Optimal Data Scheduling and Admission Control for Backscatter Sensor Networks

Dinh Thai Hoang, Dusit Niyato, Ping Wang, Dong In Kim, and Long Bao Le

Abstract

This paper studies the data scheduling and admission control problem for a backscatter sensor network (BSN). In the network, instead of initiating their own transmissions, the sensors can send their data to the gateway just by switching their antenna impedance and reflecting the received RF signals. As such, we can reduce remarkably the complexity, the power consumption, and the implementation cost of sensor nodes. Different sensors may have different functions, and data collected from each sensor may also have different status, e.g., urgent or normal, thus we need to take these factors into account. Therefore, in this paper, we first introduce a system model together with a mechanism in order to address the data collection and scheduling problem in the BSN. We then propose an optimization solution based on the Markov decision process framework and a reinforcement learning algorithm based on the linear function approximation method, with the aim of finding the optimal data collection policy for the gateway. Through simulation results, we not only show the efficiency of the proposed solution compared with other baseline policies, but also present the analysis for data admission control policy under different classes of sensors as well as different types of data.

Index Terms

Backscatter communications, passive RFID, sensor networks, MDPs, Q-learning, SARSA with linear function approximation.

I. Introduction

Radio frequency identification (RFID) and wireless sensor networks (WSNs) are two important technologies which have a lot of practical applications in current and future wireless communication systems. While RFID systems are used to detect and identify tags attached to objects, WSNs are employed to provide information about monitored objects. Each technology has its own advantages as well as disadvantages, and thus if such two technologies can be combined together, they can complement each other to improve the overall functionality and capacity significantly [1]. As a result, a new type of wireless networks has been introduced recently, called

backscatter sensor network (BSN), which integrates backscatter communication techniques used in RFID systems into WSNs. In BSNs, sensors first collect data through sensing processes. Then, the sensors send their data to the gateway node, i.e., the sink node, just by reflecting RF signals received from the gateway, thereby minimizing energy consumption, complexity, as well as the implementation cost of the sensors.

Although BSNs are very useful and they have many applications in practice such as environment monitoring [2], body area sensor networks [3], vehicle networking [4], and real-time tracking [5], the networks have to face an important problem, that is how to collect data from sensors. Different from conventional WSNs where sensor nodes are assumed to be able to transmit data directly to the gateway node, in BSNs, data from sensors is collected based on the passive backscattering technique [6]. In particular, in the networks, the gateway first transfers RF signals to the sensors. The sensor receives and reflects the signals with modulated information using the same antenna. The reflected signals are then received by the gateway and processed to extract the modulated information. Different sensors may have different priorities, and different collected data may have different status, e.g., urgent or normal. Therefore, we need to find efficient solutions to the data collection problem in BSNs such that the network performance is maximized, while the important packet loss is minimized.

In this paper, we first introduce a system model together with a data scheduling mechanism for the BSN which take priorities of sensors as well as the status of packets into considerations. We then formulate an optimization problem of the gateway as a Markov decision process (MDP). The optimization aims to find an optimal data transmission policy that minimizes the cost defined in terms of the average packet delay (which also mitigates the average packet loss of the system). We then apply the State-Action-Reward-State-Action (SARSA) learning algorithm to obtain the optimal policy. The learning algorithm overcomes the curse-of-model issue when some parameters may be unknown in advance and helps the gateway make optimal decisions in an online fashion. After that, we investigate the use of approximation functions for the learning algorithm, called SARSA with linear function approximation. This algorithm addresses the curse-of-dimensionality when the state space of the problem becomes large. We also analytically prove the convergence of the learning algorithm.

The paper is organized as follows. In Section II, we give an overview of related works in BSNs. Section III describes the system model together with the proposed data scheduling strategy. We then present the MDP formulation for the optimization problem in BSNs in Section IV. In

Section V, we introduce learning algorithms together with using function approximations for MDPs. Performance evaluation results are presented in Section VI and conclusions are presented in Section VII.

II. Related Work and Main Contributions

In this section, we first give a brief overview about BSN and some related work. We then highlight main contributions and the novelty of the paper.

A. Backscatter Sensor Networks

Backscatter sensor network (BSN) is a new kind of wireless communication networks which allows wireless sensor nodes to transmit data to a gateway/sink node by using the backscatter communication technique. In particular, a BSN consists of multiple wireless sensor nodes which want to transmit data to a gateway node as illustrated in Fig. 1 (a). Different from the conventional wireless sensor networks (CWSNs) where sensors have to transmit data directly to the gateway, in BSNs when the gateway wants to collect data from a sensor, it will transmit RF signals to that sensor through the transmitter antenna (Tx). After receiving such RF signals, the sensor will reflect these signals back to the gateway by simply switching on/off the RF impedance circuit which is well connected to its antenna as shown in Fig. 1 (b). The switch circuit of the sensor is controlled by a controller which generates a sequence of one and zero bits according to the current data of the sensor. For example, if the sensor wants to transmit a bit one, the controller will switch the RF impedance circuit to mode "On", i.e., the sensor will reflect back the received signals. After that, the gateway will use the receiver antenna (Rx) to receive the modulated signals from the sensor. Through a demodulator and an analog-digital converter, the gateway can extract the useful information from that sensor.

B. Related Work

Integrating backscatter communication techniques into CWSNs will bring many benefits for BSNs, e.g., reducing the complexity, power consumption, and implementation cost for sensor nodes, because the BSNs can take advantages of both sensor networks and backscatter techniques. However, they also have to face some inherited limitations as highlighted in [1], [9], [10]. Therefore, some research works studied solutions with the aim of improving the performance as well as implementing BSNs in practice.

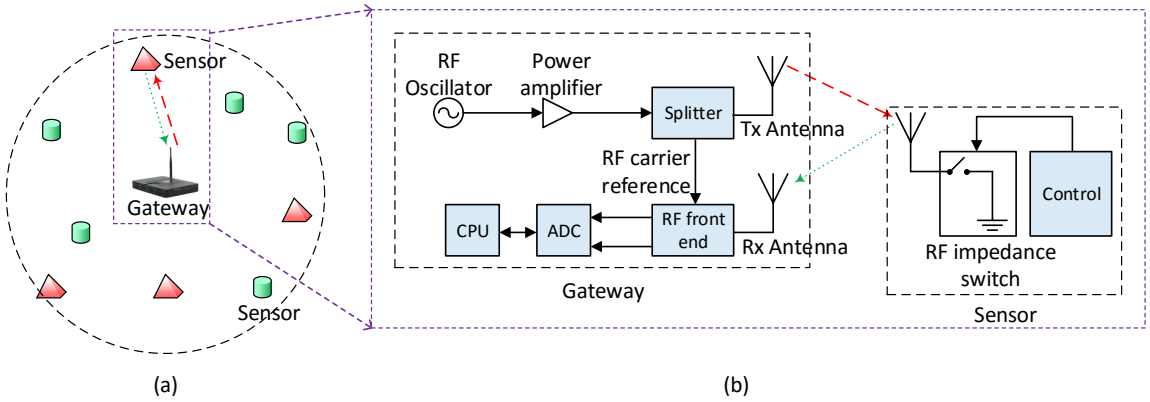


Fig. 1. (a) A typical model of backscatter sensor networks and (b) a general architecture of backscatter communication.

1) Implementation: In [14], an experimental system was developed in practice for a BSN in an indoor environment. In this test, the authors implemented the BSN including multiple sensors transmitting data to a gateway at a range of approximately 15 meters. Fig. 1 (b) shows block diagrams for the BSN implemented in [14] where the sensors are designed simply to minimize its energy consumption with only one RF impedance switch circuit which is connected to an antenna and controlled by a controller. Meanwhile, the gateway is equipped with a TX antenna to transmit RF signals, and an Rx antenna to receive the backscattered signals. After receiving the modulated signals, the gateway will use a homodyne detector to detect frequency-modulated radiation and an ADC to decode information. In addition, to address the multi-access problem in the BSN, the authors adopted minimum-shift keying (MSK) (a special case of frequency-shift keying) to modulate signals for the sensors. With MSK, signals from sensors will be backscattered at different subcarrier frequencies to avoid collision (interference) at the gateway. Real experiments in [14] showed that, the sensors consume just 5 mW to transmit data simultaneously at the bit rate of 10 bps to the gateway. In addition, the deployment problem of BSNs was analytically investigated through studying the communication coverage problem in [15]. It was pointed out that the optimal number of required RF-sources depends on the event field, frequency, and transmission power. This analysis is especially useful in designing optimal deployment strategies for sensors and RF sources in BSNs in practice.

2) Energy management: In BSNs, sensors do not need to equip batteries for data transmission, but they still need energy to remain their operations such as sensing and processing. To do so, there were some proposed solutions using RF energy harvesting techniques to help sensor nodes remain their operations without causing negative impacts to backscatter communication process

as well as extra implementation costs. In particular, when the gateway transmits RF signals, the sensor nodes can backscatter these signals to transmit data or receive it and apply RF energy harvesting techniques to harvest energy. Therefore, the authors in [16] introduced an optimal energy scheduling for sensor nodes through using Markov chain model. The proposed scheduling allows sensor nodes to balance the time for backscattering data and harvesting energy. Thus, the sensors can perform either communicating with the gateway or harvesting energy to supply for their operations. In [17], the authors proposed a solution which allows sensors to perform backscatter communications with the gateway and harvest energy simultaneously. In particular, it is assumed that sensors can receive signals from two different sources at two different frequencies, and hence they can perform separately backscattering signals and harvesting energy concurrently at two different frequencies. In [19], a new design of wireless backscatter platform was introduced, which minimizes the power consumption for sensor nodes, while enabling bandwidth to scale up to support data rates of hundreds of Kbps. The core idea of this platform is to eliminate the overheads of sensing, data handling, and communications in BSNs, thereby minimizing the whole system power consumption and increasing the data rates.

3) Communication improvement: Different from RFID systems where the communication range between tags and the tag reader is just within few meters and the bit rate is not really high, in BSNs the distance between sensors and the gateway is often longer and the bit rate is required higher than those of RFID systems. As a result, there were some solutions proposed to improve the communication capacity of BSNs. In [18], the authors proposed a solution using the multisine waveform (instead of a pure sinusoid) to enhance the communication range for BSNs without a need of increasing the transmission power at the gateway. The paper showed that the shape of the signal used to power the sensor nodes has a significant impact to the operating distance, and thus if we can set up the RF source to radiate a multisine waveform, then the communication range can be extended noticeably (with the same transmission power). Different from [18], the authors in [20] proposed a novel coding mechanism to achieve a long communication range for ambient backscatter systems. The key idea of this coding mechanism is using longer chip sequences which can increase the signal-to-noise ratio for the transmitter-receiver pair significantly, thereby enabling long-range communication for backscatter sensor systems. In [21], the authors studied how a backscatter link layer should be optimized for data transfer from sensor devices. Specifically, the authors in [21] designed a high link layer framework, named BLINK, for backscatter communications which can reduce the channel

probing overhead, select the optimal bitrate for data transfer, and optimize the use of channel for communications. The experimental results then showed that BLINK can achieve up to 3 times greater goodput than those of other mechanisms.

Recently, a new communication architecture has been introduced, namely Bistatic Scatter Radio [11], [12], [13], with the aim to extend the communication range for BSNs, while minimizing the implementation cost for sensors. In the bistatic scatter networks, RF-signal emitter is detached from the receiver, and the sensor acts as a signal modulator. When the sensor receives signals from the emitter, it will scatter these signals to the receiver with modulated information. Then, based on signals received from the emitter and the sensor, the receiver can decode and obtain the useful information. Through experimental results, the authors showed in [12] that the tag-to-reader communication range can be extended up to 150 meters with transmission power of just 20 milliwatts at the emitter.

4) Multiple access: Different from CWSNs where sensors actively transmit data to the gateway, in BSNs data from all sensor nodes is collected through energy transfer process of the gateway. When the gateway transmits signals to the sensors, these nodes will reflect these signals back to the gateway to transmit data. Therefore, how to control interference/collision among received modulated signals from different sensors at the gateway is a challenge. In [14], MSK modulation technique was adopted for sensors at which the sensors will be indicated to backscatter signals at different frequencies to avoid the interference at the gateway when the gateway demodulates received signals. However, this method requires that the subcarrier frequency of each sensor node must be sufficiently different from all others to keep the interference minimal. Therefore, the authors in [26] alleviated the interference at the gateway by analyzing, and mathematically quantifying the impact of aggregate bandwidth savings for BSNs. Nevertheless, the bit rate of this solution is still very low (10 bps) and it has to face some problems such as multi-frequency, and high bit-error rate. Another approach which adopts CSMA/CA medium access control mechanism was presented in [27], but the bit rate is still not really high (approximately 106 packets/second). Alternatively, TDMA was also considered to be adopted, and it seems to be a promising solution for multi-access problem in BSNs when the bit rate of backscatter communication between the gateway and the sensors can achieve up to many hundreds of Kbps [19]. In addition, a bulk transmission protocol for RFID-scale sensors was developed in [22] which maximizes channel utilization and minimizes energy loss and collisions. Through simulation results, the authors demonstrated that the proposed protocol

can achieve 4.5x and 9.2x more goodput than that of the EPC Gen 2 protocol when there are three and five tags transferring data concurrently, respectively.

Recently, some concurrent backscatter transmission mechanisms, e.g., [23], [24], and [25], have been introduced to improve the network throughput for backscatter sensor systems. The key point of these mechanisms is to design efficient decoding schemes to support high-speed concurrent backscatter communications. However, concurrent backscatter transmission mechanisms often suffer from complex decoding processes which consume much more energy than those of single-tag backscatter transmission mechanisms. For our considered system, the gateway is an energy-limited device which needs to utilize the harvested energy efficiently. Hence, single-tag backscatter transmission is more suitable.

C. The Novelty and Main Contributions

From all aforementioned works and others in the literature, there are some important problems which have not yet been considered and addressed in BSNs, and they are the motivations for this paper. Firstly, one of the most important characteristics of BSNs, which is different from RFID systems, is that data collected at the gateway can be the real-time data obtained through sensing processes of sensor nodes. Therefore, the gateway needs to take the urgency of data into consideration. Secondly, different sensors may perform different functions, and thus they may have different priorities in transmitting data to the gateway. Thirdly, in all existing works, it is always assumed that the gateway has an abundant energy source. Nevertheless, in practice the gateway may also be mobile which needs to harvest or receive energy from an external source, e.g., a charger, to sustain its operation. Therefore, the gateway needs to have efficient energy management policies to balance between the amount of harvested energy and consumed energy for the data collection. Fourthly, communication links between the sensors and the gateway are dependent on the states of wireless channels. At different time, the sensors' communication links may have different status, e.g., good or bad channel, which will impact the performance of BSNs[1], and hence the gateway must also take current channel status of sensors into account when it collects data.

This paper is dedicated to investigating and addressing the four major aforementioned challenges in BSNs. The main contributions therefore are threefold. Firstly, we present a system model using the MDP framework which takes all factors, i.e., the urgency of data, the priorities

[1]The impact of SNR to BER in BSNs can be found in [14].

of the sensors, the energy constraint of the gateway, and the channel states of communication links, into considerations. Secondly, we introduce a data scheduling scheme which can minimize not only the packet loss, but also the delay for important packets collected at sensors. Thirdly, we study the SARSA learning algorithm with linear function approximation to obtain the optimal policy for the gateway. This learning algorithm not only overcomes the curse-of-model issue when some parameters may be unknown in advance, but also addresses the curse-of-dimensionality when the state space of the optimization problem becomes too large. Additionally, we perform extensive experiments to demonstrate the efficiency of the proposed solution and provide the proof of convergence for the proposed learning algorithm. To the best of our knowledge, this is the first work which studies the optimization problem for a BSN with impacts from the sensors, the gateway, and the communication channels carefully considered.

III. System Model and Data Scheduling

In this section, we first describe the system model for the BSN. Then, we introduce a data scheduling strategy for the sensor nodes and a data collection mechanism for the gateway.

A. System Model

We consider a BSN as illustrated in Fig. 2. The network consists of N sensors communicating with a gateway using the passive backscattering communication technique. We assume that time is slotted and each sensor has a data queue to store data before transmitting it to the gateway. The data is classified by the sensor based on the importance of the information. We consider two types of data, namely, normal and important data. For example, for a room temperature sensor, if the average measured temperature per time unit is higher than 40 degree, the collected data will be classified as important data. Otherwise, it is a normal data. The maximum data queue size of sensor n ($n = 0, 1, \ldots, N$) is D_n, and the packet arrival probability of node n for normal data is p_n^{m} and for important data is p_n^{i}, respectively. Different sensor nodes may have different priorities, and we denote the priority factor of node n by σ_n. Similarly, we denote the weights of important and normal packets by δ^{i} and δ^{m}, respectively. The gateway is assumed to be able to harvest energy wirelessly from an external energy source, and the harvested energy will be stored in the energy storage of the gateway before it can be used to collect data from the sensors. The maximum battery size of the gateway is denoted by E units of energy.

We then consider the channel conditions between the sensors and the gateway, and between the gateway and the energy source. We refer to the former and the latter as "transmitting channel"

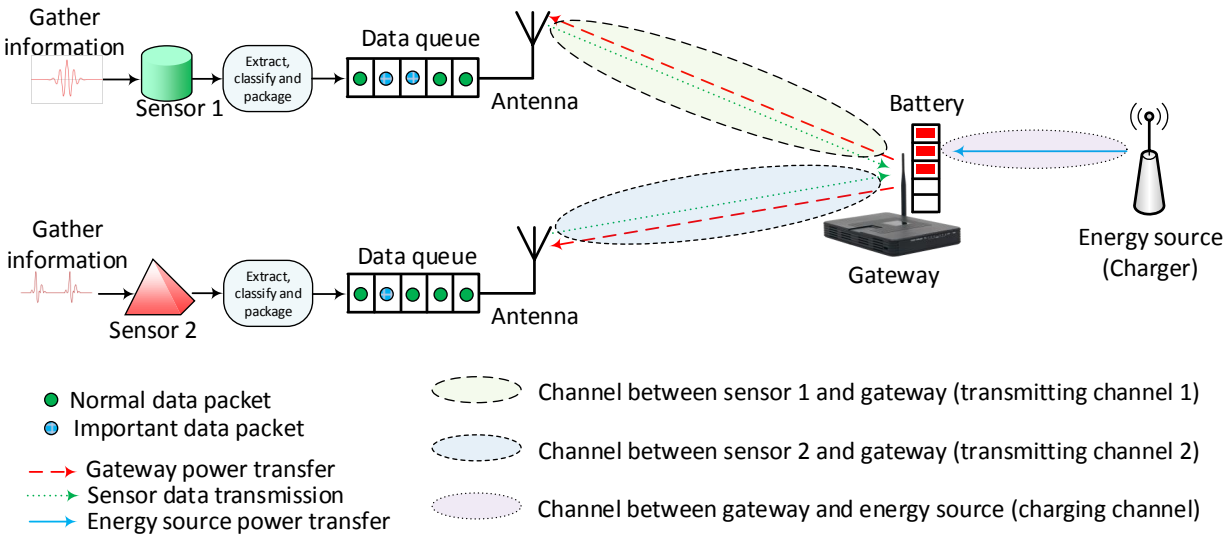


Fig. 2. The system model of a backscatter sensor network in the case with two sensors.

and "charging channel", respectively. The transmitting channel is used for data transmission through backscattering from the sensor to the gateway, while the charging channel is used for energy transfer from the energy source to the gateway. We denote $c_n(t)$ as the transmitting channel condition between the sensor n and the gateway at time slot t, and we consider two cases, i.e., good and bad channels in which $c_n(t) = 1$ and $c_n(t) = 0$, respectively. If the channel is good, the gateway is required to use e^{n} units of energy to collect data, while it is required to use e^{f} units of energy when the channel is bad. The probability that the transmitting channel is good or bad is independent over time slots and denoted as p^{g} and p^{b}, respectively. We denote $v(t)$ as the charging channel condition between the gateway and the energy source at time slot t. The probability that the charging channel is good, i.e., $v(t) = 1$, at each time slot is denoted as p^{c}, and when the channel is good, the gateway can harvest $e_{\mathrm{g}}(t)$ units of energy from the charger, i.e., the energy source, with the probability p^{e}. If the charging channel is bad, the gateway cannot harvest energy from the charger.

B. Data Scheduling

To optimize the performance for the considered BSN, in this section, we present strategies and mechanisms with the aims of helping the sensors to store data and the gateway to collect data efficiently.

1) Data scheduling at sensors: After data is collected and classified at a sensor, it will be packetized and stored in the data queue if the data queue of that sensor is not full as shown in

Fig. 3. If the data queue is full, and a normal packet arrives, the normal packet staying longest in the queue will be removed from the queue, and the new arriving normal packet will be put into the data queue. However, if the data queue is full with important packets, the new arriving normal packet will be dropped. Similarly, when a new important packet arrives and the data queue is full, the normal packet staying longest in the queue will be removed from the queue and the new arriving important packet will be put into the data queue. However, if a new important packet arrives and the queue is full with important packets, the important packet staying longest in the queue will be removed from the queue so that the new incoming important packet can be stored in the queue. Additionally, if the sensor is allowed to transmit a packet and the data queue has both normal and important packets, the sensor will transmit an important packet first. This queuing policy is designed to ensure that the important data has a higher priority, and the data will be transmitted to the gateway at the earliest.

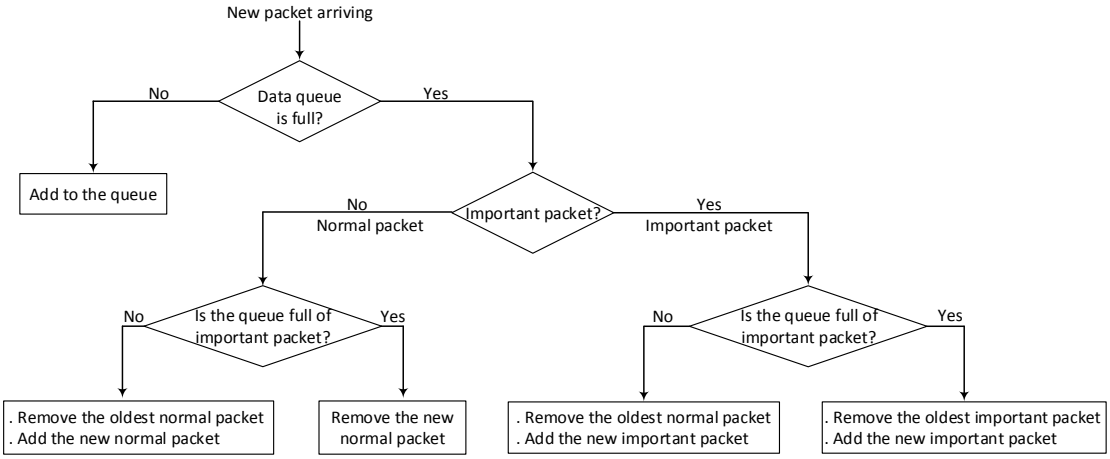


Fig. 3. Data scheduling at each sensor.

2) Energy transfer scheduling at the gateway: For the energy transfer and data transmission scheduling, there are three phases in each time slot as shown in Fig. 4. In the first phase, the gateway collects information from all sensors in the network based on the multiple-access procedures with anti-collision [6] (Chapter 7) which are widely used in practice. Specifically, the gateway first broadcasts power to sensor nodes and then these sensors use such energy to send back information to the gateway. To avoid collisions among data flows sent back to the gateway, the gateway can adopt one of the anti-collision multiple-access methods, e.g., TDMA. The information which is sent from a sensor to the gateway including the status of the data queues, and the type of sensor is very small in size and thus we can assume that the energy used

in the first phase by the gateway is negligible. Moreover, based on the reflected signals from sensors, the gateway can evaluate the quality of the current transmitting channels [7]. Then, from all information collected, in the second phase, the gateway will make a decision to sleep, to allow one of the sensors to transmit data, or to receive energy from the charger. In the third phase, the gateway performs the action determined in the second phase and updates its observed information.

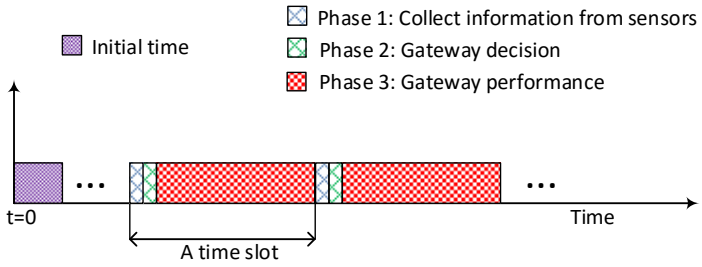


Fig. 4. Time slot structure.

Here, we adopt the aforementioned energy transfer strategy for the BSN for the following reasons. Firstly, in the considered system model, data obtained from sensors may have different states at each time slot, e.g., important or normal, and thus the gateway must use the first phase to examine the current status of the sensors in the entire network. Secondly, to avoid the collision/interference among sensor nodes, we allow only one sensor to backscatter signals to the gateway at a time. This method can achieve high backscatter communication bit rates compared with the MSK modulation technique as explained in Section II-B4. In addition, with the proposed scheme, when a sensor is communicating with the gateway, other sensors in the network can still receive signals from the gateway, and thus they can harvest this RF energy [8]. Here, we note that although sensors are not required to be equipped with a battery for data transmissions, they may still need a small energy storage, e.g., a capacitor, to store harvested energy. This energy is used for their operations, e.g., sensing and processing, as explained in Section II-B2.

IV. Optimization Formulation

In this section, we formulate the cost optimization for the scheduling problem of the considered BSN as a Markov decision process (MDP). The MDP is defined by a tuple $< \mathcal{S}, \mathcal{A}, r >$ where $\mathcal{S}$ is the system state space, $\mathcal{A}$ is the action space, and r is the reward or cost function.

A. State Space

The state space of the system is defined as follows:

$$\mathcal{S} \triangleq \mathcal{S}^{\dagger} \times \mathcal{E} \times \mathcal{V}, \tag{1}$$

where $\times$ is the Cartesian product, $\mathcal{S}^{\dagger} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \times \cdots \times \mathcal{S}_N$ is the composite state space of all sensors in the network, $\mathcal{E} = \{0, 1, \ldots, E\}$ represents the set of energy levels of the gateway, and $v \in \mathcal{V} = \{0, 1\}$ expresses the charging channel state between the gateway and the energy source. When $v = 1$, i.e., the charging channel is good, the gateway can harvest e_{g} units of energy from the energy source. When $v = 0$, i.e., the charging channel is bad, the gateway cannot harvest energy. $\mathcal{S}_n$ is the state space of sensor n that is defined by $\mathcal{S}_n = \mathcal{D}_n^{\mathrm{m}} \times \mathcal{D}_n^{\mathrm{i}} \times \mathcal{C}_n$, where $d_n^{\mathrm{m}} \in \mathcal{D}_n^{\mathrm{m}} = \{0, 1, \ldots, D_n\}$ is the number of normal packets, $d_n^{\mathrm{i}} \in \mathcal{D}_n^{\mathrm{i}} = \{0, 1, \ldots, D_n\}$ is the number of important packets in the data queue of node n ($d_n^{\mathrm{m}} + d_n^{\mathrm{i}} \leq D_n$), and $c_n \in \mathcal{C}_n = \{0, 1\}$ is the transmitting channel state between the sensor n and the gateway with 0 and 1 corresponding to the good and bad states, respectively. For $c_n = 0$ or $c_n = 1$, the gateway needs to use e^{n} or e^{f} ($e^{\mathrm{f}} > e^{\mathrm{n}}$) units of energy to transfer wireless energy and collect data from the sensor, respectively.

B. Action Space

The action space is defined by:

$$\mathcal{A} \triangleq \Big\{ a : a \in \{0, \mathbf{1}, 2\} \Big\}, \tag{2}$$

where

$$a = \begin{cases} 0, & \text{the gateway does nothing (sleep),} \\ \mathbf{1}, & \text{the gateway allows one of sensors to transmit data,} \\ 2, & \text{the gateway harvests energy from the charger.} \end{cases}$$

Note that $\mathbf{1} = (x_1, \ldots, x_n, \ldots, x_N)$ where $x_n \in \{0, 1\}$ and $\sum_{n=1}^{N} x_n = 1$. In other words, there is at most one sensor allowed to transmit data in a time slot. Additionally, the action space given the current state $s \in \mathcal{S}$ of the system, i.e., $\mathcal{A}_s$, comprises all possible actions that do not make

a transition to the state that is not allowed. We can express the action space $\mathcal{A}_s$ as follows:

$$\mathcal{A}_s = \begin{cases} \{0\}, & \text{if } v=0 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})=0, \\ & \text{OR if } v=0 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } \prod_{n=1}^{N} c_n=0 \text{ and } e<e^{\mathrm{n}}, \\ & \text{OR if } v=0 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } \prod_{n=1}^{N} c_n>0 \text{ and } e<e^{\mathrm{f}}, \\ & \text{OR if } v=0 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})(1-c_n)=0 \text{ and } e<e^{\mathrm{f}}, \\ & \text{OR if } v=1 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})=0 \text{ and } e=E, \\ \{\mathbf{1}\}, & \text{if } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } e=E\ , \\ \{2\}, & \text{if } v=1 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})=0 \text{ and } e<E\ , \\ & \text{OR if } v=1 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } \prod_{n=1}^{N} c_n=0 \text{ and } e<e^{\mathrm{n}}\ , \\ & \text{OR if } v=1 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } \prod_{n=1}^{N} c_n>0 \text{ and } e<e^{\mathrm{f}}\ , \\ & \text{OR if } v=1 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})(1-c_n)=0 \text{ and } e<e^{\mathrm{f}}, \\ \{0,\mathbf{1}\}, & \text{if } v=0 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } \prod_{n=1}^{N} c_n=0 \text{ and } e^{\mathrm{n}}\le e<E\ , \\ & \text{OR if } v=0 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } \prod_{n=1}^{N} c_n>0 \text{ and } e^{\mathrm{f}}\le e<E\ , \\ \{\mathbf{1},2\}, & \text{if } v=1 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } \prod_{n=1}^{N} c_n=0 \text{ and } e^{\mathrm{n}}\le e<E\ , \\ & \text{OR if } v=1 \text{ and } \sum_{n=1}^{N}(d_n^{\mathrm{m}}+d_n^{\mathrm{i}})>0 \text{ and } \prod_{n=1}^{N} c_n>0 \text{ and } e^{\mathrm{f}}\le e<E\ . \end{cases} \tag{3}$$

Here, we do not have two combinations, i.e., $\{0,2\}$ and $\{0,\mathbf{1},2\}$ because they are the same as the cases of $\{2\}$ and $\{\mathbf{1},2\}$, respectively. The reason is that when the charging channel is good and the energy storage is not full, the gateway will prefer to receive energy from the charger or to transfer energy to a sensor to collect data instead of doing nothing.

C. Reward Function

The gateway aims to minimize the weighted queue length corresponding to data of different types which also indirectly minimizes the average packet loss for the system. Therefore, we define the immediate cost function as follows:

$$\mathcal{C} = \sum_{n=1}^{N} \sigma_n(\delta^{\mathrm{m}} d_n^{\mathrm{m}} + \delta^{\mathrm{i}} d_n^{\mathrm{i}}), \tag{4}$$

where σ_n is the priority factor of node n. δ^{m} and δ^{i} are the weights of normal and important packets, respectively. d_n^{m} and d_n^{i} are the numbers of normal and important packets of node n, respectively. Therefore, the reward function at state s_t after action a_t is taken and the system transits to state s_{t+1} is defined as follows:

$$r(s_t, a_t, s_{t+1}) = -\sum_{n=1}^{N} \sigma_n\big(\delta^{\mathrm{m}} d_n^{\mathrm{m}}(t+1) + \delta^{\mathrm{i}} d_n^{\mathrm{i}}(t+1)\big). \tag{5}$$

Here, the reward to be maximized is the negative cost.

V. The Optimal Policy for the Gateway

A. The Q-Learning Algorithm

In the BSN, we aim to find an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ for the gateway to minimize the overall cost for the system. Accordingly, we first define value function $\mathcal{V}^\pi : \mathcal{S} \rightarrow \mathbb{R}$ that represents the expected value obtained by following policy π from each state $s \in \mathcal{S}$. The value function $\mathcal{V}$ for policy π quantifies the goodness of the policy through an infinite horizon and discounted MDP that can be expressed as follows:

$$\mathcal{V}^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t(s_t, a_t) | s_0 = s \right] = \mathbb{E}_\pi \Big[r_t(s_t, a_t) + \gamma \mathcal{V}^\pi(s_{t+1}) | s_0 = s \Big], \tag{6}$$

where $0 \leq \gamma < 1$ is a discount factor, and $r_t(s_t, a_t)$ represents the immediate reward achieved at state s_t after action a_t is taken. Since we aim to find optimal policy π^*, an optimal action at each state has to be found through the optimal value function expressed as follows:

$$\mathcal{V}^*(s) = \max_a \Big\{ \mathbb{E}_\pi \big[r_t(s_t, a_t) + \gamma \mathcal{V}^\pi(s_{t+1}) \big] \Big\}. \tag{7}$$

If we denote $\mathcal{Q}^*(s, a) \triangleq r_t(s_t, a_t) + \gamma \mathbb{E}_\pi \big[\mathcal{V}^\pi(s_{t+1}) \big]$ as the optimal Q-function for all state-action pairs, then the optimal value function can be written as follows:

$$\mathcal{V}^*(s) = \max_a \big\{ \mathcal{Q}^*(s, a) \big\}. \tag{8}$$

Now, the problem is reduced to determining an optimal value of Q-function, i.e., $\mathcal{Q}^*(s, a)$, for all state-action pairs, and this can be done through making samples iteratively. In particular, the Q-function is updated according to the following rule:

$$\mathcal{Q}_{t+1}(s, a) = \mathcal{Q}_t(s, a) + \alpha_t \Big[r_t(s, a) + \gamma \max_{a'} \mathcal{Q}_t(s, a') - \mathcal{Q}_t(s, a) \Big]. \tag{9}$$

The core idea behind this rule is to find the temporal difference between the predicted Q-value, i.e., $r_t(s, a) + \gamma \max_{a'} \mathcal{Q}_t(s, a')$ and its current value, i.e., $\mathcal{Q}_t(s, a)$. In (9), the introduction of the learning rate α_t is to determine the impact of new information to the existing value. The learning rate can be chosen to be a constant, or it can be adjusted dynamically during the learning process. However, it must satisfy the Assumption 1 to guarantee the convergence for the Q-learning algorithm.

Assumption 1. *The step size α_t is deterministic, nonnegative and satisfies the following conditions:*

$$\alpha_t \in [0,1], \quad \sum_{t=1}^{\infty} \alpha_t = \infty, \quad \textit{and} \quad \sum_{t=1}^{\infty} (\alpha_t)^2 < \infty. \tag{10}$$

The step size adaptation $\alpha_t = 1/t$ is one of the most common examples used in reinforcement learning. More discussions for selecting an appropriate step size can be found in [28].

The detail of the Q-learning algorithm is provided in Algorithm 1.

Algorithm 1 The Q-learning algorithm

Input: For each state-action pair (s, a), initialize the table entry $Q(s, a)$ arbitrarily, e.g., to zero. Observe the current state s, initialize a value for the learning rate α and the discount factor γ.

for $t := 1$ to T **do**

From the current state-action pair (s, a), execute action a and obtain the immediate reward r and a new state s'.

Select an action a' based on the state s' and then update the table entry for $Q(s, a)$ as follows:

$$\mathcal{Q}_{t+1}(s,a) \leftarrow \mathcal{Q}_t(s,a) + \alpha_t \Big[r_t(s,a) + \gamma \max_{a'} \mathcal{Q}_t(s',a') - \mathcal{Q}_t(s,a) \Big] \tag{11}$$

Replace $s \leftarrow s'$.

end for

Output: $\pi^*(s) = \arg\max_a Q^*(s, a)$.

Once either all Q-values converge or a certain number of iterations is reached, the algorithm will be terminated. This algorithm yields an optimal policy indicating an action to be taken at each state such that $Q^*(s, a)$ is maximized for all states in the state space, i.e., $\pi^*(s) = \arg\max_a Q^*(s, a)$. Under the assumption of the step size (i.e., Assumption 1), it was proved in [29] that "*Q-learning converges to the optimum action-values with probability one so long as all actions are repeatedly sampled in all states and the action-value are represented discretely.*"

***Remark* 1.** *In practice, selecting action "a" can be done through using a popular method, i.e., ϵ-greedy strategy [33], [35]. This strategy introduces a parameter ϵ which suggests for an agent in choosing a random action with probability ϵ, otherwise the agent will select an action that maximizes the $Q(s, a)$. This strategy is necessary in exploring the whole state space. Thus, in Q-*

learning algorithm, we need to balance between the exploration time, i.e., ϵ, and the exploitation time, i.e., $1-\epsilon$.

B. SARSA: An Online Q-Learning Algorithm

Although using the Q-learning algorithm we can find an optimal policy for the gateway without requiring the knowledge about the environment, the algorithm works in an offline fashion. In particular, Algorithm 1 can obtain the optimal policy only after the iterations are terminated, i.e., when Q-values converge. Therefore, in this section, we consider an alternative online learning algorithm, i.e., the SARSA algorithm (Algorithm 2).

Algorithm 2 SARSA: An online Q-learning algorithm

Input: For each state-action pair (s, a), initialize the table entry $Q(s, a)$ arbitrarily, e.g., to zero. Observe the current state s, initialize a value for the learning rate α and the discount factor γ.

for $t := 1$ to T **do**

From the current state-action pair (s, a), execute action a and obtain the immediate reward r and a new state s'.

Select an action a' from $\mathcal{A}_{s'}$ using a policy derived from Q-learning policy, e.g., ϵ-greedy, and then update the table entry for $Q(s, a)$ as follows:

$$\mathcal{Q}_{t+1}(s,a) \leftarrow \mathcal{Q}_t(s,a) + \alpha_t\Big[r_t(s,a) + \gamma\mathcal{Q}_t(s',a') - \mathcal{Q}_t(s,a)\Big] \tag{12}$$

Replace $s \leftarrow s'$ and $a \leftarrow a'$.

end for

Different from the Q-learning algorithm, the SARSA algorithm is an online algorithm which allows the gateway to choose an optimal action at each time epoch in a real-time fashion without waiting until the algorithm converges. In the Q-learning algorithm, the policy is updated based on the maximum reward of available actions regardless of which policy is applied, i.e., an offline method. In contrast, the SARSA algorithm interacts with the environment and updates the policy directly from the actions taken, i.e., an on-policy method. Note that the SARSA algorithm updates Q-values from the quintuple $Q(s, a, r, s', a')$, where s and a are the current state and action, respectively, r is the immediate reward obtained after action a is taken, and s' and a' are the new state-action pair. The parameters α and γ have the same meanings as in Algorithm 1.

Since the SARSA algorithm is an online learning algorithm, its conditions for convergence are much dependent on the selected learning policy. Therefore, to achieve the convergence to the optimality for SARSA algorithm, it was shown in [30] that the selected learning policy must be greedy in the limit of infinite exploration (GLIE). In particular, a policy is GLIE if all states are infinitely visited and each action is executed with an infinite number of times. One example of GLIE is the ϵ-greedy policy as mentioned in Remark 1. Then, the convergence of the SARSA algorithm as given in Algorithm 2 is provided in Theorem 1.

THEOREM 1. *The sequence $\{Q_t\}$ converges to Q^* with probability one, and the sequence $\{\pi_t\}$ converges to π^* if the following conditions are satisfied.*

1) *The Q-values are stored in a lookup table.*
2) $\alpha_t \in [0,1]$, $\sum_{t=1}^{\infty} \alpha_t = \infty$, *and* $\sum_{t=1}^{\infty} (\alpha_t)^2 < \infty$.
3) $\mathrm{Var}[r_t(s,a)] < \infty$.
4) $\{\pi_t\}$ *is greedy in the limit of infinite exploration (GLIE).*

In Theorem 1, the second condition guarantees that the step size of Algorithm 2 approaches zero when the time goes to infinity. The third condition is for the variance function of the immediate reward function to be lower than infinity. This condition implies that the immediate reward function is bounded. The fourth condition enforces learning policies to follow the GLIE. The detailed proof of Theorem 1 can be found in [30].

C. SARSA with Linear Function Approximation

With the SARSA algorithm, we can apply optimal policies for the gateway in an online fashion such that at each time slot the gateway can select the best action to optimize its objective function without having knowledge about the environment. However, when we implement Algorithm 2 for the gateway, it requires the gateway to store a lookup table for all Q-values. For example, if we have two sensor nodes with the maximum data queue size of 5 packets, and the maximum energy storage capacity of the gateway is 10 units of energy, the total number of states is $42^2 \times 22 = 38,808$. Then, if we have 4 actions per state, then the number of Q-values in the lookup table will be $155,232$, which is intractable especially when the number of sensors grows. Additionally, a large state space requires much more time of experiments to collect enough information for a successful learning process. To address the curse-of-dimensionality issue, we adopt a common approach from machine learning for representing values, i.e., the

linear function approximation.

By adopting an appropriate feature function $\phi(s,a)$ which may yield similar feature vectors for similar state-action pairs, we can provide a generalization over the state space and the action space for a specific task. If such a function is available, the Q-value function can be approximated as a linear combination of these features as follows:

$$Q^{\pi}(s,a) \approx Q^{\pi}_{\boldsymbol{\theta}}(s,a) = \sum_{f=1}^{F} \phi_f(s,a)\theta_f = \boldsymbol{\phi}^{\top}(s,a)\boldsymbol{\theta}, \tag{13}$$

where F is the total number of features, θ_f is a weight of feature f, and $^{\top}$ represents the transpose operation. Here, we denote $\boldsymbol{\phi}^{\top}(s,a) = \begin{bmatrix} \phi_1 & \cdots & \phi_f & \cdots & \phi_F \end{bmatrix}$. Now, instead of learning the optimal Q-values for all state-action pairs directly, it is sufficient to learn the vector of weights $\boldsymbol{\theta}^{\top} = \begin{bmatrix} \theta_1 & \cdots & \theta_f & \cdots & \theta_F \end{bmatrix}$ which can lead to an approximation of the Q-function.

Given the approximation function, we aim to minimize the mean-squared error (MSE) over a probability distribution $\mathcal{Z}$ of the state space $\mathcal{S}$, i.e.,

$$\zeta(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{Z}}\Big[(Q^{\pi}(s,a) - Q_{\boldsymbol{\theta}}(s,a))^2\Big], \tag{14}$$

where $Q^{\pi}(s,a)$ is an actual value of Q-function over the policy π, and $Q_{\boldsymbol{\theta}}(s,a)$ is the estimated value of Q-function given the vector of weights $\boldsymbol{\theta}$. Note that different from the steady state distribution π, the probability distribution $\mathcal{Z}$ represents a frequency of revisiting a pair of state-action. In other words, the probability distribution $\mathcal{Z}$ describes how often a pair of state-action is revisited.

However, it is intractable to compute the mean-squared error for a system with large state space. Thus, we can find a local minimum for the mean-squared error by using the *Stochastic Gradient Descent* (SGD) method[2]. The SGD method allows us to calculate the updated value of $\boldsymbol{\theta}'$ by following an approximation of the gradient of the error function. Specifically, the learning rate factor α will be used to adjust the step size of the SGD method and prevent overshooting. Then, we can update the weight vector $\boldsymbol{\theta}$ in the direction of the gradient as follows:

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \frac{1}{2}\alpha_t \nabla_{\boldsymbol{\theta}_t}\zeta(\boldsymbol{\theta}_t), & (15)\\ &= \boldsymbol{\theta}_t + \alpha_t \mathbb{E}_{\mathcal{Z}}\Big[\big(Q^{\pi}(s,a) - Q_{\boldsymbol{\theta}_t}(s,a)\big)\nabla Q_{\boldsymbol{\theta}_t}(s,a)\Big]. & (16) \end{aligned}$$

In (15)-(16), to compute the second term, i.e., $\mathbb{E}_{\mathcal{Z}}\Big[\big(Q^{\pi}(s,a) - Q_{\boldsymbol{\theta}_t}(s,a)\big)\nabla Q_{\boldsymbol{\theta}_t}(s,a)\Big]$, we need to know the probability distribution $\mathcal{Z}$ which may not be possible in practice. This leads

[2]The gradient methods for minimizing mean-squared error are commonly used in the literature, especially in machine learning. More information of using gradient methods can be found in [33].

to the idea of adopting the bootstrapping method [31], [32] for estimating the second term. Bootstrapping is a statistical method for estimating the sampling distribution of an estimator by sampling with replacement from the original sample, most often with the purpose of deriving robust estimates of standard errors. Now, instead of calculating over the whole distribution $\mathcal{Z}$, we can update the value of $\boldsymbol{\theta}$ at each sample gradient as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \Big(Q^{\pi}(s_t, a_t) - Q_{\boldsymbol{\theta}_t}(s_t, a_t) \Big) \nabla Q_{\boldsymbol{\theta}_t}(s_t, a_t). \tag{17}$$

Since each sample gradient is an unbiased estimate of the true gradient, this converges to a local minimum of the MSE if the step size α decreases appropriately with t, e.g., α satisfies Assumption 1 [33].

In (17), by using the Q-value approximation function in (13), we derive the following result:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \Big(Q^{\pi}(s_t, a_t) - Q_{\boldsymbol{\theta}_t}(s_t, a_t) \Big) \boldsymbol{\phi}(s_t, a_t). \tag{18}$$

We are now ready to present an online learning algorithm with linear function approximation for the gateway of the BSN as in Algorithm 3.

In Algorithm 3, κ_t denotes the temporal difference which is used to adapt the prediction for the agent by comparing the Q-value at the current state and the prediction at the next state. Vector $\boldsymbol{e}$ denotes the approximation of the gradient $\nabla_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}}(s, a)$. We then introduce the following theorem.

THEOREM 2. *Let the learning policy $\pi_{\boldsymbol{\theta}}$ be an ϵ-greedy policy with respect to $\boldsymbol{\theta}$ and let C be the Lipschitz constant of the learning policy $\pi_{\boldsymbol{\theta}}$ with respect to $\boldsymbol{\theta}$. Assume that the step size satisfies Assumption 1. Then, there is $C_0 > 0$ such that, if $C < C_0$, the Algorithm 3 converges with probability one.*

The proof of Theorem 2 is provided in Appendix A.

***Remark* 2.** *Algorithm 3 is straightforward to implement for the gateway. It just requires a small amount of the gateway's memory to store the parameters with very simple computations at each time step to update the parameters. However, one of the challenges in Algorithm 3 is to find appropriate features for the linear function approximation. Chosen features must not only indicate the best action to execute, but also convey the information about what future states are useful. For example, in our proposed system, we aim to minimize the average delay and the packet loss especially for important data in the system. Therefore, we need to have the features for different types of packets at different sensors. One example is given in (21) for*

Algorithm 3 SARSA with linear function approximation

Input:

Initialize a set of features $\boldsymbol{\phi}^\top(s,a) = \begin{bmatrix} \phi_1 & \cdots & \phi_f & \cdots & \phi_F \end{bmatrix}$ along with arbitrary weights $\boldsymbol{\theta}^\top = \begin{bmatrix} \theta_1 & \cdots & \theta_f & \cdots & \theta_F \end{bmatrix}$ with the same dimensionality F as the feature vector $\boldsymbol{\phi}$.

We set $Q_{\theta_0}(s,a) = 0$ for all state-action pairs of (s,a) at the beginning.

Observe the current state s, initialize the learning rate α and the discount factor γ.

for $t := 1$ to T **do**

From the current state-action pair (s,a), execute action a and obtain the immediate reward r and a new state s'.

Select an action a' from $\mathcal{A}_{s'}$ using a policy derived from Q-learning policy, e.g., ϵ-greedy. Then, let

$$\begin{aligned} \boldsymbol{e}_t &= \boldsymbol{\phi}(s,a), \text{and} \\ \kappa_t &= r(s,a) + \gamma Q_{\boldsymbol{\theta}_t}(s',a') - Q_{\boldsymbol{\theta_{t-1}}}(s,a) \end{aligned} \tag{19}$$

Then, the weights are updated as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \kappa_t \boldsymbol{e}_t \tag{20}$$

Replace $s \leftarrow s'$ and $a \leftarrow a'$.

end for

the performance evaluation. Further discussions on building features for the linear function approximation can be found in [33], [34], [35].

We provide the comparison for the complexity in terms of the storage requirement and the computation of Algorithm 1, Algorithm 2, and Algorithm 3 in Table I to show the efficiency of Algorithm 3. In Table I, while Algorithm 3 requires to store only F variables corresponding to F features, Algorithm 1 and Algorithm 2 require $|\mathcal{S}||\mathcal{A}|$ variables, in which $|\mathcal{S}||\mathcal{A}|$ is the total number of states multiplied with the total number of actions. Consequently, the computation complexities of Algorithm 1 and Algorithm 2 also become intractable as the number of states is very large. By contrast, with Algorithm 3, we need to update only few equations at each iteration depending on the number of features that we define.

TABLE I

THE COMPLEXITY OF LEARNING ALGORITHMS

Algorithm	The storage requirement	The computation complexity
Algorithm 1 (Q-learning)	$\|\mathcal{S}\|\|\mathcal{A}\|$ variables	$\mathcal{O}(\|\mathcal{S}\|\|\mathcal{A}\|)$
Algorithm 2 (SARSA)	$\|\mathcal{S}\|\|\mathcal{A}\|$ variables	$\mathcal{O}(\|\mathcal{S}\|\|\mathcal{A}\|)$
Algorithm 3 (SARSA with linear function approximation)	F variables	$\mathcal{O}(F)$

VI. PERFORMANCE EVALUATION

A. Parameter Setting

We perform the simulations using MATLAB to evaluate the performance of the proposed learning algorithm, i.e., Algorithm 3 under different parameter settings. First, we consider the case with two sensor nodes. The packet arrival probabilities of sensor nodes 1 and 2 are 0.5 and 0.4, respectively. 30% and 20% of the arriving packets at sensor nodes 1 and 2 are important packets, respectively. The maximum data queue size of both nodes are 5 packets. We assume that node 1 has higher priority than that of node 2, and thus the priority factors are set to 1.3 and 1, respectively. Furthermore, the probability that the transmitting channel between node 1 (node 2) and the gateway is good is 0.7 (0.9). The weight of the important packet is 1.5 and that of the normal packet is 1. For the gateway, the maximum energy storage size is 10 units of energy. The gateway uses 1 unit and 2 units of energy to collect data from a sensor when the transmitting channel is in good and bad condition, respectively. The good charging channel probability is 0.6. The probability that the gateway can harvest 1 unit or 2 units of energy from the charger is 0.5 when the charging channel is good. The parameters of the learning algorithm are set as follows. The discount factor is 0.98, and the exploration parameter is 0.01. For the feature function, one of the most effective ways is to find key features based on the objective function. In our proposed system, the objective function is the cost defined in (4). From (4), the feature function can be defined as follows:

$$Q_{\boldsymbol{\theta}}(s,a) = \sum_{n=1}^{N} \theta_n^{\mathrm{i}} \phi_n^{\mathrm{i}}(s,a) + \theta_n^{\mathrm{m}} \phi_n^{\mathrm{m}}(s,a), \tag{21}$$

where N is the number of sensor nodes and in our case $N = 2$. ϕ_n^x $(n = 1, \ldots, N)$ are numerical features corresponding to the types of packets of the nodes, and θ_n^x are the weights of such features. Here, $x = \mathrm{i}$ and $x = \mathrm{m}$ correspond to the cases of important packet and normal packet, respectively. We choose an initial arbitrary weight of 5, i.e., $\theta_n^x = 5$.

Based on actions of the gateway presented in (2), the feature functions can be defined as follows.

- Action = "Charging"
  - if $d_n^x = 0$: $\phi_n^x = \sigma_n \delta^x$
  - if $d_n^x > 0$: $\phi_n^x = (-e/E)\sigma_n \delta^x + (1 - c_n)$
- Action = "Do nothing"
  - $\phi_n^x = -(\sigma_n \delta^x + (1 - c_n))\mathbf{1}(d_n^x)$, where $\mathbf{1}(y) = 0$ if $y = 0$ and $\mathbf{1}(y) = 1$ if $y > 0$
- Action = "Accept one important packet from node n"
  - $\phi_n^{\mathrm{i}} = \sigma_n \delta^{\mathrm{i}} + (2 - c_n)$ and $\phi_n^{\mathrm{m}} = 0$
  - $\phi_{\overline{n}}^x = -\sigma_{\overline{n}} \delta^x \mathbf{1}(d_{\overline{n}}^x)$ (where $\overline{n}$ represents all nodes except node n)
- Action = "Accept one normal packet from node n"
  - $\phi_n^{\mathrm{m}} = \sigma_n \delta^{\mathrm{m}} + (2 - c_n)$ and $\phi_n^{\mathrm{i}} = 0$
  - $\phi_{\overline{n}}^x = -\sigma_{\overline{n}} \delta^x \mathbf{1}(d_{\overline{n}}^x)$

We consider the case that the sensor nodes and gateway have no information about the environment, e.g., the good channel probabilities. Therefore, to compare and evaluate the performance of the proposed learning algorithm, we consider two baseline schemes, i.e., a greedy-myopic (GM) policy and greedy-charging (GC) policy. For the GM policy, the gateway always takes action "transmit" as long as it has enough energy. For the GC policy, the gateway always takes action "charging" when the charging channel is good, and it always takes action "transmit" if it has enough energy and the charging channel is bad.

B. Simulation Results

We first evaluate the performance of the proposed learning algorithm with the linear function approximation defined in (21) by analyzing the convergence results and the obtained policy. We then vary the probability of charging channel to be good, and the priority factor of sensor node 1 to show the efficiency of the proposed learning algorithm compared with the GC policy and the GM policy. Additionally, to further gain more insight, we present results obtained by the learning algorithm with linear function approximation defined as follows:

$$Q_{\boldsymbol{\theta}}(s, a) = \theta_g \phi_g(s, a) + \sum_{n=1}^{N} \theta_n^{\mathrm{i}} \phi_n^{\mathrm{i}}(s, a) + \theta_n^{\mathrm{m}} \phi_n^{\mathrm{m}}(s, a). \tag{22}$$

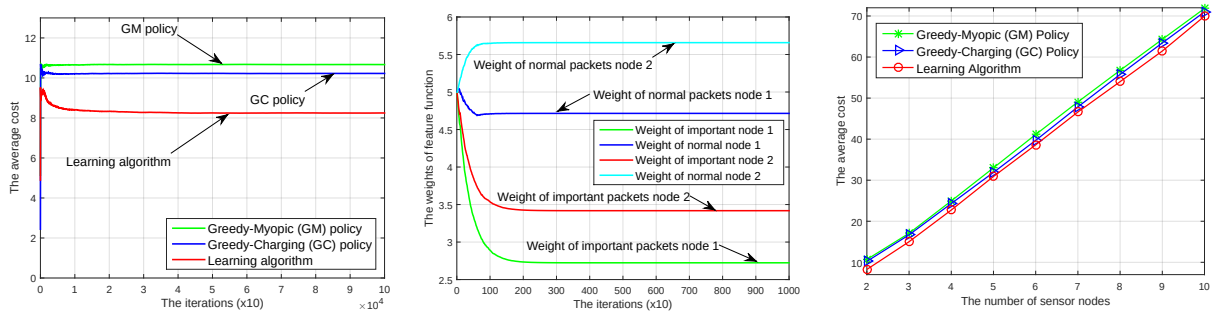
In (22), we define a new feature, denoted as ϕ_g, together with its weight θ_g, the aim of which is to show that by defining more specific feature for the system, we can obtain better performances.

The feature θ_g can be expressed as the feature of energy levels of the gateway, and it can be defined as follows:

$$\phi_g = \begin{cases} 1 - e/E, & \text{if action = “Charging”} \\ 0, & \text{otherwise.} \end{cases} \tag{23}$$

Note that we denote the linear function approximation defined in (21) and (22) as feature function 1 and feature function 2, respectively.

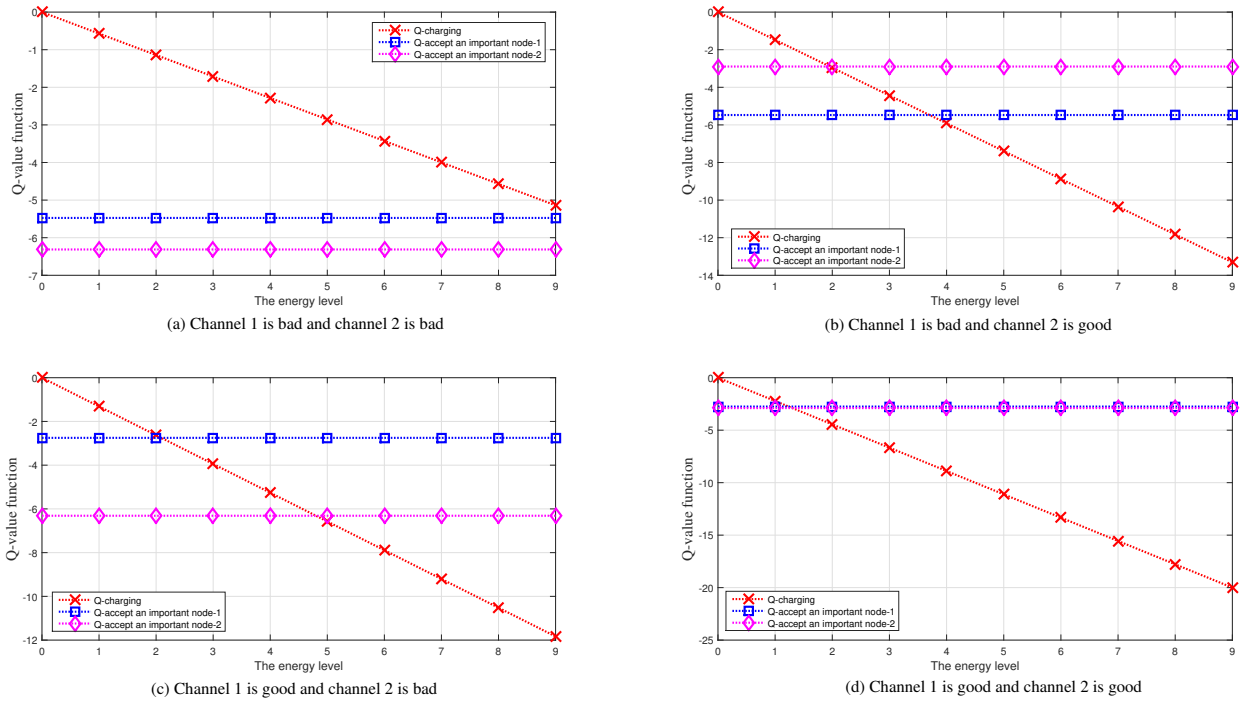
1) Convergence and Policy: We first show the convergence and the optimal policy obtained by the proposed algorithm, i.e., Algorithm 3. The convergence in terms of the average cost is shown in Fig. 5 (a) and the weights of the feature function is shown in Fig. 5 (b). The convergence rates of the weights of the feature function are relatively fast with around 4×10^3 iterations, while the convergence of the average cost of the proposed learning is slightly slower with around 4×10^5 iterations. The reason is because the average cost converges only after the weights converge. In particular, in the first 10^4 iterations, the average cost of the learning algorithm increases to 9.5. Then, in the next 10^5 iterations, it reduces gradually and becomes stable at 8.35 after 4×10^5 iterations. Although the convergence rate of the learning algorithm is relatively slower than those of the GM and GC policies, the average cost obtained by the proposed learning algorithm is much lower, i.e., 8.35 versus 10.6686 and 10.2254, respectively. Then, in Fig. 5 (c), we increase the number of sensor nodes and compare the average costs among algorithms. As shown in Fig. 5 (c), the learning algorithm always achieves the best performance (i.e., the average cost is lowest) compared with the GC and the GM policies.



(a) (b) (c)

Fig. 5. The convergence of (a) average cost of the system and (b) the parameters of weights in the feature function, and (c) the average cost of the system as the number of nodes is increased.

After the converged weights are reached for the learning algorithm, we obtain the policy through Algorithm 3 with the Q-function from (21). Next, we evaluate the common scenarios, i.e., when all two nodes have both important and normal packets in the data queue. We analyze



(a) Channel 1 is bad and channel 2 is bad

(b) Channel 1 is bad and channel 2 is good

(c) Channel 1 is good and channel 2 is bad

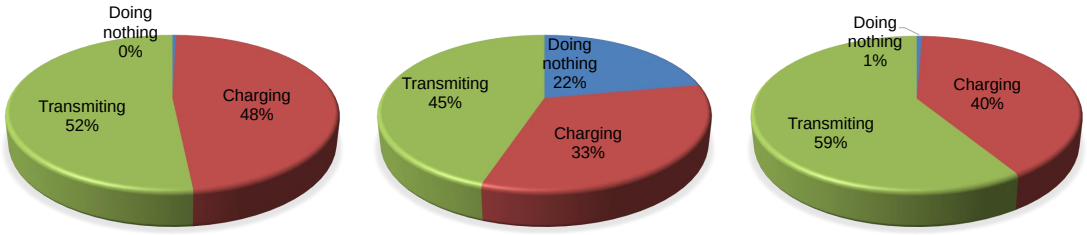
(d) Channel 1 is good and channel 2 is good

Fig. 6. The Q-value functions of (a) both transmitting channels are bad, (b) transmitting channel 1 is bad and transmitting channel 2 is good, (c) transmitting channel 1 is good and transmitting channel 2 is bad, and (d) both transmitting channels are good.

the impact of the energy and transmitting channel conditions to the policy. In particular, in Fig. 6, we consider four cases, i.e., when both transmitting channels are bad (Fig. 6 (a)), when transmitting channel 1 is bad and transmitting channel 2 is good (Fig. 6 (b)), when transmitting channel 1 is good and transmitting channel 2 is bad (Fig. 6 (c)), and when both transmitting channels are good (Fig. 6 (d)). In these figures, the Q-values of different actions are shown, and the gateway will choose the action that has the highest Q-value. In Fig. 6 (a), when both transmitting channels are bad, the gateway will take action "charging" to reserve energy for future transmissions. However, when both transmitting channels are good, i.e., Fig. 6 (d), the gateway only takes action "charging" when the energy level is lower than 2 units. The gateway will select node 1 for data transmission if the current energy level is higher than 1 unit. In the cases when one transmitting channel is good, the gateway will choose the node with good transmitting channel for data transmission if the energy level is higher than or equal to 2 units.

In Fig. 7, we compare the percentages of the gateway taking different actions. As shown in Fig. 7, by adopting the learning algorithm, the gateway is able to balance between actions "charging" and "transmitting" while action "do nothing" is minimally chosen. With enough energy, the learning algorithm is able to take action "transmitting" more frequently than the

baseline schemes, resulting in higher throughput. In particular, the charging rate of the learning algorithm is 40% which is lower than that of GC policy, i.e., 48%. However, the data transmitting rate of the learning algorithm is higher than that of the GC policy, i.e., 59% versus 52%.

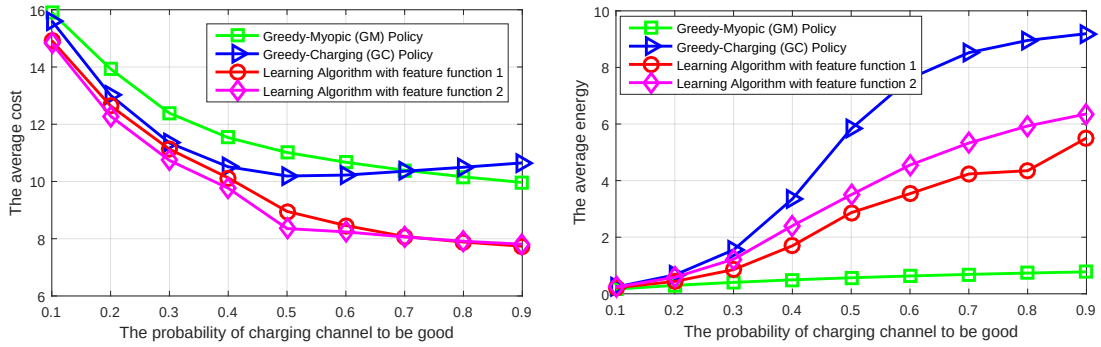


(a) (b) (c)

Fig. 7. The percentages of taking different actions obtained by (a) the GC policy, (b) the GM policy, and (c) the learning algorithm.

2) Average cost and energy: In the aforementioned analysis, the most important factor influencing the learning policy is energy of the gateway. The energy is limited, and thus the gateway has to balance between taking the charging action and the data transmission action. To obtain more insights, in Fig. 8, we vary probability of charging channel to be good from 0.1 to 0.9 and compare the average cost (Fig. 8 (a)) and the average available energy of the gateway (Fig. 8 (b)). As shown in Fig. 8 (a), as probability of charging channel to be good increases from 0.1 to 0.5, the average costs of the GM policy, the GC policy, and the learning algorithms decrease gradually to 11, 10.2 and 8.9, respectively. However, as this probability increases from 0.5 to 0.9, the average costs of the GM policy and the learning algorithms keep decreasing while that of the GC policy slightly increases. When probability of charging channel to be good is 0.9, the average cost of the learning algorithm becomes 7.8 which is lower than those of the GM and GC policies around 22% and 27%, respectively. The reason for the cost increment of the GC policy is that when probability of charging channel to be good is high, the received energy is more than that required since the GC policy always takes "charging" action as shown in Fig. 7. For the learning algorithm, although the available energy of the gateway is not as high as the GC policy, the algorithm is able to balance between energy receiving and data transmission so that the best performance can be achieved (as shown in Fig. 7). Consequently, we observe that the average energy of the learning algorithms are between those of the GM and GC policies as shown in Fig. 8 (b). Furthermore, when probability of charging channel to be good is not high, the average cost obtained by the learning algorithm with feature function 2 is slightly lower than that of the leaning algorithm with feature function 1. This reveals that the

gateway can reserve more energy to achieve better performance when the energy is scarce.



(a) (b)

Fig. 8. (a) The average cost of the system and (b) the average available energy of the gateway when probability of charging channel to be good is varied.

3) Average packet loss: Next, we examine the impact of the available energy to the packet loss probability and the average number of packets waiting in the data queues in Fig. 9. In the system under consideration, the average loss of important packets is always kept at a very small level. Specifically, when probability of charging channel to be good is 0.1, the packet loss probability for important packets of all four algorithms is relatively high, i.e., around 10% due to the lack of enough energy. However, this probability drops dramatically to around 3% when probability of charging channel to be good is 0.2 and is approximately zero when probability of charging channel to be good is higher than 0.2. This result clearly shows that the important packets are well prioritized. However, normal packets can experience high packet loss probability. This is to yield a transmission opportunity to the important packets. In particular, when probability of charging channel to be good is low, i.e., 0.1, the packet loss probability of the normal packets is very high (around 67%), and this probability decreases as the good channel probability increases for the GM policy and the learning algorithm. For the GC policy, because it spends too much time for charging, when probability of charging channel to be good is high, the time for data transmission increases. Consequently, the average packet loss rate is reduced. For all the cases, the learning algorithm always achieves the best performance in terms of the minimal packet loss.

We now examine the impact of the priority factor to the average packet loss in Fig. 10. In particular, we fix probability of charging channel to be good at 0.6, keep the priority factor of node 2 at 1, and vary the priority factor of node 1 from 0.2 to 1.8. As the priority factor of node 1 increases, the average packet losses of node 1 obtained by the GC and GM policies have a

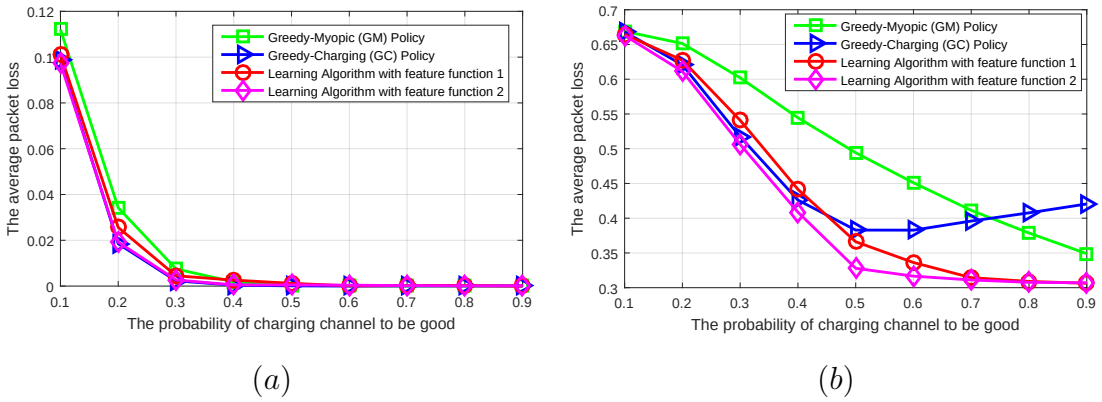
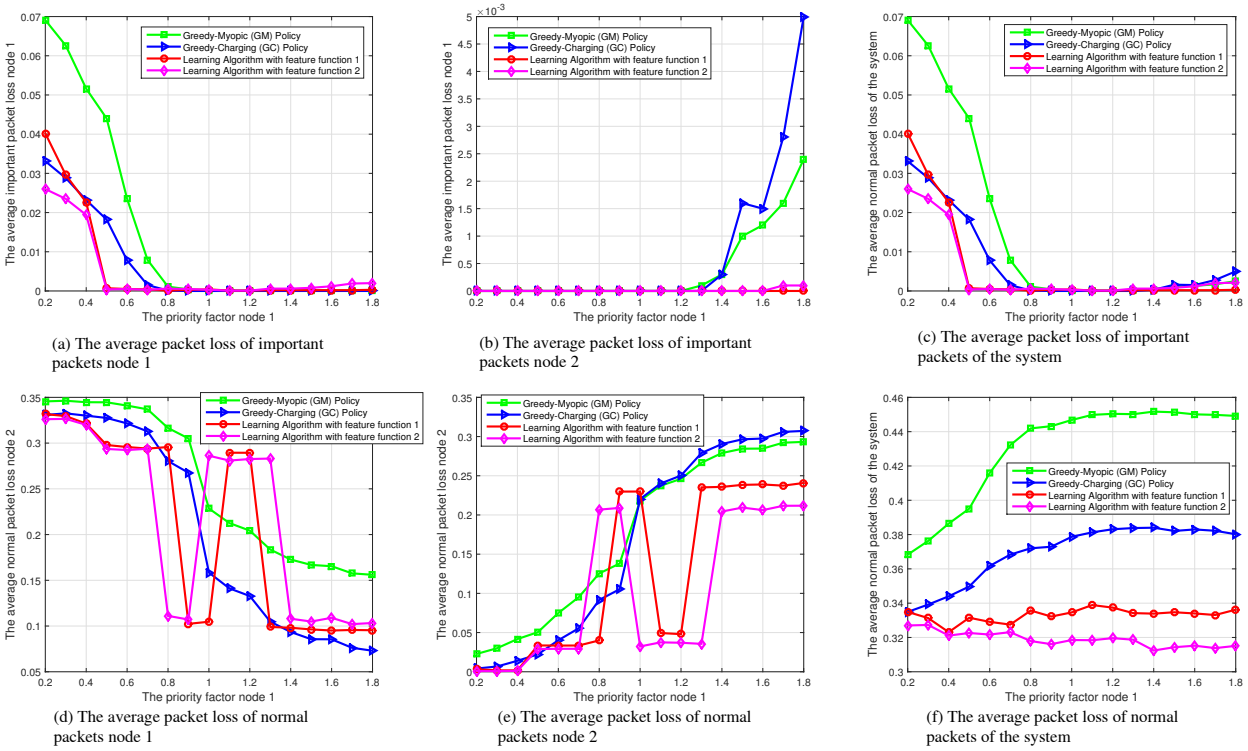


Fig. 9. The average loss of (a) important packets and (b) normal packets in the system.

down trend, while those of node 2 have an up trend as shown in Figs. 10 (a), (b), (d), and (e). This implies that the node with higher priority factor has more opportunities to transmit data, and thus its average packet loss will be lower. Similar trends are observed for average packet losses of node 1 and node 2 obtained by the learning algorithm. However, when the value of the priority factor of node 1 is close to 1, i.e., equals to the value of priority factor of node 2, the average normal packet losses of node 1 and node 2 are fluctuated. The reason is that when the priority factors of both nodes are close to each other, the learning algorithm can randomly choose one of the nodes to transmit data as long as the cost function is minimized. Thus, the average packet loss of normal packets obtained by the learning algorithm for the whole system is much lower than those of the GC and GM policies as shown in Fig. 10 (f). Through this result, we are able to set appropriate priority factors of the nodes to achieve performance requirements.

VII. Conclusion

In this paper, we have considered the backscatter sensor network (BSN) in which sensors communicate with the gateway node using the backscatter communication technique. To minimize the delay together with the packet loss, we have first formulated the energy transfer and data transmission scheduling problem as a Markov decision process. We then have adopted learning algorithms to find optimal policies for the gateway to avoid the curse-of-model issue. Moreover, we have also investigated linear approximation functions integrating to the learning algorithms, which can significantly reduce the complexity as well as the memory storage for the gateway in finding the optimal policy. Simulation results have validated the convergence and the efficiency of the proposed learning algorithm.



(a) The average packet loss of important packets node 1

(b) The average packet loss of important packets node 2

(c) The average packet loss of important packets of the system

(d) The average packet loss of normal packets node 1

(e) The average packet loss of normal packets node 2

(f) The average packet loss of normal packets of the system

Fig. 10. The average packet loss when the priority factor of node 1 is varied.

APPENDIX A
THE PROOF OF THEOREM 2

To prove Theorem 2, we adopt the Ordinary Differential Equations (ODE) approach. From (20), we can rewrite the update for the weight parameters as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \boldsymbol{\phi}(s,a)\big(r(s,a) + \gamma Q_{\boldsymbol{\theta}_t}(s',a') - Q(s,a)\big). \tag{24}$$

Recall that the standard *Euler scheme* for numerically approximating a trajectory of the ODE $\dot{x} = h(x(t))$ would be $x_{n+1} = x_n + \alpha_t h(x_n)$ (Chapter 2 [36]). Then, from (13), (15), (16) and (24), the associated ODE of $\boldsymbol{\theta}$ can be expressed as follows:

$$\begin{aligned} \dot{\boldsymbol{\theta}} &= \mathbb{E}_{\boldsymbol{\theta}}\Big[\boldsymbol{\phi}(s,a)\big(r(s,a) + \gamma Q_{\boldsymbol{\theta}}(s',a') - Q(s,a)\big)\Big], \\ &= \mathbb{E}_{\boldsymbol{\theta}}\Big[\boldsymbol{\phi}(s,a)\big(r(s,a) + \gamma \boldsymbol{\phi}^\top(s',a')\boldsymbol{\theta} - \boldsymbol{\phi}^\top(s,a)\boldsymbol{\theta}\big)\Big]. \end{aligned} \tag{25}$$

In (25), we omit the dependence of $\boldsymbol{\theta}$ on t for the convenience of presentation. Now, we will follow Theorem 17 from [37] and show that if the associated ODE of $\boldsymbol{\theta}$ has a globally asymptotically stable equilibrium point, Algorithm 3 will converge with probability one. To determine the global asymptotic stability for the associated ODE of $\boldsymbol{\theta}$, we rewrite (25) in the following form:

$$\dot{\boldsymbol{\theta}} = \mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\theta} + \boldsymbol{b}_{\boldsymbol{\theta}}, \tag{26}$$

where $\mathbf{A}_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta}}\big[\boldsymbol{\phi}(s,a)(\gamma\boldsymbol{\phi}^\top(s',a') - \boldsymbol{\phi}^\top(s,a))\big]$ and $\boldsymbol{b}_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{\theta}}\big[r(s,a)\boldsymbol{\phi}(s,a)\big]$.

Denote $\boldsymbol{\theta}^*$ as the equilibrium point of (26)[3] and let $\widetilde{\boldsymbol{\theta}}(t) = \boldsymbol{\theta}(t) - \boldsymbol{\theta}^*$. Then, from the Lyapunov stability theory, we have the following statement. Given ODE: $\dot{\vec{x}} = f(\vec{x})$, if there exists function $\mathcal{V}(\vec{x}) : \mathbb{R}^n \to \mathbb{R}$ such that

$$\mathcal{V}(\vec{x} = \vec{0}) = 0, \mathcal{V}(\vec{x}) > 0 \ \ (\forall \vec{x} \neq \vec{0}), \text{and} \ \ \mathcal{V}(\vec{x}) \to \infty \ \ \text{as} \ \ \vec{x} \to \infty, \tag{27}$$

then if $\dot{\mathcal{V}}(\vec{x}) < 0 \ \ (\forall \vec{x} \neq \vec{0})$, ODE $\dot{\vec{x}} = f(\vec{x})$ is globally asymptotically stable.

Now, let $\vec{x} = \widetilde{\boldsymbol{\theta}}$ and $\mathcal{V}(\vec{x}) = \| \widetilde{\boldsymbol{\theta}} \|_2^2$, it is straightforward to verify the conditions of $\mathcal{V}(\widetilde{\boldsymbol{\theta}})$ in (27). Hence, if we can prove that $\dot{\mathcal{V}}(\widetilde{\boldsymbol{\theta}}) < 0$, then the ODE (26) is globally asymptotically stable.

We derive the following results:

$$\begin{aligned}
\frac{d}{dt} \| \widetilde{\boldsymbol{\theta}} \|_2^2 &= 2\widetilde{\boldsymbol{\theta}}^\top \frac{d}{dt}\widetilde{\boldsymbol{\theta}}, \\
&= 2\widetilde{\boldsymbol{\theta}}^\top \frac{d}{dt}\big(\boldsymbol{\theta}(t) - \boldsymbol{\theta}^*\big), \\
&= 2\widetilde{\boldsymbol{\theta}}^\top (\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\theta} - \mathbf{A}_{\boldsymbol{\theta}^*}\boldsymbol{\theta}^* + \boldsymbol{b}_{\boldsymbol{\theta}} - \boldsymbol{b}_{\boldsymbol{\theta}^*}), \\
&= 2\widetilde{\boldsymbol{\theta}}^\top (\mathbf{A}_{\boldsymbol{\theta}}\boldsymbol{\theta} - \mathbf{A}_{\boldsymbol{\theta}^*}(\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}}) + \boldsymbol{b}_{\boldsymbol{\theta}} - \boldsymbol{b}_{\boldsymbol{\theta}^*}), \\
&= 2\widetilde{\boldsymbol{\theta}}^\top \mathbf{A}_{\boldsymbol{\theta}^*}\widetilde{\boldsymbol{\theta}} + 2\widetilde{\boldsymbol{\theta}}^\top (\mathbf{A}_{\boldsymbol{\theta}} - \mathbf{A}_{\boldsymbol{\theta}^*})\boldsymbol{\theta} + 2\widetilde{\boldsymbol{\theta}}^\top (\boldsymbol{b}_{\boldsymbol{\theta}} - \boldsymbol{b}_{\boldsymbol{\theta}^*}), \\
&\leq 2\widetilde{\boldsymbol{\theta}}^{\mathbf{A}}_{\boldsymbol{\theta}^*}\widetilde{\boldsymbol{\theta}} + 2 \| \widetilde{\boldsymbol{\theta}} \|_2 \| (\mathbf{A}_{\boldsymbol{\theta}} - \mathbf{A}_{\boldsymbol{\theta}^*})\boldsymbol{\theta} \|_2 + 2 \| \widetilde{\boldsymbol{\theta}} \|_2 \| \boldsymbol{b}_{\boldsymbol{\theta}} - \boldsymbol{b}_{\boldsymbol{\theta}^*} \|_2, \\
&\leq 2\widetilde{\boldsymbol{\theta}}^\top \mathbf{A}_{\boldsymbol{\theta}^*}\widetilde{\boldsymbol{\theta}} + 2 \| \widetilde{\boldsymbol{\theta}} \|_2 \sup_{\boldsymbol{\theta}} \| \mathbf{A}_{\boldsymbol{\theta}} - \mathbf{A}_{\boldsymbol{\theta}^*} \|_2 \| \boldsymbol{\theta} \|_2 + 2 \sup_{\boldsymbol{\theta} \neq \boldsymbol{\theta}^*} \frac{\| \boldsymbol{b}_{\boldsymbol{\theta}} - \boldsymbol{b}_{\boldsymbol{\theta}^*} \|_2}{\| \boldsymbol{\theta} - \boldsymbol{\theta}^* \|_2} \| \widetilde{\boldsymbol{\theta}} \|_2^2, \\
&\leq 2\widetilde{\boldsymbol{\theta}}^\top \mathbf{A}_{\boldsymbol{\theta}^*}\widetilde{\boldsymbol{\theta}} + \vartheta \| \widetilde{\boldsymbol{\theta}} \|_2^2 \sup_{\boldsymbol{\theta}} \| \mathbf{A}_{\boldsymbol{\theta}} - \mathbf{A}_{\boldsymbol{\theta}^*} \|_2 + 2 \sup_{\boldsymbol{\theta} \neq \boldsymbol{\theta}^*} \frac{\| \boldsymbol{b}_{\boldsymbol{\theta}} - \boldsymbol{b}_{\boldsymbol{\theta}^*} \|_2}{\| \boldsymbol{\theta} - \boldsymbol{\theta}^* \|_2} \| \widetilde{\boldsymbol{\theta}} \|_2^2,
\end{aligned} \tag{28}$$

where ϑ is a positive small constant. Then, let

$$\lambda_A = \sup_{\boldsymbol{\theta}} \| \mathbf{A}_{\boldsymbol{\theta}} - \mathbf{A}_{\boldsymbol{\theta}^*} \|_2, \ \ \text{and} \ \ \lambda_B = \sup_{\boldsymbol{\theta} \neq \boldsymbol{\theta}^*} \frac{\| \boldsymbol{b}_{\boldsymbol{\theta}} - \boldsymbol{b}_{\boldsymbol{\theta}^*} \|_2}{\| \boldsymbol{\theta} - \boldsymbol{\theta}^* \|_2} \tag{29}$$

where λ_A is the operator norm (induced form) and λ_B is the operator norm corresponding to the 2-norm for vectors, i.e., the regular Euclidian norm [39]. Then, we derive the following results:

$$\frac{d}{dt} \| \widetilde{\boldsymbol{\theta}} \|_2^2 \leq 2\widetilde{\boldsymbol{\theta}}^\top \mathbf{A}_{\boldsymbol{\theta}^*}\widetilde{\boldsymbol{\theta}} + 2(\frac{\vartheta}{2}\lambda_A + \lambda_B) \| \widetilde{\boldsymbol{\theta}} \|_2^2 . \tag{30}$$

Denote $\lambda = \frac{\vartheta}{2}\lambda_A + \lambda_B$, then we have

$$\frac{d}{dt} \| \widetilde{\boldsymbol{\theta}} \|_2^2 \leq 2\widetilde{\boldsymbol{\theta}}^\top (\mathbf{A}_{\boldsymbol{\theta}^*} + \lambda \mathbf{I})\widetilde{\boldsymbol{\theta}}. \tag{31}$$

[3]The existence of such an equilibrium point was provided in [38].

Due to the fact that learning policy is assumed to be a Lipschitz constant with regard to vector $\boldsymbol{\theta}$ and the uniform ergodicity of the corresponding induced chain implies that $\mathbf{A}_{\boldsymbol{\theta}}$ and $\boldsymbol{b}_{\boldsymbol{\theta}}$ are also Lipschitz constants with regard to vector $\boldsymbol{\theta}$. This implies that λ_A and $\lambda_B \rightarrow C$. Additionally, from Theorem 1 of [40], it is shown that $\mathbf{A}_{\boldsymbol{\theta}}$ is a negative definite matrix. Therefore, given a sufficiently small value of C, $(\mathbf{A}_{\boldsymbol{\theta}^*} + \lambda \mathbf{I})$ is a negative definite matrix. Consequently, the ODE in (26) is globally asymptotically stable.

The proof is completed.

REFERENCES

[1] H. Liu, M. Bolic, A. Nayak, and I. Stojmenovic, "Taxonomy and challenges of the integration of RFID and wireless sensor networks," *IEEE Network*, vol. 22, no. 6, pp. 26-35, Dec. 2008.

[2] E. Kampianakis, J. Kimionis, K. Tountas, C. Konstantopoulos, E. Koutroulis, and A. Bletsas, "Wireless environmental sensor networking with analog scatter radio and timer principles," *IEEE Sensors Journal*, vol. 14, no. 10, pp. 3365-3376, Oct. 2014.

[3] W. Xu, R. J. Piechocki, and G. Hilton, "Probabilistic data association for wireless passive body sensor networks," in *IEEE 15th International Conference on e-Health Networking, Applications & Services*, pp. 140-144, Lisbon, Portugal, Oct. 2013.

[4] W. C. Wilson and P. D. Juarez, "Emerging needs for pervasive passive wireless sensor networks on aerospace vehicles," in *The 5th International Conference on Emerging Ubiquitous Systems and Pervasive Networks*, pp. 101-108, Halifax Nova Scotia, Canada, Sept. 2014.

[5] C. Liu, D. Fang, Z. Yang, H. Jiang, X. Chen, W. Wang, and L. Cai, "RSS distribution-based passive localization and its application in sensor networks," *IEEE Transactions on Wireless Communications* vol. 15, no. 4, pp. 2883-2895, Apr. 2016.

[6] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*, (3rd ed.) West Sussex, UK: John Wiley & Sons. 2010.

[7] I. I. Veen, Q. Liu, P. Pawelczak, and A. Parks, "BLISP: Enhancing backscatter radio with active radio for computational RFIDs," in *IEEE International Conference on RFID*, pp. 1-4, Orlando, USA, May, 2016.

[8] R. Correia; N. B. Carvalho, S. Kawasaki, "Continuously power delivering for passive backscatter wireless sensor networks," in *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 11, pp. 3723-3731, Nov. 2016.

[9] L. Zhang and Z. Wang, "Integration of RFID into wireless sensor networks: Architectures, opportunities, and challenging problems," in *Fifth International Conference on Grid and Cooperative Computing Workshops*, pp. 463-469, Hunan, China, Oct. 2006.

[10] O. B. Akan, M. T. Isik, B. Baykal, "Wireless passive sensor networks," *IEEE Communications Magazine*, vol. 47, no. 8, pp. 92-99, Aug. 2009.

[11] J. Kimionis, A. Bletsas, J. N. Sahalos, "Increased range bistatic scatter radio," *IEEE Transactions on Communications*, vol. 62, no.3, pp. 1091-1104, Mar. 2014.

[12] N. Fasarakis-Hilliard, P. N. Alevizos, and A. Bletsas, "Coherent detection and channel coding for bistatic scatter radio sensor networking," *IEEE Transactions on Communications*, vol. 63, no. 5, pp. 1798-1810, May 2015.

[13] P. N. Alevizos, and A. Bletsas, "Noncoherent composite hypothesis testing receivers for extended range bistatic scatter radio WSNs," in *IEEE International Conference on Communications*, pp. 4448-4453, London, UK, Jun. 2015.

[14] G. Vannucci, A. Bletsas, and D. Leigh, "Implementing backscatter radio for wireless sensor networks," in *18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1-5, Athens, Greece, Sept. 2007.

[15] A. Bereketli, O. B. Akan, "Communication coverage in wireless passive sensor networks," *IEEE Communications Letters*, vol. 13, no. 2, pp. 133-135, Feb. 2009.

[16] F. Iannello, O. Simeone, and U. Spagnolini, "Energy management policies for passive RFID sensors with RF-energy harvesting," in *IEEE International Conference on Communications*, pp. 1-6, Cape Town, South Africa, May. 2010.

[17] R. Correia, N. B. de Carvalho, G. Fukudat, A. Miyajit, and S. Kawasakit, "Backscatter wireless sensor network with WPT capabilities," in *IEEE MTT-S International Microwave Symposium*, pp. 1-4, Phoenix, US, May. 2015.

[18] R. D. Fernandes, A. S. Boaventura, N. B. Carvalho, and J. N. Matos, "Increasing the range of wireless passive sensor nodes using multisines," in *IEEE International Conference on RFID-Technologies and Applications*, pp. 549-553, Sitges, Spain, Sept. 2011.

[19] P. Zhang, P. Hu, V. Pasikanti, and D. Ganesan, "EkhoNet: High speed ultra low-power backscatter for next generation sensors," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, pp. 557-568, Maui, Hawaii, Sept. 2014.

[20] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith, "Turbocharging ambient backscatter communication," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 619-630, Aug. 2014.

[21] P. Zhang, J. Gummeson, and D. Ganesan, "Blink: A high throughput link layer for backscatter communication," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 99-112, Ambleside,, UK, Jun. 2012.

[22] J. Gummeson, P. Zhang, and D. Ganesan, "Flit: a bulk transmission protocol for RFID-scale sensors," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 71-84, Lake District, UK, Jun. 2012.

[23] J. Ou, M. Li, and Y. Zheng, "Come and Be Served: Parallel Decoding for COTS RFID Tags," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 500-511, Paris, France, Sept. 2015.

[24] P. Hu, P. Zhang, and D. Ganesan, "Laissez-faire: Fully asymmetric backscatter communication," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 255-267, Aug. 2015.

[25] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and reliable low-power backscatter networks," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp. 61-72, Helsinki, Finland, Aug. 2012.

[26] A. Bletsas, S. Siachalou, and J. N. Sahalos, "Anti-collision backscatter sensor networks," *IEEE Transactions on Wireless Communications*, vol. 8, no. 10, pp. 5018-5029, Oct. 2009.

[27] M. T. Isik and O. B. Akan, "PADRE: Modulated backscattering-based passive data retrieval in wireless sensor networks," in *IEEE Wireless Communications and Networking Conference*, pp. 1-6, Budapest, Hungary, Apr. 2009.

[28] W. Dabney, *Adaptive step-sizes for reinforcement learning*, Doctoral Dissertations, May. 2014.

[29] C. J. C. H. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[30] S. Singh, T. Jaakkola, M. L. Littman, C. Szepesvari, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 287-308, 2000.

[31] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1-26, 1979.

[32] M. Geist and O. Pietquin, "A brief survey of parametric value function approximation," *Technical Report*, Sept. 2010.

[33] R. S. Sutton, and A. G. Barto. *Reinforcement learning: An introduction*, Cambridge: MIT press, 1998.

[34] J. N. Tsitsiklis and B. V. Roy, "Feature-based methods for large scale dynamic programming," *Machine Learning*, vol. 22, no. 1-3, pp. 59-94, Mar. 1996.

[35] A. Geramifard, T. J. Walsh, S. Tellex, G. Chowdhary, N. Roy, and J. P. How, "A tutorial on linear function approximation for dynamic programming and reinforcement learning," *Foundations and Trends®in Machine Learning*, vol. 6, no. 4, pp. 375-451, 2013.

[36] J. C. Butcher, *The Numerical Methods for Ordinary Differential Equations*, Hoboken : John Wiley & Sons, Ltd., 2008.

[37] A. Benveniste, M. Metivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations*, ser. Application of Mathematics. Berlin: Springer-Varlag, vol. 22, 1990.

[38] D. P. De Farias and B. V. Roy, "On the existence of fixed points for approximate value iteration and temporal-difference learning," *Journal of Optimization Theory and Applications*, vol. 105, no. 3, pp. 589-608, Jun. 2000.

[39] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins University Press Baltimore, MD, USA ©1996.

[40] J. N. Tsitsiklis and B. V. Roy, "An analysis of temporal-difference learning with linear function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674-690, May, 1997.