Applied Computing and Intelligence

https://www.aimspress.com/journal/aci

Applied Computing and Intelligence
4(2): 300–327.
DOI: 10.3934/aci.2024018
Received: 16 December 2024
Revised: 26 December 2024
Accepted: 26 December 2024
Published: 31 December 2024

Research article

Frequency-hopping scheduling algorithm for energy-efficient IoT, long-range, wide-area networks

Jui Mhatre, Ahyoung Lee* **and Ramazan Aygun**

Department of Computer Science, Mathematics, Kennesaw State University, Marietta, GA 30060, USA

* **Correspondence:** Email: ahyoung.lee@kennesaw.edu.

Academic Editor: Jun Shen

Abstract: A long-range, wide-area network is a cost-effective, energy-efficient technology for wide-area sensor networks. But the massive Internet of Things (IoT) brings challenges such as increased traffic and energy consumption. Thus, there is a pressing need to design a scheduling strategy to improve network energy efficiency without compensating for its reliability. We have proposed a deep deterministic policy gradient-based scheduling algorithm with a frequency-hopping spread spectrum that avoids repeated collisions and retransmissions. Frequency-hopping divides frequency channels into subchannels, allowing multiple devices to operate simultaneously. This makes it a favorable scheduling strategy for dense networks, as it reduces collisions and energy consumption. Scheduling in a long-range, wide-area network involves selecting transmission parameters for each device, which can be cumbersome. We used the deep deterministic policy gradient algorithm to optimize schedule generation for high-density networks, enhancing energy efficiency. In this paper, we compared the performance of the frequency-hopping spread spectrum with other heuristic and machine learning-based algorithms using the LoRaSim simulator. We observed a 42% increase in the packet delivery ratio and a 17% improvement in energy efficiency with our solution, along with detailed results on the transmission time and collision reduction.

Keywords: deep deterministic policy gradient; energy efficiency; frequency hopping spread spectrum; LoRaWAN; massive IoT; scheduling

1. Introduction

The discussion in the *Worldwide Internet of Things (IoT) Spending Guide* [1] shows how rapidly the IoT is spreading around the world. For applications such as smart agriculture and tracking glacier melting, where devices are battery-powered and installed remotely, frequently changing their batteries is not feasible. IoT devices require networks with very low energy consumption and long transmission distances in these cases. One of the best possible solutions is a long-range (LoRa) network. It has advantages such as being long range and having low battery consumption [2]. LoRaWAN is a low-power, wide-area network (LPWAN) technology for IoT applications. LoRa is suitable for alert messaging, notification systems, data collection, vehicle monitoring systems, and much more, which makes it the best choice for applications with low transfer rates. The LoRaWAN physical layer uses an ALOHA-based MAC protocol for the shared channel, a simple random access protocol. However, this protocol lacks multi-access handling, leading to a higher collision probability. As a result, LoRaWAN becomes unreliable when many devices transmit data simultaneously. The network uses retransmission to handle collisions, leading to increased energy consumption. This energy consumption is much higher than the energy invested in control message exchanges for scheduling strategies. Therefore, scheduling strategies should be designed to minimize the consumption of energy for the massive IoT.

Other challenges with LoRaWAN are network scalability and its duty cycle restriction. LoRa uses an ISM band with 868–870 MHz frequency and has about eight channels and a 1% of duty cycle, which means each channel could be used only for 36 seconds out of one hour [3]. Each device waits for a chance for transmission, and as the number of devices increases, the waiting period increases. Moreover, the duty cycle limits the use of channels and causes higher collisions. Collision probability rises with an increasing number of IoT devices in the network. Thus, large network devices and duty cycles increase retransmission and higher energy consumption. Interference or collision in the network occur when two or more devices choose the same transmission parameters, such as the channel and spread factor (SF), to transmit at any particular time. For example, the devices in the same circular band shown in Figure 1 choose the same SF. A proper scheduling strategy helps in selecting a unique combination of transmission parameters to avoid collision and interference. This gives rise to a pressing need to design a scheduling strategy for LoRaWAN.

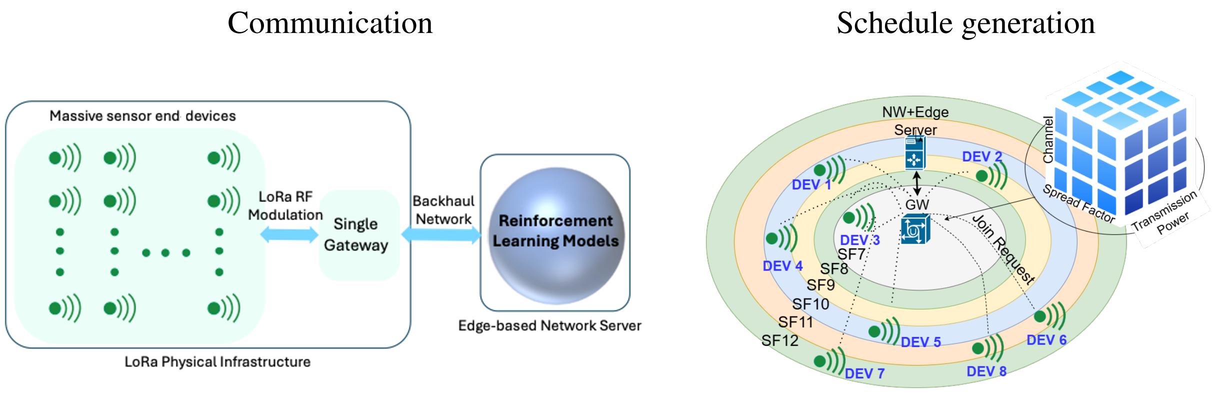


Figure 1. LoRaWAN network communication with a scheduling strategy.

The frequency-hopping spread spectrum (FHSS) modulation technique is used with LoRa in [3] to improve network capacity and robustness from that interference or collision. Long-range, frequency-hopping spread spectrum (LR-FHSS) increases the number of channels to select by dividing each channel into subcarriers, and the device can hop among the frequencies and transmit concurrently using a hopping sequence. The device pseudo-randomly determines the hopping sequence itself without sharing it with other devices. Thus, a device transmits at any time without the knowledge of other devices transmitting with them and ends up in a collision. However, another issue is the selection of the SF, which is affected by the distance of the device from the gateway. SFs help to define the number of chips required to transmit the symbol. There are a total of six SFs from SF7 to SF12. The higher the spreading factor, the lower the data rate. If the data is sent slowly, the signal is more accessible to receive and distinguish from noise. On the other hand, sending data slowly means the data spends a longer time in the air and keeps the devices highly utilized, leading to bigger power (battery) consumption. Thus, the SF selection is a critical parameter for adjusting the data rate and noise balance. It also affects the system's energy consumption. Hence, when determining a scheduling policy, it is essential to take SF allocation into consideration as it is a very crucial parameter.

Apart from the SF, transmission power is also a significant transmission parameter to trade-off with energy consumption [4]. So, transmission power is related to energy consumption. By increasing the transmission power, we increase the signal strength, which accumulates the energy consumption. In [5], the authors showed global optimal transmission power for maximizing energy consumption achieved by configuring transceivers of devices by optimizing parameters. LoRa transmission signal strength is affected by indoor use or environmental topography, such as mountains, trees, and transmitting devices. Antenna gain, beamwidth, and the frequency of operation have a significant impact on the transmission range. Adequate transmission power is affected by cable loss and antenna gain. The higher the transmission power, the better the received signal strength indication (RSSI) and signal-to-noise ratio (SNR). These two quantities help to decide whether the received signal should be discarded or accepted as actual data. We use adequate transmission power as the reference [6]. Thus, antenna selection plays a vital role in LoRa communication.

Furthermore, we have shown the need for edge computing with LoRaWAN, called 'edge-enabled LoRaWAN' for real-time monitoring and control solutions of massive IoT data overhead. Massive IoT technology has a significant setback because of the lack of any network with ubiquitous coverage and the capability to deal with vertical IoT use cases. IoT applications such as vehicular IoT, intelligent cities, smart metering, parking, street lighting, and many more come under the exact network requirements [7]. One emerging solution for such use cases is 5G technology, which is still in its early stages and needs time to evolve. Another noncellular solution, LoRaWAN, has proved itself to be efficient for IoT networks with the capability to coexist with 5G shortly to form hybrid networks. 5G networks can complement LoRaWAN as access networks or backhaul data links from the gateway to a remote cloud for IoT applications.

For deciding the scheduling strategy, not just one device but the entire network status is required. This helps to design a strategy that selects transmission parameters for a device concerning what is set for other devices. With the advantages of frequency-hopping, as discussed above, we can combine it with intelligent transmission parameter scheduling to reap more benefits. Also, the decisions regarding the channel and other parameter selections should be made by a central entity, which can decide for all devices to avoid collisions and make optimized choices. We assume that the central control system is located on an edge computing server.

We propose a scheduling strategy for edge-enabled LoRaWAN with a frequency-hopping spread spectrum (DP-FH) radio wireless technology. The FHSS technology is used to switch transmitting radio signals among several frequency channels rapidly [3]. With subcarrier-hopping, we focus on selecting the SF and TP while finding the schedule. Transmission power defines the power used by the transceiver to improve the range of communication, but higher power impacts energy consumption. Different powers give different RSSI values, which are used to decide the impact of the performance of the link, as well as the self-configuration of the power transmission received on an antenna [8]. Transmission power ranges from -4 dBm to 20 dBm in LoRa end devices. We extend this idea in our system to dynamically select the transmission power. The SF is also a factor to consider when minimizing energy consumption because the higher the SF, the more range, air time, and energy consumption. We use edge-computing servers in our network in conjunction with the LoRa gateway to compute a scheduling strategy. The IoT environment is highly dynamic, and its communication capabilities often result in sudden variations. For decision-making in complex and dynamic problem spaces with unpredictable environments, reinforcement learning approaches have been observed to provide optimal solutions through interaction with the environment. We used a reinforcement learning algorithm to find the optimal scheduling strategy. By dynamically scaling the physical layer parameters, we would like to improve the energy efficiency of the network by reducing the number of collisions. Figure 1 provides an overview of our proposed architecture, showing the different SFs assigned to devices and the edge computing server responsible for creating schedules for all of them.

In this paper, we propose a dynamic reinforcement learning-based scheduling strategy for LoRaWAN. We designed a network such that scheduling strategy generation takes place in an edge-computing server near the gateway's premises. Our scheduling strategy generates an optimized selection of occupied bandwidth (OBW), SF, and time slot for each device with transmission power, leading to minimal energy consumption. The learning agent generates the schedule using location, battery, and data buffer information provided by the IoT devices. Our goal is to reduce collisions and reduce the energy consumption of IoT devices in LoRaWAN. The contributions of this article are summarized as follows:

(1) We propose a novel approach for LoRa transmission scheduling using edge-enabled networks that reduce collisions and retransmissions and improve the energy consumption of IoT devices.
(2) We use reinforcement learning to schedule FHSS-based LoRaWAN. A deep deterministic policy gradient (DDPG) algorithm predicts the optimal transmission power, channel, and SF.
(3) We demonstrate through extensive simulations that reinforcement learning with frequency-hopping outperforms existing scheduling algorithms regarding energy conservation and data collection time.

The remaining sections of this paper are organized as follows: Section 2 provides an overview of existing technical performance studies of LoRaWAN. Subsection 3.1 presents the system model of the scheduling strategy, while Subsection 3.2 discusses the simulation network and formulates the optimization problem. In Section 4, we propose algorithms and discuss them. Section 5 presents the simulation setup, simulation results, and system analysis. Finally, Section 7 concludes the paper.

2. LoRaWAN scheduling algorithms

LoRaWAN works well with IoT devices that do not have bulk data to transmit. The lower energy consumption, such that a battery can be used for up to 10 years, is a major characteristic of LoRaWAN to promote its use for sensor networks. Despite these advantages, some challenges this network faces must be worked on. Duty cycle limitation lowers the channel usage to just 1–10%. The legacy ALOHA network with MAC protocol is used at the physical layer, which increases the chances of collision due to pseudo-random channel allocation. A proper channel allocation strategy must be designed to utilize the channel to the fullest and reduce the time to transmit the data from all devices in the network. There are various channel scheduling algorithms in the existing work. They consider minimizing transmission time or energy consumption as their cost function.

2.1. Minimizing transmission time

Previous work has been done on scheduling algorithms in the LoRa network. In [9], authors design an algorithm for achieving a shorter time for total uplink transmission from all end nodes to the gateway. Using data size, minimum SF, and payload size, it generates a list of end devices, SF, and time slots maintaining the duty cycle restrictions. The global algorithm computes the schedule for all transmissions using a single frame. The algorithm in [10] formulates the scheduling problem as the minimization of the transmission time. They associate forbidden slots in the same time frame and the same or different SF (overlapping frequency as in [11]) as well as the following 100 slots (1% duty cycle). To reduce the total time, they formulated a minimization problem to minimize the length of the most extended frame among all SFs. The minimization equation suggests finding the total time slot, which gives the time duration of the frame or time required for all data to be transmitted. An adaptive solution to determine the best LoRaWAN parameter settings, aiming to reduce channel utilization and maximize the number of packets delivered, is proposed in [12]. However, the paper does not discuss the potential trade-offs or drawbacks of the proposed adaptive solution, such as increased computational complexity or potential impact on other network performance metrics.

2.2. Reducing energy consumption

A scheduling algorithm, fine-grained scheduling for reliable and energy-efficient data collection (FREE), in [13] is meant for cases where delay requirements are not stringent and IoT devices use data buffers at their end. The algorithm aims to minimize energy consumption while following data cycle restrictions. Thus, this solution is suitable for scalable applications because it eliminates collisions and grouping acknowledgments. It also shows that using longer packet lengths is better than using short ones regarding energy usage. MAC headers impact more than more extended packet headers. It further defines two cost functions, one for energy consumption and another for collection time. It uses partial knowledge to select online schedules greedily. Our preliminary study about using a machine learning algorithm for scheduling is shown in [14] to generate scheduling in LoRaWAN and minimize energy consumption. It shows how the machine learning approach improves energy efficiency. The work in [15], focused on the performance evaluation of SF assignment schemes in LoRaWAN networks and proposed a novel smart SF assignment strategy using machine learning techniques. The proposed smart SF assignment techniques, which use a support vector machine (SVM) and a decision tree classifier (DTC), show promising simulation results in terms of energy efficiency. However, it does not consider other factors affecting network performance, such as congestion or varying channel conditions.

2.3. Frequency-hopping mode for reducing collisions

Reference [16] observed long battery life, communication range, and cost in LoRa-LPWAN compared to ALOHA-based LoRa. It also showed the distribution of the SF tiers based on the distance between the device and the gateway. This implementation and simulation were performed using

Omnet++ [17]. The LoRa network with frequency-hopping was simulated to enable connectivity between IoT devices and low earth orbit satellites in [18]. This was a case study for large-scale networks. They showed that FHSS suits a large-scale network with infrequent data collection devices, such as ones that collect information every 15 minutes. Furthermore, they demonstrated that path loss was majorly due to the loss of headers, and that the capture effect can boost performance. Another reference for studying the scalability of the LoRa system can be found in [19]. Their experiments showed dynamic communication parameter selection and that increased sinks improve scalability.

2.4. Frequency-hopping and resource allocation based on intelligent solutions

LoRa-DRL proposed in [20] is a deep reinforcement learning-based resource allocation and scheduling algorithm for dense LoRa networks. They both use the double deep Q-learning network to learn the scheduling parameters. These parameters are sent to devices using control packets. If devices do not receive these packets, the physical layer continues using the previous parameters. They tested the framework in large-scale jamming networks, where it proved quite effective. [21] designed another DDPG-based reinforcement learning method for scheduling. The learning agent makes scheduling decisions on channel and SF selection at each time step to minimize the overall energy.

Based on our literature studies, we observed research issues summarized in Table 1. We aim to design a scheduling strategy in an edge-enabled LoRa network to overcome its challenges by reducing the collision and improving the throughput and data rate while minimizing the energy consumption of devices and increasing their battery life.

Table 1. Comparative analysis of existing literature of scheduling algorithms.

Objective	**References**	**Characteristics**
Reducing latency	[9–12]	+ Reduce the latency using data size, minimum SF, and payload size. + Utilize the unused time slots and formulate the minimization equations to minimize the frame length. − Focus on selecting the SF only to minimize the transmission time. − They do not consider a selection of channels, data size, or the pseudo-orthogonality of the SF.
Reducing energy consumption	[13, 15, 22]	+ Usage of data buffers where delay requirements are not stringent. + Eliminate collisions and group acknowledgments for larger packet sizes. + Partial knowledge of greedy scheduling. − Do not consider transmission parameters based on scheduling.
Reducing collisions	[3, 18]	+ Simulate frequency-hopping for large-scale networks. − Not suitable for frequent data transmission.
Resource allocation	[20, 21]	+ Learn the scheduling parameters using reinforcement learning algorithms in a frequency-hopping mode. − Computation overhead on the LoRa gateway.

Our scheduling strategy is summarized as:

(1) Reinforcement learning process and strategy calculation occur at the edge server to lower gateway overhead.
(2) We schedule all registered devices with equal transmission opportunities using four unique parameters: channel, SF, transmission power, and time slot.
(3) Unique slot allocation in each frame to all devices also lowers the collision to reduce retransmission and increase the overall data collection rate.
(4) Our strategy dynamically allocates the SF, channel, transmission power, and time slot such that overall air time and, thus, energy consumption of the system is minimized.
(5) We focus on improving energy efficiency without degrading the network's latency and packet delivery ratio (PDR).

3. System model and problem formulation

The IoT devices set transmission parameters received from the network server and use them while transmitting of signals. These parameters determine signal strength, energy consumption, and the data rate of signal. In this paper, we design a scheduling strategy on the network server for the selection of transmission parameters and send it to IoT devices.

3.1. System model

We design our scheduling strategy on the LoRaWAN network. Figure 1 shows the LoRaWAN unit under consideration consisting of end devices communicating with a gateway. All the transmissions obtained at the gateway are collected and sent to the edge-based network server. The edge server runs a reinforcement learning algorithm to find optimal transmission parameters for IoT devices. The gateway uses backhaul networks such as cellular and WiFi networks to communicate with the network server. The network server is also responsible for MAC layer responsibilities such as collision handling and transmission parameter setting using MAC commands. We propose to use an edge-enabled LoRa network and a new reinforcement learning-based scheduling strategy.

Scheduling strategy generation for unique transmission parameter selection in each time slot for each device is shown in Figure 2. The proposed network has N IoT devices for data collection. We consider that data is generated periodically. These devices use uplink channels C to transmit the packets to the LoRa gateway using SF f at transmission time slot t with transmission power p. Packets are divided into fixed-size fragments transmitted in fixed time slot t, also known as the fragment duration. We use FHSS to increase network capacity with LoRa. In FHSS, we use network setup parameters such as the number of channels, packet and fragment size, data rate, and coding rate similar to the LR-FHSS [3]. The number of operating channel width (OCW) channels depends on the data rate. Since our setup uses DR8/9, we have eight OCW channels C, 137 kHz each. Data rates decide the number of OCW used in FHSS as shown in Table 1 in [3]. Each OCW channel is divided into 280 occupied bandwidth (OBW) channels c with a bandwidth of 488Hz, sub-channels of the selected channels. Each device can use any channel in the ISM band supported by LoRa with duty cycle restrictions d_c, which differ for each channel. Duty cycle d_c denotes the time for which a particular channel can be occupied by a device while using SF f. Figure 2 shows different SFs allocated to devices with different color bands. The six SFs range from SF7 to SF12. At any time, any device can transmit data over a single OBW channel c.

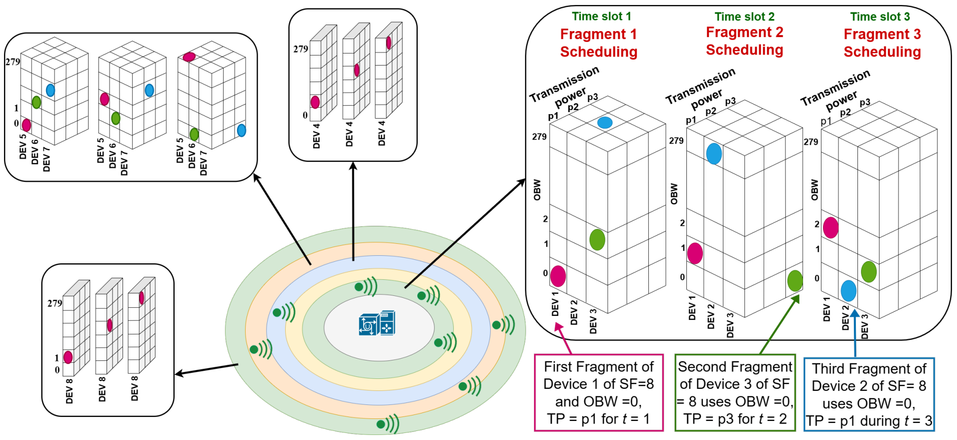

Time slot 1
Fragment 1 Scheduling
Time slot 2
Fragment 2 Scheduling
Time slot 3
Fragment 3 Scheduling
Transmission power
p1
p2
p3
OBW
279
DEV 1
DEV 2
DEV 3
DEV 4
DEV 5
DEV 6
DEV 7
DEV 8
First Fragment of Device 1 of SF=8 and OBW =0, TP = p1 for t = 1
Second Fragment of Device 3 of SF = 8 uses OBW =0, TP = p3 for t = 2
Third Fragment of Device 2 of SF= 8 uses OBW =0, TP = p1 during t = 3

Figure 2. Scheduling strategy generation.

Signals transmitted over a channel face interference if at least two of them select the same OBW channel c and the same SF f for transmission at the same time slot t. Thus scheduling a packet transmission involves the selection of a 4-tuple (c, f, t, p) that is (*channel*, *SF*, *time slot*, *transmission power*) for each packet from each device that joined to LoRa gateway, such that no two schedules can overlap in OBW and SF at a time. This is due to the orthogonality of spreading factors, channels, and time slots. Also, transmission power should be selected to ensure minimal energy consumption. Moreover, as explained in [11], some spreading factors interfere, which should be considered while scheduling. For frequency-hopping all packets for each device with payload L and MAC header H are divided into 50 ms, which are called fragments (RP2-1.0.2 LoRaWAN ®Regional Parameters) [23]. Each fragment is transmitted on a separate OBW, and then the device hops to the next OBW. Moreover, the duty cycle limitation of 3.9 kHz minimum separation, which is eight OBWs (each of 488 Hz), should be maintained. In Figure 2, we show that our DDPG agent in the edge server assigns a unique schedule to all registered devices with different SFs, time slots, and channels to all fragments so that none interfere with each other. Transmission power selection also occurs such that all parameters contribute to the minimal energy consumption of devices when all fragments from all devices are transmitted successfully.

While transmitting the packet, as per LR-FHSS, we transmit three replicas of the headers, each of 233 ms header duration T_H and waiting time T_W of 2 bits transmission time. The replicas of the headers are required to confirm that at least one of them is received by the receiver. Reference [18] gives the time-on-air T_{air} of a packet for L bytes physical layer payload at LR-FHSS as

$$T_{air} = 3 * T_H + T_W + 0.102\left\lceil \frac{L+2}{M} \right\rceil, \tag{1}$$

where $M = 2$ for DR8/DR10, and 0.102 ms is the payload duration. Fragments are several bytes transmitted in one hop period t, 50 ms [18]. Thus, the number of fragments N_F for a given packet with payload L and 2 ms guard time is

$$N_F = \left\lceil \frac{0.102\left\lceil \frac{L+2}{M} \right\rceil}{t} \right\rceil. \tag{2}$$

Energy is consumed for each packet that is transmitted. We calculate the energy consumed to transmit all data packets D using the equation defined in [13] given by

$$E = (1 + R) * D * T_{air} * I * V, \tag{3}$$

where R is the number of times packets are retransmitted, and I and V are the average current and voltage in the transceiver chip during transmission, respectively [13]. We designed our system to make it collision-free and have a retransmission factor $R = 0$. Unregistered IoT devices can transmit and interfere with registered devices, which causes a retransmission.

We observe that scheduling sequence generation constrained by minimal energy consumption should consider factors such as OBW channel allocation c, SF f, transmission power p, and time slot allocation t. We aim to generate a scheduling sequence for IoT devices in the LoRa network by assigning a 4-tuple (c, f, t, p) to each registered IoT device. The unique 4-tuple confirms that the schedule is collision-free for devices that have sent join requests to the gateway and registered themselves, and reduces energy consumption. Collision reduction leads to better data transmission rates and lower energy consumption due to fewer retransmissions.

Before the communication starts, every device must register itself to obtain its transmission parameters from our algorithm using a join request sent to the LoRa gateway. The device sends information such as location, data buffer, and the join request. All registered device information is combined and sent to the edge server to generate the schedule. Our learning model generates the schedule and sends it back to the LoRa gateway that communicates to all devices. Devices follow this schedule and transmit their data packets. The LoRa gateway then collects the energy dissipation information from all devices and sends it to the edge server as its reward. The learning DDPG model learns a better scheduling strategy for the next time based on the received reward. This process continues, and eventually, the devices get an optimized scheduling strategy. Other than interference and collision, packet loss is another factor for retransmission. It is calculated using the received signal strength indicator (RSSI) and sensitivity. RSSI [24] measures the power in the radio signal, which is an approximate value for the signal strength received,

$$RSSI = p - PL(d0) + 10 \times \gamma + \log(d/d0) + \mu, \tag{4}$$

where p is the transmission power, $PL(d0)$ is the path loss and can vary for different environments for a given distance d concerning the reference distance $d0$, γ is the path loss exponent [25], and μ is unknown-but-bounded (UBB) noise with zero mean.

We decide sensitivity based on the SF discussed in [13]. If RSSI is less than the sensitivity, the packet is considered lost and required to retransmit. However, the RSSI equation does not consider the antenna parameters and the noise level. Therefore, besides RSSI, the noise level should be considered to obtain the SNR. Given the antenna specifications, the friis formula provides another metric for the received signal power. This metric can be used with the RSSI metric by considering the noise level using SNR.

From Figure 3, the received power at a distance R from the transmitter is given to be

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi\psi)^2}, \tag{5}$$

where G is the gain of the antenna, λ is the wavelength, and ψ is the distance between transmitting antennas. However, this equation only states the gain of the antennas, which is a function of the antenna efficiency and directivity. For antenna efficiencies, we use

$$G_t = \eta_t D_t, \tag{6}$$

$$G_r = \eta_r D_r, \tag{7}$$

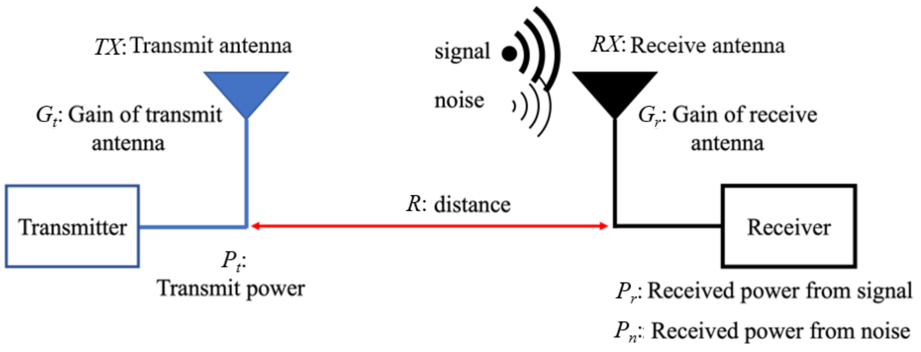
where D is the directivity and η is the efficiency of the antenna. The receiver antenna picks up both signal and noise at temperature (Ω), which can be expressed in terms of the bandwidth (BW),

$$P_n = K\Omega_{sys}BW, \tag{8}$$

where P_n is the receiver noise power, K is Boltzmann's constant 1.38 x 10^-23 (J/k), Ω_{sys} is the system noise temperature (Kelvin), and BW is the receiver bandwidth (Hz). Now that we have the receiver signal power and the receiver noise power, the SNR can be defined as

$$SNR = \frac{P_r}{P_n} = \frac{P_t G_t G_r \lambda^2}{(K\Omega_{sys}BW)(4\pi R)^2}, \tag{9}$$

$$SNR(dB) = 10\log P_t G_t G_r \lambda^2 - 10\log(K\Omega_{sys}BW) - 20\log(4\pi R)^2. \tag{10}$$



TX: Transmit antenna
G_t: Gain of transmit antenna
Transmitter
P_t: Transmit power
signal
noise
R: distance
RX: Receive antenna
G_r: Gain of receive antenna
Receiver
P_r: Received power from signal
P_n: Received power from noise

Figure 3. Friis formula for received signal power.

With a higher SNR, RSSI will also increase due to the relationship: $SNR = RSSI - RF$ background noise. Background RF noise is the collective RF signal strength in the particular frequency we are measuring. RF noise can come from non-802.11 devices, or it can come from other transmitters using the same frequency in [26].

An array antenna can be used to increase the RSSI, which improves the SNR and increases the antenna's gain. The array antenna has the advantages of beam steering and beamforming to improve the gain to IoT devices that are farther away in distance or have a low antenna gain or transmit power. By controlling the amplitude and phase of each feed of the elements in an array antenna, the direction of the beam can be steered and formed to maximize the transmission power between IoT devices and the gateway. This will increase the RSSI of the signal the gateway receives, thus improving the SNR, resulting in lower dropped or discarded packets. This eliminates the need for retransmission, which saves energy and improves the energy efficiency of the LoRa network. Machine learning algorithms can be used to optimize the beam steering and beamforming to optimize energy efficiency. This can be done by controlling the amplitude and phase of each element in the antenna array such that the beams are formed so that the RSSI is above the threshold for the minimum SNR needed to transmit a data packet successfully. By optimizing the feeds for the array antenna using machine learning, the minimum amount of energy can be used to maximize packet transmission and minimize retransmission. However, this array antenna is out of scope in this paper, but it will be our near future work for supporting a next-generation LoRaWAN.

3.2. Problem formulation

Our goal is to generate a scheduling sequence for each device that is ready to transmit so that the average energy consumption of each device is minimal. Each schedule is represented as a 3-dimensional matrix having eight channels, six SFs, and transmission power as three dimensions for each time slot. Once we select a channel, OBW in the corresponding channel is assigned sequentially. We aim to find optimal values for three quantities: *channel*, *SF*, and *transmission power* for each device, and generate a schedule of all devices such that no device with the same parameters transmits in the same time slot. Also, every device gets a chance in an order such that the overall energy consumption is the lowest with the selected corresponding parameters. Thus, we can formulate the following problem:

$$\min \sum_{n=0}^{N} \sum_{b=0}^{B_n} r_b * T_{air} * I * V, \tag{11}$$

s.t.

$$\sum_{n=0}^{N} \sum_{t=0}^{\tau} M[t][n] = \sum_{n=0}^{N} B_n, \tag{12}$$

$$\frac{\sum_{t=0}^{\tau} M[t][n] * T_{air}}{\tau} \leq d \quad \forall n \in N, \tag{13}$$

where B_n denotes the buffer size in the number of packets for device n and r_b is the retransmission factor for packet b. We assume that the buffer accommodates all the data to be transmitted; hence, the buffer size is no less than the data to be transmitted. Equation (11) is an energy minimization equation for all N devices, and I and V are the average current and voltage in the transceiver chip during transmission, respectively. Equation (12) confirms that all B_n packets of each n device are transmitted in total τ time slots, and each time slot t transmits $M[t][n]$ packets from device n. The most important duty cycle restriction is satisfied by LoRa given by (13) where d denotes the duty cycle percentage allowed (typically 1%).

We formulate the scheduling problem for energy optimization as a Markov decision problem (MDP). It is a 4-tuple problem defining the parameters: (state (S), action (A), reward (R), transition matrix (X)) to solve MDP.

(1) *State:* For each frame consisting of time slots, the system observes a state that forms as input to the system. The state consists of

$$S = \{B, Loc\}, \tag{14}$$

where $B = B_1, B_2, ..., B_N$ denotes the data in the device's buffer, and Loc is the location of the devices.

(2) *Action:* Action consist of actions taken to change the state.

$$A = A_1, A_2, ..., A_N, \tag{15}$$

Action in our system, denotes the scheduling decision for each device n and can be represented as $A_n = \{c, f, p\}$ where c is the channel, f is the SF and p is the transmission power. These parameters are selected for given sets of time slots t. These values denote the transmission parameter settings that combine to set the quality of signal transmitted by each device n when it uses the set transmission parameters.

(3) *Reward:* Reward helps in learning the DDPG model to decide the action and update the learning policy according to the reward value. Our reward is based on energy consumption E given as

$$R = -E. \tag{16}$$

The negative sign shows that as energy reduces, reward increases, and vice versa. Our DDPG model will learn that reward should increase, implying that energy consumption decreases.

(4) *Transition Matrix:* It defines the policy to find the next state $s_{(i+1)}$ given the current state s_i and action A taken. It is a function that provides a mapping from among the states for the action chosen.

4. Proposed algorithms

We propose two algorithms. The first is LR-DP (Algorithm 1), which is a reinforcement learning-based algorithm to generate a schedule finding OBW channel c, SF f, and transmission power p. This algorithm does not do time scheduling, and hence leads to collisions. It predicts the unique 3-tuple (c, f, p) for minimum energy consumption with collision. It uses the DDPG algorithm to predict the 3-tuple by calculating rewards to minimize energy consumed by all devices. A detailed explanation of how the DDPG model in LR-DP optimizes (Eq (11)) is given in Section 4.2.

Algorithm 1 LR-DP.

```
1:  Randomly initialize the critic network Q(s,a|$\theta_Q$) and actor network $\mu$(s|$\theta_\mu$) using weights $\theta_Q$ and $\theta_\mu$.
2:  Initialize replay buffer R_buff
3:  for $episode = 1..M$ do
4:      Initialize random noise $N_0$ for action exploration
5:      Receive initial observation state $S$
6:      $A = \mu(S|\theta_\mu) + N_0$
7:      Calculate reward $R$ by executing $A$ on state $S$
8:      Store transition in buffer R_buff.add($S_t, A, R, S_{t+1}$)
9:      if $learning == True$ then
10:         Sample minibatch of $K$ transitions from R_buff
11:         Set $y' = r(S_t, A_t) + \gamma * Q'(s'_{t+1}, \mu'(s'_{t+1}))$
12:         Update critic by minimizing loss $L_Q$
13:         Update actor policy using policy gradient $\nabla_{\theta_\mu} L_{\theta_\mu}$
14:         Update target networks $\theta_{\mu'}$ and $\theta_{Q'}$
15:     end if
16: end for
```

In addition to intelligent learning, our second algorithm, DP-FH (Algorithm 2), generates a time-based schedule as well by predicting the unique 4-tuple (c, f, t, p). Thus, we make a collision-free scheduling policy with minimal energy consumption. The frequency-hopping approach becomes challenging since it needs synchronization. Since we have a single server generating schedules for all the nodes, the allocation is such that all the settings are unique. Moreover, synchronization is achieved by time-based scheduling with OBW scheduling.

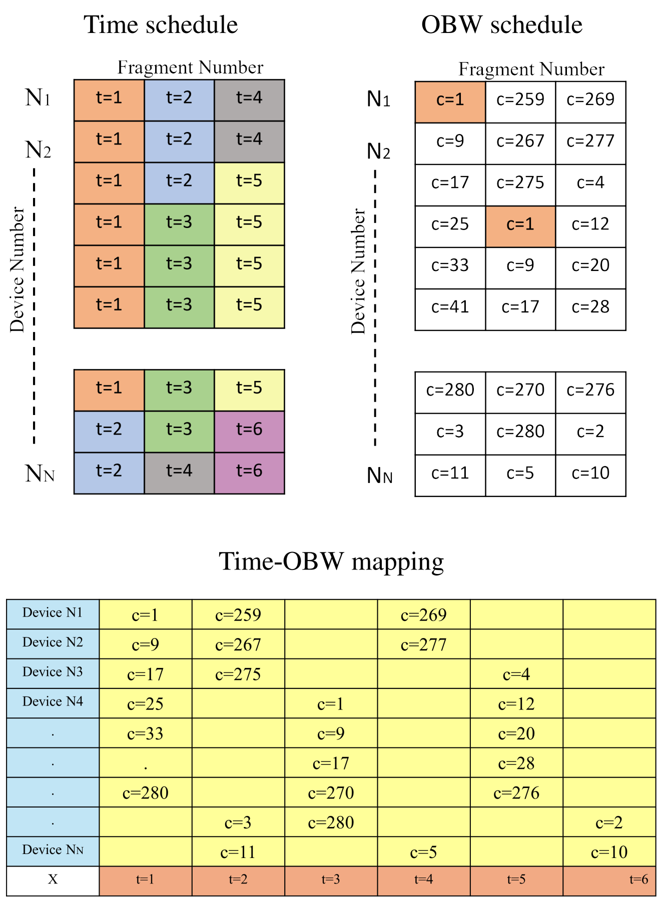
Algorithm 2 DP-FH.

```
1: Initialize the MaxEpisodes
2: for episodes in MaxEpisodes do
3:     Initialize the network
4:     [($C$, $f$, $p$)] x N = LR-DP (State) (Algorithm 1)
5:     device_to_sf_txpow = group devices with the same SFs and transmission powers
6:     reward_ep = 0
7:     for device_list   in device_to_sf_txpow do
8:         time_schedule, obw_schedule = generateSchedule()
9:         Simulate transmission
10:        reward_f = calculateReward()
11:        reward_ep += reward_f
12:    end for
13:    Learn DDPG network based on $reward\_ep$
14: end for
15: calculateReward():
16: Check overlap in the generated schedule
17: Test collision in simulation
18: Test path loss using RSSI and sensitivity
19: if collision then
20:     reward = MIN_INT
21:     return reward
22: end if
23: Calculate energy $E$
24: reward = -1 * $E$
25: return reward
26: generateSchedule(): # adds time parameter to scheduling apart from ($c$, $f$, $p$)
27: time = 0
28: for fragment in device_list do
29:     for device in device_list do
30:         if obw == 280 then
31:             obw  = 0
32:             time  += 1
33:         end if
34:         obw +=1
35:         obw_schedule[device][fragment] = obw
36:         time_schedule[device][fragment] = time
37:     end for
38: end for
```

Collision testing in LR-DP and DP-FH involves checking interference at multiple levels. Frequency collision involves comparing whether the channels overlap in their frequencies. The next level is to verify that the SFs overlap. If multiple packets at the gateway are received that overlap in all the above checks, then the collision is announced, and the packet is made to retransmit. Overall time and energy of retransmission increase compared to the non-collision scenario, and hence the reward decreases. Our DDPG model learns this information and produces (c, f, p) for the next time slot t to increase the reward and lower energy consumption. Unlike LR-DP, DP-FH selects OBW and checks whether there is an overlap of minimum separation of eight OBW channels with the SF at any given time slot. We evaluate our algorithms using LoRaSim, which has collision detection explained in [19].

4.1. Unique schedule generation

DP-FH has a method called *generateSchedule*(), which takes nodes having the same SF and channel (collision as per LR-DP) and assigns each fragment of each node a unique time and OBW combination such that none of them collide. Figure 4 gives examples of time and OBW schedules, each having length N_F as described in (2).



Device Number	Fragment Number		
N1	t=1	t=2	t=4
N2	t=1	t=2	t=4
	t=1	t=2	t=5
	t=1	t=3	t=5
	t=1	t=3	t=5
	t=1	t=3	t=5
	t=1	t=3	t=5
	t=2	t=3	t=6
NN	t=2	t=4	t=6

Device Number	Fragment Number		
N1	c=1	c=259	c=269
N2	c=9	c=267	c=277
	c=17	c=275	c=4
	c=25	c=1	c=12
	c=33	c=9	c=20
	c=41	c=17	c=28
	c=280	c=270	c=276
	c=3	c=280	c=2
NN	c=11	c=5	c=10

Device N1	c=1	c=259		c=269		
Device N2	c=9	c=267		c=277		
Device N3	c=17	c=275			c=4	
Device N4	c=25		c=1		c=12	
·	c=33		c=9		c=20	
·	.		c=17		c=28	
·	c=280		c=270		c=276	
·		c=3	c=280			c=2
Device NN		c=11		c=5		c=10
X	t=1	t=2	t=3	t=4	t=5	t=6

Figure 4. Occupied bandwidth (OBW)-time mapping using schedule generation.

In the time schedule, each cell corresponds to the transmission time slot and in the OBW schedule, each cell corresponds to the OBW assigned to each fragment (column) of a given device (row). In the time schedule, each cell value corresponding to the n^{th} device and k^{th} fragment number denotes the time slot when the fragment k of device n is transmitted. Similarly, in the OBW schedule, each cell value corresponding to the n^{th} device and k^{th} fragment number denotes the subchannel number of frequency-hopping to be used while transmitting the fragment k of device n during time slot $time\_schedule[n][k]$. For $c = 1$, we observe that each $c = 1$ in the OBW schedule is assigned for different time slots per the time schedule. This confirms that the schedule of devices with the same SF is collision-free and does not use the same time slot and OBW during frequency-hopping. We have combined both schedules as a time-OBW mapping for a better understanding. Every cell corresponds to device n and time slot t will be using a subchannel corresponding to the cell value schedule $[n][t]$.

4.2. Deep deterministic policy gradient

We use the DDPG-based reinforcement learning algorithm for the minimization of energy consumption by scheduling transmission parameters. As shown in Eq (11), we aim to reduce the overall energy consumption so that all the data in the device buffer is transmitted and the duty cycle restrictions are followed. The minimization equation involves reducing the time on air and reducing the number of retransmissions due to collisions. This can be achieved by selecting optimal transmission parameters that avoid collisions and reduce energy consumption. There are 6720 transmission parameter settings [19]. Selecting unique settings for each device makes the selection space very large. Optimal transmission parameter selection is a computationally complex problem and can be categorized as an NP-hard problem. The reinforcement learning algorithm is a suitable solution for optimization where the agent is required to perceive and interpret its environment and take action accordingly. Our problem does not produce any data to analyze. Instead, it provides an environment to take action and optimize actions based on rewards and punishments for every action. Hence, reinforcement learning is our first choice for solving our problem. When deciding on the optimal scheduling strategy, the parameters selected are channels (among subchannels of eight channels), six SFs ranging from SF7 to SF12, and transmission power. All these parameters are discrete except transmission power, which is continuous. The deep Q-network algorithm is suitable for discrete action space since it predicts Q-values for each state-action pair, which are discrete, and learns policy from Q-values. However, we expect deterministic decisions instead of distribution over actions generated using the deterministic policy gradient algorithm.

The DDPG algorithm is an off-policy algorithm combining the deterministic policy gradient and the deep Q-network. Hence, we use the DDPG algorithm applied to Algorithm 2. Policy gradient algorithms such as DDPG [27] require a policy function μ(s) which finds the best action to be taken to reach the next state from state s. At each time slot t, an actor network evaluates the current state s_t using μ to choose action a_t, and the critic network uses the value function to calculate the Q-value $Q_\theta(s_t, a_t)$ of the selected action. In addition to an actor network (μ with parameters θ_μ) and a critic network (Q with parameters θ_Q), DDPG has a target actor (μ' with parameters θ'_μ) and a target critic network (Q' with parameters θ'_Q) to manage stability while training. The critic network computes the loss as the mean-squared error between the temporal-difference (TD)-target and the Q-value estimate of the current state and corresponding action. The TD-target y is generated in target networks. The next state s_{t+1} and the associated action predicted by the target actor μ' are provided as input to the target critic Q' [28] formulated as

$$y = r(s_t, a_t) + \Gamma * Q'(s_{t+1}, \mu'(s_{t+1})), \tag{17}$$

where $r(s, a)$ is the reward obtained when the actor network chooses action a in state s, Γ is the discount factor, and Q'$(s_{t+1}, \mu'(s_{t+1}))$ is the maximum future return of the next state. DDPG uses a buffer to sample a minibatch of size m for calculating the loss function of the critic and then learns by minimizing the loss function given by

$$L_Q = \frac{1}{m}\sum_{i=1}^{m}(y_i - Q(s_t, \mu(s_t)))^2, \tag{18}$$

where Q$(s_t, \mu(s_t))$ is the Q-value estimate of the current state and corresponding action at time t. The actor network maximizes the Q-value. For a continuous action space, we cannot find the max Q-value for a set of actions. For continuous values, maximizing a function is the same as minimizing the negative value of that function. Hence, the loss function of the actor network is

$$L_\mu = -Q(s_t, \mu(s_t)). \tag{19}$$

Here, the loss function is in terms of Q and not θ; hence, we apply the chain rule to optimize it:

$$\nabla_\theta L_\mu = \frac{1}{m}\sum_{i=1}^{m}(\nabla_a Q(s_t, \mu(s))\nabla_\theta \mu(s)). \tag{20}$$

Parameters of the actor and critic network are updated using these optimized loss functions. Finally, the DDPG agent does a soft update to the parameters of the critic target network and actor target network using a small constant δ such that

$$\begin{aligned} \theta_{\mu'} &= \theta_\mu + (1-\delta)\theta_{\mu'}, \\ \theta_{Q'} &= \theta_Q + (1-\delta)\theta_{Q'}. \end{aligned} \tag{21}$$

The next step s_{t+1} is calculated for the remaining computation of the task on obtaining action a_t.

4.3. Deep deterministic reinforcement learning algorithm

In Algorithm 2, we use the frequency-hopping technique by increasing the number of usable channels. The channel, SF, and transmission power selection are made similarly as in Algorithm 1 using LR-DP. The difference lies in the selection of the central frequency. In LR-DP, we select central frequencies among OCW, and in DP-FH, we select from OBWs, thus enabling the frequency-hopping technique. We first group devices with the same SF and transmission powers allocated by LR-DP, which are then passed to $generateSchedule()$. We then assign each node fragment a unique time and OBW combination so that none of them collide. Thus, the algorithm adds the time parameter to the schedule generated to reduce collision and retransmissions and make it more energy efficient.

The transmission parameter selection and setting process is as follows: IoT devices must join the network to participate in transmission. The LoRa gateway collects the state of each device and sends it to the edge server to find the optimal transmission parameter setting using our proposed algorithm. The entire schedule generated for all time slots is sent to the devices. MAC commands request information from end devices and set each device's parameters. MAC commands interact between the device and network server in [29]. The network server obtains the setting from the edge server and sends separate MAC commands to each end device. Devices use these settings for future transmissions, which again consume some energy. Future model training is done based on the amount of consumed energy using the newly set parameters.

5. Experimental setup

We implement simulations LR-DP and DP-FH using LoRaSim in [19], which uses a Simpy library from Python. For machine learning, we use Python's Tensorflow libraries. We evaluate and compare the results with FREE algorithm results in [13] for various performance metrics such as energy consumption, packet delivery ratio, transmission time, and collision probabilities.

For evaluation, we use the parameters for the simulation discussed in Table 2. Our learning agent gets input device information and learns the policy to generate the optimal channel, SF, and transmission power. We have extended this learning with the frequency-hopping strategy in LoRaSim to include OBW and time slot selection to avoid a collision. After every learning episode, we generate rewards as a negative energy consumption value. The learning agent learns that optimal parameters generate maximum reward, meaning learning takes place to minimize energy. Thus, our algorithm optimizes energy.

Table 2. Simulation parameters.

Parameter	Value
Number of IoT devices (N)	100–4000
OCW channels (EU863-870) (C)	867 MHz, 867.3 MHz, 867.5 MHz, 867.7 MHz, 867.9 MHz, 868.1 MHz, 868.3 MHz, 868.5 MHz
OCW bandwidth (OC_{BW})	137 KHz
OBW bandwidth (OB_{BW})	488 Hz
OBW minimum separation	3.9 kHz
Coding rate (CR)	1/3
Data rate (DR)	DR8
Spread factor (f)	[7, 8, 9, 10, 11, 12]
Transmission power (p)	-4 to 20 dBm
Payload fragment duration	50 ms
Payload duration	102 ms
Uplink channel duty cycle (d)	1%
Battery capacity	1000 mAh
Maximum data packet size	50 bytes
Mean path loss at the reference distance d0 ($PL(d0)$)	127.41 dB

6. Results

We gathered results to compare our implementation with existing algorithms. In the first stage of our implementation, we used only reinforcement learning to optimize the selection of transmission parameters (the algorithm mentioned in LR-DP [22]) such as central frequency, bandwidth, SF, and transmission power. We combined reinforcement learning and frequency-hopping in the second stage of our proposed approach (Algorithm 2). We trained the model for around 3000 iterations to obtain the results shown in this section. Figure 5 shows the training process of the DP-FH RL algorithm. This training was for 4000 devices and data for 10 bytes. The reward is an energy consumption function denoted in Eq (16). Training the model increases the reward, which reduces the energy consumption of the devices. We made multiple training runs to find the maximum number of episodes required for training. They range from approximately 1800 to 3000. In the learning process, we observed a few dips at around 1300 due to exploration used in the DDPG algorithm. The performance evaluation based on collisions, transmission time, energy consumption, and packet delivery ratio is shown in Figures 6–10.

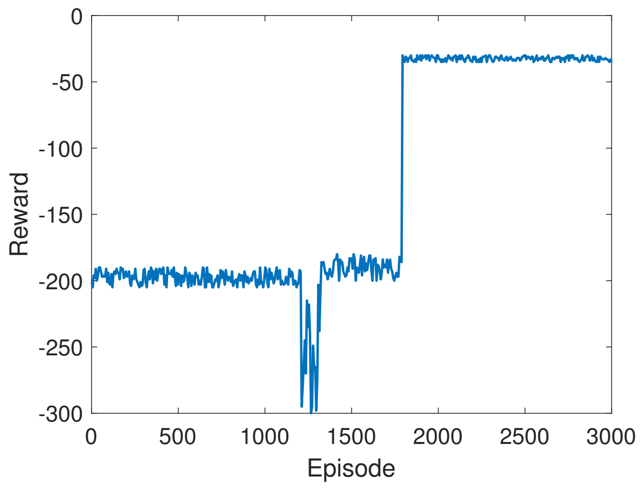


Figure 5. Training of the DP-FH model for 3000 episodes.

We compare our results with various algorithms as shown in Table 3: the heuristic approach, FREE in [13], LoRa-based frequency-hopping, LR-FHSS in [3], SF assignment using a support vector machine (SF_Smart_SVM), SF assignment using a decision tree classifier (SF_Smart_DTC) [15], mixed integer linear programming (MILP) called an optimization model for LoRaWAN resource allocation for IoT applications (MARCO), and a heuristic for adaptive resource allocation on LoRaWAN for IoT applications (CORRECT) [12].

Table 3. Algorithms for comparison with the proposed algorithm DP-FH.

Reference	Name	Intelligent/Heuristic
Proposed	DP-FH	Intelligent
[13]	FREE	Heuristic
[22]	LR-DP	Intelligent
[15]	SF_Smart_SVM	Intelligent
[15]	SF_Smart_DTC	Intelligent
[12]	MARCO	Intelligent
[12]	CORRECT	Intelligent
[3]	LR-FHSS	Heuristic

Our evaluation is in four phases:

(1) We evaluate the results for various parameters by increasing the number of devices.
(2) We evaluate the results for various parameters by increasing the data size to be transmitted.
(3) We evaluate the effect of the ML algorithm with the frequency-hopping spread spectrum.
(4) We make signal strength-specific evaluations, including RSSI and SNR.

Each phase is discussed below with a detailed analysis of the results.

6.1. *Increasing number of devices*

In this step of evaluation, we record the total time to transmit the data (ms), the number of collisions in transmission, measurements of energy consumed (J), and the packet delivery ratio (PDR). Figure 6 is a comparison of total collisions and total transmission time for an increasing number of devices from 100 to 4000 for 20 bytes of data. The transmission time involves all the packets that are transmitted even after retransmission. But if the device decides to drop the transmission of packet fragments due to exceeding the number of allowed retransmission attempts, the packet will not be received by the gateway. This recorded transmission time is the average of the total time each device takes to transmit all possible packets. It should be noted that the collisions are counted each time the packet is interfered irrespective of whether it was a first-time transmission or retransmitted packet. Thus, as the number of collisions increases, the number of retransmissions and total time to transmit, taken by each device, also increases. It can be observed that both go hand in hand. Also, DP-FH has the lowest collision probability with the lowest time to transmit. With more devices and elevated network traffic, the number of collisions increases. However, our proposed algorithm still maintains the trend of lowest collisions and lowest time to transmit. FREE is a heuristic algorithm and is observed to have a higher collision ratio than all other existing algorithms. Machine learning-based algorithms, such as MARCO [12], LR-DP [22], and the adaptive algorithm CORRECT [12] perform better than FREE. Transmission time for DP-FH is about 58% lower than other machine learning and adaptive algorithms. The collision count in DP-FH is about 9% lower than the existing state-of-the-art MARCO algorithm.

Figure 7 shows a comparison for the energy consumed by the devices to send all their data and the PDR achieved in this transmission. It runs from 100 to 4000 for 10 bytes of data with a maximum gateway range of 5000m. We compare the energy consumed and PDR using our proposed algorithm DP-FH with FREE, LR-DP, SF_Smart_SVM, and SF_Smart_DTC. It shows evaluation results for the network by increasing the device density by increasing its count from 100 to 4000. The distance between the gateway and the device is a minimum of 5000m. Figure 7 shows the energy consumed by the algorithms under consideration. Energy consumption is the average energy a device consumes to transmit a packet [19]. We observe that the machine learning algorithms perform better than the heuristic FREE algorithm regarding energy consumption. DP-FH consumes about a minimum of 17% lower energy compared to the existing algorithms. Another aspect is the packet delivery ratio, which is affected if energy consumption is reduced. But here we see that the DP-FH still has comparable ratios as FREE, LR-DP, SF_Smart_SVM, and SF_Smart_DTC. Using our proposed approach, the PDR is improved by around 42%. The improvements in energy consumption, packet delivery ratio, and total time to transmit are due to the handling of collisions. The frequency-hopping technique LoRa uses gives a spectrum of frequencies for various devices to transmit. Since this avoids overlap in frequencies, the collisions are reduced. Moreover, the massive IoT, which may cause collisions even in frequency-hopping, is further reduced due to the appropriate scheduling of transmission parameters. This helps mitigate collisions and retransmission, thus reducing energy consumption, latency, and the packet delivery ratio.

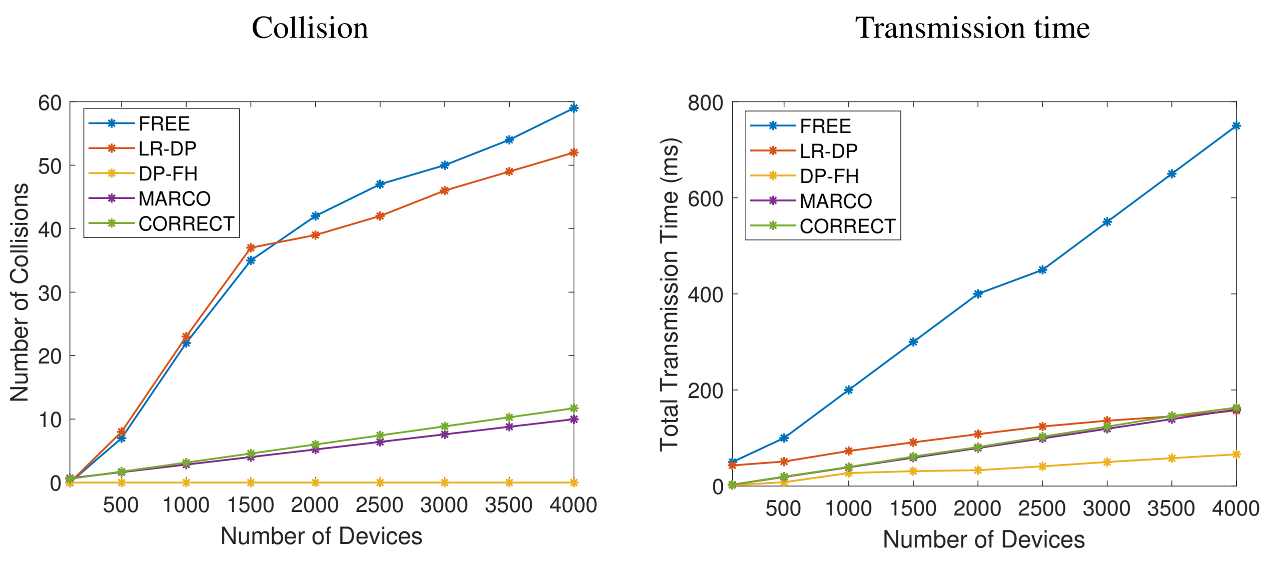


Figure 6. Comparison of total collisions and total transmission time for increasing the devices.

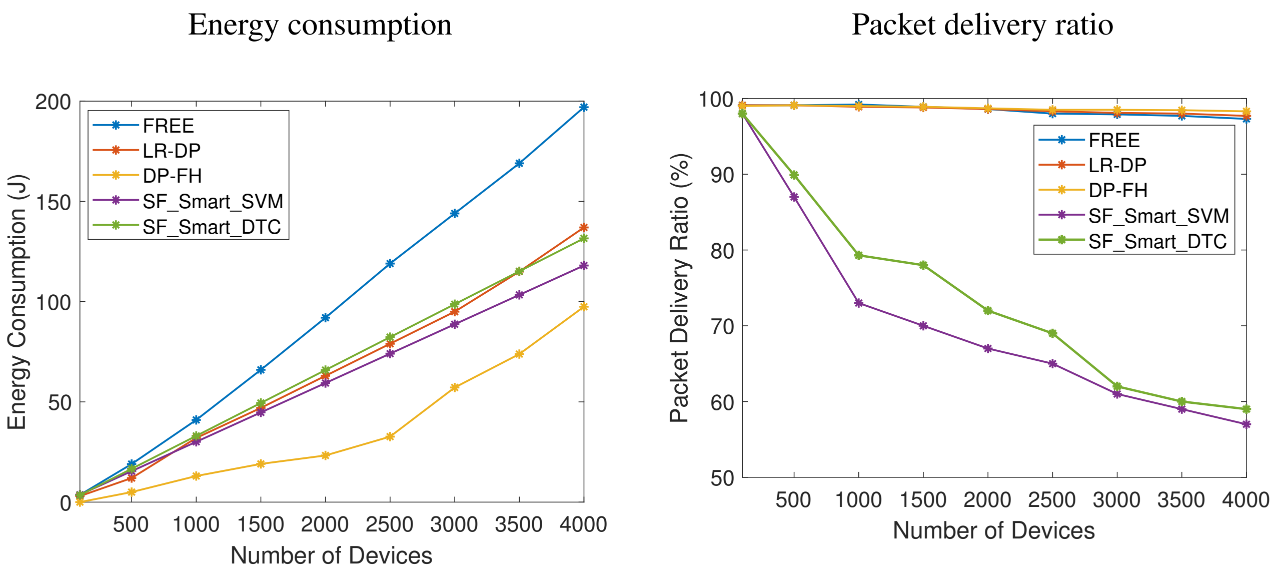


Figure 7. Energy consumption and packet delivery ratio for increasing the devices.

The transmission parameters, specially SF, need to be allocated and tweaked to manage energy consumption. Figure 8 shows SF distributions: the existing heuristic FREE approach and the proposed DP-FH. When using DP-FH for scheduling, we minimize energy consumption that is closely related to spreading factor selection. Since we focus on minimizing energy consumption, we observe that the DDPG-based scheduling strategy learns the impact of the SF on energy consumption and allocates minimal SFs. If we increase the SF, transmission time increases, and more energy is consumed. Hence, more devices use a lower SF. However, if devices are far away, they must be assigned higher SFs to ensure that transmission is successful because the signals with lower SFs do not travel longer distances. Hence, we see that after higher usage of SF7, more devices use SF SF11, and less use SF12. The SFs SF8, SF9, and SF10 are observed to be allocated minimally, and instead, transmission powers or data rates are adjusted to avoid collisions and increase the range of devices with SF7. This is because the algorithm mostly assigns SF7 to reduce energy; it plays on other parameters, such as transmission power, to try and tweak them to achieve acceptable RSSI and keep the SF low to SF7. However, at a certain distance between the gateway and the device, signals with SF7 cannot reach the gateway. Here, SF11 is allocated and we try to avoid SF12 for the same reason as SF8 to SF10. The FREE algorithm has designed an SF allocation algorithm to minimize the cost function (energy minimization for $\alpha = 0$) in [13] such that RSSI is higher than the device sensitivity for validation packet loss. However, it is observed that while minimizing energy consumption, FREE fails to consider transmission time in Figure 7. We increased the number of devices from 100 to 2000 and transmitted 50 bytes of data. In DP-FH (titled as XXX-FHSS in Figure 8, where XXX is the number of devices), as the number of devices increases, the percentage of devices allocated to SF7 (avg. 75%) reduces and that to higher SFs increase. This shows that as the number of devices increases, the allocation of unique transmission parameters forces the scheduling strategy to allocate devices to SF7 up to a threshold (Table 4). Beyond that, all remaining devices are allocated higher SFs. However, the FREE algorithm heuristically allocates transmission parameters to the device and does not have enough unique transmission parameters to allocate. Thus, it ends up assigning higher SFs to about 40% of the devices.

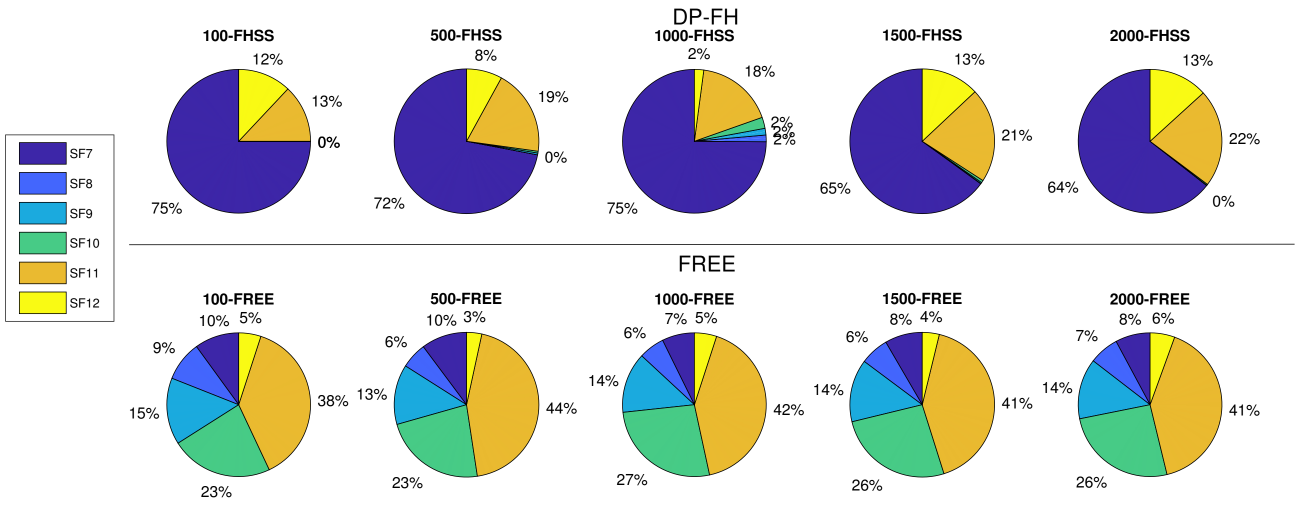


Figure 8. SF distributions for an increasing number of devices.

Table 4. Interference thresholds [13].

	SF7	SF8	SF9	SF10	SF11	SF12
SF7	1	-8	-9	-9	-9	-9
SF8	-11	1	-11	-12	-13	-13
SF9	-15	-13	1	-13	-14	-15
SF10	-19	-18	-17	1	-17	-18
SF11	-22	-22	-21	-20	1	-20
SF12	-25	-25	-25	-24	-23	1

6.2. Increasing data sizes

In this section, we evaluate how our proposed algorithm performs when the amount of data to be transmitted increases. When the amount of data to transmit increases, the number of fragments to transmit also increases. DP-FH utilizes the subchannels due to FHSS and transmits data from devices more frequently than the other two algorithms, which do not use FHSS. This reduces the overall waiting time for all devices and improves transmission latency for our approach. Figure 9 shows transmission time and energy consumption. It uses from 10 to 50 bytes using 1000 devices transmitting this data to the gateway. Our proposed algorithm takes about 14% less time to transmit the same data as others. Figure 9 compares energy consumption (J) for three scheduling strategies showing the lowest energy consumption in DP-FH compared to FREE and LR-DP. For smaller data sizes, energy consumption for LR-DP is higher than FREE but lower for larger data packets. For energy consumption evaluation in Figure 9, we compare our proposed approach with FREE; we observe that for increasing data sizes, energy consumption is more in FREE as well as LR-DP but frequency-hopping reduces the energy consumption by around 98%.

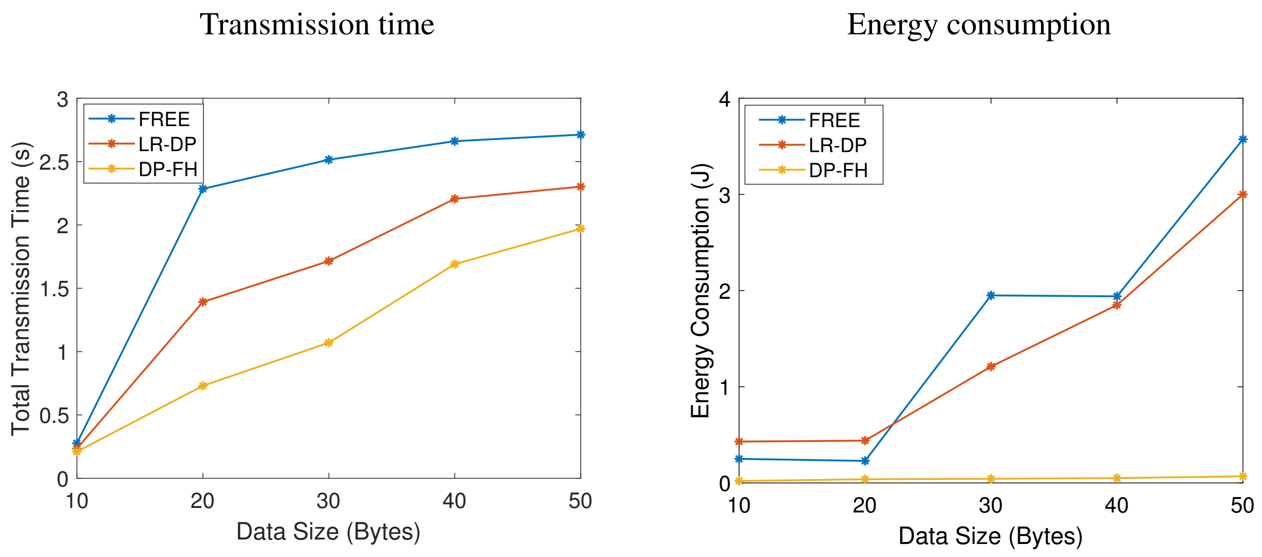


Figure 9. Comparison of time on air and energy consumption for increasing data size.

We compare SF distribution for increasing data sizes with 500 devices and data sizes from 10 to 50 bytes for DP-FH and FREE in Figure 10. This is a crucial result since we need to observe how the

SF distribution takes place to collect more data from data-intensive applications such as reporting systems. This can help us extend LoRa applications to a larger spectrum of use cases rather than just notification purposes or small data applications. As we know, more data can be sent over the channels with transmission parameters having a smaller SF. A smaller SF selection is better for a network with good reception at the receiver gateway and a higher SNR ratio since it lowers energy consumption. DP-FH allocates most of the devices to lower SFs (avg. 71.4%) since we intend to minimize energy consumption. We observe the same for all data sizes with multiple fragments to transmit, unlike the FREE algorithm, which uses sensitivity and RSSI to select SFs and proves inefficient in conserving energy.

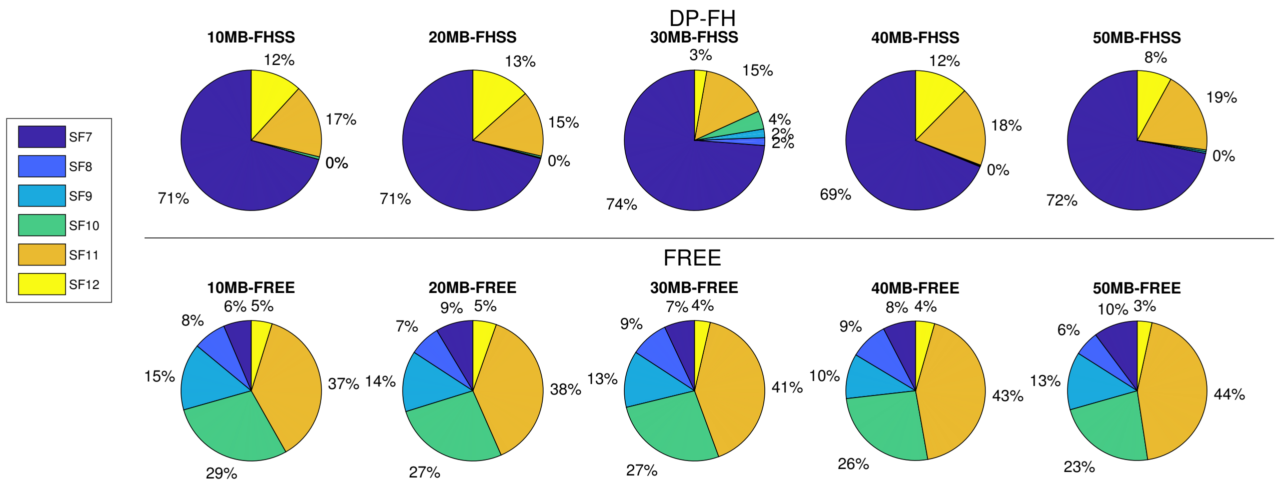


Figure 10. SF distribution of devices for increasing data packet size.

6.3. Enhancing the frequency-hopping spread spectrum using ML algorithms

When the amount of data to be transmitted increases, the number of fragments sent by each device increases, increasing the total transmission time. Thus, it is necessary to compare the transmission times of the existing frequency-hopping technique, LR-FHSS in [3], an FHSS-based approach, and DP-FH which uses DDPG-based scheduling in addition to FHSS. We compare the results of the time on air (ToA) per device and increasing data sizes ranging from 10 to 50 bytes. ToA is the time for which the transmission unit of the end device is kept ON so that the signal reaches the gateway. Figure 11 shows a comparison of the ToA of frequency-hopping-based strategies, heuristic LR-FHSS, and DP-FH in [3] that uses reinforcement learning with data rate DR8/10 (head replication = 3) abiding by the EU 868–870 MHz band regulation, which imposes a 1% duty cycle per device and channel. We observe that using the reinforcement learning DDPG algorithm for scheduling frequency-hopping improves the ToA per device. This reduces data collection time and improves the efficiency of the system. The increase in the ToA for the LR-FHSS algorithm per device is more (about 97%) [3] as packet size increases compared to the ToA taken by DP-FH. The hopping sequence suggested using reinforcement learning thus gives an optimal schedule compared to LR-FHSS.

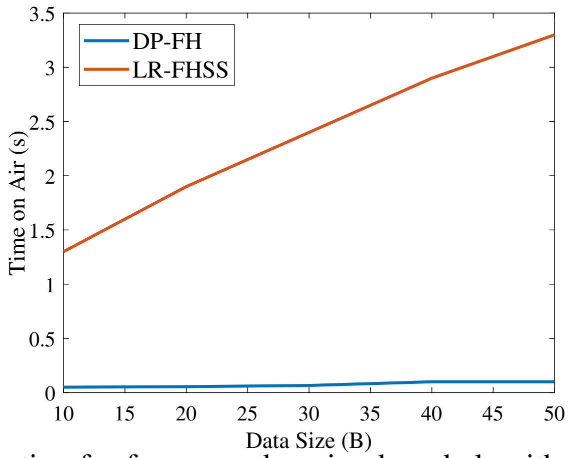


Figure 11. Time evaluation for frequency-hopping-based algorithms for an increasing data size.

6.4. Signal-specific metrics

Figure 12 plots RSSI values for distances of devices from the gateways in km. RSSI ≥ -120 dBm is required for the signal to be received without the need of a repeater in [30]. We observe that DP-FH has RSSI ≥ -120 dBm for 87.2% of the times, whereas, for the FREE approach, it is just 35%. As devices move farther (≥ 20 Km), signal strength reduces in FREE, but for DP-FH, it remains above -120 dBm, maintaining signal strength. Figure 12 plots the SNR of packets received from the corresponding number of devices. A favorable SNR means that the signal power is greater than the noise power, i.e., the receiver can demodulate the signal. An SNR below -30dB is unacceptable [31]. We observe that in algorithms LR-DP and DP-FH, received packets from most of the devices have positive SNR values as opposed to the FREE algorithm that has near zero but a negative SNR ratio. A negative ratio makes demodulation at the receiver difficult. Thus, implementing an intelligent selection of transmission power helps in improving SNR.

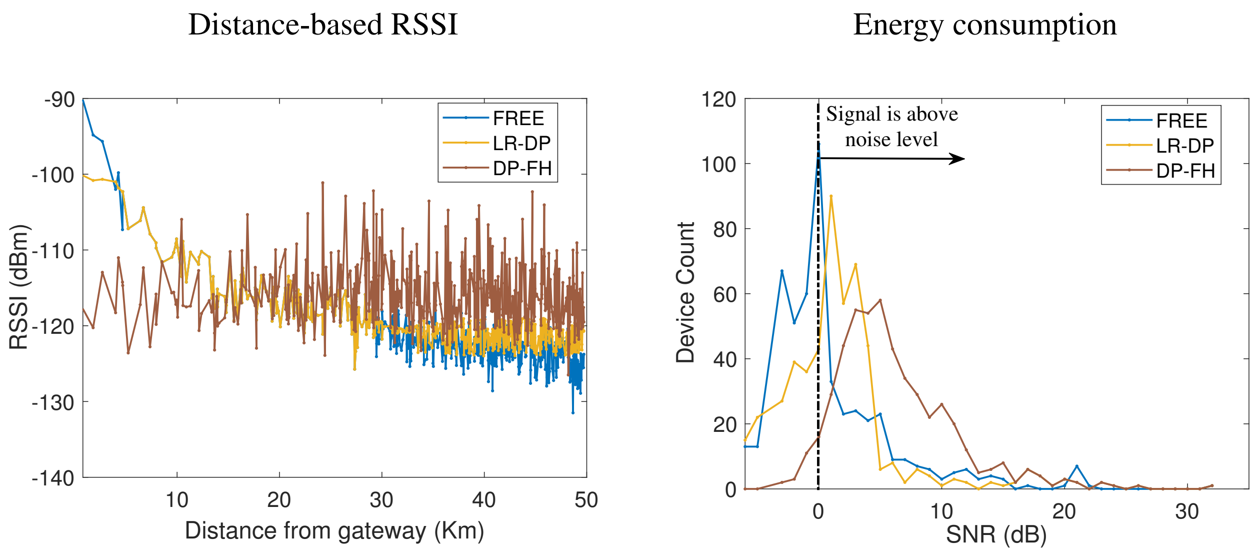


Figure 12. RSSI and SNR of devices in our proposed DP-FH algorithm.

7. Conclusions

In this paper, we propose an intelligent and energy-efficient frequency-hopping-based scheduling algorithm, named DP-FH, which is based on a deep deterministic policy gradient algorithm and incorporates a frequency-hopping spread spectrum for LoRa networks. It uses a DDPG algorithm to schedule transmission parameters in frequency-hopping LoRaWAN. Using FHSS increases the channels available for transmission, which enables more devices to transmit at the same time without collisions. This reduces the waiting time of devices and reduces the overall transmission time. Moreover, the DDPG-based scheduling algorithm generates unique transmission parameter allocation strategies for each device, ensuring collision avoidance and a higher packet delivery ratio. This reduces the amount of retransmissions and improves the energy efficiency of the system. We also show that the overall energy efficiency is increased due to lower SF allocations to devices. We compare existing heuristic as well as machine learning-based algorithms for scheduling in LoRaWAN and compare their results to show that DP-FH improves its energy efficiency and packet delivery ratio, still maintaining lower transmission times and a lower collision rate. We observe a 42% improvement in PDR, 17% more energy efficiency, 58% speed increase, and 9% lowered collisions. We have also shown that unlike FREE and LR-DP, signal strength in our approach is maintained even when devices are far away from the gateway. Thus, it is safe to specify that the proposed algorithm, DP-FH, is suitable for reducing energy consumption in LoRaWAN with the massive IoT with increased coverage.

As part of our future work, we plan to improve LoRa with frequency-hopping for communication with increased data to transmit, thus opening more use cases for LoRa. Regarding beamforming and beam steering, we plan to utilize machine learning to optimize the beamforming for the array antenna to improve the RSSI and maximize overall communication reliability while improving energy efficiency for massive IoT networks. The SNR and RSSI data from the IoT devices will be continuously monitored, and based on the RSSI of certain IoTs at specific distances from a gateway device, the beamforming and steering will be controlled to improve the data packet transmission. Simultaneously improving data packet transmission reliability, minimizing retransmission, and increasing data transmission can increase energy efficiency and communication reliability.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

Preliminary results for this work have been published in the IEEE Conference on Standards for Communications and Networking (IEEE CSCN) 2023 [14].

References

1. *M. Torchia, A. Shrivastava, K. Chinta, E. Elshewy, M. Fang, N. Guo, et al., Worldwide internet of things spending guide*, IDC Corporate, 2019. Available from: `https://www.idc.com/getdoc.jsp?containerId=IDC_P29475`.

2. *Semtech, What is LoRa*, Semtech Corporation, 2024. Available from: `https://www.semtech.com/lora/what-is-lora`.
3. G. Boquet, P. Tuset-Peiró, F. Adelantado, T. Watteyne, X. Vilajosana, Lr-fhss: overview and performance analysis, *IEEE Commun. Mag.*, **59** (2021), 30–36. https://doi.org/10.1109/MCOM.001.2000627
4. Y. Li, J. Yang, J. Wang, Dylora: towards energy efficient dynamic lora transmission control, *Proceedings of IEEE INFOCOM 2020—IEEE Conference on Computer Communications*, 2020, 2312–2320. https://doi.org/10.1109/INFOCOM41043.2020.9155407
5. L. Tu, A. Bradai, Y. Pousset, A. Aravanis, Energy efficiency analysis of LoRa networks, *IEEE Wirel. Commun. Le.*, **10** (2021), 1881–1885. https://doi.org/10.1109/LWC.2021.3084996
6. *SonicWall, Wireless: SNR, RSSI and noise basics of wireless troubleshooting*, SonicWall, 2019. Available from: `https://www.sonicwall.com/support/knowledge-base/wireless-snr-rssi-and-noise-basics-of-wireless-troubleshooting/180314090744170/`.
7. R. Sanchez-Iborra, J. Sánchez-Gómez, J. Santa, P. J. Fernández, A. F. Skarmeta, IPv6 communications over LoRa for future IoV services, *Proceedings of IEEE 4th World Forum on Internet of Things (WF-IoT)*, 2018, 92–97. https://doi.org/10.1109/WF-IoT.2018.8355231
8. A. Gloria, C. Dionisio, G. Simões, P. Sebastião, Lora transmission power self configuration for low power end devices, *Proceedings of 22nd International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2019, 1–6. https://doi.org/10.1109/WPMC48795.2019.9096197
9. D. Zorbas, K. Q. Abdelfadeel, V. Cionca, D. Pesch, B. O'Flynn, Offline scheduling algorithms for time-slotted lora-based bulk data transmission, *Proceedings of IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, 949–954. https://doi.org/10.1109/WF-IoT.2019.8767277
10. D. Zorbas, C. Caillouet, K. Abdelfadeel Hassan, D. Pesch, Optimal data collection time in lora networks—a time-slotted approach, *Sensors*, **21** (2021), 1193. https://doi.org/10.3390/s21041193
11. C. Pham, A. Bounceur, L. Clavier, U. Noreen, M. Ehsan, Radio channel access challenges in LoRa low-power wide-area networks, *LPWAN Technologies for IoT and M2M Applications*, 2020, 65–102. https://doi.org/10.1016/B978-0-12-818880-4.00004-1
12. J. Moraes, N. Matni, A. Riker, H. Oliveira, E. Cerqueira, C. Both, et al., An efficient heuristic LoRaWAN adaptive resource allocation for IoT applications, *Proceedings of IEEE Symposium on Computers and Communications (ISCC)*, 2020, 1–6. https://doi.org/10.1109/ISCC50000.2020.9219600
13. K. Q. Abdelfadeel, D. Zorbas, V. Cionca, D. Pesch, *Free*—fine-grained scheduling for reliable and energy-efficient data collection in lorawan, *IEEE Internet Things*, **7** (2020), 669–683. https://doi.org/10.1109/JIOT.2019.2949918

14. J. Mhatre, A. Lee, H. Lee, Frequency hopping scheduling algorithm in green lorawan: reinforcement learning approach, *Proceedings of IEEE Conference on Standards for Communications and Networking (CSCN)*, 2023, 216–221. https://doi.org/10.1109/CSCN60443.2023.10453154
15. T. Yatagan, S. Oktug, Smart spreading factor assignment for lorawans, *Proceedings of IEEE Symposium on Computers and Communications (ISCC)*, 2019, 1–7. https://doi.org/10.1109/ISCC47284.2019.8969608
16. M. El-Aasser, A. Gasser, M. Ashour, T. Elshabrawy, Performance analysis comparison between LoRa and frequency hopping-based LPWAN, *Proceedings of IEEE Global Conference on Internet of Things (GCIoT)*, 2019, 1–6. https://doi.org/10.1109/GCIoT47977.2019.9058411
17. A. Varga, R. Hornig, An overview of the OMNeT++ simulation environment, *Proceedings of 1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2010, 1–10. https://doi.org/10.4108/ICST.SIMUTOOLS2008.3027
18. M. A. Ullah, K. Mikhaylov, H. Alves, Analysis and simulation of LoRaWAN LR-FHSS for direct-to-satellite scenario, *IEEE Wirel. Commun. Le.*, **11** (2022), 548–552. https://doi.org/10.1109/LWC.2021.3135984
19. M. C. Bor, U. Roedig, T. Voigt, J. M. Alonso, Do LoRa low-power wide-area networks scale? *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2016, 59–67. https://doi.org/10.1145/2988287.2989163
20. I. Ilahi, M. Usama, M. O. Farooq, M. U. Janjua, J. Qadir, Intelligent resource allocation in dense lora networks using deep reinforcement learning, arXiv: 2012.11867. https://doi.org/10.48550/arXiv.2012.11867
21. R. Hamdi, E. Baccour, A. Erbad, M. Qaraqe, M. Hamdi, LoRa-RL: deep reinforcement learning for resource management in hybrid energy lora wireless networks, *IEEE Internet Things*, **9** (2022), 6458–6476. https://doi.org/10.1109/JIOT.2021.3110996
22. J. Mhatre, A. Lee, Dynamic reinforcement learning based scheduling for energy-efficient edge-enabled lorawan, *Proceedings of IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2022, 412–413. https://doi.org/10.1109/IPCCC55026.2022.9894340
23. *LoRaWAN, Notice of use and disclosure*, LoRa Alliance, Inc., 2020. Available from: `https://lora-alliance.org/wp-content/uploads/2020/11/RP_2-1.0.2.pdf`.
24. L. Zhang, B. Yang, X. You, Received signal strength indicator-based recursive set-membership localization with unknown transmit power and path loss exponent, *IEEE Sens. J.*, **21** (2021), 26175–26185. https://doi.org/10.1109/JSEN.2021.3118536
25. J. Miranda, R. Abrishambaf, T. Gomes, P. Gonçalves, J. Cabral, A. Tavares, et al., Path loss exponent analysis in wireless sensor networks: Experimental evaluation, *Proceedings of 11th IEEE international conference on industrial informatics (INDIN)*, 2013, 54–58. https://doi.org/10.1109/INDIN.2013.6622857

26. *D. Hermans, When it comes to Wi-Fi coverage, green is not always a good color*, Cambium Networks, 2018. Available from: `https://www.cambiumnetworks.com/blog/when-it-comes-to-wi-fi-coverage-green-is-not-always-a-good-color/`.

27. D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, *Proceedings of the 31st International Conference on Machine Learning*, 2014, 387–395.

28. Y. Wang, W. Fang, Y. Ding, N. Xiong, Computation offloading optimization for UAV-assisted mobile edge computing: a deep deterministic policy gradient approach, *Wireless Netw.*, **27** (2021), 2991–3006. https://doi.org/10.1007/s11276-021-02632-z

29. *RF Wireless, LoRaWAN MAC layer message formats*, RF Wireless World, 2024. Available from: `https://www.rfwireless-world.com/Tutorials/LoRaWAN-MAC-layer-inside.html`.

30. K. Koriakin, R. K. Boughton, Vaginal birthing sensors as a tool to monitor calving on large scale applications, *Comput. Electron. Agr.*, **182** (2021), 106035. https://doi.org/10.1016/j.compag.2021.106035

31. *The Things Network, RSSI and SNR*, The Things Industries, 2024. Available from: `https://www.thethingsnetwork.org/docs/lorawan/rssi-and-snr/`.