AI-based Rate Control in IIoT Networks for Improved Energy Efficiency

BJORN VUURENS, University of Twente, The Netherlands

In a Wi-Fi-based industrial IoT (IIoT) network, there are sensors and devices with different Quality of Service (QoS) requirements. In Wi-Fi, one of the methods to achieve QoS requirements is rate control. Many existing algorithms for rate control focus on achieving the highest possible performance. However, in IIoT, it might be beneficial to sacrifice some performance for lower energy consumption, therefore improving the lifespan of IIoT devices and reducing the operation cost of the network. The improvement in energy efficiency is particularly important in IoT networks because devices are usually battery-constrained. This research proposes a deep reinforcement learning (DRL) algorithm to improve energy efficiency while maintaining the QoS requirements. The DRL algorithm adjusts the Modulation and Coding Scheme (MCS) and transmission power to optimise for energy efficiency. We show that the algorithm is able to reduce energy consumption while maintaining QoS requirements.

Additional Key Words and Phrases: Energy Efficiency, Modulation Coding Scheme, MCS, Industrial IoT-networks, IIoT, Rate Control, Quality of Service, QoS, Deep Reinforcement Learning, DRL, NS3.

1 INTRODUCTION

Over the past decades, Industrial IoT (IIoT) has become increasingly important, allowing for data collection, exchange, and analysis. This can improve efficiency and productivity as it reduces the need for labour.

One application of IIoT is real-time production monitoring in a factory environment, allowing companies to identify defects or deviations from their standards [3]. However, this is not the only use case: IIoT powers Autonomous Mobile Robots (AMR), sensors, and AMR video fusion, to name a few. (see figure 1) These various use cases have divergent requirements. For example, AMRs typically require low latency and a significant throughput rate, while moving. On the other hand, sensors have vastly different requirements, only requiring a small throughput and are often being power-constrained [5]. As demonstrated by the examples, all these devices have different Quality of Service (QoS) requirements which have to be taken into account.

The IoT devices in an IIoT require connectivity to share information. Many different types of networks are used for communication. Some popular ones include Wi-Fi, LTE-M, NB-IoT, and Bluetooth (LE) [2]. In this paper, we will focus on Wi-Fi.

In Wi-Fi networks, similarly to other types of wireless networks, channel conditions can differ significantly from time to time. Wi-Fi makes use of the very populated 2.4GHz, 5GHz and 6GHz bands. Because of this, other sources of noise, such as Wi-Fi networks or Bluetooth devices, can significantly impact the amount of noise on these bands, which negatively influences the Signal to Noise Ratio (SNR). To improve robustness against large amounts of noise in the channel, the Modulation and Coding Scheme (MCS) is used. MCS represents the data rate at which the packet is transmitted. When the channel experiences a high noise level, a low MCS will be used, thus limiting the data rate but reducing packet loss. Commonly used algorithms such as Minstrel and Thompson Sampling focus on getting the highest performance from the channel [8, 11].

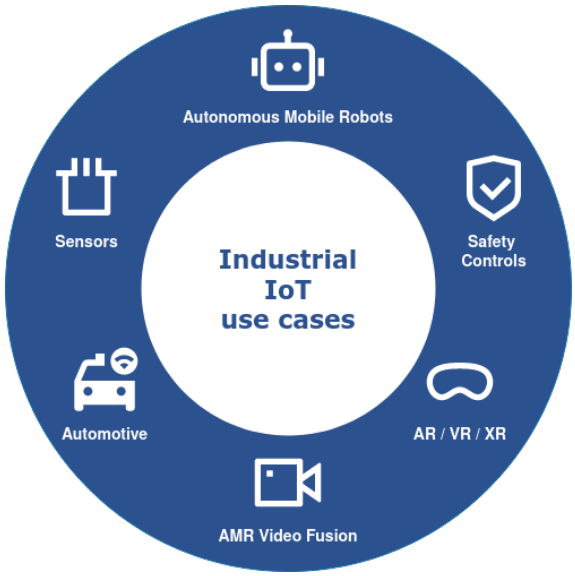

Fig. 1. Wi-Fi IIoT use cases

Although many algorithms exist for rate control, none focus on energy efficiency while meeting QoS requirements. We seek to address this gap by using Deep Reinforcement Learning (DRL), a combination of Reinforcement Learning, where an agent learns by trial and error, combined with Deep Learning, a type of neural network, to allow the agent to decide based on unstructured input data. DRL has the advantage of being able to adapt to different kinds of IIoT networks and it has the potential to deal with the complex decision-making in Wi-Fi. Our DRL algorithm uses a small neural network with a small number of layers since a large network would make the application more power-hungry.

In IIoT networks, energy-efficient devices offer significant benefits by bringing costs down and keeping the operation more sustainable. Although there has been quite some research into optimising rate control in Wi-Fi-based industrial IoT for increased performance, there has not been much research into doing this to reduce power consumption.

This paper focuses on Wi-Fi-based industrial IoT networks and proposes a solution to improve energy efficiency while maintaining the QoS requirements. Our proposed algorithm aims to strike a balance between energy efficiency and spectrum efficiency. In the end, we analyse the results and discuss our findings.

TScIT 41, July 5, 2024, Enschede, The Netherlands

The main research question (RQ) that we address in this paper is: How can Deep Reinforcement Learning (DRL) be effectively utilised for rate control and transmission power control in Industrial IoT networks to optimise energy efficiency while maintaining quality of service (QoS) requirements?

We answer this research question with 2 smaller sub-questions:

- **RQ1**: How can DRL be effectively utilised for MCS selection in industrial IoT to optimise energy efficiency while maintaining the Quality of Service requirements?
- **RQ2**: To what extent can the algorithm from RQ1 be improved by transmission power control?

After first discussing the state-of-the-art and previous works in section 2, we elaborate on the overall setup for the simulations in section 3. Afterwards, we discuss the results in section 4 and future work in section 5. Lastly, we conclude everything in section 6.

2 RELATED WORKS

In recent years, significant work has been done on the problem of rate control in networks. Traditional approaches such as Minstrel and Thompson Sampling focus on improving performance [8, 11]. Minstrel, for example, relies on an algorithm that dynamically adjusts the data rate based on acknowledgement feedback. Therefore, estimates of future success are based solely on past success rates at that data rate, while occasionally exploring new rates to adapt to changing network conditions [8]. Minstrel and Thompson Sampling both spend a significant amount of time on the exploration of the network. Traditional rate control algorithms face difficulties adjusting to changing channel conditions and predicting future states [12].

DRL-based algorithms offer advantages by better adjusting to changing channel conditions and predicting future states. There has already been some research that discusses improving the performance of Wi-Fi by rate control using DRL. In 2022, Lin et al. [2022] proposed a DRL-based solution to improve performance on congested Wi-Fi networks by adjusting the data rate [7]. This solution consistently outperforms Minstrel in terms of throughput. In the same year, Ribeiro [2022] also concluded that they were able to make an algorithm that greatly outperforms Minstrel, both in terms of throughput and in terms of how fast it adapts to SNR variations [10]. Both of these papers show that significant performance improvements can be made.

Most researchers have focused on improving throughput and reducing packet loss in networks while developing rate control algorithms. However, they did not focus on energy efficiency, which can be quite important in IoT networks. The few researchers that focused on energy efficiency looked at different types of networks. A paper from A. Parsa et al. [2022] proposes an algorithm that utilises DRL to improve energy efficiency in future-generation networks. They were able to achieve approximately 1.5x the amount of bits per Watt compared to the QL-AMC solution [9].

Other works discuss energy efficiency for IoT in mobile networks. Al Homssi et al. [2018] have looked into 5G and found that a significant amount of energy could be saved by implementing adaptive control on the transmit power and rate while the device is in transmission mode [1]. Similar research has been done for LTE where Bruno et al. [2014] improved MCS selection using reinforcement learning [4].

Although there are a lot of works that focus on rate control and energy efficiency, now work has been proposed that focuses on energy efficiency while meeting QoS requirements in WiFi-based IoT networks. Therefore, we aim to employ DRL to develop an energy-efficient rate and power control algorithm to meet diverse QoS requirements in an Industrial IoT network.

3 METHODOLOGY

In order to test our deep reinforcement learning (DRL) algorithm, we simulate an indoor factory environment with several IIoT devices, because we believe this is a common use case scenario. This scenario consists of different types of IoT devices randomly scattered through the factory. We simulate this scenario using Network Simulator 3 (NS-3) and we use Python for our DRL agent.

Category	Data Rate (Mbps)
Sensors	0.265
AMR AVG	1
Safety Controls	0.512
Automotive	4
AMR Video	10
AR / VR / XR	2

Table 1. Data Rates for Various IIoT Use Cases

3.1 Network Simulator 3 (NS-3)

NS-3 is used to simulate an IIoT network with several devices, which have different QoS requirements. We have opted for 6 different types of use cases. These 6 use cases can be seen in figure 1. For every use case, we deploy 3 nodes. These nodes use bandwidth as can be seen in Table 1. They receive data from the access point at this rate for two-thirds of the time, using an exponential random variable function. In total, there are 18 nodes connected to 1 access point. The 18 nodes are spread randomly around the access point, an example of how this might look can be seen in Figure 2. The nodes will communicate with the access point over Wi-Fi 4 5Ghz (802.11n).

While we run our simulation, we gather data about the state and quality of the network. We do this in set time steps of 0.1 seconds. We use 0.1 seconds in order to gather enough data about transmitted and received packets within that time period while keeping the time period as short as possible in order to be able to quickly react, should the channel conditions change. In this time period, we keep track of the following things: throughput, packet loss, SNR, and power usage. Every round, we get a new set of settings from the agent. In our observations, we filter packets based on if they are sent with the same settings as put into this agent, in order to lower the noise in the observations. For tracking the SNR, we take the SNR values of every packet which is sent and received within the time slot. To calculate the throughput, we track every packet that is sent at the IP layer. In addition, we track every packet that is received at the IP layer. At the end of the time step, we check how many packets are (1) sent and received, (2) only sent, and (3) what the last sent

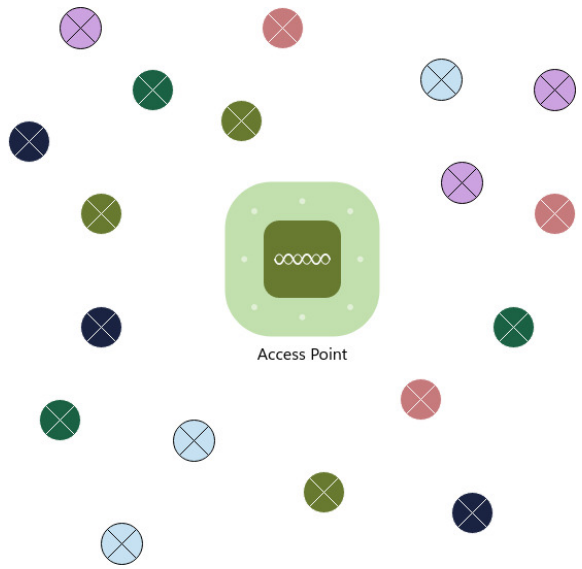

Fig. 2. Possible setup of the access point and the nodes in NS-3 (Every colour represents a different use case)

and received packet is. Since there is a significant chance that if a packet is sent near the end of the current time slot, it will be received in the next time slot, we cannot consider all sent but not received packages 'lost'. To approximate the actual throughput, we assume that the proportion of packets arriving and being lost after the last successfully sent-and-received packet is roughly the same as the ratio observed in the previous part of the time slot. Figure 3 shows an example of such a scenario. Packet 2 is sent during the first time step, but arrives in the second. Therefore, we cannot use the information for time step 1, since the second time step already started and we cannot use it for the second time slot, since it was sent using the settings of time slot 1. With this approach, we can calculate the approximate throughput and packet loss of the channel during a particular time slot.

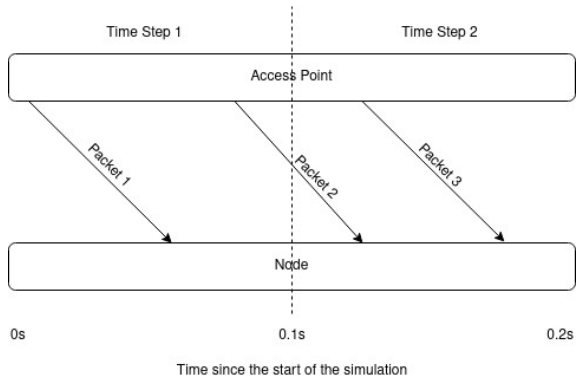

Fig. 3. A packet sent in a certain time step can arrive in the next one.

The last metric that we aim to gather is energy efficiency. For this, we use the NS-3 Energy Framework. This framework consists of multiple classes that help simulate power- or battery-constrained devices. In this research, we use this framework to calculate power draw within a time slot. This allows us to request the total energy usage every round in order to calculate the difference from the previous round and report the energy used in that time step.

3.2 Python DDQN Agent

For the calculation of the MCS and transmit power, AI is used. We use a Double Deep Q-Network (DDQN) agent, a Deep Reinforcement Learning (DRL) solution. We use a model with 2 dense layers containing 256 units each. Below in Table 2 the configuration for the DRL agent is shown.

Parameter	Value
Learning rate	0.005
Gamma	0.5
Epsilon	0.99
Epsilon decrement	0.998
Epsilon end (minimum)	0.01
Batch size	128
Memory size	5000
Replace target	25
Hidden layer activation function	ReLU
Optimizer	Adam

Table 2. Settings in the configuration of the DRL Agent

3.3 Interaction between Python and NS3

Every time slot or step starts with an action determined by the agent based on the data of the previous observation. The simulator executes this action and gathers data. This is sent to the Python Open-Gym agent as the observation alongside the reward of the action, which we discuss in greater detail in section 3.4.3. This interaction is displayed in Figure 4. For interaction between the NS3 simulation script and the Python script, we use the `ns3gym` [6] library.

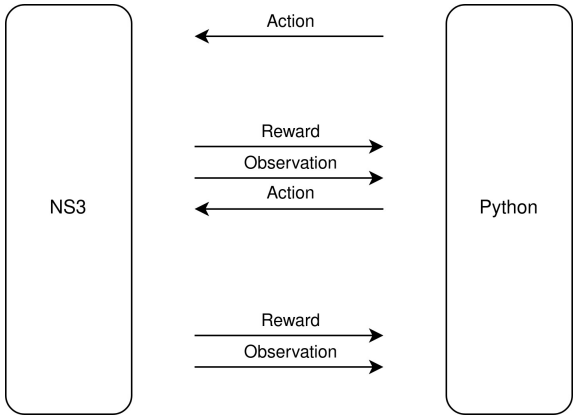

Fig. 4. Communication between Python and NS-3

3.4 Markov Decision Process

As mentioned in section 1, we use rate control and adjust the transmit power to improve throughput and reduce power consumption. This problem can be modelled as a Markov Decision Process (MDP). The agent aims to find a policy π that maximises the cumulative reward over time.

3.4.1 State. The state is represented by throughput, packet loss, SNR, MCS, transmit power & energy consumption. The C++ NS-3 simulator file gathers all this information and sends it to the agent using ns3gym [6].

3.4.2 Action. Based on the data received from NS-3, the agent decides what to do. In our program, the agent can choose 3 actions for the transmit power: decrease, keep the same, and increase. The transmit power can be increased/decreased with steps of 1 with a minimum of 10 and a maximum of 20. For the MCS the agent can choose it directly, it can choose any value between 0 and 7.

In the beginning, the epsilon will be high (see Table 2), this means that the actions of the agent are mostly random. Over time the epsilon decreases, meaning that the neural network gets to decide a larger percentage of the actions.

3.4.3 Reward. The reward function plays a crucial role in evaluating the performance of the AI agent. For this problem, we want the agent to aim for the highest possible throughput while maintaining a low packet loss ratio and energy usage. Therefore, we calculate the reward by taking the throughput score and subtracting the packet loss and energy scores. We defined the following reward function:

$$Reward = (w_T \cdot T) - (w_P \cdot P) - (w_E \cdot E)$$

where:

- T: Throughput
- P: Packet Loss Ratio
- E: Energy Consumption
- w_T, w_P, w_E: Weights for the throughput, packet-loss and energy consumption respectively.

In order to calculate the scores, the values for the throughput and energy consumption are first normalised.

4 RESULTS & DISCUSSION

In order to analyse the performance of our DRL algorithm, we take logs of the throughput, packet loss, transmit power, energy consumption, signal-to-noise ratio (SNR), and reward. We plot these values in Figure 5, 6, and 7 to show how our algorithm performs.

In Table 3 we display the weights we have used for the reward function.

Weight	Value
w_T	1
w_P	0.2
w_E	1

Table 3. The weights for the reward function.

After simulating roughly 3000 steps, since there are 10 steps per second this resulted in 300 seconds (in simulator time), we have averaged our results per 5 steps.

The reward of our DRL agent is plotted against the time in Figure 5. The figure shows that the agent's reward stabilises after about 2000 steps (200 seconds). The initial instability is due to the exploration of the environment by the DRL agent. The agent takes random actions at the start to learn the environment, over time it uses the learned information to make better decisions. In Figure 5 this can be seen as the reward increases over time.

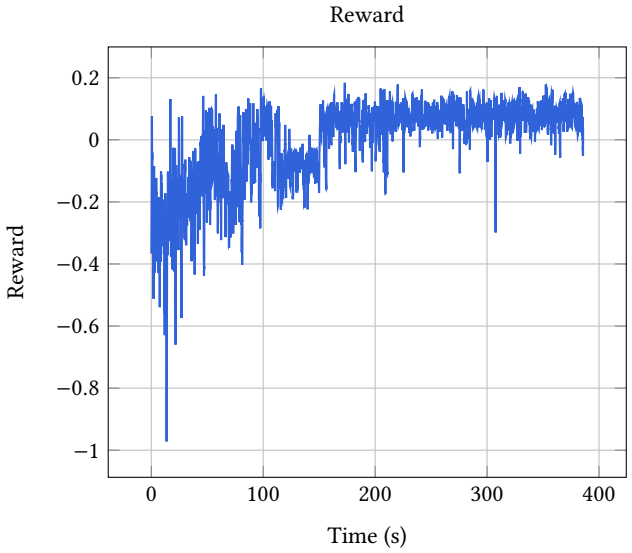

Fig. 5. Plot of the reward against the time.

The increase in reward is also reflected in the increase in throughput of the network, as seen in Figure 6. As the agent learns to make better decisions, the throughput increases around the 150s mark. On the other end, Figure 7 shows us that the energy consumption also fluctuates at the start. We can see it stabilises around the 150s mark to around 120J. This is because, with our chosen weights for the reward function, the agent cares more for throughput improvement (spectrum efficiency) than energy savings (energy efficiency).

4.1 Aggressive energy conservation

In order to examine the agent's learning for energy efficiency, we updated the weights for the reward function and increased the importance of energy consumption, Table 4 shows the updated weights. We ran the simulations with the same QoS requirements and plotted the reward of the DRL agent as shown in Figure 8. The increasing reward function indicates that the agent is learning, however, this time the agent is trying to conserve more energy while achieving less throughput indicating its response to the increased energy efficiency requirement.

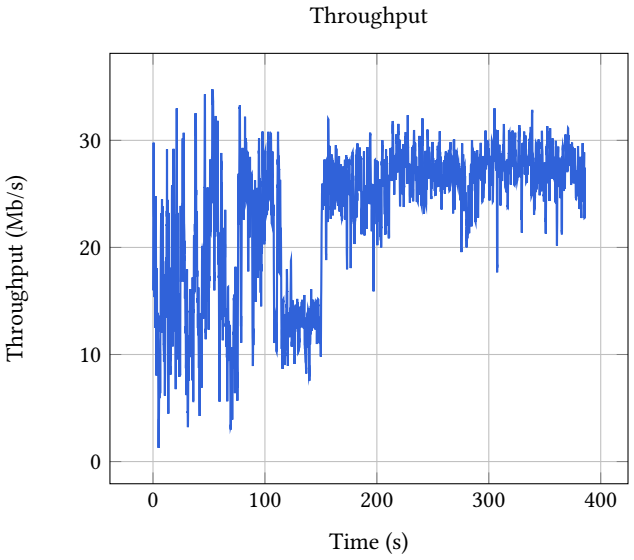

Fig. 6. Plot of the throughput against the time.

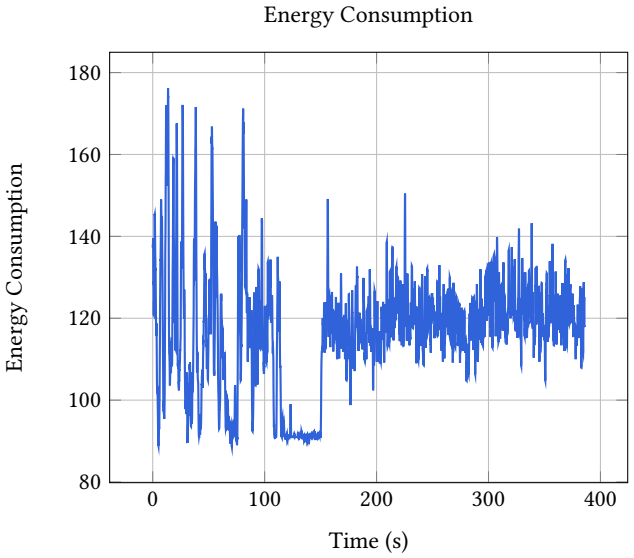

Fig. 7. Plot of the energy consumption against the time.

Weight	Value
w_T	1 (same as before)
w_P	0.2 (same as before)
w_E	3

Table 4. The weights for the reward function.

Figure 9 shows the throughput achieved in the network by the DRL agent in the aggressive energy efficiency case. We can notice that the throughput of the network has reduced, however, if we look at the energy efficiency in Figure 10, there is a considerable energy efficiency gain achieved by the agent.

From the results, we can see that there is a trade-off between energy efficiency and spectrum efficiency in an IoT network and the DRL agent is able to capture this trade-off. By optimally setting up the weights of the reward function of the DRL agent, this trade-off can be exploited depending on the requirements in the underlying IoT network.

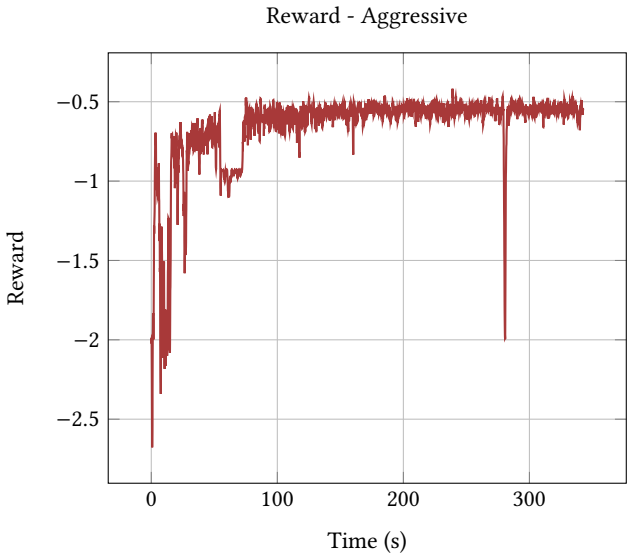

Fig. 8. Plot of the reward against the time when using an aggressive energy-conserving reward function.

4.2 Discussion

Even though our DRL agent was not able to provide optimal results regarding the MCS selection and transmit power values in given scenarios, we believe our research offers significant contributions. We show that it is possible to save energy using this method, while still meeting QoS requirements. However, more work is needed to refine the parameters for the DRL agent, to achieve better results.

Additionally, we show that the DRL agent is capable of learning how to balance throughput and energy efficiency based on the reward function. This can be used to optimise the agent for specific IIoT networks.

One aspect worth mentioning is that when we use transmit power directly as a measurement of energy consumption, the agent's reward stabilised more quickly, and also was more stable near the end than when using direct energy consumption measurements. This can be explained by looking at the data coming from the simulator. Transmit power seems to be a stable value compared to energy consumption, which seems to be more noisy. The energy consumption measurements take the actual power draw, which is dependent on the transmit power but also the fraction of the time the node or AP was in transmission mode versus, for example, idle mode. There

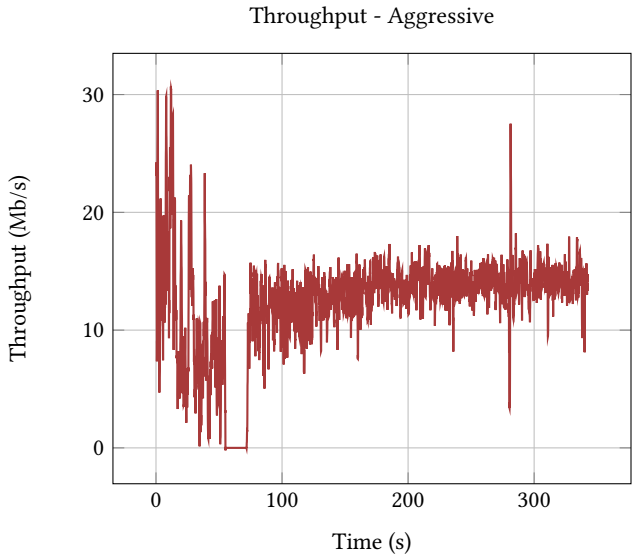

Fig. 9. Plot of the throughput against the time when using an aggressive energy-conserving reward function.

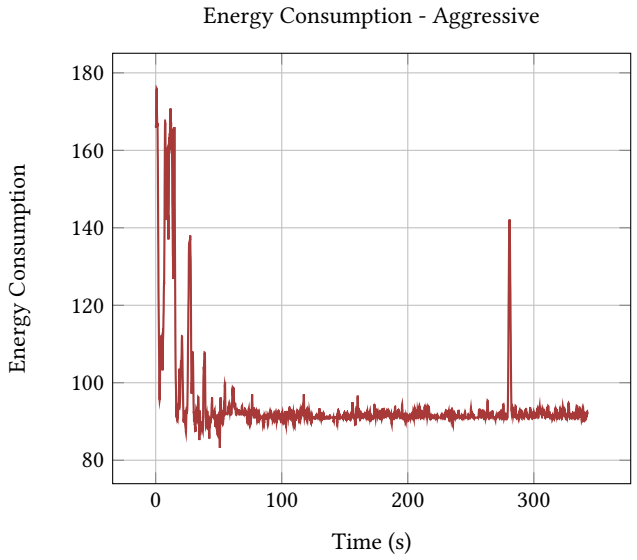

Fig. 10. Plot of the energy consumption against the time when using an aggressive energy-conserving reward function.

are situations where measuring the actual energy usage can make it more difficult for the DRL agent to learn. For example, in our log data, we found scenarios where the agent chose to reduce the transmission power (and keep the MCS the same) but the energy consumption went up. This was, among other things, due to more packets being sent. However, only using transmit power can fail to show improvement in some scenarios. For example, we have state 1 where we use MCS 0 with a transmit power of 16, and in state 2, we use MCS 7 with a transmit power of 16. If we send 10 packets in both slots and all of them arrive, state 2 probably was more efficient since the node was in transmission mode for a shorter period of the time step.

Since the DRL agent was deployed on the Access Point (AP), it does not consider every node individually, but rather the whole network at once. It may be beneficial to treat the nodes individually to maintain different MCS and transmit power values per node because every node is at a different distance from the AP and has a different data rate. This could reduce energy consumption while maintaining QoS requirements since closer nodes can use lower transmit powers.

5 FUTURE WORK

Our approach to improving throughput and energy efficiency in IIoT networks has shown promising results, however, more work is needed in order to further optimise the network performance even more.

In addition, we have only tested down-link scenarios, whereas it is more realistic to also include up-link in the scenarios, however, due to time constraints we have not been able to include it in this research.

As mentioned before in section 4.2, the agent changes the MCS and transmit power for the network as a whole in order to optimise the network. Future work is needed to see if it would be beneficial to treat every node separately. However, this requires a more complex algorithm and is therefore not included in our research.

Lastly, we have not been able to spend enough time optimising the parameters for the DDQN agent and the weights of the reward function. Optimising these might offer significant improvements in the throughput and power savings.

6 CONCLUSIONS

The increase in IIoT networks in recent years has made it increasingly important to develop ways to lower energy consumption in IIoT networks. Therefore, this research proposes a Deep Reinforcement Learning (DRL)-based algorithm designed to improve energy efficiency in an Industrial IoT network by leveraging rate control and adjusting the transmit power.

We developed an algorithm that adjusts rate control and transmit power. We show that it can achieve QoS requirements while conserving energy. In addition, we show that we can adjust it to different network requirements, by changing the weights of the reward function.

Although these results indicate that it is possible to optimise for IIoT networks using a DRL-based algorithm in order to gain a reduction in energy consumption while maintaining throughput requirements, more research is needed to further optimise the algorithm.

A DECLARATION OF USAGE OF AI AND AI-ASSISTED TECHNOLOGIES

During the preparation of this work, the author used Grammarly in order to reduce spelling and grammar mistakes. During the research itself, the author used ChatGPT to improve the understanding of the NS3 documentation and small sections of the theory. After using

these tools/services, the author reviewed and edited the content as needed and takes full responsibility for the content of the publication.

REFERENCES

[1] Bassel Al Homssi, Akram Al-Hourani, Karina Gomez Chavez, Sathyanarayanan Chandrasekharan, and Sithamparanathan Kandeepan. 2018. Energy-Efficient IoT for 5G: A Framework for Adaptive Power and Rate Control. In *2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*. IEEE, Cairns, Australia, 1–6. https://doi.org/10.1109/ICSPCS.2018.8631733

[2] Havot J. Albeyboni and Ismail A. Ali. 2023. LPWAN TECHNOLOGIES FOR IOT APPLICATIONS: A REVIEW. *Journal of Duhok University* 26, 1 (April 2023), 29–42. https://doi.org/10.26682/sjuod.2023.26.1.4 Number: 1.

[3] Hugh Boyes, Bil Hallaq, Joe Cunningham, and Tim Watson. 2018. The industrial internet of things (IIoT): An analysis framework. *Computers in Industry* 101 (Oct. 2018), 1–12. https://doi.org/10.1016/j.compind.2018.04.015

[4] Raffaele Bruno, Antonino Masaracchia, and Andrea Passarella. 2014. Robust Adaptive Modulation and Coding (AMC) Selection in LTE Systems Using Reinforcement Learning. In *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*. IEEE, Vancouver, BC, Canada, 1–6. https://doi.org/10.1109/VTCFall.2014.6966162

[5] WBA Wi-Fi 6/6E for IIOT Project team. 2022. Wi-Fi 6/6E for Industrial IOT Enabling Wi-Fi determinism in an IOT world.

[6] Piotr Gawłowicz and Anatolij Zubow. 2019. ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research. In *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)* (Miami Beach, USA). http://www.tkn.tu-berlin.de/fileadmin/fg112/Papers/2019/gawlowicz19_mswim.pdf

[7] Wenhai Lin, Ziyang Guo, Peng Liu, Mingjun Du, Xinghua Sun, and Xun Yang. 2022. Deep Reinforcement Learning based Rate Adaptation for Wi-Fi Networks. In *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*. IEEE, London, United Kingdom, 1–5. https://doi.org/10.1109/VTC2022-Fall57202.2022.10012797

[8] Andrew Mcgregor and Derek Smithies. 2010. Rate Adaptation for 802.11 Wireless Networks: Minstrel. 14. https://blog.cerowrt.org/papers/minstrel-sigcomm-final.pdf

[9] Ali Parsa, Neda Moghim, and Pouyan Salavati. 2022. Joint power allocation and MCS selection for energy-efficient link adaptation: A deep reinforcement learning approach. *Computer Networks* 218 (Dec. 2022), 109386. https://doi.org/10.1016/j.comnet.2022.109386

[10] Héber Miguel Severino Ribeiro. 2022. Using Deep Reinforcement Learning Techniques to Optimize the Throughput of Wi-Fi Links. (2022).

[11] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. 2020. A Tutorial on Thompson Sampling. https://doi.org/10.48550/arXiv.1707.02038 arXiv:1707.02038 [cs].

[12] Ibrahim Sammour and Gerard Chalhoub. 2020. Evaluation of Rate Adaptation Algorithms in IEEE 802.11 Networks. *Electronics* 9, 9 (Sept. 2020), 1436. https://doi.org/10.3390/electronics9091436