Park *et al. EURASIP Journal on Wireless Communications and Networking* (2020) 2020:176
https://doi.org/10.1186/s13638-020-01783-5

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Network resource optimization with reinforcement learning for low power wide area networks

Gyubong Park†, Wooyeob Lee† and Inwhee Joe*

*Correspondence:
iwjoe@hanyang.ac.kr
†Gyubong Park and Wooyeob Lee contributed equally to this work.
Department of Computer Science, Hanynag University, 222 Wangsimni-ro, Seongdong-gu, Seoul, 04763, South Korea

Abstract

As the 4th industrial revolution using information becomes an issue, wireless communication technologies such as the Internet of Things have been spotlighted. Therefore, much research is needed to satisfy the technological demands for the future society. A LPWA (low power wide area) in the wireless communication environment enables low-power, long-distance communication to meet various application requirements that conventional wireless communications have been difficult to meet. We propose a method to consume the minimum transmission power relative to the maximum data rate with the target of LoRaWAN among LPWA networks. Reinforcement learning is adopted to find the appropriate parameter values for the minimum transmission power. With deep reinforcement learning, we address the LoRaWAN problem with the goal of optimizing the distribution of network resources such as spreading factor, transmission power, and channel. By creating a number of deep reinforcement learning agents that match the terminal nodes in the network server, the optimal transmission parameters are provided to the terminal nodes. The simulation results show that the proposed method is about 15% better than the existing ADR (adaptive data rate) MAX of LoRaWAN in terms of throughput relative to energy transmission.

Keywords: LPWA, LoRa, Reinforcement learning, Resource optimization, DQN

1 Introduction

As the 4th industrial revolution using information becomes an issue, information and communication technologies such as the IoT (Internet of Things), big data, and AI (artificial intelligence) have become popular around the world. In particular, the IoT is a very important part of the technology that generates information that is the basis of the 4th industrial revolution. As a result, the scale of the IoT will increase in the next few years, and the current wireless communication environment will not be able to meet the demands of society. It is necessary to research the Internet of Things to meet the demands of the future society.

Even if the number of connected devices is large, there is an IoT environment in which the amount of data generated by each device is relatively small. This environment

opens up the possibility of large-scale connectivity and LPWA (low power wide area) networks [1]. Compared with traditional wireless technology, LPWA technology aims to provide a compromise between low power consumption, range, and data rate to address a wider range of requirements than traditional IoT applications. There are several commercial technologies such as LoRaWAN, SIGFOX, NB-IoT, INGENU, and TELENSA in this LPWA. This paper is aimed at LoRaWAN, which discloses standard protocols.

LoRa specification includes two layers: physical layer (LoRa RF) and link layer (LoRaWAN) [2]. LoRa enables low-power, long-distance communication through modulation schemes using chirped spread spectrum technology at the physical layer. And the link layer is designed to consume the minimum power of the end node using pure ALOHA method. This design makes LoRa possible for low-power long-distance communication, but there are disadvantages due to the simplicity of the terminal node. The possibility of packet collision has increased due to the increase of time on air of packets in LoRa system, and LoRa also has near-far problem depending on the location of gateway and terminal node. Therefore, in order to minimize these side effects, appropriate spreading factor values and transmission power values should be set. LoRaWAN provides ADR to find an appropriate factor value.

This paper attempts to use reinforcement learning to find the appropriate parameter values. Deep learning technology is currently used to solve a variety of problems, including Deep Mind's AlphaGo. In particular, deep learning is widely used in network fields such as network resource allocation and load balancing. In this paper, we take a deep approach to the problem of the distribution of network resource optimization such as spreading factor, transmission power, and channel allocation in LoRaWAN.

This paper is organized as follows. Section 2 discusses LoRa and the existing network resource allocation methods, discusses deep reinforcement, and explains the motivation of the proposed method. Section 3 describes the proposed method. Section 4 analyzes the performance of the proposed method through simulation. Finally, Section 5 concludes this paper.

2 Related work

2.1 LoRa

LoRa is an abbreviation for Long Range and is a long-range wireless communication system [2]. LoRa uses a shared frequency band called unlicensed spectrum and consists of LoRa terminal nodes, gateways, and network servers as shown in Fig. 1. The terminal and the gateway can communicate with 1-Hop, so it is composed of star-shaped network topology. The terminal and the gateway transmit data through LoRa connection, and the gateway and the network server exchange data received from the terminal through wire. LoRa consists of two layers, LoRa and LoRaWAN. LoRa can send data at low power and long distance because of the unique modulation schemes in the LoRa physical layer. Chirp spread spectrum (CSS) modulation is used for the physical layer, which uses signals that increase or decrease with the increase of time called chirp.

The symbol duration of LoRa is based on spreading factor (SF) and bandwidth (BW). The symbol of each LoRa is composed of convolutions for the entire bandwidth, and the symbol duration (T_s) is as shown in Eq. 1.

$$T_s = \frac{2^{SF}}{BW} \tag{1}$$

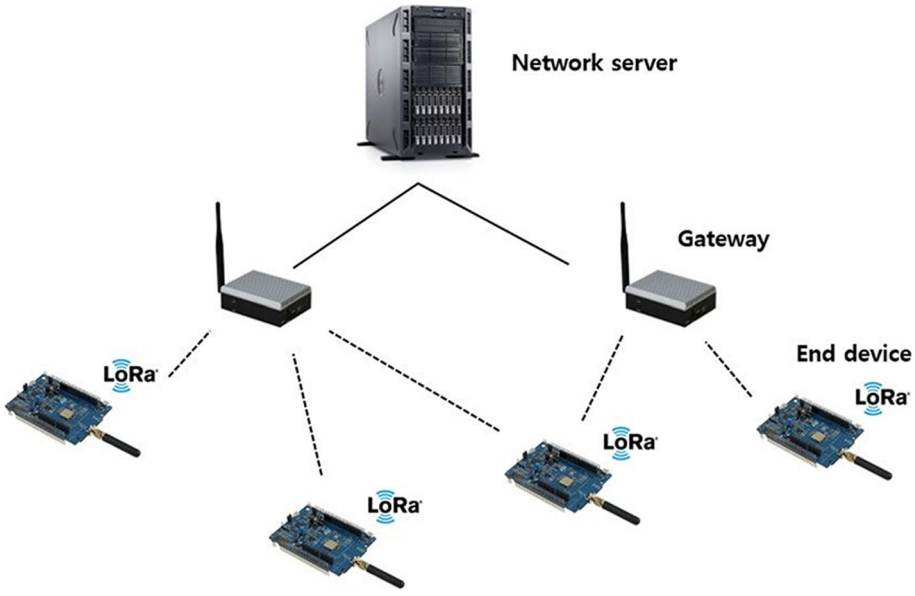

Fig. 1 LoRa network

Therefore, the symbol duration is doubled according to the value of SF when having a fixed bandwidth. This means that the higher the SF, the greater the probability of collision between packets due to the symbol duration, and also doubles the data rate over the same time. In addition, as symbol duration increases, more energy is consumed in signal encoding, increasing battery consumption. On the contrary, as the symbol duration increases, the message transmission time (ToA) increases, and communication can be performed at a relatively longer distance than the low SF. Since SF has orthogonality, the gateway can receive signals encoded with different SF values in the same channel. In Table 1, we can see the bit rate and sensitivity as SF increases.

The second network layer, called LoRaWAN, processes the media access control layer (MAC). LoRaWAN supports three devices: class A, class B, and class C. Class A pursues low power, so it uses pure ALOHA instead of carrier sense multiple access (CSMA) to check if media is in use. Class B allocates additional slots to increase the downlink success rate from the gateway. Allocating additional slots increases the probability of data reception but consumes more power than class A. Class C always allocates downlink slots in an externally powered format.

Table 1 Bit rate and sensitivity according to LoRa modulation

Mode	Bit rate (kb/s)	Sensitivity (dBm)
LoRa SF = 11	0.537	−134.5
LoRa SF = 10	0.976	−132
LoRa SF = 9	1757	−129
LoRa SF = 8	3125	−126
LoRa SF = 7	5468	−123

2.2 Performance issues of LoRa and ADR

LoRa enables low-power, long-range wireless communication due to the simple modulation of the special modulation scheme and pure ALOHA. However, this structure not only has advantages but also disadvantages. There is no carrier sensing for packets sent from other terminal nodes, and the message transmission time due to high SF increases the probability of packet collision as the number of terminals increases.

LoRaWAN also has a near-far problem like other wireless communication environments. If the received signal strength indicator (RSSI) of the packet transmitted from the LoRa gateway is very large, the packet transmitted from the terminal node far from the gateway may not be recognized by the gateway. LoRa has this performance problem [3].

LoRaWAN has this problem, so it provides adaptive data rate (ADR) to solve the above problems. ADR can control symbol duration, data rate, and energy consumption to minimize side effects as the number of LoRa terminals increases. As shown in Algorithm 1, the ADR collects approximately 20 packets initially and then analyzes the received packet strength to determine the SF increase/transmission power (TP) increase. By adjusting SF and TP properly, near-far problem can be solved by minimizing packet collision and increasing the strength of transmission signal to the node far from gateway [3].

Algorithm 1 LoRa ADR Algorithm

```
i ← 0
offset ← 10
history[j] ← 0 for all j in [ 0, 19]
threshold ← {20.0, 17.5, 15.0, 12.5, 10.0, 7.5}

if i = 20 then
   margin ← max(history) - threshold[DR] - offset
   steps ← round(margin/3)

   if step >0 then
      increase DR by steps until DR = 5
      decrease TP by remaining steps until TX = 0
   else
      increase TP by stps until TX = 5
   end if
else
   history[i] = mSNR
   i ← i+1
end if
```

The ADR uses the signal-to-noise ratio (SNR) of the collected packets to determine SF and TP. The ADR varies greatly depending on the number of packets collected and how to set the representative values. As a result, much research has been conducted to maximize the performance of ADR. The research by Hauser et al. [4] changed the existing ADR logic

to make network resource distribution more efficient. Reynders et al. [5] significantly reduced the retransmission rate of packets by adding logic to change the channel according to the distance between the gateway and the terminal node to the ADR. There are several characteristics that can lead to an unbalanced distribution of data extraction rate (DER) between nodes. Since SF does not have perfect orthogonality, it is not possible to detect weak packet signals due to strong packet signals. For this reason, packets of nodes with large signal attenuation are not delivered well. In view of these issues, Abdelfadeel et al. [6] proposed the Fair Adaptive Data Rate Algorithm (FADR). FADR is an ADR in which LoRa achieves an effective data extraction rate with an appropriate spreading factor and transmission power while excluding excessively high transmission power. Kim et al. [7] suggested congestion estimation through logistic regression to improve the transmission performance degradation of LoraWAN without considering congestion. They proposed a congestion classifier based on logistic regression, which improves performance in terms of transmission delay. When using the same data rate and not considering collision, throughput is greatly reduced. Therefore, Kim and Yoo [8] proposed a contention-aware adaptive data rate for throughput optimization and applied a gradient projection method to find the optimal value. Through this, the throughput was improved to a value close to the theoretical maximum value. El-Aasser et al. [9] proposes the algorithms of sensitivitySF and assignmentSF, which are smarter SF setting techniques than the existing ADR, to increase both the throughput between the nodes of the same tier and the overall throughput, and achieve higher throughput than the existing ADR. For fair SF allocation, Cuomo et al. [10] proposed ordered water-filling-based EXPLoRa-KM (K-means) and EXPLoRa-TS (time symbol) techniques. KM mitigates the critical region where the collision occurs seriously, and TS plays a role to make the traffic load of each SF constant. Through this, fair allocation was achieved. Zhou et al. [11] proposed Data Rate and Channel Control (DRCC) to support massive number of LoRa nodes and improve resource utilization. The channel was evaluated based on data extraction rate (DER), SF allocation was performed based on DER, and load balancing of the channel was performed in consideration of packet collision according to node density. Through this, it was proved that the proposed technique is excellent in dense deployment scenarios. Ta et al. [12] derived optimized SF for fair resource allocation using Exponential Weights for Exploration and Exploitation (EXP3) algorithm to support large-scale LoRa nodes. Through the simulation using real data such as non-uniform node distribution and inter-spreading factor, collision was reflected, and as a result, fair resource allocation was achieved.

As such, we are conducting advanced research to modify the existing ADR or add technologies such as machine learning for efficient and reliable usage of network resources.

2.3 Deep reinforcement learning

Reinforcement learning is an algorithm that derives the best value for the situation through interaction with the environment. Reinforcement learning agents recognize state in the environment and take appropriate action. From this behavior, the agent learns through rewards to determine if the action is appropriate and to maximize its future reward.

This environment of reinforcement learning is generally modeled as a Markov decision process (MDP). An agent operates according to a policy, which is expressed as an actionable distribution according to each state defined in the MDP. The goal of a agent is to

continually refine the policy to maximize the total future reward $G_t = \sum_{i=t}^{T} R_{i+1}$ attained at any time t.

Most reinforcement learning algorithms define a state s_t in t time, a value r_{t+1} obtained from the environment by performing an action a_t in that state s_t, and a new state s_{t+1} transformed from this continuous interaction into a tuple $(s_t, a_t, r_{t+1}, s_{t+1})$. An agent learns from these consecutive sequential tuples. This learning is usually done through action-valued functions called Q values [13], which is usually expressed as Eq. 2.

$$Q(s,a) = E^{\pi}[\, G_t \mid S_t = s, A_t = a] \tag{2}$$

Equation 2 is the expected value obtained by taking action on the state according to the policy. On-policy and off-policy are divided according to whether the agent recognizes the current policy. If the current policy is continuously followed, it is on-policy.

In general, off-policy reinforcement learning is used because it is better at finding similar optimal values than on-policy reinforcement learning. Among these off-policy reinforcement learning is Q-learning. Q-learning is a simple structure that learns policy from greed policy that takes action from the highest estimated Q value in each state. In 2014, the Deep Q-Network Algorithm [14] was created when deep learning combined with Q-learning in Deep Mind. Q-learning updated the value function as shown in Eq. 3.

$$\begin{aligned} Q(s_t, a_t) &= Q(s_t, a_t) \\ &\quad + \alpha(r_t + \gamma max'_a Q(s'_t, a'_t) - Q(s_t, a_t)) \end{aligned} \tag{3}$$

On the other hand, Deep Q-Network (DQN) works by adding artificial neural network parameters to define the loss function as shown in Eq. 4 and repeating it i times to reduce the cost of the loss function through gradient descent.

$$\begin{aligned} L_i(\theta_i) &= E_s, a, r, s'[\, (y_i^{DQN} - Q(s, a; \theta))^2] \\ &\quad with, y_i^{DQN} = r + \gamma max_{a'}(s', a'; \theta_i^-) \end{aligned} \tag{4}$$

It also prevents divergence by using experience replay, which randomly extracts the collected experiences and learns them. DQN has been able to solve the high dimensional observation space problem by integrating deep learning technology with existing Q-learning.

By incorporating deep learning into traditional reinforcement learning, it was able to develop more practically than before. However, this deep learning may not solve all real problems. The world we live in consists of multi-agents and is complicated by the cooperation or competition of these agents [15, 16].

Areas such as autonomous driving, robot soccer, and wireless networks are composed of one environment and partially aware agents, and these agents achieve their goals through negotiation or competition with each other. In the case of deep learning, the state of the MDP model is designed on the assumption that the whole environment can be recognized. However, the multi-agent system has no choice but to understand the environment only partially by the agent. Therefore, MDP in multi-agent environment is called Partially Observable Markov Decision Process (POMDP) [17]. We need to be able to apply deep learning in the POMDP environment so that we can solve more real-world problems. Centralized RL (reinforcement learning) and distributed multi-agent RL (DMRL) are models designed to solve this real-world multi-agent problem.

The centralized RL deploys one agent to recognize all states centrally and to select successive actions. The advantage of this method is that the agent can select actions collectively without designing cooperation or competition. The disadvantage is that it takes a plurality of actions and thus has a relatively large amount of action space. How to effectively handle a relatively large number of states and actions will be required for centralized reinforcement learning methods.

On the other hand, in distributed multi-agent RL, each agent exists independently and each agent learns its own policy through cooperation and competition. However, since they exist independently of each other, a system for cooperation and competition must be additionally implemented. DMRL has this disadvantage, but it has a fixed action space because the agent is separated.

2.4 Motivation of the proposed method

In order for LoRa to consume the minimum transmission power while guaranteeing the maximum data rate, an appropriate spreading factor (SF) and transmission power (TP) must be selected. Well-selected and distributed SFs significantly increase the total throughput of LoRa network [19] [18]. However, using the same parameters selected by ADR among LoRa nodes to increase the data rate may increase the loss rate of transmitted packets. In addition, the distance between the gateway and LoRa nodes should be taken into account. As the distance increases, parameters such as transmission power and encoding method need to be changed. Like this, it is not easy to estimate the optimal value considering everything. LoRa has an ADR proposed by LoRaWAN to solve this network resource problem, and complementary algorithms are proposed. However, to minimize the complexity of algorithm, those methods compute the parameters with very limited number of samples and simply set the user-defined threshold value to get the parameters. They may increase the total throughput compared to the normal class A, but the energy consumption may increase too. Therefore, to decrease the energy consumption and increase packet arrival rate, we devised an appropriate transmission parameter method for LoRaWAN by using deep learning. Information about the LoRa terminal is transmitted to the central network server. By adjusting the network transport parameters through the reinforcement learning agent on the network server, the overall network situation of LoRa can be identified and network resources allocated more efficiently than the existing ADR method. The contributions of our proposed method are as follows: (1) we considered not only the throughput of nodes but also the energy efficiency of nodes; (2) we considered the distances among the nodes and gateway, and we grouped the nodes by distance and their samples are learned on the same agent; and (3) if the sufficient learning is done, our methods provide the appropriate parameters immediately.

3 Proposed method

We recognized LoRaWAN as a multi-agent problem. We used a mixture of centralized RL and distributed multi-agent RL methods. Since LoRa has a star-shaped network, information about each terminal is collected in a central network server. Therefore, like the centralized RL method, one controller can determine the information about each terminal and distribute network resources to each terminal. However, in this case, the number N of terminals and the number of actions of each terminal must be considered, that is, N action space must be considered. If only one controller is configured, the number of

action spaces in the controller varies depending on the number of terminals. As the number of terminals increases, the number of terminals has a relatively large number of action spaces. This is high. Therefore, reinforcement learning agent corresponding to each LoRa terminal is created in the network server to create multiple controllers, reducing action space and giving common reward for each action so that each terminal has fixed action space.

Our model is to create reinforcement learning agents corresponding to LoRa's terminals in the network server and learn them. Therefore, like centralized RL, state information about each node is centralized. The model presented in this paper takes the most appropriate action based on the information of all the centrally gathered nodes. In this case, we use independent information corresponding to LoRa terminal nodes like distributed multi-agent RL rather than centralized RL to prevent the increase of action space. It is designed to receive high rewards when creating an agent and taking action considering all LoRa terminals through common rewards. We explain how the model presented is applied to LoRa environment in detail in order of reinforcement learning agent, state, action, and reward.

3.1 System model

In this section, we describe the system model. Figure 2 demonstrates our reinforcement work. It is assumed that one LoRa gateway and multiple LoRa nodes are distributed in a two-dimensional grid of size 1500 × 1500. LoRa gateway is located at the center of the grid, and nodes are uniformly distributed in the grid. Basically, LoRa gateway and nodes adjust transmission power and spreading factor according to ADR, and transmission method follows class A protocol. In this paper, in order to derive the transmission power and spreading factor through reinforcement learning, an additional role for reinforcement learning was given to gateway. LoRa gateway is a container for the agent for the node and basically creates an agent according to the number of nodes. However, if there are many nodes, the nodes located at similar distances are grouped and managed by one agent. Each agent performs training through training data in the learning stage, and the nodes transmit messages every 30 min in the test stage to obtain transmission power and spreading factor through the agent in the gateway and apply it to the ADR.

3.2 Reinforcement learning agent

The goal of this paper is to analyze LoRa network situation using reinforcement learning agent and find the optimal network parameters accordingly. For this reason, MDP is applied. LoRa system is composed of N terminal nodes. s_t is the node information of t time zone, a_t is the network parameter selected at one terminal node and r_t is the compensation value obtained through corresponding network parameter, and last s_{t+1} is applied to action. This is defined as the state of the LoRa network. The experience gained thus made tuples $(s_t, a_t, r_{t+1}, s_{t+1})$ to learn reinforcement learning agents.

And all the reinforcement learning agents of the proposed model are composed of Deep Q-Network (DQN) [14]. Therefore, the end of the reinforcement learning was defined as the end of the divided episode as shown in Fig. 3. Since the model is composed of DQN, we update the value function as shown in Eq. 4. Because DQN updates the value function in this way, we can find the convergence value quickly, maximizing future compensation by updating the Q value every step before termination.

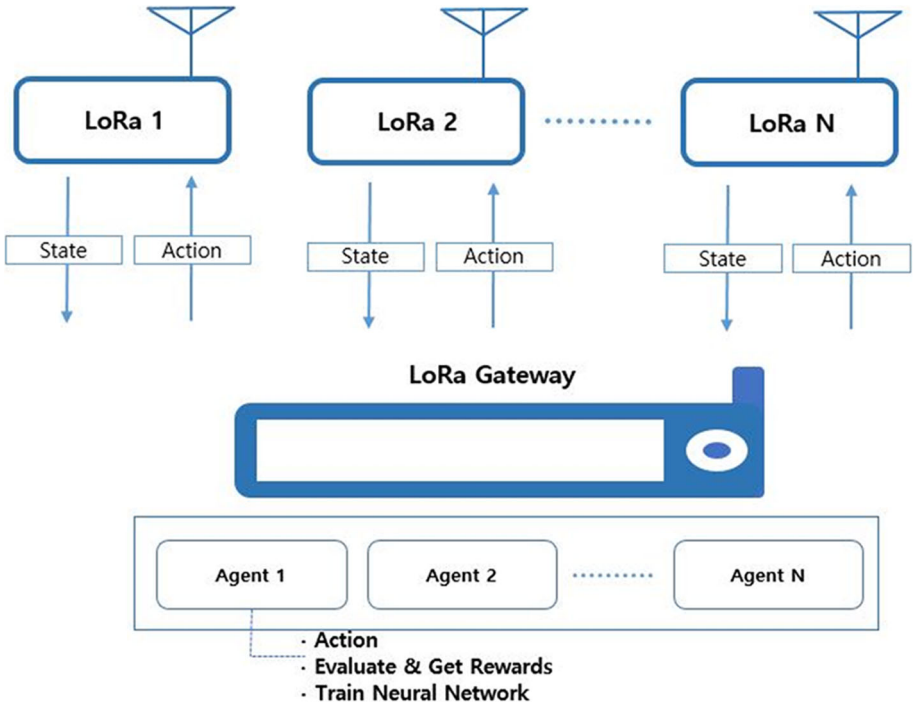

Fig. 2 LoRa reinforcement learning model

3.2.1 State

We should give reinforcement learning agent enough information about the environment, so the agent can recognize the differences by state and take appropriate action for each situation. We have included attribute values such as terminal information, packet information, and learning information in state so that the reinforcement learning agent can fully understand network status of LoRaWAN. Figure 4 shows the state information of reinforcement learning. The state is composed of a matrix of terminal information, packet information, and learning information. Each row of the matrix represents the information of the terminal node and is continuously updated until the learning is completed as the environment changes.

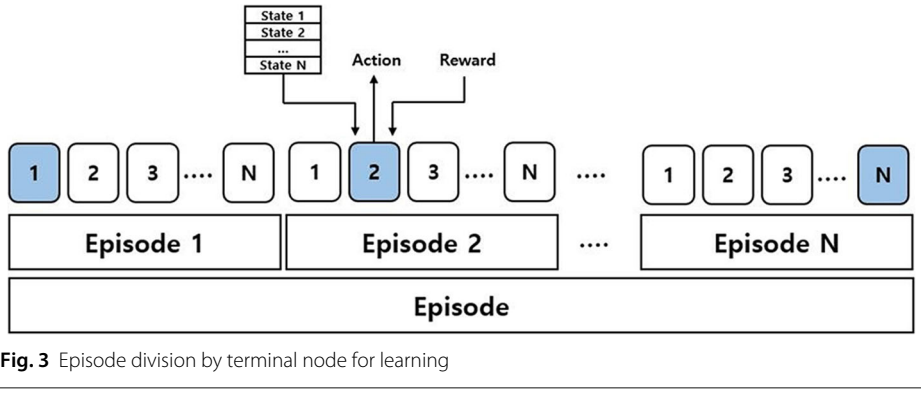

Fig. 3 Episode division by terminal node for learning

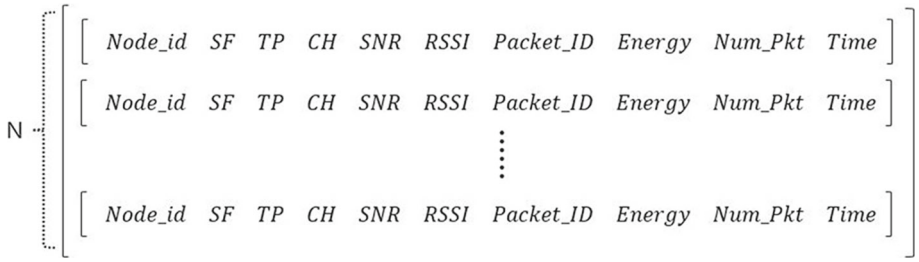

Fig. 4 State of the proposed model

Node ID, SF, TP, and CH of Fig. 4 are information about terminal node. Node ID is a value mapped to LoRa terminal node. SF denotes a spreading factor, TP denotes a transmission power of a terminal, and CH denotes a channel value currently used by a terminal. These values like SF, TP, and CH are in one-hot encoding format. SNR, RSSI, Packek ID, and Energy are information on currently received packets. SNR is the signal-to-noise ratio. RSSI is the received signal strength indicator, signal strength of packet. Packek ID is the unique number generated each time packet is sent from terminal node, and Energy is the energy consumed at the terminal node. We calculate it based on the research by Callebaut et al. [22]. Num_Pkt represents the sum of unique packets received at the gateway. Time is information to help reinforcement learning by including the entire simulation time in state information.

3.2.2 Action

In the LoRa system model used in this paper, the parameters affecting network conditions are SF, TP, and channel. So SF, TP, and channel are designed as the action of reinforcement learning. In LoRa, SF should be one of 7 to 12; TP should be one of 2 dBm, 5 dBm, 8 dBm, 11 dBm, and 14 dBm as the transmission power [20]; and the channel should be one of 868.1 MHz, 868.3 MHz, and 868.5 MHz. Therefore, it has 90 fixed actions which are $SF\{7,8,9,10,11,12\} \times TP\{2,5,8,11,14\} \times Freq\{868.1 \times 10^6, 868.3 \times 10^6, 868.5 \times 10^6\}$. In this paper, one of these 90 actions can be selected as an action.

3.2.3 Reward

The reward for action is given as shown in Eq. 5.

$$\text{Reward} = \frac{\alpha \sum_{i=0}^{N} P_i}{\sum_{i=0}^{N} E_i} \tag{5}$$

In Eq. 5, N is the number of nodes. P is the sum of the packets received over time, and E means the sum of energy consumed by the terminal. By designing a reward like this, the reward increases as the total packet increases, and the reward decreases as more energy is consumed. As the reward changes according to the state of the entire node, the reward gets closer to the full optimization rather than the partial optimization. The packet success rate is given priority over the energy consumption by multiplying α which is a very large value.

3.3 Learning method

In this paper, LoRa terminal is treated as one agent. As a result, each agent has its own policy. However, in case of LoRa, the terminal nodes are not mobile, and if the terminal and the gateway are located in a similar position, they will have similar policies as other nodes. Therefore, as shown in Fig. 5, we form groups in tracks according to the distance between the terminal and the gateway in order to share a common learning weight. In the proposed method, the number k of groups was previously determined to create fixed groups. The criteria for dividing groups are distances, and each group has a certain distance range in tracks. If the number of sectors per group is variable, a new sector must be learned, so a fixed sector is assumed. Algorithm 2 represents group creation and assignment. First, the boundary of group is derived by calculating the range of each group through the communication radius, which is set to 750, given the number of groups. Then, after calculating the distance between each node and the gateway, each node joins the corresponding group. In the learning process, nodes placed in each group were randomly divided between 0 and 30, and they were continuously repeated to learn to cope with various deployment situations. In the evaluation process, 6 nodes were placed in each group.

In this way, agents with similar policies can learn reinforcement learning agents more efficiently because they share the weights of neural network models with each other. Figure 6 is the overall learning procedure of the proposed scheme. The learning manager in the LoRa gateway gathers the distances of nodes then creates the learning group by similar distances, then generates RL agents per each group. If the gateway receives a packet, it converts node's information to the current state and determines the corresponding agent by pre-determined group. After that, it computes the current action for the current state and does current action. Then, it calculates the reward for the current action and updates

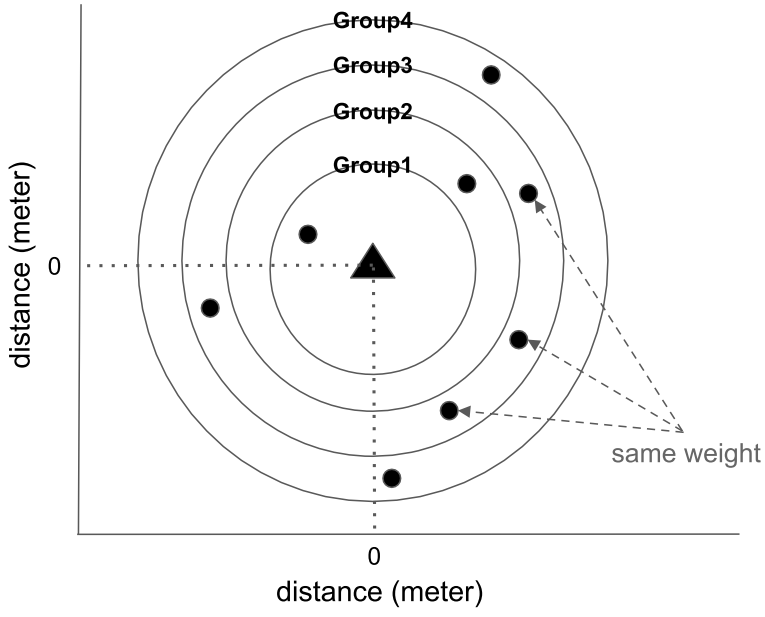

Fig. 5 Group by distance between terminal and gateway

Algorithm 2 Grouping Algorithm

```
i ← 0
j ← 0
n ← 30
k ← 5
radius ← 750
gateway ← (x, y)
nodeList[] ← (NodeID, x, y, 0)
groups[] ← 0
while i < k do
  groups[i] ← (i, i*radius/k, (i+1)*(radius/k))
  i ← i+1
end while
while j < n do
  distance ← norm2(gateway, nodeList[i])
  i← 0
  while i < k do
    if groups[i].min ≤ distances < groups[i].max then
      nodeList[j].group = groups[i].gid
      Break
    end if
    i ← i+1
    if i = k then
      nodeList[j].group = groups[i].gid
    end if
  end while
  j ← j+1
end while
```

the neural network weights. This procedure is repeated until the maximum simulation time.

And as shown in Fig. 3, we split one episode into several smaller episodes, and then, we load and learn only one reinforcement learning agent into GPU memory for each episode. The reason for designing the learning method is as follows. Firstly, if the number of terminals increases, it is impossible to put all reinforcement learning in the GPU memory. Second, since only one reinforcement learning agent learns for each divided episode and ADR is performed, the entire system of LoRa is operated during reinforcement learning except the reinforcement learning agent.

4 Experiments and results

4.1 Test setup

In the previous work, we implemented LoRa simulation in OPNET simulator [21]. However, to integrate with the RL tested implemented using Keras which is the deep learning platform written in Python code, we used other published LoRa simulator [22]. In our

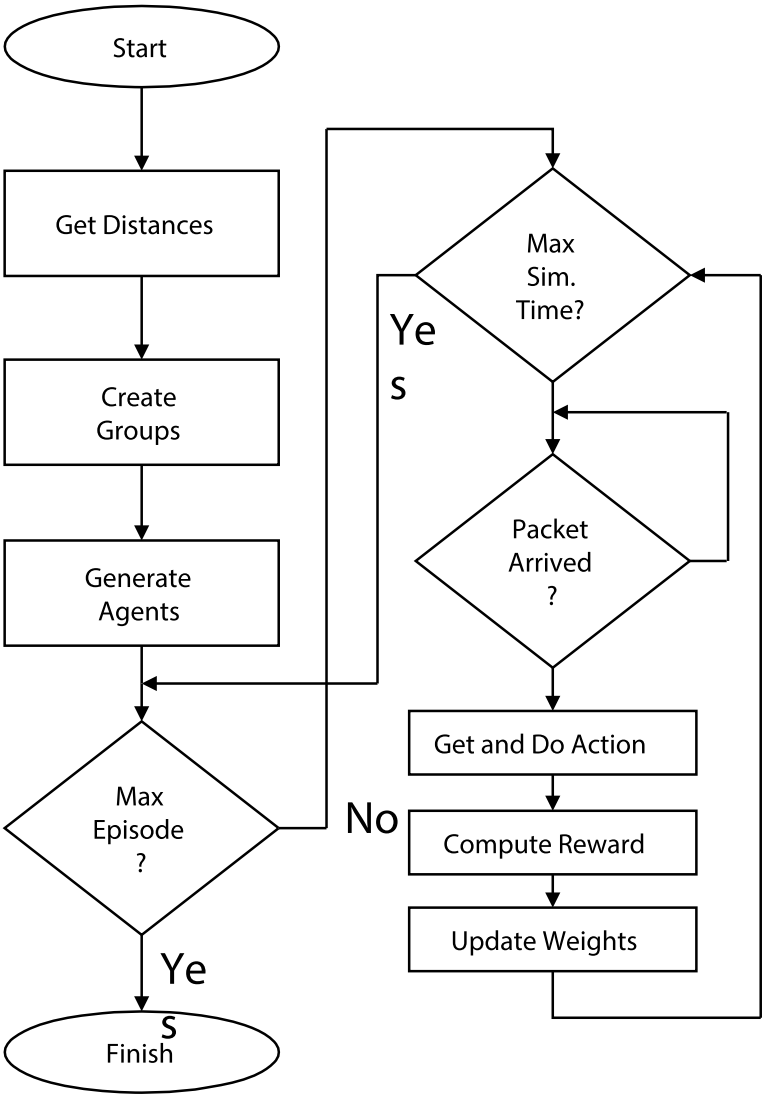

Fig. 6 Learning procedure

simulation, LoRaWAN is composed of 30 LoRa terminal nodes and 1 gateway, and suburbs are assumed. All LoRa terminals are configured as class A, and the payload of the packet is fixed to 10 bytes. In addition, one packet is generated every 30 min at each node, and more packets are generated than before. In addition, the path loss and the energy consumption of the terminal node are set as in the research by Callebaut et al. [22].

Basically, as explained in the system model section, the learning agent is generated in proportion to the number of terminals, but if there are many terminals, it is impossible to create agents for all terminals, so the terminals are grouped according to their respective distances and one agent was created for each group. Figure 7 shows the test scenario with the group applied. The triangle in the center represents the gateway, and the circles around it represent the nodes. In the test scenario, the sector was divided into five groups, with six nodes in each group.

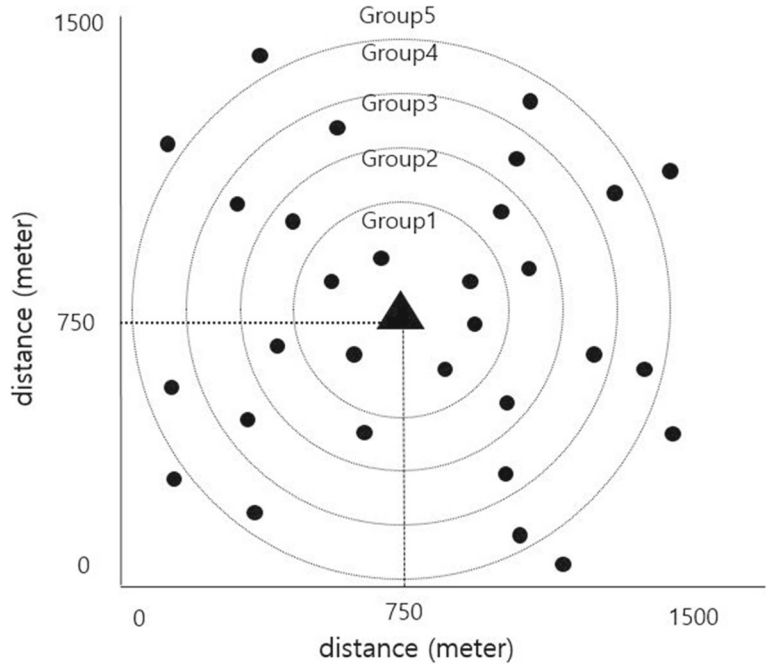

Fig. 7 LoRa terminal node layout

Tables 2 and 3 are the LoRa simulation parameters and the reinforcement learning parameters, respectively. The simulation environment of the proposed method has a topology of 1500 × 1500 size, 5 groups are created, and a total of 30 nodes were divided and arranged so that 6 nodes could belong to one group. LoRA gateway is located at the center of the topology. Theoretically, the spreading factor can be set from a minimum of 6 to a maximum of 12, but because there are many products that do not support SF 6, this paper tested the values from 7 to 12. The transmission power is set to a total of five steps up to 14 dbm according to the specification [2] as the uplink transmission power of the node. The data transmission rate is a value obtained by converting the frequency of data generation to bps, and a slow environment is assumed. The simulation time is 150 h in total. Since a low transmission rate is assumed, the simulation time is set long to collect sufficient information. Replay buffer size is generally determined according to the size of the sample that occurs in the learning process. In this simulation, it is assumed that the frequency of occurrence of the sample is low. So, if the replay buffer is large, learning will not proceed. In the same context, the minibatch is also set to a small enough value,

Table 2 LoRa simulation parameters [22]

Parameters	Values
Topology size (m^2)	1500 × 1500
Spreading factors	7–12
Transmission power (dBm)	2, 5, 8, 11, 14
Data transmission rate (bps)	0.02
Number of LoRa nodes	30
Number of LoRa gateway	1
Number of groups	5
Simulation time (hours)	150

Table 3 Hyperparameters for RL

Hyperparameters	Values
Replay buffer size	50
Minibatch size	8
Activation function	Relu
Optimizer	Adam
Learning rate	0.01
Epsilon decay	0.99
Hidden layers	30×58
Learning time (hours)	10

because a part of the replay buffer is sampled and trained, but if it is set to a size similar to the replay buffer, the learning may be unevenly biased. As the activation function, a non-linear function relu was used. Because it is an environment that does not need to normalize the value, we chose relu, which treats only negative numbers as 0 and outputs the rest as it is. The learning rate was set to 0.01, and the epsilon decay was set to 0.99 to set the learning to converge quickly because the nodes were grouped and divided into multiple agents, so it was determined that the learning difficulty of each agent was not high. The total learning time of the proposed method is 10 h, and the Adam optimizer is used.

4.2 Experimental results

We confirmed the difference between the proposed model and ADR. In Algorithm 1, ADR collects approximately 20 packets and then determines SF and TP based on the collected SNR. Depending on where the reference point is set in the SF and TP determination, it can be classified into SNR(MAX) and SNR(MIN). SNR(MAX) determines SF and TP based on the highest SNR among the collected packets, and SNR(MIN) classifies SF and TP based on the lowest SNR among the collected packets. There are various ADR methods depending on how the packet is classified based on the collected packets. So, we compare SNR(MAX) and SNR(MIN) with the reinforcement learning method proposed in the paper. In the simulation, 10 h of simulation time, the results of ADR(MAX), ADR(MIN), and the proposed model were examined.

When ADR(MAX) is applied, the SF and TP distributions selected by all LoRa terminals are shown in Figs. 8 and 9. Looking at Fig. 8, it can be seen that LoRa terminals adopting ADR(MAX) mainly selected SF7. SF8, SF9, SF10, and SF11 are also about 400 times, but SF7 is selected about 800 times. This is because SF is the first priority when designing ADR. As SF increases, symbols are designed to use lower values of SF because they consume more energy for data rate and encoding. In the case of TP distribution, ADR(MAX) was mainly selected as 2 dBm and 14 dBm. The distribution of SF and TP with ADR(MIN) is more extreme than ADR(MAX). All packets were selected SF7 and 14 dBm. ADR(MIN) can be seen that the focus is on obtaining more data rate at the expense of packet reception probability. The proposed method in this paper analyzes all LoRa network conditions and decides SF and TP. As shown in Fig. 9, the proposed method shows that the absolute number of packets is smaller than the ADR and the SF value is concentrated on a large value. The reason is that because the SF value is large, the length of the symbol

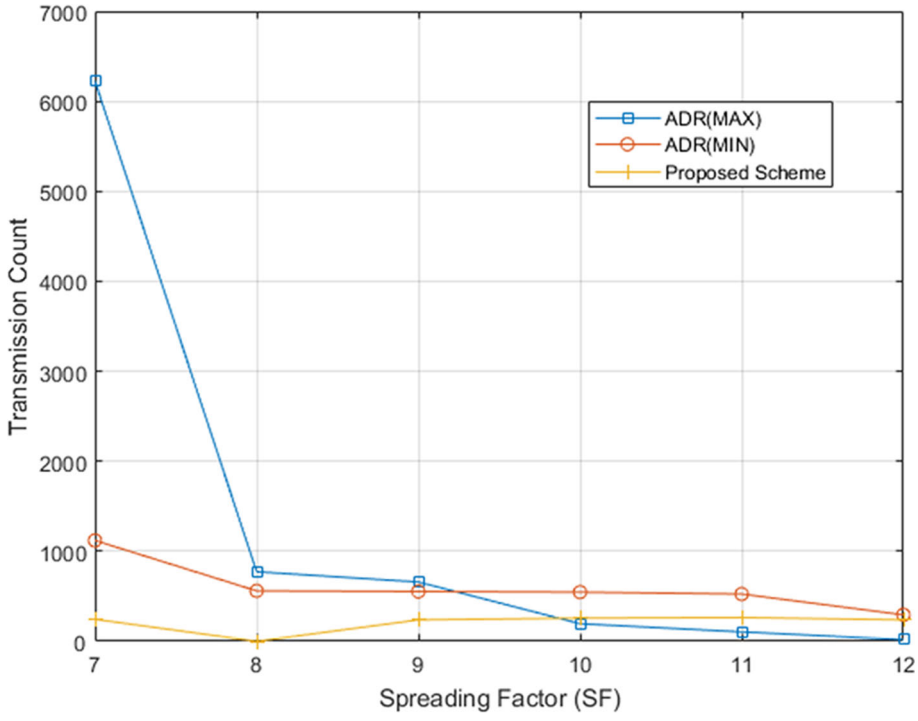

Fig. 8 SF distribution of LoRa terminal nodes

required for transmission is increased and the time on air is increased, so the number of transmitted packets is absolutely smaller than that of the ADR.

Next is the simulation statistics. This can be seen in Table 4, which is divided into information measured at the terminal and information measured at the gateway. Statistical information is shown as an average value divided by the number of terminals. The information obtained by the terminal includes total packets issued by the terminal and total energy consumed by the terminal. The information obtained from the gateway includes non-overlapping number of received packets (unique packets), number of weak packets in uplink transmission (UL weak packets), and number of packets lost during downlink transmission (DL packets lost).

The actual throughput in Table 4 is unique packets. In order of high throughput, ADR(MAX) received the most packets with 33.13, followed by ADR(MIN) with 18.97 and 11.84. However, ADR actually issued a lot of packets and the ratio of UL weak packets and DL packets lost is relatively higher than the proposed model. When comparing the energy consumed at the terminal node, the proposed model consumed the least energy at 1497.94 (mJ), followed by ADR(MAX) at 13,577.30 (mJ) and ADR(MIN) at 22,268.20 (mJ).

We are interested in throughput and energy consumption. Thus, we compared how our model differs from the conventional methods in terms of throughput and energy consumption. We compared ADR(MAX) and ADR(MIN) with the energy consumption compared to throughput. The proposed method is 2.8 times and 1.60 times less than ADR(MAX) and ADR(MIN) respectively in throughput and 3.24 times and 9.28 times

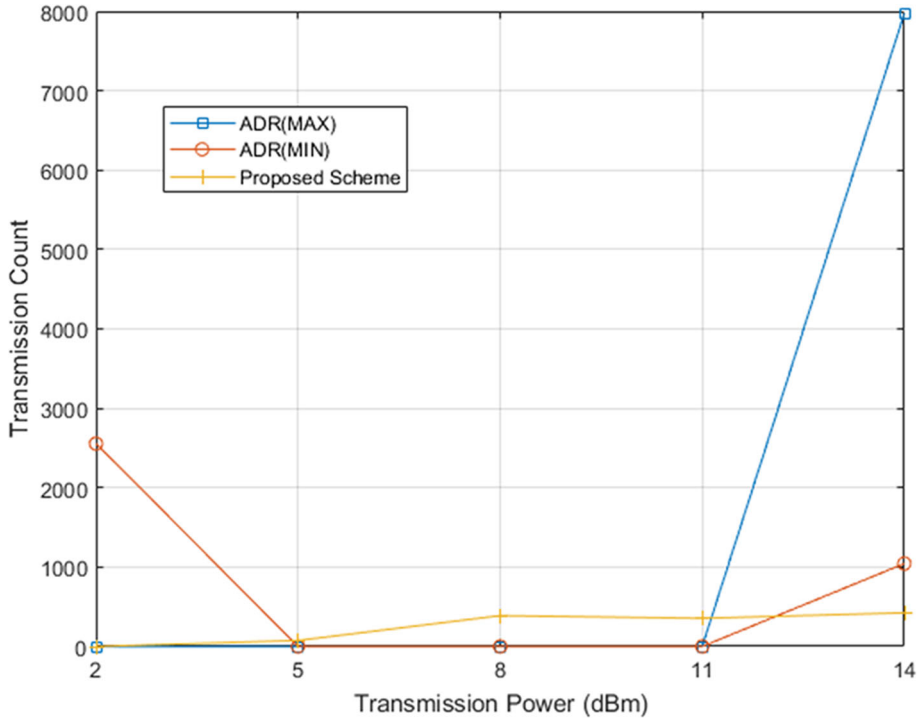

Fig. 9 TP distribution of LoRa terminal nodes

better in terms of energy consumption. Arithmetic calculations show that the proposed method is 1.15 times better than ADR(MAX) and 5.8 times better than ADR(MIN).

Figure 10 shows the packet arrival rates of uplink and downlink packets. The proposed method is better than the other two methods by performing the learning considering the energy consumption so that it can transmit reliably with lower power than the other two methods. Specifically, in the case of the downlink, almost all messages were delivered to the node so that the proposed method has an arrival rate close to 1. In the case of class A protocol, downlink is performed twice after one uplink. In the case of the proposed method, it was already confirmed through Fig. 8 that the SF was set very conservatively, so it can be deduced that most of the first downlink succeeded. In addition, since the transmission power of all packets can always be set to the maximum value in the gateway, the influence by the transmission power value is very limited. In the case of ADR(MAX) and ADR(MIN), ADR(MIN), which sets the SF more conservatively, showed a slightly higher arrival rate than ADR(MAX), but it was less than the proposed method. In the case of the uplink, there is a risk of collision because the node tries to transmit at any time, and thus, the transmission power has a great influence. In this context, ADR(MIN) has

Table 4 Simulation statistics result

	Terminal total packets	Terminal total energy (mJ)	Gateway unique packets	Gateway UL weak packets	Gateway DL packets lost
ADR(MAX)	265.84	13,577.30	33.13	169.80	47.97
ADR(MIN)	119.97	22,268.20	18.97	33.87	28.27
Proposed method	41.40	1497.94	11.84	26.57	3.64

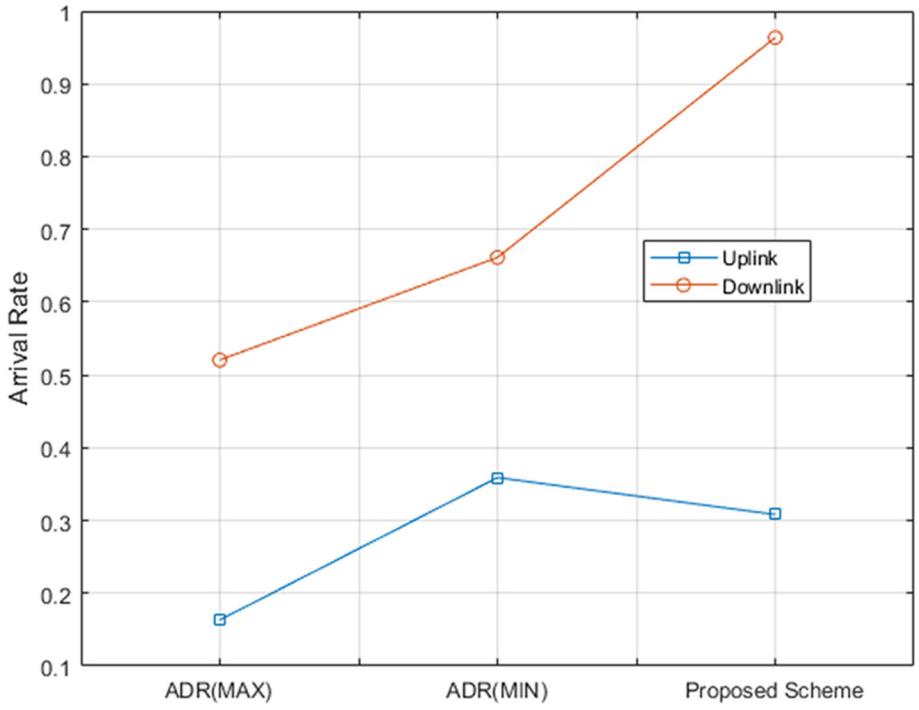

Fig. 10 Packet arrival rate

a slightly higher performance than the proposed method because it attempts uplink with high SF and high transmission power.

Figure 11 shows the energy efficiency per packet. Since the proposed method derives the transmission power and spreading factor by setting the reward considering the energy efficiency compared to the transmission time, it can be confirmed that the energy efficiency is better than other methods. In the case of ADR(MIN), the arrival rate and throughput are guaranteed to a certain level, but it can be seen that the energy efficiency is very poor; in the case of ADR(MAX), it shows the observed energy efficiency. Therefore, in an environment that needs to be used at a low speed for a very long period of time, the proposed method is suitable. ADR(MIN) may be effective in an environment where energy consumption is not significantly considered, and ADR(MAX) may be effective in an environment in which transmission failure is tolerated.

5 Conclusions

We applied reinforcement learning to allocate network resources in LoRaWAN. In the case of ADR, which is the existing method, the optimal value between the LoRa terminal and the gateway is considered, while the proposed method using reinforcement learning has centrally distributed network resources considering the network conditions of all the terminal nodes. As a result, it was arithmetic improved about 15% more than ADR(MAX).

But the method we have presented is not good. Our method shows good performance in terms of energy consumption but decreases in throughput. And also while the existing method works with only 20 sample data, our method took a lot of time to learn. The results presented in Section 4 were acquired by learning 10 h in real time and 150 h in simulation time. So, if we want to get the same results in the real world, it may need 150 h

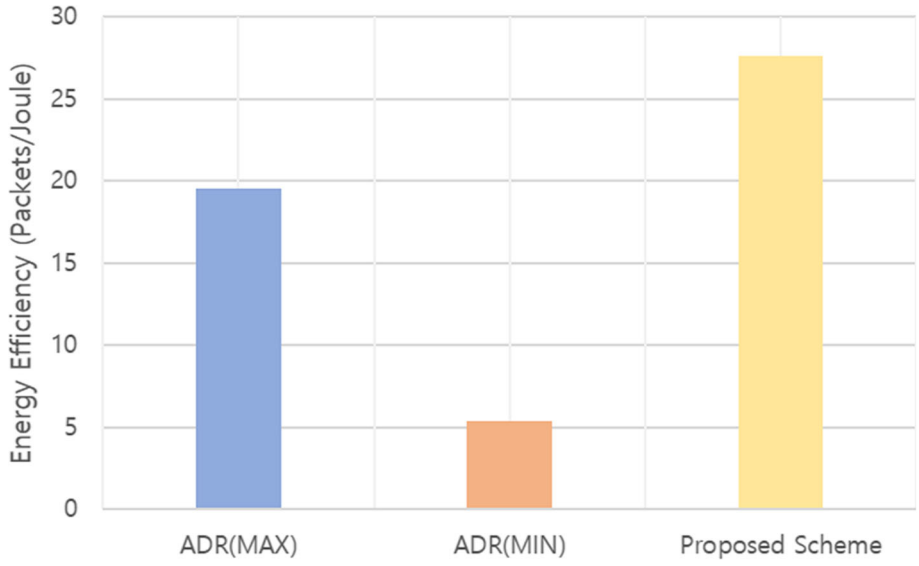

Fig. 11 Energy efficiency

of operation time to gather the same number of sample experiences. We can decrease this operation time by pre-learning with generated samples, but the necessary amount of samples are very huge if the number of candidate LoRa nodes is not fixed. Since our RL agent works on the LoRa gateway, the gateway must be more powerful than the current gateway device, and as the results of the ADR procedure must be reported to the LoRa network server to process the LoRaWAN procedure, it is necessary for researches about the protocol-level approach to extend the cooperation of the gateway and the network server.

Although there are some disadvantages, the proposed method can execute various scenarios depending on what information is given to the state.

In the case of sufficient learning, the proposed method is able to derive the result according to a wider range of trends in a short time, unlike the existing ADR that only sees the last 20 data. In addition, it is possible to ensure that the ADR operates effectively even in an environment where the data generation rate is very low and the transmission time is very long. In this paper, learning was performed in terms of energy efficiency compared to transmission time, but it has the potential to develop according to the throughput and transmission success rate depending on the setting of the reward.

This study complements the abovementioned shortcomings as an initial study, and it will be able to meet the needs of various network environments required by the future society. This study has this potential, so we think it is worth studying.

Abbreviations

LPWA: Low power wide area; IoT: Internet of Things; CSS: Chirp spread spectrum; MAC: Media access control; CSMA: Carrier sense multiple access; ADR: Adaptive data rate; SF: Spreading factor; TP: Transmission power; MDP: Markov decision process; CRL: Centralized reinforcement learning; DMRL: Distributed multi-agent reinforcement learning;

Acknowledgements

This work was supported partly by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2020-0-00107, Development of the technology to automate the recommendations for big data analytic models that define data characteristics and problems), and partly by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1009894).

Authors' contributions
GBP is in charge of the major idea, theoretical analysis, and experimental simulation and wrote the manuscript. WYL is in charge of part of the idea, theoretical analysis, manuscript writing, and whole revision process. IWJ is in charge of part of the theoretical analysis and given suggestions to the manuscript writing. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Data sharing is not applicable to this article as no data sets were generated or analyzed during the current study.

Competing interests
The authors declare that they have no competing interests.

Received: 3 January 2020 Accepted: 14 August 2020
Published online: 10 September 2020

References

1. U. Raza, P. Kulkarni, M. Sooriyabandara, Low power wide area networks: an overview. IEEE Commun. Surv. Tutorials. **12**(2), 855–873 (2017)
2. Semtech Corporation, *AN1200.22 LoRa Modulation Basics, Application note Revision 2(AN1200.22)*. (Semtech Corporation, 2015), pp. 1–26
3. A. Augustin, et al., A study of LoRa: long range & low power networks for the internet of things. Sensors. **16**(9), 1466 (2016)
4. V. Hauser, T. Hegr, *Proposal of adaptive data rate algorithm for LoRaWAN-based infrastructure, 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*. (IEEE, 2017), pp. 85–90
5. B. Reynders, W. Meert, S. Pollin, *Power and spreading factor control in low power wide area networks, 2017 IEEE International Conference on Communications (ICC)*. (IEEE, 2017), pp. 1–6
6. K. Abdelfadeel, C. Cionca, D. Pesch, *Poster: A Fair Adaptive Data Rate Algorithm for LoRaWAN, Proceedings of the 2018 International Conference on Embedded Wireless Systems and Networks (EWSN Š18)*. (Junction Publishing, 2018), pp. 169–170
7. Kim, et al., Adaptive data rate control in low power wide area networks for long range IoT services. J. Comput. Sci. **22**, 171–178 (2017)
8. Kim, et al., Contention-aware adaptive data rate for throughput optimization in LoRaWAN. Sensors. **18**(6), 1716 (2018)
9. M. El-Aasser, T. Elshabrawy, M. Ashour, *Joint spreading factor and coding rate assignment in LoRaWAN networks, 2018 IEEE Global Conference on Internet of Things (GCIoT)*. (IEEE, 2018), pp. 1–7
10. Cuomo, et al., Towards traffic-oriented spreading factor allocations in LoRaWAN systems, 2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net) (IEEE, 2018), pp. 1–8
11. Q. Zhou, J. Xing, L. Hou, R. Xu, K. Zheng, *A novel rate and channel control scheme based on data extraction rate for LoRa networks, 2019 IEEE Wireless Communications and Networking Conference (WCNC)*. (IEEE, 2019), pp. 1–6
12. D. Ta, K. Khawam, S. Lahoud, C. Adjih, S. Martin, LoRa-MAB: Toward an Intelligent Resource Allocation Approach for LoRaWAN, 2019 IEEE Global Communications Conference (GLOBECOM) (IEEE, 2019), pp. 1–6
13. R. S. Sutton, A. G. Barto, *Reinforcement learning: an introduction, Adaptive Computation and Machine Learning series (Textbook)*. (The MIT Press, 2014), pp. 1–352
14. V. Mnih, et al., Human-level control through deep reinforcement learning. Nature. **518**(7540), 529 (2015)
15. J. Foerster, I. A. Assael, N. de Freitas, S. Whiteson, *Learning to communicate with deep multi-agent reinforcement learning, Advances in Neural Information Processing Systems*. (The MIT Press, 2016), pp. 2137–2145
16. A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, Multiagent cooperation and competition with deep reinforcement learning. PLoS ONE. **12**(4), 1–15 (2017)
17. J. K. Gupta, M. Egorov, M. Kochenderfer, in *International Conference on Autonomous Agents and Multiagent Systems*, Cooperative multi-agent control using deep reinforcement learning (Springer, Cham, 2017)
18. F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini, P. Pisani, *EXPLoRa: EXtending the Performance of LoRa by suitable spreading factor allocations, 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. (IEEE, 2017), pp. 1–8
19. M. Asad Ullah, J. Iqbal, A. Hoeller, R. D. Souza, H. Alves, K-means spreading factor allocation for large-scale LoRa networks. Sensors. **19**(21), 1–19 (2019)
20. G. Ottoy, et al., LoRaWAN EFM32 (2017). https://github.com/DRAMCO/LoRaWAN_EFM32. DRAMCO, Github
21. A. D. Jun, S. Hong, W. Lee, K. Lee, I. Joe, *Modeling and simulation of LoRa in OPNET, Advanced Multimedia and Ubiquitous Engineering*. (Springer, 2017), pp. 551–559
22. G. Callebaut, G. Ottoy, L. Van der Perre, *Cross-layer framework and optimization for efficient use of the energy budget of IoT nodes, 2019 IEEE Wireless Communications and Networking Conference (WCNC)*. (IEEE, Marrakesh, 2019), pp. 1–6

Publisher's Note