Bringing Intelligence to Wireless Sensor Nodes: Improving Energy Efficiency and Communication Reliability in Sensor Nodes

DELAINO TODOROVIC, University of Twente, The Netherlands

Wireless Sensor Networks (WSN) are becoming more dense to enable many new smarter use cases in industry, healthcare and agriculture. WiFi-based wireless sensor networks are becoming very attractive because of high-bandwidth, large coverage and low-powered sensors being cost-effective. Even though WiFi offers low power consumption, resources are still limited in wireless sensor networks and the identification of how to efficiently use the energy of a wireless sensor node has been an open research topic for years. In this paper a lightweight distributed reinforcement learning framework for wireless sensor networks is presented. This framework allows sensor nodes to control their transmit power in such a way that they still communicate reliably with minimum energy consumption which increases the network life-span.

Additional Key Words and Phrases: Wireless Sensor Network, Wireless Sensor Node, Distributed Machine Learning, Energy Efficiency, Communication, Reliability

1 INTRODUCTION

Wireless sensor networks (WSNs) have been of great interest among both consumers and manufacturers for a long time now [10]. This technology is widely used for collecting and analysing data and is considered one of the most promising technologies, because of its size, cost-effective and easily deploy-able nature [1][7]. There are many different applications for wireless sensor networks such as in healthcare, environment and agriculture, military, industry, and transportation systems. One example is an underwater sensor network created for long-term monitoring of coral reefs and fisheries [18]. It is expected that these networks work autonomously for a long period of time.

However, wireless sensor networks have, just like any other technology, limitations. A wireless sensor network consists of multiple autonomous, small sensor nodes that monitor, gather information and transmit that information to a base station (sink) for further analysis. A single node is generally equipped with a limited and unchangeable power source, a battery, and has low computational power. Therefore, energy is the most valuable resource and must be conserved as much as possible. Identifying how to efficiently use this energy and extend the lifetime of the network is a critical issue [7]. It is the communication between nodes and the sink that requires the most energy [5]. The transmission power can be responsible for up to 70% of the total power consumption. It can affect other important aspects like latency and throughput as well[17]. In the desired situation the node transmits at such power that it can still communicate reliably while minimising the amount of energy used. Reliable communication meaning guaranteed packet delivery. By assigning a optimal transmit power per-node a lot of energy of sensor nodes can be saved.

Transmission power control (TPC) in wireless sensor networks is a popular research topic, however only few have applied machine learning to TPC. The different applications of WSNs share common challenges such as dynamic environments and goals such as network longevity [6]. This is where machine learning can play an important role. Machine learning techniques can be applied at network and node level which enables intelligent behaviour and adaptability [14]. Machine learning allows a sensor network to learn from previous experiences so it can learn and adapt to the dynamic environment. By learning and adapting the transmission power to the environment, sensor nodes can autonomously choose the optimal transmit power as per their desired communication requirements. Machine learning, however, does come with a price which are energy and computational power. The predictions and computations cost a considerable amount of energy which could do more harm than good and shorten the network's lifespan [3]. But this is only when we speak of a centralized system which means that all computations are done by one node in the whole network. To solve the computational power issue it is possible to distribute the machine learning workload across multiple machines. This means that instead of a centralized system where a central node does all the computational work, a decentralized system is used where the workload is distributed across multiple nodes. The benefit of distributed machine learning is that it requires less computational power and energy consumption since the nodes only consider information from their own environment and not the whole network [7].The machine learning techniques applied in WSNs generally speaking are only Reinforcement Learning and Fuzzy. However, most contributions are mainly focused on routing protocols[1] and sleep scheduling [12].

In this paper a deep reinforcement learning algorithm (deep Q-learning) is proposed and tested. The algorithm is implemented in NS-3 which is a network simulator and the de-facto standard for academic and industry studies in the areas of networking protocols and communication technologies. With the help of NS-3-Gym [8] important information like throughput, transmission power and transmission energy cost is fed to the algorithm. Network performance is then evaluated by analyzing energy conservation of each node while ensuring their throughput requirements. In this paper is the following research question answered: "How can distributed machine learning help wireless sensor nodes communicate reliably with minimal energy consumption?".

2 PARTIALLY OBSERVABLE MARKOV DECISION PROCESS & REINFORCEMENT LEARNING

Like mentioned earlier are WSNs usually deployed in uncertain, dynamic environments. With the help of a mathematical framework called Markov decision process (MDP) can these dynamics

TScIT 37, July 8, 2022, Enschede, The Netherlands

be modeled for decision making under uncertainty. Reinforcement learning is used to solve the made MDP models [2]. In RL there is an agent or multiple agents that interact with an environment. The sensor nodes in this case serve as agents that interact with the uncertain environment. At each decision time (T) the agent takes in the environment which is represented as a state (S) and with the knowledge that it has, it will perform an action (A) to reach a certain specified end-goal. Each action taken has either a positive or negative impact on the system. This impact is represented as a reward (R) (or punishment). When the reward is received, the agent will take in the next state (S') at the next decision time. The MDP is defined by a tuple (S, A, R, T) where,

- S is a finite set of states,
- A is a finite set of actions,
- R is the reward obtained after action A, and
- T is a finite or infinite set of decision epoch

The goal of the agent is to maximize the expected reward over a number of decision epochs[2, 4, 17].

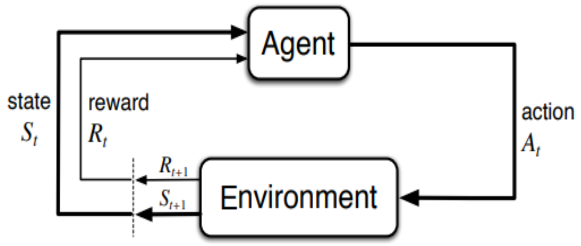

Fig. 1. Reinforcement learning: single agent model

In multi-agent MDP, each agents gets a full observation of the environment state. However, in our distributed scenario each agents only receives part of the environment state instead of the full state. To be more specific, each agent only receives local information of the environment state. Each agent will execute actions not knowing what the other agents' actions are. This is called Partially Observable Markov Decision Process (POMDP), because each agents only observes a specific part of the whole environment. When communication is costly, like in WSNs, POMPD is a good framework [2].

3 RELATED WORK

Increasing energy efficiency with the use of machine learning in WSNs is a widely studied topic. In the previous section is explained how RL is one of the possible solutions to MDP. It is therefore important to focus on other contributions that have addressed MDP for power control and used RL for increasing energy efficiency in general. Alsheikh et al. [2] divided all contributions into categories of applications of MDP in WSNs. Most contributions use classic MDP with some contributions using POMDP. The authors of [16] use POMPD in their paper where an agent faces the issue that it does not know the state of the channel. To get a better idea of the state of the channel it carries out belief states as an estimation of the environment to be able to make better decisions. The agent carries out actions like transmit in low power, transmit in high power, wait idle, or listen to the channel. For instance, if the channel is busy then the probability of successful packet delivery is low. If the channel is in an idle state then the probability of successful packet delivery is high and it should transmit. Every actions taken by the agent has a cost and will affect its reward. The authors of [16] have done another paper where they use classic MDP instead of POMDP. The problem they want to solve this time is at what power level should a node transmit to maximize the chances of successful packet delivery when there is interference. Since this is MDP, the node gets all the information of the environment including the interfering nodes which makes the problem a bit easier to solve than POMDP [17]. The authors of [4] created QL-TPC, a Q-learning algorithm for TPC. They want to increase energy efficiency of sensor nodes while emphasising on Quality of Service (QoS) meaning that they want to conserve as much energy as possible while still having reliable communication. Their RL algorithm is also based upon POMDP. They tested their QL-TPC in both NS-3 and in a real-life scenario. In their NS-3 scenario they use IEEE 802.15 wireless personal ares network (WPAN) as communication protocol which is ideal for communication within a short range and it is private so there is not interference from other devices. Their algorithm also makes use of game theory meaning that all nodes work towards a common objective, minimal energy consumption. The spacing between the nodes is around 2 to 4 meters and the packet size is 50 bytes. However, 2 to 4 meters is a fairly short distance meaning that the probability of a successful packet delivery is higher at lower transmission power level than if the distance between transmitter and receiver were to be further. Also the amount of states is denoted as 68 meaning that it is a discrete space and not a continuous space where the number of states can be infinite. The authors of [11] focus on maximizing average throughput per total consumed energy. They use a RL method called actor-critic where the node is an actor and every time the actor chooses an action the critic will tell the actor whether that action was good or not. The actor selects an action according to the Gibbs softmax method. They propose a single agent point-to-point communication scenario and a multi-agent scenario. In their multi-agent scenario the agents learn the optimal transmission power and modulation level by checking the channel gain of the previous transmission, number of packets in the queue and the level of interference. In the point-to-point scenario the only difference is that interference is not considered. The reward function is set as the total amount of successful transmission over the total energy consumption. In [10] the authors considered a centralized approach where certain power levels cannot be used when the battery is at a certain percentage level. The problem is formulated as a MDP where full information about the environment is available. The central node calculates the optimal policy through solving the bellman equation and sends it to all other nodes. The authors of [9] introduce a reinforcement learning-based sleep scheduling algorithm. They formulate their problem as a multi-agent MDP. All nodes receive information about the whole network. The agents learn when to switch on and off based upon computational task and residual energy. This algorithm consists of both a centralized and offline distributed implementation.

These papers have researched TPC in WSNs and formulated the problem as a MDP and solved it using RL. However, most papers consider only one agent in a point-to-point communication model. Another problem is that a centralized approach where information from nodes is send to one sink node to do calculations is unfeasible. If only one node would have to do all calculations in a large WSN it would require a lot of computational power and energy, two things sensor nodes to not possess. WSNs are deployed in uncertain environments and have to deal with that, this is why a decentralized (distributed) approach is more appropriate. Each node only has to deal with local information and does not have to be synchronized with the whole network. Also, no papers have a solution of a continuous observation space where there are many states. This is why deep Q-learning is used, to deal with a large number of state spaces. The last point is that simulation parameters of the papers are not realistic. A transmitter-node distance of 2-4 meters is not far enough if transmit power level wants to have any significant value, even though WPAN is used.

The contributions of this paper are:

- A decentralized Deep Reinforcement Learning algorithm (Deep Q-learning) applied to TPC in WSNs based on Partially Observable Markov Decision Process. Since distributed learning is used all nodes will only receive information about their own local environment and not about the whole network which makes the network partially observable.
- Utilization of Deep Q-learning method so that the POMPD can be solved with a continuous state space.
- Large-scale network applicability such that networks improve their battery life while performing their desired function.
- An algorithm tested on IEEE 802.11 WiFi communication protocol.
- Realistic simulation environment with nodes distanced up to 40 meters and with interference. The nodes will all try to fight for the channel so they can transmit their packets to the access point.

4 METHODOLOGY

4.1 Environment

It is important to create a good realistic environment to test the algorithm. Without the environment no information can be passed to the agents. So to begin with, the environment is created in NS-3 to the following scenario: the network consists of seven nodes of which six are WiFi (IEEE 802.11a) stations that act as sensors and one is an access point which will serve as a base station (Fig 2). All six nodes send traffic to the base station. The distances between the nodes and base station varies in such way that transmission power has effect on the network. Also, because all six nodes try to transmit to the access point at the same time there will be interference. It is important that the traffic is not too light so if any packets are lost, the sensor nodes can try to re-transmit them. Network performance is evaluated by analyzing the throughput and energy consumption of each node and the aggregated throughput of the whole network.

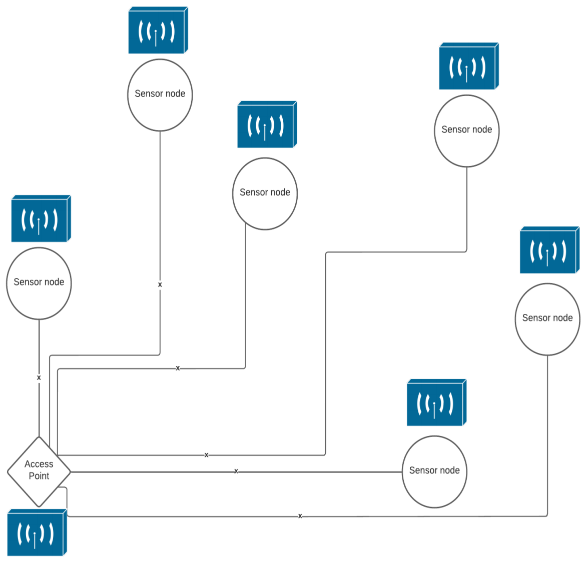

Fig. 2. Wireless sensor network model

4.2 Deep Q-learning

Deep Q-learning (DQL) is an extension of the Q-learning algorithm that uses a deep neural network to calculate what the best action would be given a certain state. The agents will act in the environment stated in 4.1 as each will adopt a wireless sensor node. However, they will not work together. All agents only receive local information and calculate the best action it can make with this local information, making it a decentralized POMDP. Like explained earlier, a MDP is defined by a tuple. This tuple of (s,a,s',r) is then saved in the replay memory of the agent.

DQL works within a continuous space with a discrete set of actions [13]. The state space is defined as a vector filled with three values representing the local environment: throughput, transmission power level (TxPower) and energy consumption.

$$State = [Throughput, TxPower, Energyconsumption]$$

However, even though throughput is continuous, it is made discrete for the sake of having a lower amount of states. Thus throughput in range of {0, 250, 300, ..., 400, 500} kilo bit per second (kbps). For energy used {0.070, 0.075, ..., 0.095, 0.100} joule has been done the same.

The action space is also discrete. The agent can choose between three actions: up the transmit power with one dBm, lower the transmit power with one dBm or keep the same transmit power. Transmit power levels range from {0, 1, ... , 14, 15}. The action space is defined as a vector filled with only one value which is either 0, 1 or 2 meaning TxPower +1, TxPower -1 or do nothing respectively.

$$Action = [actionValue]$$

A wireless sensor node does not have much computational power and memory. This is why the deep neural network only consists of two hidden layer being of length 32 and 16, respectively.

For action choosing the epsilon-greedy strategy is applied. This way the agent will explore new situations from which it can learn. The epsilon value determines the probability of taking a random action over the best action. Epsilon will start out at 1.0 and will decrement every iteration with 0.005 until it reaches a value of 0.1.

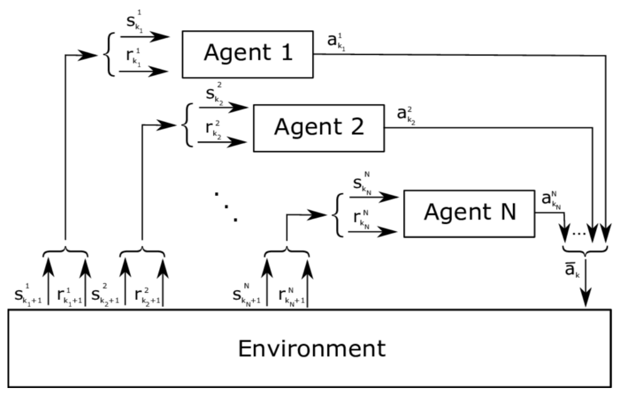

Fig. 3. Reinforcement learning: multi-agent model

5 NETWORK MODEL & NS-3 SIMULATION SETUP

A more in-depth explanation is given about the environment in this section.

5.1 Setup

The setup consists of seven nodes that use (WLAN) protocol IEEE 802.11a for communication. Six are stations that act as sensors and one is an access point which will serve as a base station (Fig 2). The sensor nodes measure the temperature of parts that are placed in an oven of a station in a production line.

5.1.1 *Mobility model.*

At the start of each simulation are the six wireless sensor nodes randomly placed with four nodes distanced between 20-40 meters and two nodes a bit closer distanced between 0-20 meters from the access point. By having four nodes further away the delay will increase which will mean that the throughput will decrease since throughput is directly affected by latency [15]. The agents will learn that they need more power which will increase the throughput so that the nodes fit the minimum throughput requirements.

5.1.2 *Traffic generation.*

All nodes generate traffic of one packet of 100 bytes with an interval of 1ms for ten seconds long, making it a total of 10.000 packets send to the access point per node. This much traffic is send, because the network needs to be under load such that the nodes need to fight for the channel since there is interference.

5.1.3 *NS-3-Gym.*

NS-3-Gym is a framework that integrates both Open AI gym and NS-3 for the usage of RL in NS-3. Open AI gym is a toolkit which is used to test and train RL models in different environments. The simulation will pass the state which is a vector of throughput, transmit power and the transmission energy cost via the NS-3-Gym framework to all the agents. All six agents are initialized with hyper-parameter values stated in Table 1

5.1.4 *Energy model.*

The energy model used is the standard WiFi energy model provided by NS-3 called *LinearWiFiTxCurrentModel*. This model calculates the energy costs of the nodes based on in what mode they are in, transmitting, receiving or idle. The energy draw of different power levels is calculated with

$$Powerlevel/(voltage * eta)$$

with eta being the efficiency of the power amplifier which is set to 0.1 by default.

Parameters	Value
Number of WiFi stations	6
Number of access points	1
Packet payload	100 bytes
Interval time	1ms
Number of transmission levels	16
Simulation time	10 seconds
Discount rate	0.99
Epsilon start	1.0
Epsilon decrement	1.0 * 0.997
Memory size	2 kB
Learning rate	0.005
Iterations	30

Table 1. Simulation parameters

5.2 Reward function

The reward function of the agents is based upon the throughput and the transmit power level. The minimum required throughput is set to 300 kilo bit per second (kbps). Any value under 300 kbps does not assure reliable communication between nodes. If the throughput is under 300 kbps the agent will be punished. The goal of this algorithm is to increase the lifespan of a sensor node. Transmitting at a higher power level costs more energy. The goal is to find the minimum available transmission power level that allows for reliable communication. Every level has its small punishment such that the agents learn that transmitting at a higher power level is not encouraged. The access point also plays a role, namely as a critic. Every agent that transmits at such power that minimum throughput requirements are met will receive an extra reward of 1. The access point will send one packet to each node every millisecond to check the throughput. There are two reward functions defined.

$$R = 0.6 * -x + 0.6 * (y - powerlevel * 0.1)(+1)$$

$$R = 0.4 * x + 0.6 * (y - powerlevel * 0.1)(+1)$$

Where x is -1, 1 or 2 if throughput < 300, 300 < throughput <= 500 and throughput > 500, respectively and where y is 1 if throughput < 300 and 300 < throughput <= 500, and 2 if throughput > 500, respectively.

The first function is for when the minimum throughput requirement of 300 kbps is not met. The second function is for when minimum throughput requirements are met. In the functions, weights are set for throughput and power level. If minimum throughput requirements are not met then it is more important to get back to that level than at which power level the node is transmitting.

This is why a weight of 0.6 is set to throughput and 0.4 to power level. However, if minimum throughput requirements are met then the weights are swapped. The agent can then focus on minimizing energy consumption.

6 RESULTS

The results of the performance of the network with the DQL-agents are compared with the performance of a network without any machine learning.

6.1 Energy consumption analysis

Transmission power level and the on-time of a node both affect the energy consumption. The model used does not have any sleeping techniques, so nodes stay on all the time. In figure 4 the transmissions costs are plotted for every node. Node 2 and 3 do consume more power, but there are nonetheless also stationed further away from the access point. What is interesting is that Node 1, also a node that is stationed further away, is consuming less energy than Nodes 2, 3 and 4. All nodes almost consume the same energy consumption. In figure 5 transmission costs are plotted of a network that does not use any machine learning. This network uses a fixed power level which is 0 by default. Interestingly though are the closest nodes the biggest energy consumers. This could be because of interference meaning that they had to re-transmit packets which leads to longer transmission and on-time.

6.2 Throughput analysis

As mentioned earlier is throughput affected by delay. In figure 6 and 7 the effect is clearly depicted. Nodes 1 and 2 get throughput up to 1 Mbps since they are closer to the access point and thus endure less delay. The DQL-agents look like they are struggling, but this is actually how they learn. By trying all sorts of variation of power levels and learning which ones are optimal. Throughput of nodes 3 to 6 is fluctuating between 180 and 450 kbps. The minimum required throughput of 300 kbps is met in over half of the iterations and will only increase with more training. Nodes 3 to 6 in the network without algorithm are fluctuating a lot between 200 and 600 kbps meaning that for many iterations they could not communicate reliably.

6.3 Memory and computational requirements

A sensor node generally does not have a lot of memory available and does not have a lot of computational power. An Esp32 micro controller has about 520 kB of memory available. The algorithm's model of 32 and 16 hidden layers uses 42.8 kB of memory. The replay memory is 2 kB which adds up to 44.8 kB. The total size of the states, actions and rewards are 120 bytes * 180 states, 5 bytes * 3 action values, and 2 bytes * 15 rewards is 21,645 kB. 44.8 + 21,645 = 66,445 kB. Which means that the minimum required memory size is roughly taken 67 kB. The algorithm can run on an Esp32.

7 DISCUSSION

The full potential of the algorithm has not been met yet. This is due to limitations of the NS-3-Gym framework. A learning algorithm needs many iterations before it shows what the improvements

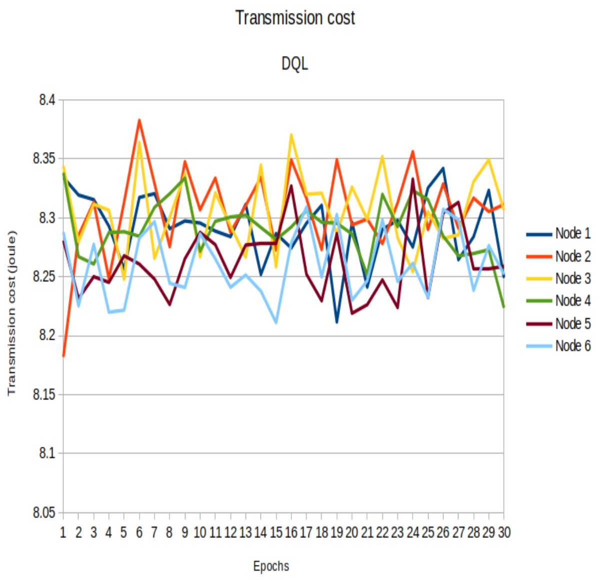

Fig. 4. Transmission costs with DQL-agents

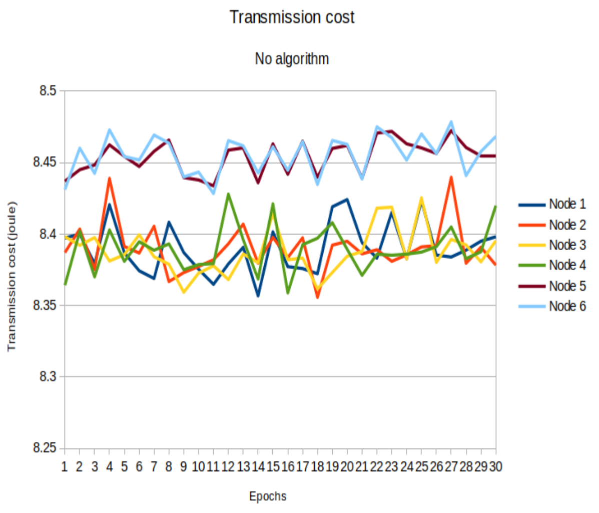

Fig. 5. Transmission costs without any RL

are, because it needs to go over every state that is available and learn what the best action is in that particular state. 30 iterations is unfortunately not enough to see actual improvement. The reason why only 30 iterations are done are because of limitations of NS-3. When the simulation is done it will destroy all objects created. A solution could be to loop the simulation, however the environment needs to be reset after each iteration in order to optimally learn. This is why the simulation has to be manually started every each iteration. To have at least 100+ iterations it would take a very long time. However, the algorithm did show that there is an improvement

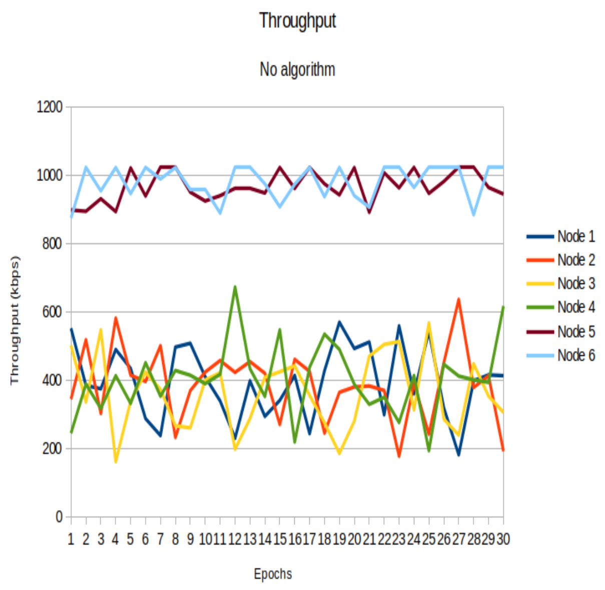

Fig. 6. Throughput withouth any RL

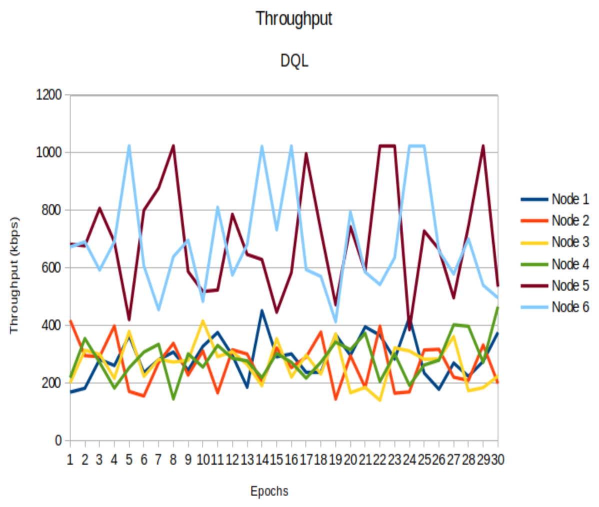

Fig. 7. Throughput with DQL-agents

in energy conservation. The agents do manage to keep the overall energy consumption lower than of the network without learning. In short-term this will not matter, however in long-term this will have a significant effect which is a longer network lifespan. The algorithm uses roughly taken 67 kB of memory with 2 layers of size 32 and 16 respectively which is very lightweight since an Esp32 520 kB memory available. The layer sizes could be increased to get better performance. A well-working autonomous sensor node with TPC should be able to react to changes in the environment. An issue that arises is when a sensor node is moved to a different location further away from the access point. This would mean that delay increases and throughput decreases. The sensor node will adapt to this change and will transmit at a higher power level. The issue is that because of the increase in power, the access point will detect more interference. A solution could be that the other nodes transmit at a lower power level, but lowering other nodes' power levels influences the network performance in a bad way. The benefit of the algorithm proposed is that the nodes learn the minimum power available for reliable communication without having to rely on previous knowledge of the environment.

8 CONCLUSION

A lightweight distributed transmission power control algorithm for wireless sensor nodes is proposed. The algorithm is tested in a NS-3 environment. The environment is formulated as a Partially Observable Markov Decision Process (POMDP) and is solved with the use of reinforcement learning. The algorithm's performance is compared to the performance of a network without any machine learning. It was shown that the algorithm performs better than when no algorithm is used. Because of limitations, the algorithm has not shown its full potential, however in its current state it conserves energy which is the most important goal. The answer to the research question is with a distributed reinforcement learning algorithm that teaches sensor nodes how to control their power effectively. By setting minimum desired communication requirements the wireless sensor nodes can autonomously choose their optimal transmit power which conserves a lot of valuable energy.

9 FUTURE WORK

Although the algorithm is not perfect, it has great potential. In this future work section it is explained how to reach this potential.

9.1 Training

First and foremost the performance of the algorithm needs to be measured when it has done more iterations. Only then we can really reflect whether the algorithm works or not.

9.2 Cooperation

Right now there is no cooperation between the agents. All take in the environment and do what they think is best for themselves. However, if the agents have the same goal they can work together towards that goal and adapt their actions to what is best for the whole network and not what is best for themselves. So future work will be implementing, for instance, game theory in such a way that the agents have a mutual goal.

9.3 Sleep-scheduling

Transmission power control in combination with effective sleep-scheduling would even conserve more energy. If the sensor nodes are in idle more for a long time they can conserve more energy by turning completely off.

9.4 Real-life application

The algorithm has only been tested in a network simulator where conditions are always perfect. In real-life conditions are not always perfect so it would be interesting to measure performance in a real-life application.

REFERENCES

[1] Ali Forghani Elah Abadi, Seyyed Amir Asghari, Mohammadreza Binesh Marvasti, Golnoush Abaei, Morteza Nabavi, and Yvon Savaria. 2022. RLBEEP: Reinforcement-Learning-Based Energy Efficient Control and Routing Protocol for Wireless Sensor Networks. *IEEE Access* 10 (2022), 44123–44135. https://doi.org/10.1109/access.2022.3167058

[2] Mohammad Abu Alsheikh, Dinh Thai Hoang, Dusit Niyato, Hwee-Pink Tan, and Shaowei Lin. 2015. Markov Decision Processes With Applications in Wireless Sensor Networks: A Survey. *IEEE Communications Surveys &amp Tutorials* 17, 3 (2015), 1239–1267. https://doi.org/10.1109/comst.2015.2420686

[3] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. 2014. Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications. *IEEE Communications Surveys &amp Tutorials* 16, 4 (2014), 1996–2018. https://doi.org/10.1109/comst.2014.2320099

[4] Michele Chincoli and Antonio Liotta. 2018. Self-Learning Power Control in Wireless Sensor Networks. *Sensors* 18, 2 (Jan. 2018), 375. https://doi.org/10.3390/s18020375

[5] Sultan Mahmood Chowdhury and Ashraf Hossain. 2020. Different Energy Saving Schemes in Wireless Sensor Networks: A Survey. *Wireless Personal Communications* 114, 3 (May 2020), 2043–2062. https://doi.org/10.1007/s11277-020-07461-5

[6] Peter Corke, Tim Wark, Raja Jurdak, Wen Hu, Philip Valencia, and Darren Moore. 2010. Environmental Wireless Sensor Networks. *Proc. IEEE* 98, 11 (Nov. 2010), 1903–1917. https://doi.org/10.1109/jproc.2010.2068530

[7] Qianao Ding, Rongbo Zhu, Hao Liu, and Maode Ma. 2021. An Overview of Machine Learning-Based Energy-Efficient Routing Algorithms in Wireless Sensor Networks. *Electronics* 10, 13 (June 2021), 1539. https://doi.org/10.3390/electronics10131539

[8] 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

[9] Zhihui Guo and Hongbin Chen. 2022. A reinforcement learning-based sleep scheduling algorithm for cooperative computing in event-driven wireless sensor networks. *Ad Hoc Networks* 130 (May 2022), 102837. https://doi.org/10.1016/j.adhoc.2022.102837

[10] A. Kobbane, M.-A. Koulali, H. Tembine, M. El Koutbi, and J. Ben-othman. 2012. Dynamic power control with energy constraint for Multimedia Wireless Sensor Networks. In *2012 IEEE International Conference on Communications (ICC)*. IEEE. https://doi.org/10.1109/icc.2012.6363971

[11] C. Pandana and K.J. Ray Liu. 2005. Near-optimal reinforcement learning framework for energy-aware sensor communications. *IEEE Journal on Selected Areas in Communications* 23, 4 (April 2005), 788–797. https://doi.org/10.1109/jsac.2005.843547

[12] S. Radhika and P. Rangarajan. 2021. Fuzzy Based Sleep Scheduling Algorithm with Machine Learning Techniques to Enhance Energy Efficiency in Wireless Sensor Networks. *Wireless Personal Communications* 118, 4 (Feb. 2021), 3025–3044. https://doi.org/10.1007/s11277-021-08167-y

[13] Nimish Sanghi. 2021. *Deep Reinforcement Learning with Python*. Apress. https://doi.org/10.1007/978-1-4842-6809-4

[14] Claudio Savaglio, Pasquale Pace, Gianluca Aloi, Antonio Liotta, and Giancarlo Fortino. 2019. Lightweight Reinforcement Learning for Energy Efficient Communications in Wireless Sensor Networks. *IEEE Access* 7 (2019), 29355–29364. https://doi.org/10.1109/access.2019.2902371

[15] M.B. Srivastava and M. Potkonjak. 1995. Optimum and heuristic transformation techniques for simultaneous optimization of latency and throughput. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 3, 1 (March 1995), 2–19. https://doi.org/10.1109/92.365450

[16] Adrian Udenze and Klaus McDonald-Maier. 2008. Partially Observable Markov Decision Process for Transmitter Power Control in Wireless Sensor Networks. In *2008 Bio-inspired, Learning and Intelligent Systems for Security*. IEEE. https://doi.org/10.1109/bliss.2008.32

[17] Adrian Udenze and Klaus McDonald-Maier. 2009. Direct Reinforcement Learning for Autonomous Power Configuration and Control in Wireless Networks. In *2009 NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE. https://doi.org/10.1109/ahs.2009.50

[18] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. 2008. Wireless sensor network survey. *Computer Networks* 52, 12 (Aug. 2008), 2292–2330. https://doi.org/10.1016/j.comnet.2008.04.002