Improvement of LoRa Communication Scalability using Machine Learning Based Adaptiveness

Alexander Valach and Dominik Macko

***Abstract*—Number of embedded devices connected to the Internet is rapidly increasing, especially in the era of the Internet of Things (IoT). The growing number of IoT devices communicating wirelessly causes a communication-parameters selection problem, implying the increasing number of communication collisions. Multiple factors of IoT networks signify this problem, such as inability to communication-channel listening prior to the transmission (due to longer distances), energy constrains (due to inability of powering devices from the grid), or limitation of duty cycle and high interference (due to usage of unlicensed band in communication technologies). This article is focused on alleviating this problem in LoRa networks, which is one of the most promising technology for long-range and low-power communication. We utilize the existing LoRa@FIIT protocol to achieve energy-efficient communication. The scalability of the LoRa network is increased by modifying the communication-parameters selection algorithm. By ensuring of quality of service mechanism at each node in the infrastructure, the application domain of the proposed architecture is widened. The simulation-based experimental results showed a significantly reduced number of collisions for mobile nodes, which reduces the channel congestion and the wasted energy by retransmissions.**

***Index Terms*—Adaptiveness, efficient communication, Internet of Things, LoRa, low power, machine learning.**

I. Introduction

THE end devices of the Internet of Things (IoT; sensor or actuator based) are usually small embedded systems with low performance, small memory, and limited energy supply (battery powered or energy harvesting). Moreover, their communication has different characteristics than in conventional IP (Internet Protocol) and Ethernet network devices (bandwidth, periodicity, wireless, mobility). Wireless communication in unlicensed bands is also limited by a duty-cycle restriction. It is defined as the maximum percentage of time during which an end-device can transmit on a selected channel [2]. In Europe, the duty cycle is regulated by ETSI EN300.220 and is limited to less than 36 s time on air per hour for each node (i.e. max 1%) [3]. To avoid the duty-cycle restriction, the IoT network can use licensed communication bands, such as narrowband IoT (NB-IoT), utilized by mobile network operators, which is however more costly. All of these factors should be considered while designing the network architecture and selecting the communication protocol.

Statista [4] estimates that by the end of 2025, there will be approximately 8.30 billion LPWAN devices connected to the Internet, compared to only 3.94 billion in 2021. To meet the requirements of modern industries, IoT network should be scalable (to handle the growing number of connected devices) and adaptive (to quickly respond to network changes). It means that such a network should be able to successfully deal with the problems of congestion, duty-cycle restrictions, interference, and collisions.

IoT offers new possibilities of gathering information and knowledge, and thus provides a way to make faster and more informed decisions. It has found its application in many areas influencing our everyday lives, such as healthcare, transportation, education, agriculture, trade, or energetics [5], [6]. For example, a typical scenario in healthcare requires a person to be wearing a small battery-powered IoT device, which can remotely monitor a blood pressure, a heart rate, an oxygen saturation, etc. The sensed and collected data are then processed, evaluated, and adequate recommendations are provided to the monitored person itself (e.g. to calm down, to sit) or to the monitoring staff (e.g. to detect a disease in its early stage, to send an ambulance).

The existing research works, summarized in Section II, are focused on optimizing the communication, especially its energy requirements, minimizing the collisions, and selecting a proper communication channel and other parameters. However, most of the solutions do not consider increased overhead at gateways, communication of which is also limited by the duty-cycle restrictions. Also, very few works integrate some sort of the quality-of-service (QoS) mechanism (i.e. classification and prioritization of messages in case of a congestion to ensure low latency and successful delivery on the first attempt for important messages) into their architectures.

This article deals with a reliable LoRa network architecture using the LoRa@FIIT and STIoT (Secured TCP for IoT) communication protocols [7]. LoRa@FIIT seems quite promising, since it offers more efficient communication in a LoRa network than the widely spread LoRaWAN. However, machine-learning based recent advancements of LoRaWAN are not yet properly researched using LoRa@FIIT (meaning that it supports just the basic adaptive data rate algorithm for communication-parameters selection). Therefore, this work is focused on overcoming the limitations of the adaptive data rate algorithm in LoRa@FIIT by using machine-learning based adaptiveness to network-condition changes. The proposed modification also ensures QoS at each component of the network and provides an effective distribution of updated

Manuscript received May 21, 2021; revised XXXX XX, 2021.

This is an extended version of the paper presented at the TSP 2020 conference [1].

This publication has been written thanks to support of the Ministry of Education, Science, Research and Sport of the Slovak Republic (Incentives for Research and Development, Grant No.: 2018/14427:1-26C0), and the Slovak University of Technology in Bratislava.

A. Valach and D. Macko are with the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava, Slovakia e-mail: alexander.valach@stuba.sk, dominik.macko@stuba.sk.

communication parameters for adapting to network changes. The key contribution of this work lies in:

- energy-efficient LoRa communication using LoRa@FIIT link layer,
- increased scalability of the LoRa network considering duty-cycle restrictions,
- better LoRa@FIIT adaptiveness to network changes,
- combination of centralized and decentralized machine learning,
- ensured quality of service at each point of the network infrastructure.

The rest of the article is organized in the following way. Section II summarizes the latest research in optimization of communication via the LoRa technology. In Section III, an overview of the LoRaWAN and LoRa@FIIT protocols is provided. Section IV includes the proposed modification of a LoRa network architecture using the LoRa@FIIT and STIoT protocols with QoS. Section V presents the experimental setup with the implemented simulator. In Section VI, the experimental results are described along with a discussion. Section VII summarizes and concludes the article.

II. Related Works

One of the greatest challenges of LoRa-based IoT networks in densely populated areas is a communication without collisions and scalability of the network as a whole. In recent years, there has been an intensive research in this field [8], [9], [10], [11]. Much research has been conducted in the area of computing the optimal communication-channel parameters used by IoT end nodes and the network server [12], [13], [14], [15]. The work has been especially focused on optimizing energy requirements of end nodes [16], [17], [18], [19], [20], selecting a communication channel [21], [22], [23], reducing the number of collisions [24], [25], [26], predicting collisions [27], or making the communication generally more efficient [28], [29], [10].

A reinforcement learning has been found helpful in IoT networks, even when non-stationary settings are present in the environment [30]. According to [23], it is also helpful in coexisting IoT networks, in which multiple communication technologies are used simultaneously (e.g. Sigfox and LoRa using the same frequency bands). To simplify evaluation of various reinforcement-learning algorithms in LoRaWAN networks, the LoRa-MAB simulator was developed, which also helped to show that the distributed learning outperforms simple heuristics in terms of network throughput [31]. The work [22] showed that even a pure UCB-based (upper confidence bound – algorithm in reinforcement learning) channel access performs similarly to more complex learning strategies.

The authors of [32], [15] have also considered a dissemination of acquired information from the network server to the end nodes. As already mentioned in previous section, all LoRa nodes communication is limited by the duty-cycle restriction. Therefore, determining a suitable timing and form of above-mentioned dissemination is an important task, when considering a huge number of nodes in a real-world use case. Using this method, the network server is able to update a reward model based on RSSI (received signal strength indicator) and SNR (signal-to-noise ratio) parameters, monitored by LoRa gateways, and distribute the updated network-wise configuration to the end nodes. A reward model consists of combinations of communication parameters and an estimated reward for transmitting using a selected combination. The goal is to get highest cumulative reward possible. The work [9] tries to use a Markov Decision Process to determine the communication parameters suitable for individual end nodes. The proposed method needs to be further tested in a real-world scenario (considering signal attenuation, e.g., due to interference). The IoTligent method [26] uses a decentralized approach, in which a spectrum learning is used at the LoRaWAN end-node side to avoid collisions, and thus prolong the battery life.

Additionally, the latest research proved that a lightweight carrier sensing can also make the communication more efficient and energy-wise in a dense and harsh environment with a lot of nodes placed close to each other [24], [33], e.g., a smart-city scenario.

To conclude, a network-wise approach instead of a node-wise approach has been used in [32], [15], both in terms of a proper communication-parameters selection and a message importance. A message importance can be considered a form of QoS, as in [21], [32]. The main idea behind this solution is to update end-nodes configuration based on the overall network congestion. End nodes prefer lower Spreading Factor (SF) values, since it means less time on air, and thus it leads to lower probability of collisions [32]. The main goal was to increase network throughput in heterogeneous IoT networks, which represent a real-world scenario much more, comparing to research, in which a homogeneous IoT network is assumed [34], [35]. A heterogeneous IoT network takes into account that nodes send messages on an event-driven basis or with different periods for each node, and messages also have a different payload length.

To the authors' best knowledge, the analyzed works did not really take into account an overhead for LoRa gateways to propagate the communication parameters to the end devices, which significantly influences their duty cycle. Besides [32], [15], other works have not considered the duty-cycle restrictions at all. It is important to find an efficient form to disseminate a control-plane data from the network server to the end devices. Also, it is an important task to compute an appropriate timing of updating the end-node communication parameters. As there can be a congestion in the network, it is important that end nodes are autonomous and are able to use different communication parameters based on the state in the network. To clarify using an example: Let's assume that a device is set to use a congested spreading factor of *SF7*. Instead of using such a spreading factor, it switches to the spreading factor of *SF8* even if it increases the transmission energy requirements and a risk of collisions. The reason is that it often leads to better energy efficiency, as there is no need for retransmissions on SF that does not suffer from congestion [34], [31], [33]. This is what we are dealing with in our work, but firstly, in the next section, we overview and compare two link-layer LoRa-based protocols in order to backup our selection in the proposed architecture.

III. LoRaWAN and LoRa@FIIT

LoRa is a long-range low-power communication technology, operating in frequency bands of 868 MHz in Europe and 915 MHz in USA [2], [36]. It uses a proprietary chirp spread spectrum modulation technique. This mechanism gives to LoRa devices the ability to communicate with minimal power consumption over long distances with a relatively high resistance to interference (mainly due to the usage of lower frequencies). LoRa offers eight bidirectional communication channels and five orthogonal SFs (referred to as sub-channels), which enable multiple devices to communicate simultaneously without interference. However, the selection of SF also influences the data rate, range of the communication, and the energy consumption.

A usual LoRa-based network architecture includes multiple end nodes (EN), multiple LoRa concentrators/gateways (represented by wireless access points (AP)), and a network server (NS) (see Fig. 1). ENs are mostly sensor-based embedded devices, which are able to monitor and process some environmental parameters and transmit the measured data using the LoRa technology. APs usually represent Internet gateways, forwarding LoRa frames to the NS via an Internet connection. Communication from ENs can be received by all the APs in range, and all of them forward the message. A NS serves as a central point, controlling the communication in the network. It is responsible for computation of duty cycles to follow duty-cycle restrictions. It is also responsible for selection of suitable communication parameters. It also decides about which AP will be used for a response in case a message was received by multiple APs and must be acknowledged.

A. LoRaWAN Overview

It is not practical to send RAW data using the LoRa technology. Therefore, a link-layer protocol is needed to provide features like ensuring security, addressing of devices, or acknowledgment of messages. The LoRaWAN protocol [36], maintained by LoRa Alliance, is the most widely used link-layer protocol in LoRa networks. LoRaWAN was designed to support roaming (i.e., the owner of the network is different from the owner of end devices) and device control, using so-called MAC (medium access control) commands. In LoRaWAN case, the roaming is an important concept, since the LoRaWAN networks are primarily owned by Internet service providers and the services are provided to customers for certain fees.

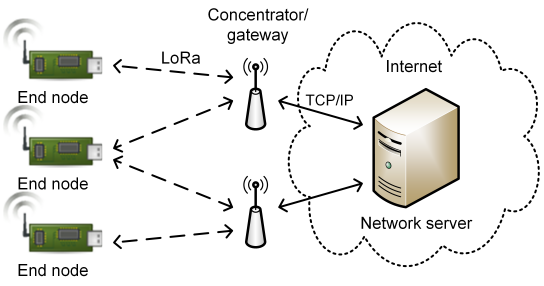

Fig. 1. A typical LoRa-based network architecture.

To secure the communication, it uses the AES-128 algorithm for data encryption. Is is a strong encryption algorithm, which was not originally designed to be used by energy constrained IoT devices with limited memory and computational power. Also, the LoRaWAN header is rather complex, in order to support all the designed features. Specifically, EN needs to send 29 B of control data to transmit 1 B of payload, which is a relatively high overhead. LoRaWAN was not designed to ensure quality of service in terms of some sort of a critical-message prioritization (e.g., an emergency message or security alert).

LoRaWAN differentiates three classes of devices. The *A* class is the most energy efficient. EN opens the receiving window for a downlink message only for a short while after it sends the uplink message. In the *B* class, the EN is able to periodically open the receiving window, upon reception of a beacon message from the LoRa gateway. The *C* class consists of ENs that keep the receiving window continuously opened besides the sending period. The highest energy requirements are put on ENs in this class; however, they have the lowest latency.

In LoRaWAN networks, a mechanism called adaptive data rate (ADR) is used by NS for updating the ENs communication parameters if required. The ADR relies on a simple comparison of the average RSSI value (calculated from a number of latest received messages) to a stated threshold value [36], [7]. Upon the average crossing the threshold, a downlink message with the updated communication settings is generated to be sent to the corresponding EN during the next opened receiving window. However, this cannot always occur in a dense environment, since the AP's duty cycle could be easily depleted.

B. LoRa@FIIT Overview

LoRa@FIIT [7] is a link-layer protocol operating above the LoRa physical layer. It was designed to provide optimized, more efficient communication than the traditional LoRaWAN in situations, in which the roaming feature is not required. Since it does not offer roaming, the owner (the administrator) of the LoRa network also needs to be the owner of the LoRa nodes. This represents a certain limitation of LoRa@FIIT usage.

However, since it was designed to overcome some LoRaWAN drawbacks, it provides several advantages:

1) *Optional acknowledgments* – The message acknowledgement is optional, and therefore, the EN does not have to open the receiving window after sending an uplink message. A type of acknowledgment (mandatory, optional, or no acknowledgement at all) can be customized for some message types (data messages), depending on application needs.
2) *XXTEA algorithm* – For data encryption in LoRa@FIIT, the XXTEA algorithm [37] is used, which requires much smaller blocks than AES-128 (32–64-bit vs. 128-bit block alignment). It was specifically designed for IoT devices, considering a constrained memory capacity. It uses a Diffie-Hellman mechanism [38] for key exchange during the registration process or key update.

3) *Energy efficiency* – The communication is more energy-efficient than LoRaWAN, since LoRa@FIIT uses shorter protocol headers, and thus has a smaller overhead. Specifically, EN needs to send only 12 B of control data to transmit 1 B of payload. This represents approximately 42% saving of energy and also smaller depletion of duty cycle [7].
4) *Sequence numbers* – Sequence numbers are used to ensure a reliable delivery of messages. The NS can also use such numbers to determine the quality of a pseudo link from an individual EN by comparing the last received sequence number with the total number of messages received from the EN since joining the network.
5) *Quality of service* – LoRa@FIIT offers a built-in mechanism ensuring QoS, which is achieved by means of a so-called emergency message. This type of a message is required to be acknowledged to ensure a reliable delivery. In the current implementation of LoRa@FIIT, it uses different frequency and separate communication parameters, and is transmitted using the maximal power and SF to increase a chance to be received by some AP. The current implementation uses ADR to optimize three separate communication-parameters groups (emergency, registration, and data messages explained below). A change in the configuration of data messages is not reflected in the configuration of registration or emergency messages.

There are multiple message types defined in the LoRa@FIIT protocol. Register messages are exchanged during an initial EN registration process. Hello messages provide a keepalive and health-check mechanism. Emergency messages are used for transmitting critical data. And finally, data messages are used for sending of usual data. Register and emergency messages are initially transmitted using the maximal power to ensure a successful delivery (or at least to increase a chance to be heard by an AP), but the parameters can be optimized. Some of the messages are illustrated in Fig. 2, the emergency and hello messages use the same headers as data messages. The header fields explanation can be found in [7].

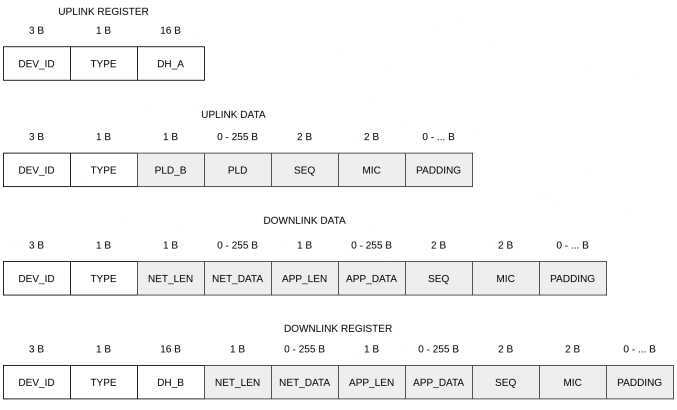

Fig. 2. LoRa@FIIT message types, protocol header fields and their sizes. Grey fields are encrypted during transmission.

Instead of using the Message Queuing Telemetry Transport (MQTT) message broker for communication between APs and NS, like in a known open-source LoRaWAN implementation [39], LoRa@FIIT uses the STIoT protocol [7]. MQTT is based on a subscriber-publisher model, while STIoT is based on a client-server model, especially focused on a secure and reliable data exchange between APs and the NS. Thanks to the used model, such an architecture is easy to deploy.

Both protocols, LoRa@FIIT and LoRaWAN, use ALOHA-based medium access methods. It means that ENs send messages whenever they are ready and use just acknowledgements to monitor whether they have been successfully received. They do not provide any other collision detection/avoidance mechanism. A responsibility to monitor the network state and a possible occurrence of collisions or interference is up to the NS (or possibly APs) instead of ENs [32].

When modifying a communication-parameters selection algorithm in the EN, the energy requirements must be taken into account. IoT ENs usually have implemented a quite simple firmware, sensible to an increased overhead, which could also increase the power consumption. Thus, a total time spent on transmitting or receiving the messages and on selecting suitable communication parameters must be carefully considered and evaluated.

IV. The Proposed Modification

The aim of the work was to provide better adaptiveness to network changes in a dynamic environment (where the end nodes are moving) and to increase the scalability of the communication technology (with regard to rapidly growing number of IoT nodes). The proposed modifications can be divided into two aspects. Firstly, a modification of the algorithm for efficient selection and propagation of communication parameters is proposed. A centralized solution was replaced by a reinforcement-learning technique that is evaluated on the end devices in case of a congestion of the sub-channel. The second modification deals with ensuring QoS at each point of the network infrastructure in order to be effective.

A. Communication-Parameters Selection

The existing communication-parameters selection implemented on the network server is based on the ADR algorithm [39]. As proved by the research of [40], [32], ADR does not perform very well with increasing number of devices and does not perform well with mobile nodes. One of the main advantages of the LoRa technology is its high resistance to interference and immunity to Doppler effect, which makes a promising solution for mobile nodes. The ADR algorithm optimizes a transmission power and SF based on the average value of RSSI for messages sent by a given node. This is also the case of the existing LoRa network server implementation [41], in which the threshold values (based on which ADR reacts) are set globally. However, a number of historical messages (which are used for evaluation of the average RSSI) is fully customizable. It is set to 20, by default, which can make the algorithm rather slow in a dynamic environment.

In this work, we propose a more modern approach for communication-parameters selection suitable for dynamic environments, which is exactly the case of mobile LoRa nodes. Instead of setting a single communication-parameters combination based on an average RSSI value, we propose the usage of a reward model. In such a model, several combinations can be selected in case of a congestion of some SF. Each combination of SF and transmission power (TP) has a predefined reward (initially set to 1), which changes dynamically in time, according to network conditions. Dynamic reward updates follows these rules to simplify the solution:

1) *Successful transmission* – If an acknowledgment for an emergency message is received, the reward value is increased by 1. The rule applies to both EN and NS.
2) *Failed transmission* – If an acknowledgment for an emergency message is not received, the reward value is decreased by 1. The reward value cannot be negative, it is rather set to zero. This applies to both EN and NS. On the NS, a failed transmission is indicated by missing sequence-number values. Therefore, there is a computation latency, as the NS cannot evaluate message loss without additional uplink messages.
3) *Unknown results* – If an acknowledgment is not required, EN is unable to update its reward model. However, the NS follows both rules for successful and failed transmissions and updates the reward model based on any type of messages, not only emergency and registration messages.

Due to the reward-model based approach, ENs are not strictly limited to use a single SF and TP combination and can now select their communication parameters using an UCB algorithm. UCB was primarily developed as a reinforcement-learning technique to solve a Multi-Armed Bandit (MAB) problem, well-known from a recommendation process. A reinforcement-learning is a field of machine learning that copes with exploration versus exploitation dilemma, in which a certain decision has to be made with only a limited knowledge of the whole situation. The MAB algorithm calculations are based on the provided reward (a feedback) that is received when a certain arm (a communication-parameters combination) is pulled (selected) by a bandit (EN). To differentiate these autonomous nodes, we introduce the term *bandit node*, which is an end node that uses a reinforcement-learning technique to select communication parameters. UCB can perform quite well for communication-parameters selection according to the performed computer simulations [30], [34], [22]. However, it was designed to deal with static rewards that do not change in time and is considered the worst-performing from all MAB algorithms [30], [34]. The decision-process of UCB is heavily dependent on the first pull, e.g., the initial SF value. It is an ϵ-greedy approach that prefers a local maximum to the globally optimal path [42]. In this work, we compare UCB (as the worst-performing but easy-to-use MAB algorithm) to ADR (as the current state of the art), considering a scenario of constantly moving mobile end nodes.

UCB uses a confidence interval for each arm, which is considered an additional reward to the empirical experience of a single EN. It follows the strategy of being optimistic about uncertain communication parameters. An overview of UCB algorithm is presented in a form of the pseudocode in Fig. 3.

Input: A list of all arms in net_data.
Output: The selected best arm.

$arm \leftarrow 0$
$max\_upper\_bound \leftarrow 0$
for $i = 0$ **to** num_of_arms **do**
  if $num\_of\_selections[i] > 0$ **then**
    $avg\_reward \leftarrow \frac{sums\_of\_rewards[i]}{num\_of\_selections[i]}$
    $exploration \leftarrow \sqrt{\frac{2\times\log(num\_tries+1)}{num\_of\_selections[i]}}$
    $upper\_bound \leftarrow avg\_reward + exploration$
  else
    $upper\_bound \leftarrow 10^{400}$
  end if
  if $upper\_bound > max\_upper\_bound$ **then**
    $max\_upper\_bound \leftarrow upper\_bound$
    $arm \leftarrow i$
  end if
end for
increment num_tries **by** 1
append arm **to** $arms\_selected$
increment $num\_of\_selections[arm]$ **by** 1
$chosen\_arm = net\_data[arm]$
increment $sums\_of\_rewards[arm]$ **by** reward of $chosen\_arm$
increment $total\_reward$ **by** reward of $chosen\_arm$
return $chosen\_arm$

Fig. 3. A pseudoalgorithm of the implemented UCB.

The algorithms that aim to solve the MAB problem are suitable for a selection of communication parameters (a combination of SF and TP), when the signal strength and message-delivery successfulness are not predefined. Using an ADR algorithm, the ENs cannot just simply select these parameters without additional help from the network server. There are two main reasons for this:

1) The proposed reward model has to be constantly updated based on the average RSSI value (indicating a TP or SF update) and the sequence number (indicating a message loss), which are not known to ENs.
2) ENs are not ready to select the parameters with limited knowledge of a network state without frequent updates and careful reward-model dissemination.

To summarize the necessary steps, NS maintains a reward model for EN and AP, and disseminates it only when there is a change in the environment or a link quality has been decreased. The model is easily updated on NS using data from APs (SNR, RSSI, and a sequence number). SNR and RSSI are used to determine a link quality, and a sequence number is used to detect a link congestion (i.e., missing sequence-number values) or a signal loss for individual ENs. To determine an environmental change, the same threshold as in case of ADR is used (i.e., the average RSSI value). ENs are responsible only

for communication-parameters selection based on the updated reward model. The main problem of the proposed solution is maintaining this model on ENs.

Because of the duty-cycle restrictions, it is impossible in LoRaWAN networks to provide a feedback to ENs immediately after an uplink message has been sent, if a dense and harsh environment with hundreds of nodes is assumed. LoRa@FIIT has a shorter header (12 B) [7] than LoRaWAN (21 B) [36] and a support for mandatory acknowledgement messages. ENs can benefit from sending messages with a mandatory acknowledgement, because they are provided with a feedback whether the message was delivered. Thus, ENs can accordingly update their reward model without an intervention from the NS.

The proposed reward model consists of SF, TP, and a reward for selecting the respective combination of both parameters. To simplify the selection process (i.e., to reduce a number of possible combinations), we propose to use only two power values (10 and 14) for each SF in the reward model. The ordered combinations follow the rule that the transmission power is increased in prior to an increase of SF, since it has lower impact on energy consumption. The reward model is stored in the JSON (JavaScript Object Notation) format. It can be encoded in a base64 string or even the BSON (Binary JSON) form to reduce the size of a downlink message. When a significant change occurs in the LoRa network, the reward models of individual ENs have to be updated. The NS prepares and schedules a downlink message with the updated combinations, which is immediately transmitted to the EN in the form of an acknowledgement with additional network data, e.g., an updated combination (sf: 12, tp: 14, rw: 5). Within an initial EN registration, the reward model of the AP that received the registration message is sent to the EN, if there is no prior model for the EN (see Fig. 4 for illustration of a dissemination process). There is a situation when a mobile EN is switching between different APs. However, we do not address this problem, as there has to be a change point detection approach implemented on NS in the network [43], [44], [45]. This issue can be addresses by an adversarial MAB algorithm of the Global Switching Thompson Sampling with Bayessian Aggregation [34].

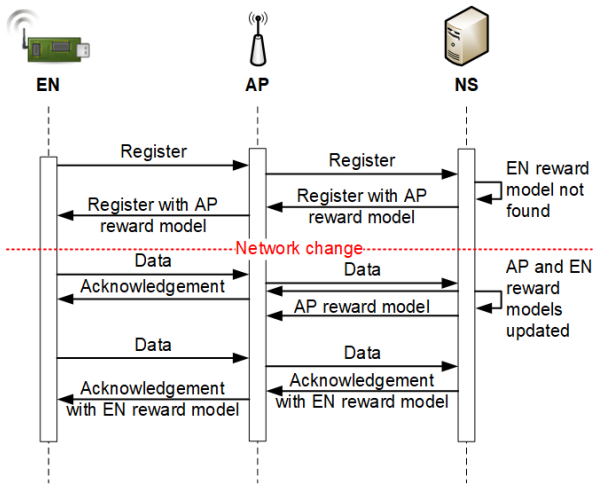

Fig. 4. A reward model dissemination.

B. Quality of Service

LoRa@FIIT protocol offers a built-in mechanism to ensure QoS and a reliable delivery. Due to optional acknowledgements, an EN does not have to open a receiving window, when an acknowledgment from the NS is not required (which saves the energy). Some messages, however, always require the acknowledgement of their delivery. Thus, some messages need to be acknowledged and some may be acknowledged (depending on the specific application requirements). This is quite beneficial in non-critical applications, in which unnecessary acknowledgements would just waste the time, energy, duty cycle, and bandwidth.

However, there are also critical events, which require the lowest possible latency. As already mentioned, LoRa@FIIT supports an emergency message, which is transmitted using the maximal power to increase the probability of receiving by any nearby AP. This can be considered as a kind of the QoS mechanism in ENs. However, the QoS implementation in the other network components (the NS or APs) is missing. Therefore, the lowest possible latency is not guaranteed, since there can be delays introduced in AP or NS.

The proposed modification assumes a priority queue for the emergency messages to be implemented in the AP as well as in the NS (illustrated in Fig. 5). The AP and the NS then process the messages from the priority queue with precedence to other types of messages in the regular queue.

Another problem in the current LoRa@FIIT implementation (with regard to QoS) is that the NS has to wait for a predefined time (500 ms by default, can be changed by the administrator) prior to acknowledging the message. It waits to receive all copies of the same LoRa message, possibly forwarded by different APs. The reason is to cope with the duty-cycle restriction. After all copies of the same message are received, the NS selects the AP with the highest available duty cycle to transmit the reply. The NS does not recognize (before the timer expiration), whether there is only a single AP in range of the EN, and thus there is no reason to wait.

In the proposed modification, the NS does not wait for the copies of messages in the priority queue, in order to minimize a delay. Thus, upon a reception of an emergency message, it is immediately processed and replied using the AP, by which the original emergency message was forwarded to the NS.

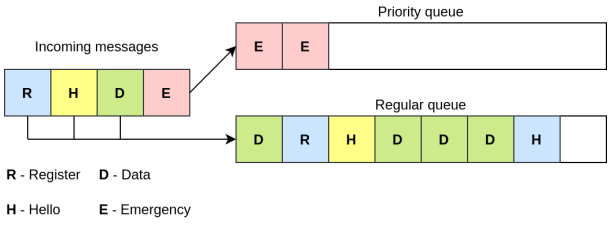

Fig. 5. A dedicated queue for the emergency messages.

V. The Experimental Setup

For being able to evaluate the proposed modifications, we have designed and implemented a LoRa@FIIT AP simulator. The proposed AP simulator consists of a custom console application, written in Python. It was developed to simulate a daily routine of ENs. The MAC layer implementation is inspired by the LoRa@FIIT library [46] and other existing components [47], [48]. It aims at simulating the functions of a single AP with multiple connected ENs. Physical ENs in the intended healthcare application would be connected for the purpose of heart-rate measuring. The simulator simulates this process by generating pseudorandom sensor-measured values, based on the heart-rate calculation algorithm currently implemented in the official library of the sensor [49]. The healthcare is considered a type of an application, in which a certain degree of QoS is required, represented by a need of an emergency message to notify critical events. The simulator is connected to a real-world network server running in a virtual environment in a cloud (see Fig. 6).

An abstract overview of the developed simulator components is illustrated in Fig. 7. It consists of several components:

1) *LoRa PHY Helper* – supports the time-on-air calculation per each message, the SNR value generation, RSSI value calculation based on the distance of EN from AP, and it detects collisions in the network.
2) *Access Point* – this module registers AP to NS using SETR STIoT message and calculates remaining duty cycle of AP.

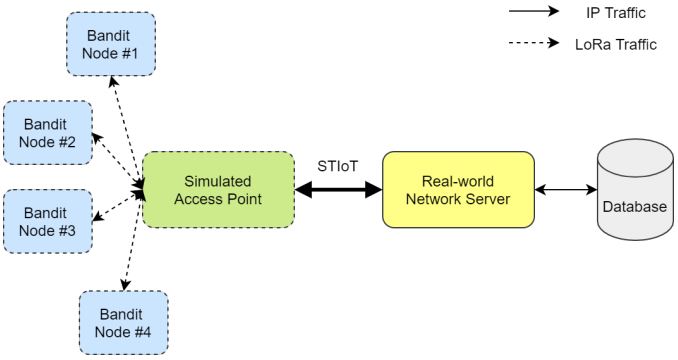

Fig. 6. An architecture of evaluation network with simulated nodes and a real-world network server.

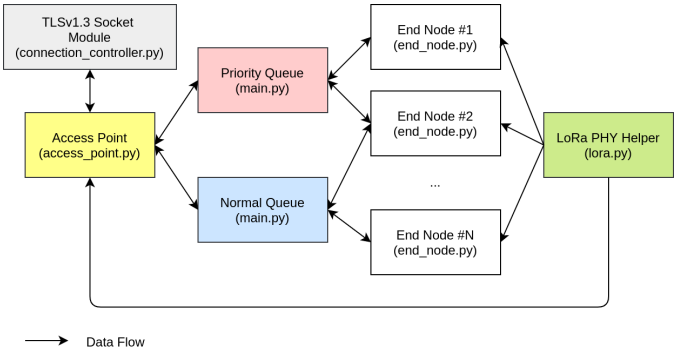

Fig. 7. The LoRa@FIIT AP simulator overview.

3) *Priority and Normal Queue* – a program uses two message queues to support the proposed QoS mechanism. The priority queue is checked first, and only when it is empty, the simulated AP processes other messages.
4) *End Node* – this module covers a device routine, its communication-parameters selection process, movement of each node and is responsible for handling retransmissions.
5) *Bandit Node* – covers the same functionality as EN module. Rather than using different communication-parameters groups (emergency, data, and normal), which is the case in ADR, it uses a common communication-parameters group for all traffic types and selects them based on the used MAB algorithm (UCB). The proposed method eliminated separated groups because when a separate channel is used for emergency messages, it is not fully utilized. In congested network, it can lead to inefficient communication, as there are channels which are rarely used and channels that are crowded.
6) *Connection Controller* – the SSL module is used to provide a secure communication with the NS using the STIoT protocol (the TCP port of 25001). Traffic is encrypted using the latest standard TLSv1.3 (Transport Layer Encryption).

The source code of the implemented LoRa@FIIT AP simulator is available on GitHub [48] along with the list of currently supported features. The application has also a command-line interface and can be modified in the future to support LoRaWAN as well, since the physical properties stays the same for both protocols – only a header, message types, and acknowledgment behaviour are changed in LoRa@FIIT. There is also no need to implement the actual XXTEA encryption, as there is no over-the-air communication between ENs and AP. However, messages are exchanged between simulator and NS using TLSv1.3 encryption.

A single LoRa NS, implemented in Java, was deployed for testing purposes [41]. Such a setup is able to test hundreds of simulated LoRa ENs and several APs. It is sufficient to simulate a situation of increasing LoRa network traffic, which might occur in a real-world scenario. However, we are unable to simulate an interference and collisions between several AP, as our packet generator simulates only a single AP.

A. The Performance Metrics

During the experiments the following performance metrics were evaluated:

1) *Collision ratio* – a number of collisions per each node (collisions explained below).
2) *Packet delivery ratio* – a ratio of successfully delivered messages to all sent messages.
3) *Distribution of SF* – a number of messages transmitted on each SF. An important aspect is to use various SF values. When a *SF7* is congested, a higher and supposedly ineffective decision should be made. Certain devices are able to mitigate collisions and save more energy by transmitting on a rarely used SF value.

4) *Active time ratio* – expresses the time spent during transmission and reception of messages along with processing time to select proper communication parameters. This value is compared to the total device uptime to calculate the ratio a device has spent in active time. Since ENs are expected to last for several years on a single battery, this ratio is expected to be very low.

In the designed semi-simulated environment with a single AP, the following events are considered collisions, as a collision generally represents a state in which a message was lost due to various causes:

1) *Packet collisions* – when multiple ENs transmit messages using the same channel (only a single frequency channel is used in experiments) and the same subchannel (i.e., SF) at the same time, or their transmitting messages (frames) are overlapping with each other.
2) *RSSI* – the RSSI value is below the receiver's sensitivity (e.g., below -120 dB). The reason is that the EN is too far away from the AP, or it communicates using a lower SF value than possible (as a result of slow or no adaptability).
3) *Noise* – a SNR value is a randomly generated number. If it is between certain thresholds (from 0 to -7.5), a message is considered unreadable, due to high interference. As a result, a receiver is unable to demodulate the message, which is discarded afterwards.

The SNR values are randomly generated in a range from -20 to 10. The RSSI values are calculated based on the distance of an EN from the AP using a Free Space Path Loss model and varies from less than -120 dB to 0 dB. As no physical obstacles are present, a larger area (10 km $\times$ 10 km) was used for the nodes to loose signal near the edges of the area. However, no signal shadowing, reflection, or interference from other technologies were considered, as it would be in a real-world environment.

A capture effect was also considered in the implemented collisions calculation. When an AP receives two messages simultaneously, it cannot normally decode them (i.e., a packet collision). However, if one of the messages has the signal stronger than the other one by at least 6 dB, it will suppress the weaker signal and successfully decode the stronger message.

B. Duty-Cycle Constrains

The duty-cycle restrictions are also considered in the implemented simulator. Both APs and ENs store their remaining duty-cycle and are not able to transmit messages if the value is too low (e.g., too low duty-cycle value is determined, when 100 ms is remaining and a message with 120 ms of time-on-air is queued). In such a case, the simulated devices have to wait for the next duty-cycle refresh. In case of AP's low duty-cycle value, it leads to higher collision rates, because it is unable to transmit the scheduled acknowledgement messages (i.e., unnecessary retransmissions occur).

Due to the duty-cycle restrictions and early experiments, we decided to limit the number of ENs up to 100. The maximum number of nodes being connected to a single AP simultaneously depends on the type of messages sent (a high amount of registration or emergency messages significantly decreases this number, as both messages requires an acknowledgment from the NS). During one hour, only 10–15 nodes can be registered to a single AP, because of the long network-configuration information that is carried in a registration-acknowledgement downlink message (0.9–1 s). To speed up the joining process, we decided to pre-register the nodes during the first run of each experiment and work with the registered nodes in the following runs.

VI. Experimental Results and Discussion

We have evaluated the proposed modifications using two different simulation scenarios, each consisting of 15 runs:

1) *Adaptive Data Rate* – the current state of the art solution of both LoRa@FIIT and LoRaWAN networks, designed for static nodes with the additional full QoS support.
2) *Upper Confidence Bound* – the proposed solution using a reinforcement-learning technique to improve adaptiveness of ENs and scalability of network. UCB is suitable to support even mobile nodes. This scenario also includes architecture with the full QoS support.

Each EN is assigned a random x and y coordinate from a uniform distribution within 100 km^2 area at the beginning. Nodes run as background processes and constantly move each second, when not transmitting, in one of the following direction; up or down, left or right. A random variable is generated and assigned a value between 0–1.4 m for both vertical and horizontal movement. In conclusion, each node is able to perform a diagonal movement, or stay still when both coordinates-movement variables are set to 0. The maximal value of 1.4 m/s represents an average speed of a moving person. When a node is close to being out of the specified area (a border), it reverses the direction.

As mentioned in the previous section, the ENs were not connected all at once, as this would easily lead to depletion of the duty cycle of the AP. Each time, when there was nothing to process in both queues on AP, a new device tried to join the network (i.e., to start sending messages). If the device was out of range of the AP, it simply could not join the network. After three failures, it switched to a sleep mode and tried to join the network again after two minutes. As a result, at the end of each experiment, a different number of ENs were connected. There were some nodes that were unable to join the network, because of simply being too far away from the AP and their inability to switch to a proper SF.

Since each node has been pre-registered before an experiment started, its reward model (consisting of bandit arms and their corresponding rewards, starting at initial value set to 1) was already aware of the situation in the network. In the UCB scenario, the same average RSSI value principle was used as in the case of the ADR evaluation. The same settings were used for both scenarios, as presented in Table I. ENs were sending messages with various payload length every 120 s to create a dense and harsh environment with frequent transmissions (with regard to the duty-cycle constrains). Each EN can send either an emergency message or a data message, but emergency messages are preferred to speed-up the learning

TABLE I
COMMON EXPERIMENT SETTINGS FOR BOTH EVALUATED ALGORITHMS

Characteristic	Value
Frequency	866.1 MHz
Coding rate	4/5
Bandwidth	125 kHz
Duration of a single run	7 hours
Number of access points	1
Number of end nodes	up to 100
Uplink transmission frequency	120 s
Payload length (range)	16–52 B
Number of runs	15

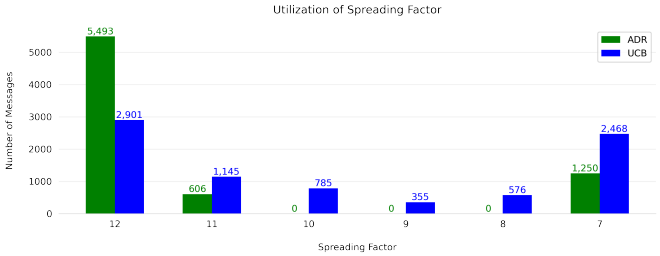

Fig. 8. Comparison of a number of messages sent on each SF.

process. The duty cycle of the AP was closely examined and it was not depleted due to a relatively low number of ENs.

The distribution of the used SF is presented in Fig. 8. Most of the ADR nodes begin transmission on *SF12* and only a portion was able to lower the value to *SF11*. Most of the SFs were not used at all. Some of the nodes started on *SF7* as a result of the device-registration process before the experiments. Lower SF values were preferred for data messages, which is expected and a suitable approach (since a lower SF value means lower energy requirements). On the other hand, the bandit nodes using UCB were able to utilize all SF values fairly, also starting on *SF12* or *SF7* depending on their previous network experience.

As illustrated in Fig. 9, many collisions have occurred in the ADR scenario, with a mean value of 212, compared to a much lower mean value of 4 for UCB. The main reason for a high collision rate in ADR is the mechanism of sending emergency messages used by ENs. Each node must send a message with the mandatory acknowledgement. If an emergency message was not acknowledged due to an uplink/downlink collision or a low duty-cycle value, the message was retransmitted up to two more times, and then the EN enters a sleep mode. The message was discarded afterwards, because after 120 s of sleep time, the coordinates carried in the payload were probably updated (explained below in path rendering).

For ADR, 95.1% of the network traffic consisted of the emergency messages with the mandatory acknowledgement and only 4.9% of the data messages with no acknowledgement. For UCB, the ratio of the emergency messages was even higher, 98.2% with only 1.8% of data messages. The data messages were randomly placed between the emergency messages. In a real-world scenario, an emergency message is only sent based on the application needs, when a certain threshold is reached.

For mandatorily acknowledged messages (i.e., an emergency message), the initial value of *SF12* is used to increase the range. Higher SF leads to higher time-on-air for a message, and thus higher probability of a collision. In a real-world scenario, mandatorily acknowledged messages would use different frequencies [7]. However, collisions mainly occurred between the same message types, e.g., the first node's message acknowledgement and the second node's message transmission, both using *SF12*. For ADR, the time-on-air, which depends on SF value and payload length, has a mean value of 2142.88 ms per message. A mean value is only 548 ms for UCB, due to an improved communication-parameters selection procedure, where lower SFs are used when getting closer to the AP.

A packet delivery ratio (PDR) was also evaluated to provide an overview of the situation in the evaluated network (illustrated in Fig. 10). Majority of ENs in ADR have a very low PDR value. A mean value was only 11.91%, compared to a much higher value of 80.85% for UCB. However, the value is still not very reliable for applications with specific QoS requirements. The overall PDR was decreased (despite a low collision rate) because of the performance of 12 nodes that were unable to join the network at all (10 nodes in case of ADR). The important metrics are also clearly presented in Table II.

A path for each EN has been rendered to verify the mobility of nodes and is presented in Fig. 11. We were unable to render the path for ENs that were unable to send uplink messages, since the coordinates were part of a payload. These coordinates used a different number of places after decimal period, resulting in a variable length of the payload. Most of the nodes did not change their direction and did not traverse the whole area. However, they were constantly moving closer

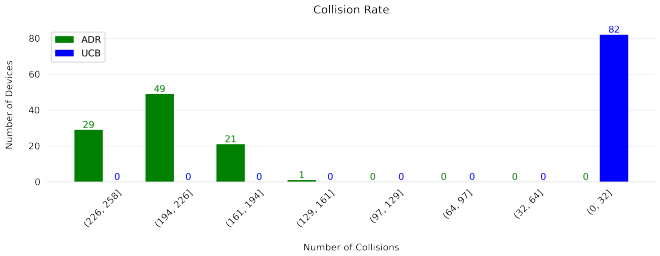

Fig. 9. Comparison of a number of collisions.

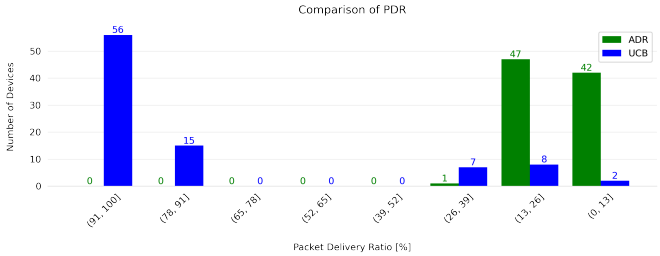

Fig. 10. Comparison of packet-delivery ratio values.

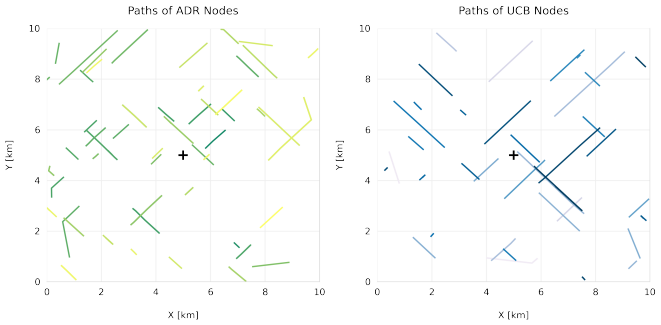

Fig. 11. Rendered paths of each node based on coordinates sent to the NS with AP in the center of the 100 km^2 area.

to or getting away from the AP, placed in the center of a defined area. The paths for UCB appear longer due to a higher PDR, as the NS was able to receive more messages from ENs.

We were unable to measure the exact energy consumption using the implemented simulator. However, we were able to calculate a ratio between the active time and the total uptime for each EN. The active time represents the time spent outside of a sleep mode. This time consists of the total time spent in transmitting and receiving mode and during a communication-parameters selection process. However, we did not distinguish between different modes with different power consumption, as it was not our goal to measure the exact power consumption.

At the end of each experiment, a program printed a list of ENs and their respective total uptime, total active time, and a number of collisons. In ADR scenario, 23.15% of total device uptime (a mean value) was spent in the active mode, compared to only 2.5% of total uptime for UCB, as presented in Table II. These results clearly show that the use of the UCB algorithm in a dense and harsh environment can lead to a more energy-wise solution. The reason is that the risk of collisions decreases with increasing ability to adapt for network changes. This is in contrast to the assumption that an additional overhead for computing a proper communication parameters automatically leads to a higher overall energy consumption. The active time ratio is extremely (since low-power devices were connected only for hours) high for ADR due to the congestion of *SF12* and missing ability to adapt without a NS intervention. A congestion of the desired SF leads to frequent massive retransmissions. A back-off time before the retransmission process might help, but is unable to solve the problem. The reason is that a message transmitted using *SF12* usually occupied the sub-channel for 0.9–1 s in the experiments.

TABLE II
COMPARISON OF BOTH ALGORITHMS USING SELECTED METRICS

Characteristic	ADR	UCB
Total connected nodes	90	88
Collisions per end node (mean)	212	4
Packet delivery ratio (mean)	11.91%	80.85%
Time spent in active mode (mean)	23.15%	2.50%
Emergency message ratio	95.1%	98.2%

TABLE III
ADVANTAGES AND DISADVANTAGES OF THE PROPOSED LORA@FIIT MODIFICATIONS

Advantages	Disadvantages
a reduced number of collisions along with energy consumption due to a better adaptiveness	the increased EN computation overhead due to computation of suitable communication parameters by the EN itself
a reduced number of retransmissions due to the reduced number of collisions and QoS support	the increased complexity of LoRa nodes due to reward model maintenance and ensuring prioritization in all ENs, APs, and the NS
the increased communication energy efficiency due to the use of LoRa@FIIT instead of LoRaWAN with reduced control overhead	no roaming support due to the use of LoRa@FIIT instead of LoRaWAN
the faster adaptiveness to network changes due to a partial independence from the NS	
the increased quality of service due to a priority queue at APs and NS	

To summarize our work, the identified advantages and disadvantages of the proposed modifications are provided in Table III. In the future, we plan to test the UCB approach in a real-world environment, where multiple access points are present and a switching environment is introduced (i.e., the previously learnt reward model is not valid when moving to another AP). Interference from the environment also comes into a consideration and the energy consumption can be accurately measured, not just estimated its increase. Also, as a further work, we plan to implement a Thompson Sampling on ENs to compare the results with the currently implemented UCB approach. There is also a possibility to select not only between SF and TP, but also to select different frequencies. However, the process of selecting the optimal communication-parameters combination gets more complex with the increasing size of an initial communication-parameters set. This can lead to a higher required maintenance of the reward model in compensation for higher throughput and reduced number of collisions.

VII. CONCLUSION

LoRa is one of the most promising technologies for long-range communication of IoT devices. However, its potential could be lost, if not using carefully. New challenges arise, such as a dramatically increasing number of connected devices, duty-cycle restrictions, and nonoptimal communication-parameters selection. These challenges must be coped with to deploy LoRa in a real-world application with thousands of connected devices. In this article, we have proposed a modification of the existing LoRa@FIIT protocol, ensuring energy-efficient, QoS-supporting and reliable communication over the LoRa technology. Inspired by the results of other existing works in this field, we have used a network-wise statistical model for individual network devices, which is maintained by the network server based on information obtained from

LoRa gateways. For the communication-parameters selection, the Adaptive Data Rate algorithm was replaced by the Upper Confidence Bound algorithm, since the statistical model represents a reward model for a kind of the Multi-Armed Bandit problem. This model is disseminated to end nodes during the registration process and they use it for optimal communication-parameters selection. The statistical model can be further updated by the network server when a significant network change is detected. Moreover, the end nodes can use acknowledgements as a feedback to adjust the model themselves, thus being partially independent from the network server, enabling them to more quickly respond to the current network conditions. The proposal has been evaluated in the implemented simulation environment. The results showed that the number of collisions is significantly reduced. Moreover, although the computation overhead of end nodes was increased, the estimated overall energy requirements of end nodes were also reduced (in contrast to assumption). This makes reinforcement-learning algorithms quite promising in communication-parameters selection process, not only in terms of efficient communication, but also with regard to energy efficiency.

REFERENCES

[1] A. Valach and D. Macko, "Optimization of LoRa devices communication for applications in healthcare," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2020, pp. 511–514.

[2] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," *IEEE Communications magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[3] ETSI, "Short range devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz; Part 1: Technical characteristics and methods of measurement," 2017, eTSI EN 300 220-1 V3.1.1 (2017-02).

[4] (2021) Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[5] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[6] N. Saleh, A. Kassem, and A. M. Haidar, "Energy-efficient architecture for wireless sensor networks in healthcare applications," *IEEE Access*, vol. 6, pp. 6478–6486, 2018.

[7] O. Perešíni and T. Krajčovič, "More efficient IoT communication through LoRa network with LoRa@FIIT and STIOT protocols," in *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE, 2017, pp. 1–6.

[8] Y. Hou, Z. Liu, and D. Sun, "A novel MAC protocol exploiting concurrent transmissions for massive LoRa connectivity," *Journal of Communications and Networks*, vol. 22, no. 2, pp. 108–117, 2020.

[9] R. M. Sandoval, A.-J. Garcia-Sanchez, J. Garcia-Haro, and T. M. Chen, "Optimal policy derivation for transmission duty-cycle constrained LPWAN," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3114–3125, 2018.

[10] D. Zorbas, G. Z. Papadopoulos, P. Maille, N. Montavont, and C. Douligeris, "Improving LoRa network capacity using multiple spreading factor configurations," in *2018 25th International Conference on Telecommunications (ICT)*. IEEE, 2018, pp. 516–520.

[11] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on 5G networks for the Internet of Things: Communication technologies and challenges," *IEEE Access*, vol. 6, pp. 3619–3647, 2017.

[12] A. L. Emmanuel, X. Fernando, F. Hussain, and W. Farjow, "Optimization of spreading factor distribution in high density LoRa networks," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, 2020, pp. 1–5.

[13] M. Parvin and M. R. Meybodi, "MABRP: A multi-armed bandit problem-based energy-aware routing protocol for Wireless Sensor Network," in *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)*. IEEE, 2012, pp. 464–468.

[14] A. Tiurlikova, N. Stepanov, and K. Mikhaylov, "Method of assigning spreading factor to improve the scalability of the LoRaWAN wide area network," in *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 2018, pp. 1–4.

[15] R. M. Sandoval, D. Rodenas-Herraiz, A.-J. Garcia-Sanchez, and J. Garcia-Haro, "Deriving and updating optimal transmission configurations for LoRa networks," *IEEE Access*, vol. 8, pp. 38 586–38 595, 2020.

[16] H. H. R. Sherazi, L. A. Grieco, M. A. Imran, and G. Boggia, "Energy-efficient LoRaWAN for industry 4.0 applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 891–902, 2021.

[17] N. Michelusi and M. Levorato, "Energy-based adaptive multiple access in LPWAN IoT systems with energy harvesting," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 1112–1116.

[18] J. Zhang, H. Jiang, Z. Huang, C. Chen, and H. Jiang, "Study of multi-armed bandits for energy conservation in cognitive radio sensor networks," *Sensors*, vol. 15, no. 4, pp. 9360–9387, 2015.

[19] A. Azari and C. Cavdar, "Self-organized low-power IoT networks: A distributed learning approach," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[20] Y. Li, J. Yang, and J. Wang, "DyLoRa: Towards energy efficient dynamic LoRa transmission control," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 2312–2320.

[21] A. Hoeller, R. D. Souza, O. L. A. López, H. Alves, M. de Noronha Neto, and G. Brante, "Analysis and performance optimization of LoRa networks with time and antenna diversity," *IEEE Access*, vol. 6, pp. 32 820–32 829, 2018.

[22] R. Bonnefoi, L. Besson, J. Manco-Vasquez, and C. Moy, "Upper-confidence bound for channel selection in LPWA networks with retransmissions," in *2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW)*. IEEE, 2019, pp. 1–7.

[23] S. Hasegawa, S.-J. Kim, Y. Shoji, and M. Hasegawa, "Performance evaluation of machine learning based channel selection algorithm implemented on IoT sensor devices in coexisting IoT networks," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–5.

[24] E. M. Rochester, A. M. Yousuf, B. Ousat, and M. Ghaderi, "Lightweight carrier sensing in LoRa: Implementation and performance evaluation," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[25] J. Pullmann and D. Macko, "A new planning-based collision-prevention mechanism in long-range IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9439–9446, 2019.

[26] C. Moy and L. Besson, "Decentralized spectrum learning for IoT wireless networks collision mitigation," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 644–651.

[27] S. Cui and I. Joe, "Collision prediction for a low power wide area network using deep learning methods," *Journal of Communications and Networks*, vol. 22, no. 3, pp. 205–214, 2020.

[28] L. H. Shen, C. H. Wu, W. C. Su, and K. T. Feng, "Analysis and implementation for traffic-aware channel assignment and contention scheme in LoRa based IoT networks," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[29] D. Magrin, M. Capuzzo, A. Zanella, L. Vangelista, and M. Zorzi, "Performance analysis of LoRaWAN in industrial scenarios," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[30] R. Bonnefoi, L. Besson, C. Moy, E. Kaufmann, and J. Palicot, "Multi-armed bandit learning in IoT networks: Learning helps even in non-stationary settings," in *International Conference on Cognitive Radio Oriented Wireless Networks*. Springer, 2017, pp. 173–185.

[31] D.-T. Ta, K. Khawam, S. Lahoud, C. Adjih, and S. Martin, "LoRa-MAB: Toward an intelligent resource allocation approach for LoRaWAN," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[32] R. M. Sandoval, A. Garcia-Sanchez, and J. Garcia-Haro, "Optimizing and updating LoRa communication parameters: A machine learning approach," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 884–895, 2019.

[33] T.-H. To and A. Duda, "Simulation of LoRa in NS-3: Improving LoRa performance with CSMA," in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.
[34] R. Kerkouche, R. Alami, R. Féraud, N. Varsier, and P. Maillé, "Node-based optimization of LoRa transmissions with multi-armed bandit algorithms," in *2018 25th International Conference on Telecommunications (ICT)*. IEEE, 2018, pp. 521–526.
[35] D. Bankov, E. Khorov, and A. Lyakhov, "Mathematical model of LoRaWAN channel access with capture effect," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 2017, pp. 1–5.
[36] LoRa Alliance, "Whitepaper: A solution for successful interoperability with DLMS/COSEM and LoRaWAN," 2019, https://lora-alliance.org/sites/default/files/2019-11/dlms-lorawan-whitepaper_v1.pdf.
[37] S. K. Vishwakarma and S. Khare, "XXTEA an optimized encryption design with high feedback substitution box architecture," *International Journal of Modern Engineering & Management Research*, vol. 2, no. 3, pp. 12–16, 2014.
[38] M. Ahmed, B. Sanjabi, D. Aldiaz, A. Rezaei, and H. Omotunde, "Diffie-Hellman and its application in security protocols," *International Journal of Engineering Science and Innovative Technology (IJESIT)*, vol. 1, no. 2, pp. 69–73, 2012.
[39] (2019) Chirpstack network server. [Online]. Available: https://github.com/brocaar/chirpstack-network-server
[40] S. Li, U. Raza, and A. Khan, "How agile is the adaptive data rate mechanism of LoRaWAN?" in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 206–212.
[41] (2019) LoRa network server. [Online]. Available: https://github.com/alexandervalach/lora-network-server
[42] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*, ser. ALT'11. Berlin, Heidelberg: Springer-Verlag, 2011, p. 174–188.
[43] J. Mellor and J. Shapiro, "Thompson sampling in switching environments with Bayesian online change detection," in *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, C. M. Carvalho and P. Ravikumar, Eds., vol. 31. Scottsdale, Arizona, USA: PMLR, 29 Apr–01 May 2013, pp. 442–450.
[44] A. Robin and R. Feraud, "EXP3 with drift detection for the switching bandit problem," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 10 2015.
[45] R. Alami, O.-A. Maillard, and R. Feraud, "Memory bandits: a Bayesian approach for the switching bandit problem," in *NIPS 2017 - 31st Conference on Neural Information Processing Systems*, 12 2017.
[46] (2017) LoRa@FIIT library. [Online]. Available: https://github.com/HalfDeadPie/LoRa-FIIT
[47] (2019) LoRa@FIIT access point. [Online]. Available: https://github.com/alexandervalach/lorafiit-access-point
[48] (2020) LoRa@FIIT access point and end nodes simulator. [Online]. Available: https://github.com/alexandervalach/lora-ap-sim
[49] (2019) SparkFun MAX301x particle sensor library. [Online]. Available: https://github.com/sparkfun/SparkFun_MAX3010x_Sensor_Library