely, the goodput and the packet loss. The reward equation is shown in (1).

$$r = \alpha * g - \beta * pl \tag{1}$$

The goodput is normalized to the range of [0,1] by dividing it by the maximum expected goodput (based on the maximum possible data rate). α and β parameters help to fine tune the reward to be compliant with the requirements of the application. For instance, increasing β will drive the agent to favor learning how to reduce the packet loss more than increasing the goodput. In this case, selecting a lower data rate would be the most probable decision.

The immediate reward may not be sufficient to determine the proper decision in the current state. The decision at any state have an impact on the future series of events. The discount factor $\gamma \in [0, 1]$ is used to determine the importance of future rewards in comparison to the immediate reward. This is represented by the cumulative reward equation (2). It predicts how much reward is expected in the future after taking an action in a certain state. For instance, choosing a low value of γ means favoring short term rewards. For example, when streaming a live video, we care more about short term rewards.

$$\overline{r} = r_t + \gamma * r_{t+1} + \gamma^2 * r_{t+2} + \cdots + \gamma^T * r_T \tag{2}$$

where T is the window size of the future rewards that we care about.

Our aim is to reach a policy in which for any observed network conditions, it selects the data rate that maximizes the *return* (cumulative reward) of the agent. Our approach is split into two phases: *Offline-training* and *Exploitation*. During the *Offline-training* phase, the agent explores the environment through different simulation scenarios until converging to a policy based on our reward design. At the end of the training, we save the model that contains our final policy to use it in the next phase. During the *Exploitation* phase, we deploy the generated model in scenarios where the agent is able to pick with a certain degree of confidence the most suitable data rate given a state. Our training method is presented in the next sub-section.

2) Finding the most suitable data rate: We employ an on-policy approach based on the actor-critic framework, namely, Proximal Policy Optimisation (PPO) with the clipping surrogate technique [29]. We chose PPO for its stability as it constraints policy updates so the learning does not diverge or fall to a local optimum. The actor-critic framework consists of two neural networks: the actor network and the critic network. The actor network is responsible for the action selection. It takes the state as an input and outputs a probability vector of the possible actions. The critic network is the value function. It takes a state as an input and outputs the expected *return*. The critic network decides if the policy (actor) is improving or deteriorating. Figure 2 shows the complete structure of the proposed approach.

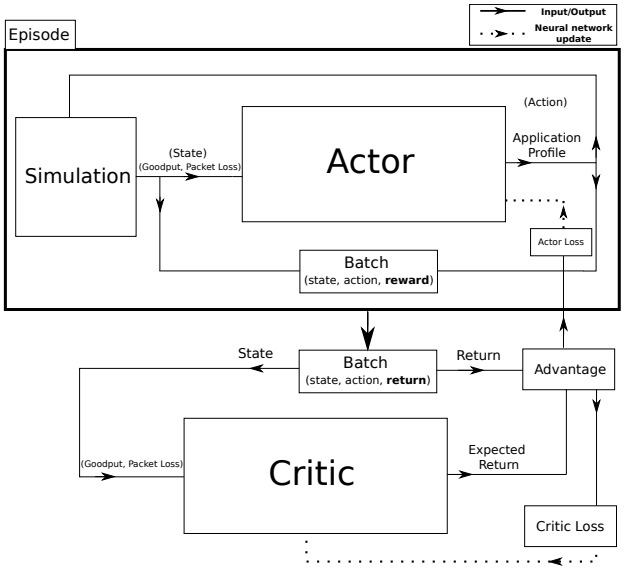

Fig. 2: The structure of the proposed approach using the PPO algorithm

In the first step of training we initialize the hyperparameters and the network weights of the actor and the critic networks, θ and ϕ respectively. Then, we iterate through multiple episodes of training. At the beginning of each episode, we initialize an empty batch **B** that will hold the (s, a, r) tuples. The tuples are used to update the actor and the critic networks at the end of each episode where the simulation environment is restarted. When the simulation starts, the density of nodes is increased gradually over time. The agent starts observing the environment by collecting the state of each node. A reward is then calculated based on the collected states. Next, the previous state, the action taken in the previous state, and the newly calculated reward are added to **B**. The actor network is then used to predict an action for the current state.

To check if our model is improving or deteriorating, we calculate the advantage function at the end of each episode shown in (3). It indicates how beneficial each action was when using the current policy. It is a comparison between the *return* when taking an action in a state and the expected *return* of the state using the previous policy. The advantage function provides an insight on the impact of the action of the agent on the *return* of the state.

$$A_t^{\pi}(s,a) = r_t + \gamma * V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \tag{3}$$

Where $V^{\pi}(s_t)$ is the critic network that gives the expected *return* of a state. The advantage function is then used to update the ϕ parameters of the critic network by performing gradient descent with respect to the loss function (4). We update the critic network parameters so that its predictions match the *return* of the policy.

$$L(\phi) = \mathbb{E}[A_t^2] \tag{4}$$

The θ parameters of the actor network are updated by performing gradient ascent with respect to the loss function shown in (5) where r_θ is the probability ratio shown in (6). r_θ will be greater than 1 when the action is more probable for the current policy than it was for the old policy, it will be between 0 and 1 otherwise. Clipping is done based on ϵ which is a hyperparameter in the loss function, used to avoid the cases where the actions between policies have a larger difference in probabilities. Thus, it prevents taking big gradient steps when updating the policy. The update of the actor parameters makes the actions that resulted in a better *return* more probable in the new policy.

$$L(\theta) = \mathbb{E}_t[\min(r_\theta * A_t, \mathrm{clip}(r_\theta, 1-\epsilon, 1+\epsilon)A_t)] \tag{5}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta old}(a_t \mid s_t)} \tag{6}$$

The actor and critic networks are updated until their losses become negligible. The critic loss becomes negligible when the new policies have no more advantage over the old ones. The actor loss becomes negligible when the policies are producing almost no difference in the predicted action probabilities for the different states. Thus the training is finished. We save the model which is now ready to be deployed in real scenarios.

IV. Simulation Results

In this section, we conduct a performance evaluation of our DRL model. We train our model offline in a simulated environment and we evaluate it in complex scenarios that were not encountered during training.

A. Simulation Environment

Our simulation procedure consists of two key parts: Wi-Fi simulation and DRL Training. Each part is implemented in its own environment. The Wi-Fi simulation is conducted in NS-3 which is an open source simulator offering a detailed Wi-Fi module [30]. The training and validation of the DRL model take place in a python environment using TensorFlow which is an open source platform for machine learning [31]. Both NS-3 and the DRL module exchange information using ZeroMQ (ZMQ) which is an asynchronous messaging library that allows to exchange information between independent applications [32]. The simulation parameters are presented in table I.

TABLE I: Simulation parameters.

Parameter	**Value**
Simulation time (Training)	180 seconds
Simulation time (Validation)	60 seconds
Runs	50
WLAN standard	IEEE 802.11ac
Path loss model	Log-distance
Fading factor	random(0, 2dB)
Traffic	UDP
Channel Width	20 MHz
Packet size	1500 Bytes
Mobility model	Random Walk 2d Mobility Model
Mobility speed	4 m/s
Topology size	Square of boundaries (-30, 30, -30, 30)
α	0.8
β	0.2
Learning rate	0.001

B. Model Training

To start the training process, we prepare a simulation scenario with a varying number of nodes. The total duration of the simulation is set to 180 seconds of NS-3 simulation seconds. We start the simulation with 10 nodes and we split each simulation scenario into 3 phases of 60 seconds. After each phase, we add 10 nodes to the scenario. The number of nodes is chosen in order for the model to explore a variety of states in different network conditions.

Our training reward design of equation 1 is set to favor improving the goodput over the packet loss $\alpha > \beta$. Note that the packet loss seen by the application can be caused not only by collisions but also by buffer overflow.

During training, each NS-3 simulation sends to the DRL module the unique identifier and the state of each node. Then, the DRL module predicts an application profile for each node based on the states and calculates the corresponding rewards. The list of available application-level data rates are [0.7 Mb/s, 1.5 Mb/s, 2.5 Mb/s, 3.5 Mb/s] corresponding to data rates that were used in [20]. Each node transmits with data rate for an interval of time which is set to 250 milliseconds. This duration allows the nodes to interact with the environment multiple times providing a better vision of the environment than a single transmission. Depending on the fault tolerance of the application, a shorter or a longer monitoring duration can be used. At the end of the simulation (Episode), the losses of the actor and the critic networks are calculated. The actor and critic networks are then updated accordingly. The training process is marked as done when the losses become stable and the return is not increasing anymore.

C. Model Validation

The offline training produces an optimized model that maximizes the return. The goal of our model is to adapt the application-level data rate to enhance the performance of the network. We validate the model, which was trained in a few static scenarios limited to 30 nodes, in a larger space of scenarios including mobile scenarios with more nodes (up to 100 nodes). The validation aims at testing the ability of the model to generalize and adapt to scenarios it has not previously encountered.

In what follows, we will compare our DRL model with E2E-MAC [20] and the highest application-level data rate. Papers dealing with the same problem as ours often lack details about the proposed model such as hyperparameters, simulation time, and the environment. This makes producing the same models for comparison almost impossible. Thus we decided to compare our approach with [20] for which the needed details were available in the paper.

First, we perform scenarios with mobility to make our simulation more realistic and to confront our model to dynamic situations. During the scenarios, the density of nodes is increased from 1 to 100, which covers the decrease in goodput for most application profiles.

Figures 3 and 4 show the goodput and the ratio of collisions respectively averaged over 50 runs and include the standard deviations.

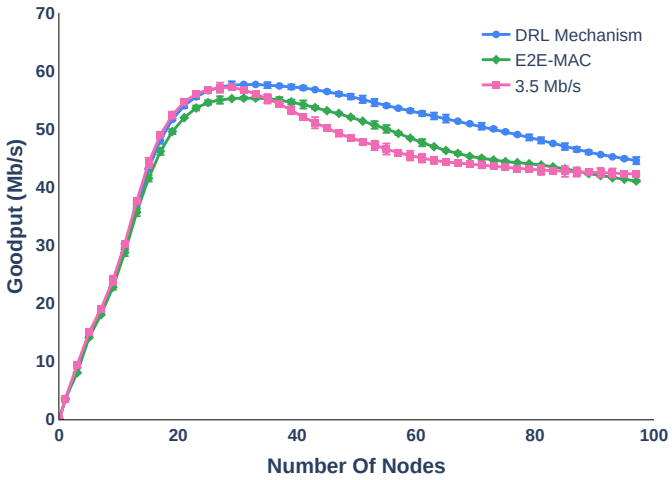

Fig. 3: Overall goodput obtained in the network using the trained DRL model, E2E-MAC, and single application profile.

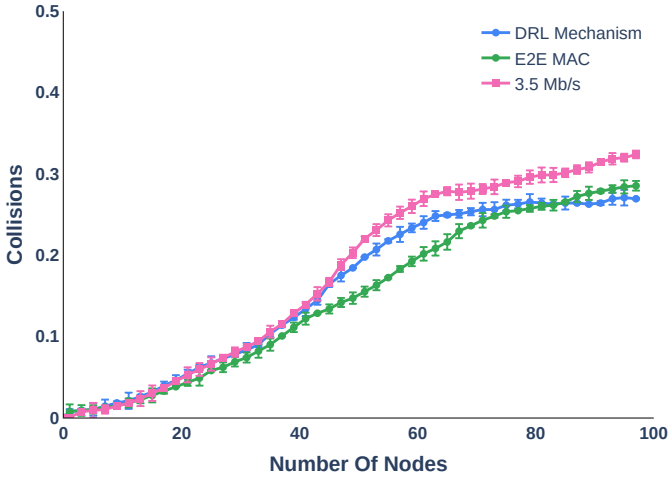

Fig. 4: Ratio of collisions obtained in the network using the trained DRL model, E2E-MAC, and single application profile.

Results in terms of goodput show that using our DRL model we are able to maintain the maximum value of goodput for almost double the number of nodes. Note that E2E-MAC does not reach the maximum goodput value reached by the maximum data rate and our DRL model but is able to maintain higher goodput values for a higher number of nodes compared to the maximum data rate case. When we only have a few nodes in the network, results of the three approaches are close. This is an expected result because the enhancement we are aiming is achievable under saturation conditions.

Results in term of number of collisions show that taking the network state into consideration reduces the overall number of losses. E2E-MAC approach aims gives better results than our DRL model for a number of nodes less than 80. This can be explained by the fact that E2E-MAC takes into account the number of MAC level retransmissions which is a closer estimation of the losses than application level packet loss. For more than 80 nodes, our DRL results in less collisions because under these scenarios most of the nodes switch to the lowest data rate which helps to reduce the overall number of collisions.

In certain situations, the data rate requirement of an application profile can be higher than the physical data rate. In these cases, the DRL model selects application-level data rates lower than the current physical data rate. This is done without knowledge to the underlying physical data rate. It is based on local observations for each node. Nodes suffering from bad network performance are more likely to choose lower application-level data rates. This kind of decisions has a double impact: first they are generating less traffic and thus suffering from less data loss, second they are occupying the channel less often, meaning that other nodes can profit from a lower overall contention and increase their application profile if their performance feedback suggests so.

Overall, the simulation results show that our DRL approach is able to maintain its peak throughput levels for more than double the number of nodes in the network. This is the main aim of the approach, and figure 3 shows this improvement. As for the number of collisions, which is not the main aim of the approach (as apposed to E2E-MAC method), our DRL method was able to reduce them compared to the baseline method, and under very high contention, the DRL method resulted in less collisions than E2E-MAC.

D. Model Complexity

In our approach, the DRL model is trained offline inside the simulator. The exported model, which is used for validation, can be deployed on end devices where calculating the computational overhead is critical. Thus, the computational overhead is important during validation and is directly related to the dimensions of the designed neural network. The overhead is the number of operations inside the neural network starting from the input layer to the output layer. The time and memory overhead of the model predictions depends on the computational power and the memory specs of the hardware. Relatively small neural networks such as our case have an insignificant overhead on end devices.

V. Conclusion and Future Work

In this paper, we explored a DRL mechanism for enhancing goodput in dense Wi-Fi networks. Wi-Fi networks are present in our every day life through many applications. Nevertheless, they are known to suffer from performance degradation under high offered load situations. We explored different saturation situations showing how Wi-Fi achieves low goodput under high offered loads.

In an attempt to increase the goodput in the network, we investigated application-level data rate adaptation using Deep Reinforcement Learning that allows to adapt the offered load depending on the locally observed network state. We proposed a DRL model that takes into account the current performance of the network in order to dynamically choose the most suitable data rate. We used network simulation to train our model offline. This approach is cost-free and can be done prior to real deployments. The DRL model was then validated in dense and more complex scenarios. The aim was to show how the model was able to cope in scenarios that it did not encounter during training. The DRL model learned adapting the application-level data rates according to the observed state of the network. We showed how the training achieved better results in term of goodput under network congestion compared to a similar learning approach and baseline cases.

In our future work, we will test our model on real devices in an attempt to validate the ability of the offline simulated training to serve for real deployments. We will further explore offline training through simulation by offering all available network metrics to the nodes. The aim is to use a reward that includes the current overall network performance instead of per node local observations. Once the model is trained, it can be exploited without the global knowledge of the network performance. Also, we will expend our approach by giving the nodes the ability to adapt parts of the Wi-Fi standard, namely

the CSMA/CA mechanism in order to achieve full potential performance. In addition to that, we will work on more specific applications such as video streaming and construct reward functions that included metrics that relate to the application such as buffer size.

REFERENCES

[1] J. F. Schmidt, D. Neuhold, C. Bettstetter, J. Klaue and D. Schupke, "Wireless Connectivity in Airplanes: Challenges and the Case for UWB," in IEEE Access, vol. 9, pp. 52913-52925, 2021.

[2] C. Shi, V. Andino-Pavlovsky, et al "Application of a sub–0.1-mm¡sup¿3¡/sup¿ implantable mote for in vivo real-time wireless temperature sensing," in Science Advances, vol. 7, 2021.

[3] Ravindranath, N.S. Singh, Inder Prasad, Ajay Rao, V.. "Study of performance of transmit beamforming and MU-MIMO mechanisms in IEEE 802.11ac WLANs," IEEE ICICCT 2017.

[4] George, A. Shaji, A.s.. "A Review of Wi-Fi 6: The Revolution of 6th Generation Wi-Fi Technology," 10. 56-65. 2020.

[5] Garcia, Laura Jimenez, Jose Taha, Miran Lloret, Jaime. (2018). Wireless Technologies for IoT in Smart Cities. Network Protocols and Algorithms.

[6] Jawad, H.M.; Nordin, R.; Gharghan, S.K.; Jawad, A.M.; Ismail, M. Energy-Efficient Wireless Sensor Networks for Precision Agriculture: A Review.

[7] Levallet, N.O'Reilly, N. Wanless, E. Naraine, M. Alkon, E. Longmire, Wade. "Enhancing the Fan Experience at Live Sporting Events: The Case of Stadium Wi-Fi," Case Studies In Sport Management. vol.8, 2019.

[8] Z. Hays, G. Richter, S. Berger, C. Baylis and R. J. Marks, "Alleviating airport WiFi congestion: An comparison of 2.4 GHz and 5 GHz WiFi usage and capabilities," Texas Symposium on Wireless and Microwave Circuits and Systems, 2014, pp. 1-4.

[9] "Cisco Annual Internet Report (2018-2023) White Paper," Cisco, San Jose, CA, USA, White Paper, 2018

[10] S. Szott, K. Kosek-Szott et al, "Wi-Fi Meets ML: A Survey on Improving IEEE 802.11 Performance with Machine Learning," 2022

[11] C. Zhang, P. Patras and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," in IEEE Communications Surveys and Tutorials, vol. 21, no. 3, pp. 2224-2287, thirdquarter 2019.

[12] K. Arulkumaran, M. Deisenroth, M. Brundage, and A. Bharath. (2017). "A Brief Survey of Deep Reinforcement Learning," in IEEE Signal Processing Magazine. 34.

[13] S. Szott et al., "Wi-Fi Meets ML: A Survey on Improving IEEE 802.11 Performance With Machine Learning," in IEEE Communications Surveys & Tutorials, vol. 24, no. 3, pp. 1843-1893, thirdquarter 2022

[14] M. Morshedi and J. Noll, "A Survey on Prediction of PQoS Using Machine Learning on Wi-Fi Networks," 2020 International Conference on Advanced Technologies for Communications (ATC), 2020, pp. 5-11

[15] R. Ali, N. Shahin, Y. B. Zikria, B. Kim and S. W. Kim, "Deep Reinforcement Learning Paradigm for Performance Optimization of Channel Observation–Based MAC Protocols in Dense WLANs," in IEEE Access, vol. 7, pp. 3500-3511, 2019.

[16] D. Kotagiri, K. Nihei and T. Li, "Distributed Convolutional Deep Reinforcement Learning based OFDMA MAC for 802.11ax," IEEE ICC 2021, pp. 1-6.

[17] S. -C. Chen, C. -Y. Li and C. -H. Chiu, "An Experience Driven Design for IEEE 802.11ac Rate Adaptation based on Reinforcement Learning," IEEE INFOCOM 2021, pp. 1-10.

[18] S. Su, W. Tan, X. Zhu and R. Liston, "Client Pre-Screening for MU-MIMO in Commodity 802.11ac Networks via Online Learning," IEEE INFOCOM 2019, pp. 649-657.

[19] M. A. Kafi, A. Mouradian and V. Vèque, "On-line Client Association Scheme Based on Reinforcement Learning for WLAN Networks," 2019 IEEE WCNC, 2019, pp. 1-7.

[20] A. S. Abdallah and A. B. MacKenzie, "A cross-layer controller for adaptive video streaming over IEEE 802.11 networks," 2015 IEEE ICC, 2015, pp. 6797-6802.

[21] A. Ganji, G. Page and M. Shahzad, "Characterizing the Performance of WiFi in Dense IoT Deployments," 2019 ICCCN, pp. 1-9.

[22] E. Ziouva, T. Antonakopoulos, "CSMA/CA performance under high traffic conditions: throughput and delay analysis," Computer Communications, Volume 25, Issue 3.

[23] Y. Edalat and K. Obraczka, "Dynamically Tuning IEEE 802.11's Contention Window Using Machine Learning," Association for Computing Machinery, 2019.

[24] N. C. Luong et al., "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," in IEEE Communications Surveys and Tutorials, vol. 21, no. 4, pp. 3133-3174, Fourthquarter 2019.

[25] F. Chiariotti, C. Pielli, A. Zanella and M. Zorzi, "QoE-aware Video Rate Adaptation algorithms in multi-user IEEE 802.11 wireless networks," 2015 IEEE ICC, 2015, pp. 6116-6121.

[26] I. Sammour and G. Chalhoub, "Evaluation of Rate Adaptation Algorithms in IEEE 802.11 Networks," Electronics 2020, 9, 1436.

[27] A. Gong, T. Zhang, H. Chen and Y. Zhang, "Age-of-Information-based Scheduling in Multiuser Uplinks with Stochastic Arrivals: A POMDP Approach," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1-6

[28] R. Sutton and A. Barto, "Reinforcement Learning: An Introduction," The MIT Press, 2nd edition, 2018

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. https://arxiv.org/abs/1707.06347

[30] G. Riley, T. Henderson, "The ns-3 network simulator," in Modeling and Tools for Network Simulation. Springer.

[31] M. Abadi, A. Agarwal et al. "Large-Scale Machine Learning on Heterogeneous Systems," White Paper 2015

[32] ZeroMQ, http://zeromq.org accessed on 14/02/2022

[33] Medepalli, K. and Tobagi, F. A. "Towards Performance Modeling of IEEE 802.11 Based Wireless Networks: A Unified Framework and Its Applications", Proceedings IEEE INFOCOM 2006.