



Minnesota State University, Mankato
Cornerstone: A Collection of Scholarly
and Creative Works for Minnesota
State University, Mankato

All Graduate Theses, Dissertations, and Other
Capstone Projects

Graduate Theses, Dissertations, and Other
Capstone Projects

2019

Optimization of Energy Harvesting Mobile Nodes Within Scalable Converter System Based on Reinforcement Learning

Chengtao Xu
Minnesota State University, Mankato

Follow this and additional works at: <https://cornerstone.lib.mnsu.edu/etds>



Part of the [Power and Energy Commons](#), and the [Systems and Communications Commons](#)

Recommended Citation

Xu, C. (2019). Optimization of energy harvesting mobile nodes within scalable converter system based on reinforcement learning [Master's thesis, Minnesota State University, Mankato]. Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato. <https://cornerstone.lib.mnsu.edu/etds/938/>

This Thesis is brought to you for free and open access by the Graduate Theses, Dissertations, and Other Capstone Projects at Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato. It has been accepted for inclusion in All Graduate Theses, Dissertations, and Other Capstone Projects by an authorized administrator of Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato.

Optimization of Energy Harvesting Mobile Nodes Within Scalable Converter System Based on Reinforcement Learning

By

Chengtao Xu

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

In

Electrical Engineering

Minnesota State University, Mankato

Mankato, Minnesota

July 2019

07.2019

Optimization of Energy Harvesting Mobile Nodes Within Scalable Converter System Based on Reinforcement Learning

Chengtao Xu

This Thesis has been examined and approved by the following members of the student's committee.

Dr. Vincent J. Winstead

Dr. Xuanhui Wu

Dr. Jianwu Zeng

Acknowledgement

I would like to be indeed grateful for many people helped and guided me in the two years study in graduate school at MNSU. The completion of this thesis could not have been possible without the support and suggestions of so many people. Especially, I wish to express my sincere gratitude to my advisor Dr. Vincent J. Winstead for providing me the chance to join the renewable energy project and keep encouraging and guiding me in the process of research.

I also want to thank my committee members, Dr. Xuanhui Wu and Dr. Jianwu Zeng for the suggestion on my thesis.

In addition, I would like to thank Dr. Han-way Huang for guiding my study in MNSU.

To my parents, fiancé, friends and others who in one way or another shared their support, either morally, financially, and physically, thank you.

Abstract

Microgrid monitoring focusing on power data, such as voltage and current, has become more significant in the development of decentralized power supply system. The power data transmission delay between distributed generator is vital for evaluating the stability and financial outcome of overall grid performance. In this thesis, both hardware and simulation has been discussed for optimizing the data packets transmission delay, energy consumption, and collision rate. To minimize the transmission delay and collision rate, state-action-reward-state-action (SARSA) and Q-learning method based on Markov decision process (MDP) model is used to search the most efficient data transmission scheme for each agent device. A training process comparison between SARSA and Q-learning is given out for representing the training speed of these two methodologies in the scenario of source-relaying-destination model. To balance the exploration and exploitation process involved in these two methods, a parameter ϵ is introduced to optimize the cost time of training process. Finally, the simulation result of average throughput and data packets collision rate in the network with 20 agent nodes is presented to indicate the application feasibility of reinforcement learning algorithm in the development of scalable network. The results show that, the average throughput and collision rate stay on the expected ideal performance level for the overall network when the number of nodes is not too large. Also, the hardware development based on Bluetooth Low Energy (BLE) is used to reveal the process of data packets transmission.

Contents

1. Introduction.....	1
1.1 Background.....	1
1.2 Related works.....	3
2. Bluetooth Low Energy (BLE) Mesh Network.....	6
2.1 BLE Mesh Network Configuration.....	6
2.1.1 Comparation between wireless communication method.....	6
2.1.2 Hardware selection.....	8
2.1.3 Energy Consumption Parameter of BLE.....	11
2.2 Embedded system Development.....	16
3. Reinforcement Learning in Energy Harvesting Mobile Network.....	18
3.1 Reinforcement Learning.....	18
3.1.1 Markov Process.....	18
3.1.2 Markov Reward Process.....	19
3.1.3 Markov Decision Process.....	21
3.2 MDP simulation.....	26
3.3 ϵ -greedy Exploration Algorithm.....	29
4. Optimization of Energy Harvesting Mobile Nodes.....	30
4.1 Energy Harvesting Node Model.....	30
4.1.1 MDP parameter setting and assumption.....	30
4.1.2 Channel random access model.....	33
4.1.3 EH mobile node parameter setting.....	34
4.2 Analysis of training method.....	35

5. Simulation and Test Results.....	40
5.1 Scalable network analysis.....	40
5.2 Test on embedded system.....	43
6. Conclusion.....	45
Reference.....	46
Appendix	49

List of Figures

Figure 1.1 Universal and scalable converter system.....	2
Figure 2.1 X-NUCLEO-IDB05A1.....	8
Figure 2.2 NUCLEO-F401RE.....	10
Figure 2.3 STM32F769I-DISCOVERY.....	11
Figure 2.4 Energy consumption estimation tool of BLE.....	12
Figure 2.5 Current consumption under connection-master mode in one period.....	13
Figure 2.6 Current consumption under connection-slave mode in one period.....	13
Figure 2.7 Current consumption under advertising mode in one period.....	14
Figure 2.8 Current consumption under scanning mode in one period.....	15
Figure 2.9 Sketch of BLE distribution system.....	16
Figure 2.10 Slave device hardware configuration.....	16
Figure 2.11 Master device hardware configuration.....	17
Figure 3.1 Map of package transferring within the mesh network.....	26
Figure 3.2 State value and step direction choosing in each of the square.....	27
Figure 3.3 Shortest route from starting point to destination.....	27
Figure 4.1 Source-Relay-Destination communication system model with finite battery space and data buffer.....	30
Figure 4.2 Slotted system model in one unit of transmission	31
Figure 4.3 Action value with SARSA training.....	35
Figure 4.4 Action value with SARSA training ($\varepsilon=0.04$)	36
Figure 4.5 Action value with SARSA training ($\varepsilon=1$)	36
Figure 4.6 Action value with SARSA training ($\varepsilon=0.01$)	37
Figure 4.7 Action value with Q-learning training ($\varepsilon=0.04$)	38
Figure 4.8 Action value with Q-learning training ($\varepsilon=0.01$)	39

Figure 5.1 Average throughput dynamics.....	41
Figure 5.2 Average collision rate dynamics.....	42
Figure 5.3 Information of data sending on client (slave) side.....	43
Figure 5.4 Information of data receiving on the server (master) side.....	44

Chapter 1

Introduction

1.1 Background

Renewable energy, such as generated from solar energy and wind energy currently play an important role under the theme of economical and sustainable development. The use of renewable sources tends to reduce the discharge of greenhouse effect gas. For a standalone source with the ability of bi-directional power supply with grid, an universal converter system is designed to be configurable and capable of building the connection with a variety of power generation and storage devices (i.e. renewable energy generators, battery systems, ultra-capacitor systems, hybrid vehicle, etc.) and provide a universal interface to the grid in the future [1]. Meanwhile, to allow for grid stability, monitoring and electricity financial exchange anticipation with the emergence of smart grid power, data collection of all these distributed power generation devices and storage is essential.

According to the configuration of a universal and scalable converter system described in [2], as the figure 1.1 shows, each universal converter transfers electricity back to the utility line through the power flow line. For the monitoring data collection between converters, each of them can be configured as a single node within this scalable network. Here, they are capable of sending, relaying, and receiving collected data packets. However, since the communication range of a single node is limited, the single hop between converter nodes cannot meet the wide range device deployment requirements [9]. Also, for single node connections between multiple nodes, this data collision would be the cause of losing of essential data in the transmission process when the

receiving node receives the packets from different resources. Meanwhile, the data packet traffic within the network could be unbalanced for each node due to the case where the potential difference of the data packet size, severe environmental interference (i.e. moisture, high temperature) and unexpected reduction in expected device life exists. In particular, the node connected to the utility communication devices (or power line communication devices) consumes its energy faster than the others since it relays all the other nodes' data packets with longer receiving, buffering and sending times. The energy harvesting network could be applied to solve the energy imbalance problem.

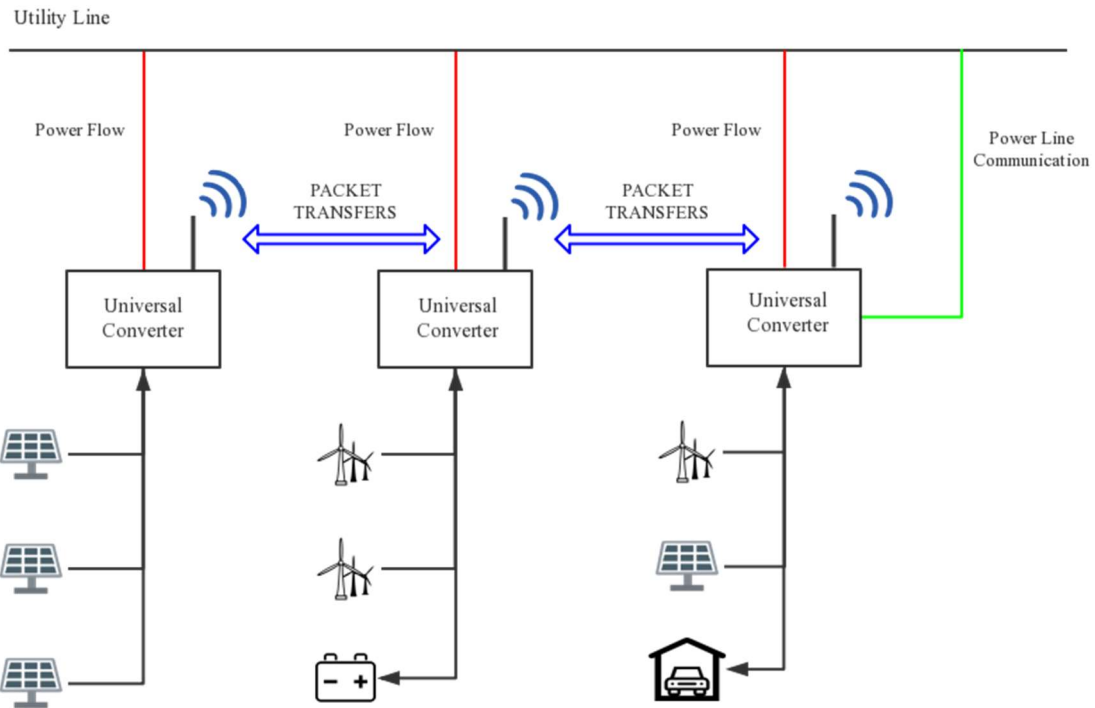


Figure 1.1 Universal and scalable converter system

There are two models considering the process of optimal performance of the sending, relaying, and receiving process in this multi-hop wireless sensor network (WHWSNs), which can be an energy constrained (EC) network or energy harvesting network (EH), both with and without

relaying cooperation [3]. Existing methodologies concerning data packet transferring in EC networks have mainly focused on solving the delay optimization, data priority, and data collecting efficiency to represent the situation of wireless sensor networks [4][5][6][7]. With the emerging technique called energy harvesting that enables wireless devices to scavenge energy physically or chemically from natural or man-made phenomena [8], additional advantages (i.e. self-sustainable capability, nearly permanent network life) have been realized in energy harvesting networks.

1.2 Related works

The study of EH network wireless communication has been based on three different approaches, namely, offline, online, and learning [10]. The offline policy assumes the stationary knowledge in a specific pattern regarding the EH processes in a distributed network. Similar works have been done by assuming the transmitter has non-casual information on arrival time and data amount [11, 12, 13]. Although the offline policy will not be able to fit in most realistic applications, the ideal network performance results will be generated by it. The online policy requests associated knowledge from the transmitter representing the statistical information of the data packets arrival time, power harvested by a rechargeable battery, and channel collision or fading. Also, prior knowledge won't be available at the beginning operating time or after topology changes in the EH process. Therefore, a learning policy has been considered to optimize the action execution when the packets need to be transmitted in an unknown operating environment. One approach we will apply to find the optimized action of package transferring in this thesis is called reinforcement learning.

In [12], a learning theoretical approach has been investigated which was applied to a point-to-point wireless communication system with finite capacity rechargeable battery. A data packet arrives at the beginning of each time slot (TS) and gets lost in the following TS if it has not been sent out. One can assign a likelihood for this assuming the transmitter will be terminated under specific conditions to make this energy harvesting (EH) network more realistic. However, the only energy consumption we consider is strictly defined in the process of transmission while ignoring the relaying and receiving process energy consumption.

In [3], both the EC network and EH network have been investigated. In this work, the authors provide the framework for finding the optimal policy by the proposed Markov decision process (MDP) under two different methods describing the data package transmission action, namely finite-horizon processes and infinite-horizon processes. Under the unified MDP framework and proposed dual linear programming based algorithm, the numerical result measuring network performance (i.e. energy harvesting rate, nonarrival package rate, battery capacity) is provided. However, the authors do not give out the convergence result by analyzing this general framework created to describe the data transmission process inside multi-hop wireless sensor network (MHWSN).

The authors of [14][15] both investigate the learning policy for agents within a network to learn the optimal strategy to balance the traffic demand and harvested energy, battery level, the drop rate of data packets and energy efficiency. The author in [14] proposed an approximated set of binary functions to approximate the expected throughput. In [15], the authors applied a Q-learning algorithm thus finding the optimized system performance in terms of drop rate, throughput, and energy efficiency. The author in [16] provided a mean-field deep reinforcement learning approach to find the online control policy which only requires the local knowledge of

the state. They achieved performance comparable with a centralized network having an offline policy.

In this thesis, based on the point-to-point EH wireless communication, the performance (i.e. data transferring latency, device node power consumption, channel collision and data packet dropping rate) of a scalable EH multi-agent wireless communication network will be investigated. The network model will be both formulized as a Markov process using historical statistical knowledge of state to find the best action of package transferring and MDP to find the optimal strategy with local knowledge of the state. Finally, I will compare simulation results of the system performance by taking Q-learning in an offline policy, state-action-rewards-state-action (SARSA) in an online policy, and greedy algorithm used in exploring the optimized policy.

The rest of the thesis will be organized as follows. In chapter 2, the system hardware deployment is presented, which includes a Bluetooth low energy technique configured mesh network to realize the embedded system test of this reinforcement learning process. In chapter 3, the EH multi-agent power allocation, data package storage and relaying, and collision rate of packets will be formulated. In chapter 4, the reinforcement learning (RL) algorithm used for EH network scenario to achieve network performance optimization is presented. In chapter 5, the final simulation results will be demonstrated and tested via the algorithm code on a STM32 embedded system. Chapter 6 provides a summary and concluding remarks.

Chapter 2

Bluetooth Low Energy Mesh Network

2.1 BLE Mesh Network Configuration

2.1.1 Comparison between wireless communication method

To develop a device which is configurable (in firmware) and capable of connecting electrically to a variety of power generation and energy storage devices (i.e. renewable energy generator, ultra-capacitor systems, hybrid vehicles, etc.) and provide a universal interface to the grid of the future [2], it is necessary to develop a comprehensive data capture and communication capability for the flexible converter to allow converter-to-converter communication of energy transfer data via wireless protocols and to enable remote diagnostic/prognostics.

The options we have for wireless communication technologies include Infrared (IR), Cellular connectivity, Near field communications (NFC), Bluetooth low energy (BLE), Zigbee, Wi-Fi, and Bluetooth classic. IR has been prevalent in the era of flip phones given a desired line of sight connection. However, the communication between two devices based on IR has some significant flaws, such as low bandwidth, short range, and a requirement that the device be positioned within line of sight of each other. This makes it impractical given the need to reposition the converter device. Similarly, regarding the other two technologies, cellular is not feasible given the need for a SIM card in devices and the costly subscription fees. Also, the NFC has a problem of short range. Wi-Fi could be a potentially viable communication method between these converters based on its high data throughput. The Wi-Fi network relies on TCP-IP protocol, which requires all the connected devices obtain their own IP address and authenticate themselves on the network. This is not suitable for the converter devices given the extra requirement for a converter, to include a physical user interface and entering a Wi-Fi password. Also, it results in potential network and data security issues.

ZigBee is a mesh network protocol designed to realize medium range wireless transmissions with a small amount of data. It transfers the message via an inner mesh topology network, which

sends meaningful information from a single node to the gateways across a group of nodes. This explains why it is fairly limited for a high throughput local network, such as in industrial IoT (Internet of Things) applications. Also, it could result in higher latency when ZigBee is applied to high transmission node density networks because of its mesh topology. The comparison between ZigBee and Bluetooth Low Energy is listed as follows.

Table 2.1 Network characteristic of ZigBee and Bluetooth Low Energy (BLE)

	BLE	ZigBee
Network type	Personal area network (PAN), which supports few nodes	Local area network (LAN), which supports many nodes
Range*	77 meters	291 meters
Operation system	Android, iOS, Windows 8, OS X	Not current compatible
Topology	Mesh and star	Mesh only
Throughput	270 kbps	250 kbps
Modulation	Frequency-hopping spread spectrum (FHSS)	Direct-sequence spread spectrum (DSSS)
Transmit power	10 mW	100 mW

A few main characteristics can be seen from table 2.1. The ZigBee protocol could fit well in medium range communication applications when the mesh node has low density network density. However, a few advantages held by BLE makes it more suitable to build a scalable mesh network in universal and scalable converter devices. First, its more flexible topology (i.e. star topology, mesh topology) gives BLE more choices in building different structured wireless connection networks. The combination of these two structures could potentially be capable of

reducing the communication latency since the synchronized scheme in star topology could effectively cut down the possibility that the data packets collide leading to the retransmission of the same packet. The frequency hopping spread spectrum modulation could also reduce the rate of packet collision since two devices share the same channel. Also, the higher throughput of the BLE is a more ideal choice for scalable converter device with higher data transmission requirements. Furthermore, the lower transmission power of BLE with higher data throughput could extend the life time of rechargeable batteries in an energy harvesting scalable network.

Therefore, the BLE based converter mesh network configuration will be discussed in this thesis.

2.1.2 Hardware selection

The X-NUCELO-IDB05A1 is a Bluetooth Low Energy evaluation board based on the STMicroelectronics SPBTLE-RF BlueNRG-MS RF module which is a shield for the STM32 NUCLEO boards. It will be used in the embedded system development.

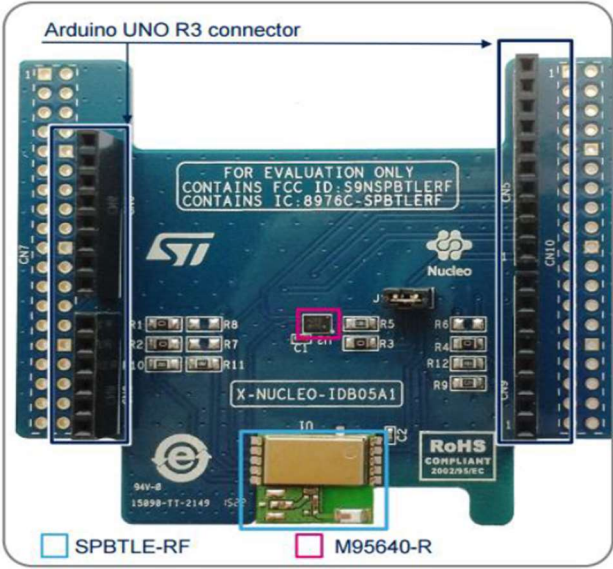


Figure 2.1 X-NUCLEO-IDB05A1

It also interfaces with the STM32 microcontroller via SPI serial communication. A few key features are listed below.

- STM32 expansion board based on the SPBTLE-RF module for SMT32 NUCLEO
- X-NUCLEO-IDB05A1 contains FCC and IC certified module SPBTLE-RF (FCC ID: S9NSPBTLERF and IC: 8976C-SPBTLERF)
- SPBTLE-RF
 - Bluetooth Low Energy FCC and IC certified module based on Bluetooth® SMART 4.1 network processor BlueNRG-MS
 - Integrated Balun (BALF-NRG-01D3) and chip antenna
 - It embeds 32 MHz and 32.768 kHz crystal oscillators for the BlueNRG-MS
- Compatible with STM32 Nucleo boards
- Equipped with Arduino UNO R3 connector
- Scalable solution, capable of cascading multiple boards for larger systems
- Free comprehensive development firmware library and example for BlueNRG-MS, compatible with STM32Cube firmware
- M95640-R has 64-kbit serial SPI bus EEPROM with high-speed clock interface

The mother board of the X-NUCLEO-IDB05A1 used in software development is the STMicroelectronics of NUCLEO-F401RE (Cortex4 Microprocessor). It provides the Arduino UNO R3 connector allowing for easy expansion of the functions of the STM32 Nucleo open development platform. The key features are listed as follows.

- SMT32 32bit microprocessor
- 1 user and 1 reset push-button
- 32.768 kHz crystal oscillator
- Board connectors: Arduino Un V3 expansion connector ST morpho extension pin headers for full access to all STM32 I/Os

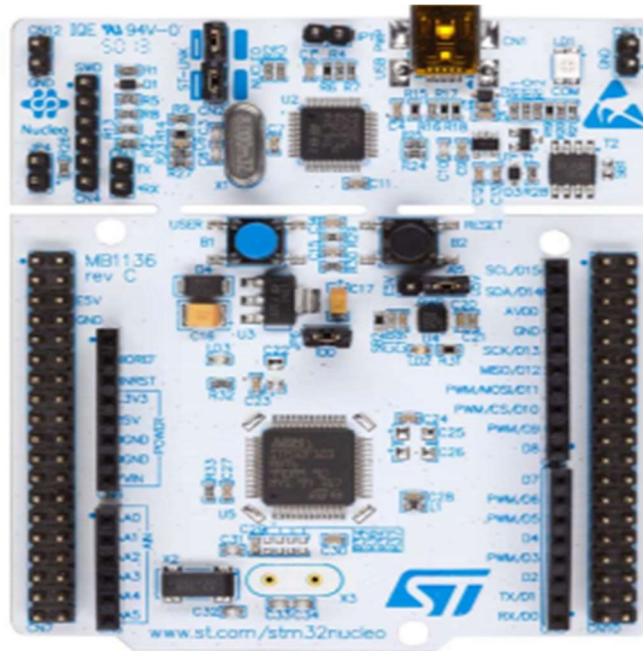


Figure 2.2 NUCLEO-F401RE

As the gateway will involve power converter transmission devices, the collecting data microprocessor requires more advanced data processing capability and larger data storage to lessen risk of receiving lost package or packets with data corruption. Therefore, the STMicroelectronics STM32F769I-DISCOVERY board will be used to collect the amount of data generated in the network. The key features of it are listed as follows.

- STM32F769NIH6 microprocessor featuring 2 Mbytes of flash memory and 512+16+4 Kbytes of RAM, in BGA216 package
- Two push buttons (user and reset)
- 512-Mbit Quad-SPI Flash memory

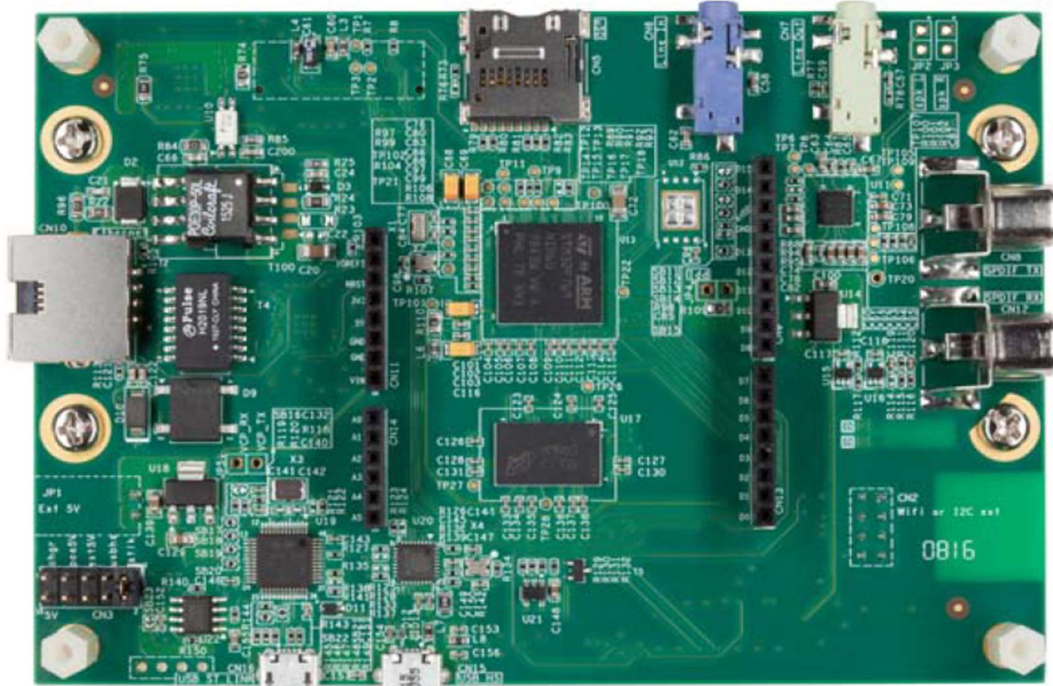


Figure 2.3 STM32F769I-DISCOVERY

2.1.3 Energy Consumption Parameter of BLE

Regarding to the energy consumption associated with BLE communication under various package transmission modes, the BlueNRG current consumption estimation tool also available from STMicroelectronics will be used in the analysis. There are four behaviors of the BLE devices requiring quantized energy which includes advertising, scanning, connection-slave, and connection-master mode. These four working modes on a BLE chip could be used to transport data packets between points, and it's not necessary for the user to specify the transmission protocol.

The BlueNRG current consumption estimation tool v.1.4 is utilized to estimate the energy consumption under the four modes of operation of BLE. figure 2.4 shows the BLE setting of work mode, device type, power supply, master SCA, slave SCA, and the performance of average current and device life-time under a specified battery capacity.

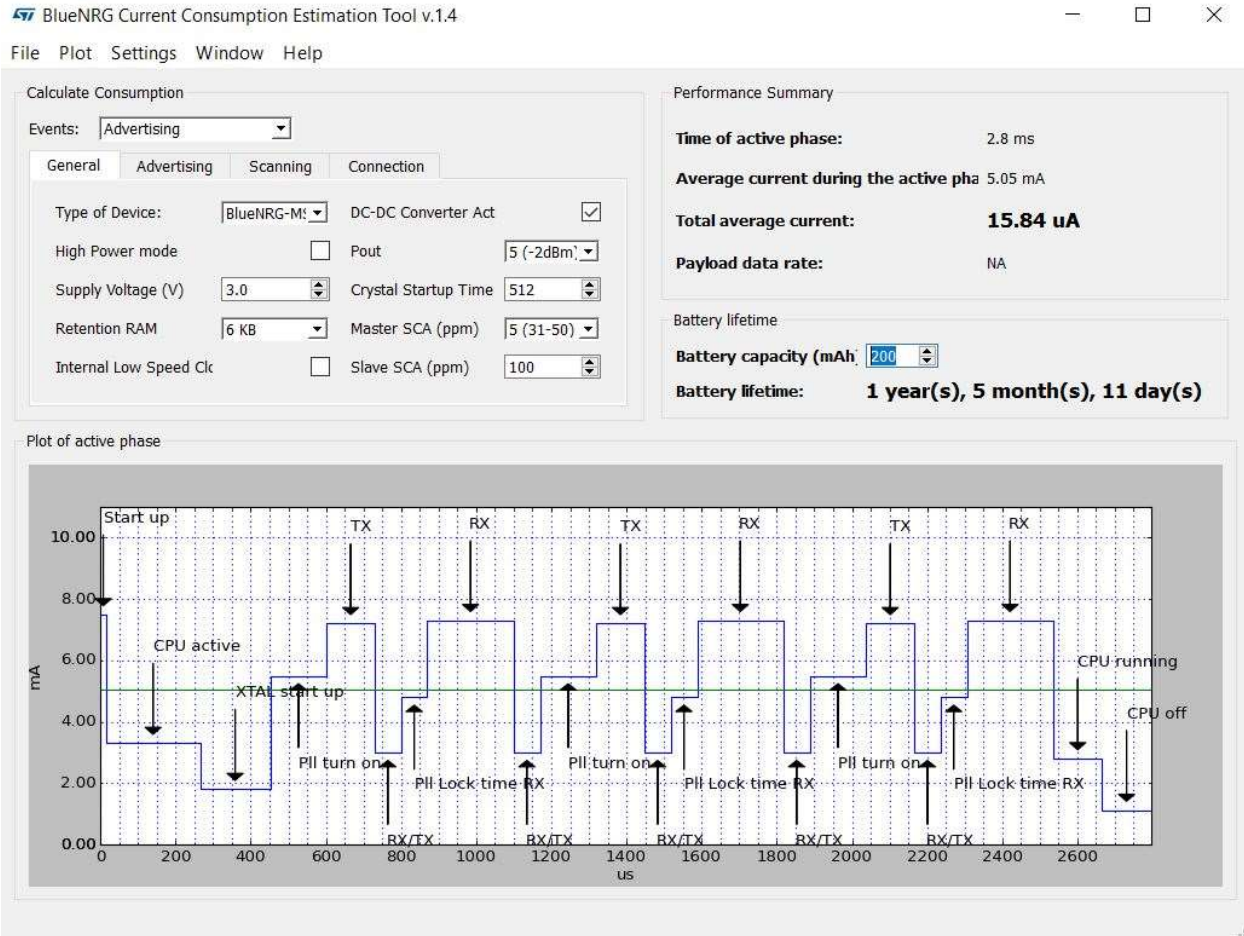


Figure 2.4 Energy consumption estimation tool of BLE

1. Connection-master mode

The master mode set in the BLE module could initiate the connection with a connection-slave mode device. Under this mode, the surrounding devices would start searching and connect with the device which needs to be connected. Figure 2.5 shows the energy cost of one period under master work mode. The average current is 6.16 uA in one period, which helps devices with a 200 mAh battery survive 3 years, 8 months and 17 days.

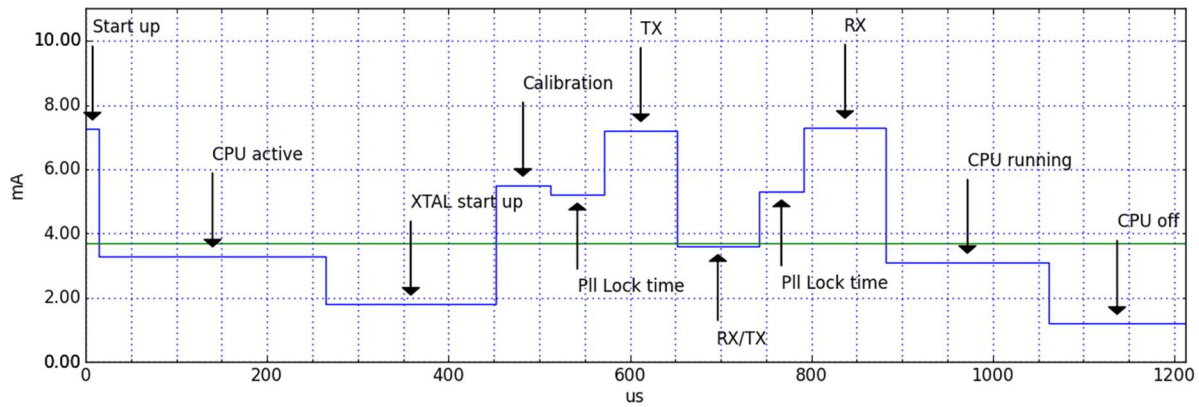


Figure 2.5 Current consumption under connection-master mode in one period

2. Connection-slave mode

BLE allows slave work mode in devices, which includes a service of serial port sending and receiving, and the user could find it with the unique UUID. The user could operate the two functions of writing and reading under the service to achieve the data transmission. Figure 2.6 shows the current consumption estimation under slave mode of BLE operation. The average current is 7.54 uA in one period, which helps devices with a 200 mAh battery survive 3 years, 0 months and 10 days.

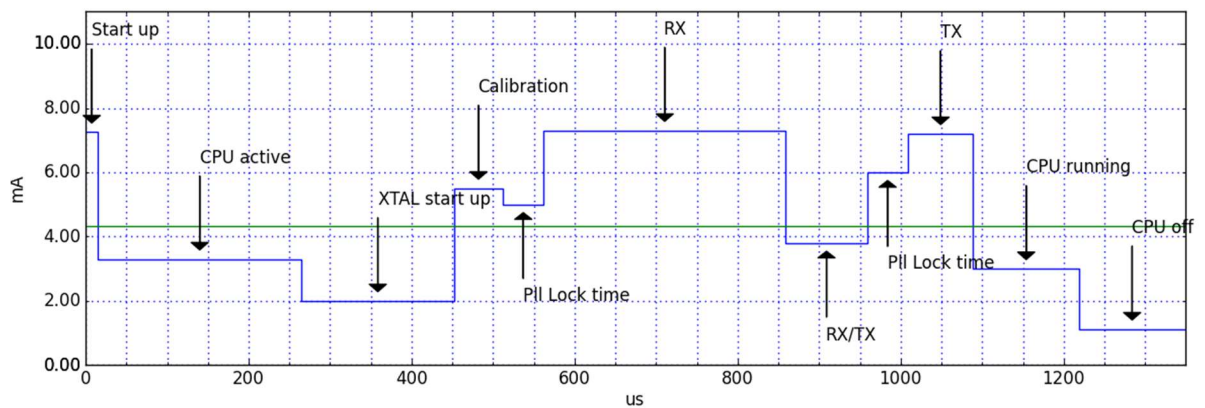


Figure 2.6 Current consumption under connection-slave mode in one period

3. Advertising mode

Under advertising mode, the user can apply an AT command to setup the module to advertise data and the module could constantly keep advertising under low power mode in an ultra-low power, small amount of data, single transmission direction scenario. For example, wireless meter reading. Figure 2.7 shows the current consumption estimation under advertising mode. The average current is 15.84 μA in one period which helps devices with a 200 mAh battery survive 1 year, 5 months and 11 days.

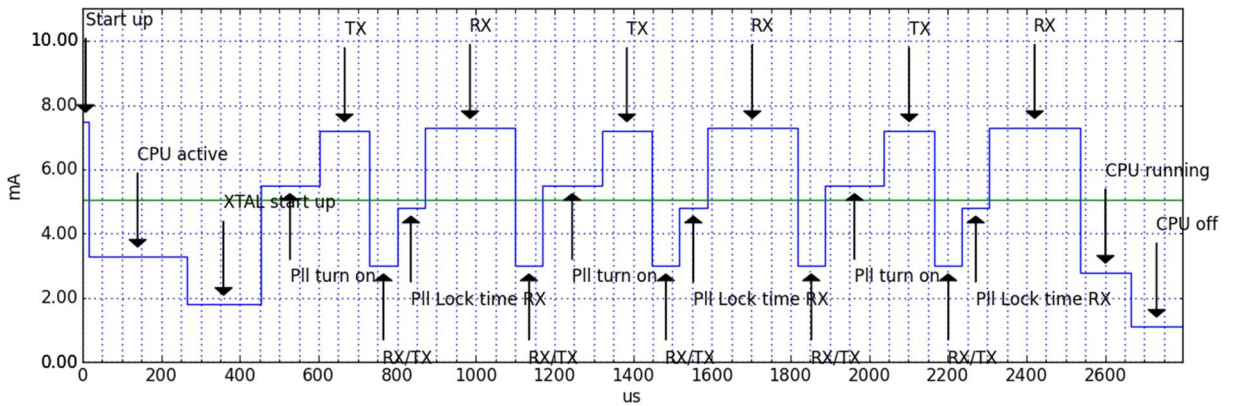


Figure 2.7 Current consumption under advertising mode in one period

4. Scanning mode

Under scanning mode, the BLE could scan the surrounding devices without establishing a connection with other devices under advertising mode. This mode applies to a few scenarios, such as, the remote control of BLE devices, receiving and retransmission of data to the server. The transmission speed under this mode could reach 1 mega bit per second. The transmission distance could span 100 meters in free space. Image 2.8 shows the current consumption estimation under scanning mode. The average current is 3615.85

uA in one period, which helps devices with a 200 mAh battery survive 0 years, 0 months and 2 days.

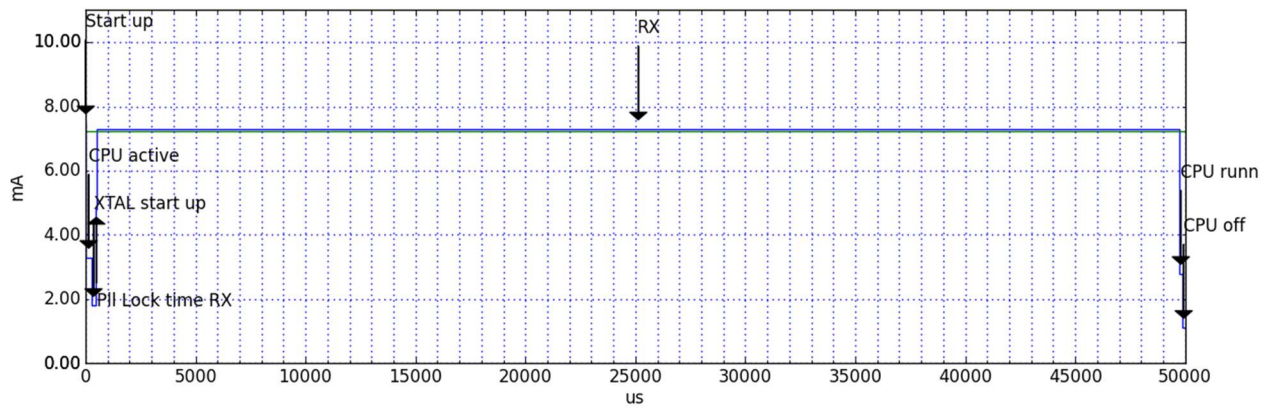


Figure 2.8 Current consumption under scanning mode in one period

2.2 Embedded System Development

The embedded system configuration includes two parts, slave devices and master devices. As shown in figure 2.9, slave devices collect the data from converter sensors and transmit it to the receiving devices or relaying devices in the middle between it and the destination. It is not necessary for the slave device to have a large amount of spare space for data storage or buffering and advanced data processing ability. However, the master device serves as gateway for the entire distributed network which requires more urgent needs of data buffering and storage.

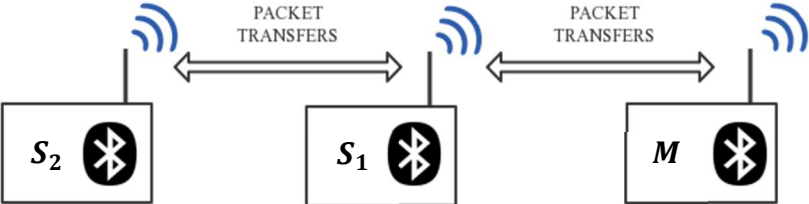


Figure 2.9 Sketch of BLE distribution system

Therefore, as shown in figure 2.10, the slave devices use the STM32F401RE as the data buffer and storage processor and mother board which connects to STM32 BLE expand low energy board X-NUCLEO-IDB05A1, by utilizing the serial peripheral interface (SPI) bidirectional communication method to transfer collected data.



Figure 2.10 Slave device hardware configuration

Similarly, in figure 2.11, considering the requirement of high data processing ability, we added the SMT32F769I-DISCOVERY board as the medium device before the data collected in the

whole network was sent to PLC communication device, which is separated from the task of collecting the data in the SMT32F401RE board. The voltage data is buffered and processed in both two STM32 boards. This configuration reduces the risk of losing packets when larger amounts of power data is collected from distributed converters after more slave nodes have been added to the network. Especially, this issue would be more serious when it is the period of power data peak moment. Also, the bidirectional UART communication method is used in talk of STM32F401RE and STM32F769I-DISCOVERY board.

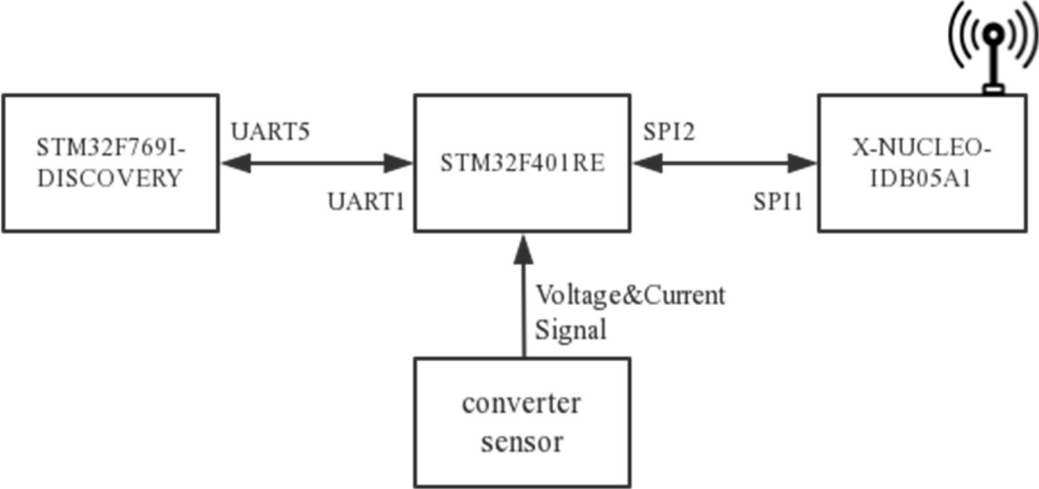


Figure 2.11 Master device hardware configuration

Chapter 3

Reinforcement Learning in Energy Harvesting Mobile Network

In this chapter, to estimate the energy harvesting mobile network energy cost and communication optimal delay, the framework of reinforcement learning, especially Markov decision process, will be discussed. First, the fundamental idea of Markov processes and the derivation of a Markov decision process is explained. Second, following the idea we presented in chapter 2, the EH mobile network model will be described in a Markov decision process (MDP), State-action-reward-state-action (SARSA), and using the Q-learning method. Finally, the exploration and exploitation iteration method to find the optimal value of performance in EH network will be discussed.

3.1 Markov Decision Processes in Reinforcement Learning

The Markov estimation method was presented by Russian mathematician Markov in the theory of stochastic process [21]. It estimates the state transition processes of an object in the system according to probability and statistical theory.

In reinforcement learning, a Markov decision process (MDP) is used to describe an observable environment. A characteristic parameter of decision making depend on the observed state. Considering the universality of MDP, we can apply the MDP into the framework of energy harvesting communication networks in a scalable converter system.

3.1.1 Markov Process

The term ‘‘Markov property’’ refers to the memoryless property of a stochastic process [22]. One can say a stochastic process has the Markov property if the conditional probability distribution depends on the present state, not on the historical event that preceded it [23]. It can be denoted as follows:

$$\mathcal{P}_{ss'} = \mathbf{P} [S_{t+1} = s' \mid S_t = s] \quad (3.1)$$

where $\mathbf{P}[\cdot]$ is the probability function.

The state transition matrix defines all the state transition probabilities:

$$\mathcal{P} = \begin{matrix} & \begin{matrix} S_1 & \cdots & S_n \end{matrix} \\ \begin{matrix} S_1 \\ \vdots \\ S_n \end{matrix} & \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \end{matrix}$$

In this case, n represents the number of states, and the sum of the elements in each row is 1.

A Markov process is known as a Markov chain. It is a stochastic process without memory. It can be represented as a pair $\langle S, P \rangle$, in which S is a finite number of states, and \mathcal{P} is the state transition probability matrix.

3.1.2 Markov Reward Process (MRP)

Markov reward processes add reward R and discount factor γ based on the Markov process.

Therefore, the Markov reward process is comprised of a tuple with four elements $\langle S, P, R, \gamma \rangle$:

S – state of agent, P – transition probability matrix, R – reward, γ - discount factor. The reward of state S denotes the expectation of reward at time $t+1$ under state S at time t :

$$R_s = E [R_{t+1} \mid S_t = s] \quad (3.2)$$

Here $E[\cdot]$ denotes the expectation function.

Discount factor

The parameter $\gamma \in [0,1]$ is the discount factor. We introduce the discount factor for describing the uncertainty of long terms benefits for this multi-agent system but not only focus on the immediate benefits. It effectively stops the process falling into an infinite loop when finding the best route.

Return

Definition: The sum of immediate rewards and rewards with discount from time t on the Markov reward chain. It can be denoted as the following,

$$G_t = R_{t+1} + \gamma R_{t+2} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (3.3)$$

where the discount factor represents the present value scale on the future rewards. The rewards earned at time K+1 moment can be represented as value of $\gamma^k R$ at time t moment. When $\gamma = 0$, it denotes the future rewards given the observed sequence has no influence on this return. When $\gamma = 1$, the future rewards give the same effects as the current rewards to the return.

Value Function

In MDP, the value function can be denoted as the long-term value of a state and an action.

$$v(s) = E[G_t | S_t = s] \quad (3.4)$$

Bellman Equation

We substitute G_t from (3.3) into (3.4) yielding

$$\begin{aligned} v(s) &= E[G_t | S_t = s] \\ &= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \\ &= E[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \dots) | S_t = s] \\ &= E[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= E[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \end{aligned}$$

In the derivation we have $G_{t+1} = v(S_{t+1})$ since the expectation of the reward is the same as the expectation of the reward's expectation. Then we have the bellman equation [19] as follows,

$$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) | S_t = s]$$

We know that $v(s)$ includes the expectation of instantaneous rewards and the expectation of the product of the rewards in the next moment and discount factor respectively. As for the expectation of next moment rewards, it is given by the transformation matrix of state and state value function associated with the next moment. Then, the Bellman equation yields:

$$v(s) = R_s + \gamma \sum_{s' \in S} \mathcal{P}_{ss'} v(s') \quad (3.5)$$

Equation (3.5) can be written in matrix form as:

$$\begin{bmatrix} v_1(s) \\ \vdots \\ v_n(s) \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} + \gamma \begin{pmatrix} P_{11} & \dots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \dots & P_{nn} \end{pmatrix} \begin{bmatrix} v_1(s') \\ \vdots \\ v_n(s') \end{bmatrix}$$

The Bellman equation is a linear equation, so it can be solved directly,

$$\begin{aligned} v &= R + \gamma \mathcal{P}v \\ (1 - \gamma \mathcal{P})v &= R \\ v &= (1 - \gamma \mathcal{P})^{-1}R \end{aligned}$$

The complexity of the computation of this form reaches $O(n^3)$, where n represents the number of states. Thus, the direct solution can only be applied to small MRPs scenarios. To solve a large scale MRP, we may use an iterative method such as dynamic programming or Monte-Carlo evaluations to find numerical solutions.

3.1.3 Markov Decision Process (MDP)

Generally, a MDP is used to describe the interaction between multi-agent systems and the environment [20]. For the energy harvesting network in this thesis, each device connected in the network is homogeneous, which means they have the same data buffer (queue), energy buffer

(queue), and channel jumping characteristic. Therefore, the following discussion of model parameters (i.e. state, action, state value function, action value function, policy) of the EH network would be the same.

The parameters of MDP include the state of agent, action, transition probability, immediate reward, discount factor, and policy. Each parameter is represented as follows.

1. State s of an agent belongs to a set of discrete states of the agent, which can be denoted as $S = \{s_1, s_2, \dots, s_{N_s}\}$, where N_s is the number of possible states. The state on time slot i denoted as s_i , in which $s_i \in S$.
2. A set of discrete actions A of the agent can be denoted as $A = \{a_1, a_2, \dots, a_{N_a}\}$, in which N_a is the number of possible actions for each of state s_i . At time slot i , the action is denoted as a_i .
3. Transition probabilities between states, where $P(s, a, s')$ is the transition probability from state s to next state s' with the action of a .
4. The immediate reward $R(s, a, s')$, which is the reward given when state s transfers to next state s' with action of a .
5. A discount factor $\gamma \in [0, 1]$. It denotes the weight of immediate reward related to the future rewards. The cumulative reward is finite when the discount factor value is less than 1 given that the immediate reward is bounded [21].
6. A policy to define action under different states s . The s_i determined policy, $\pi(s)$, can be defined as mapping from state to action. In reinforcement learning, policy used to find the best action of state s_i to reach the optimal cumulative value of the agent. The expected cumulative reward is given by:

$$E \left[\sum_{i=1}^{\infty} \gamma R(s_i, a_i, s_{i+1}) \mid a_i = \pi(s_i) \right] \quad (3.6)$$

7. The state value function v_π is the expected reward given by the policy started from state s ,

$$v_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{j+k+1} \mid s \right] \quad (3.7)$$

Here, R_{j+k+1} represents the reward received in the $(j+k+1)$ -th step, given that j is the starting time step.

8. The action value function, is the expected cumulative reward starting from state s with action a defined by policy π ,

$$q_\pi(s, a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{j+k+1} \mid s, a \right]. \quad (3.8)$$

Therefore, the optimal state value function and optimal action value function could be given by,

$$v_{\pi^*}(s) = \max_{\pi} v_\pi(s) \quad \forall s \in \mathcal{S} \quad (3.9)$$

$$q_{\pi^*}(s, a) = \max_{\pi} q_\pi(s, a) \quad \forall a \in A, \forall s \in \mathcal{S} \quad (3.10)$$

From (3.7) (3.8) (3.9) (3.10), we have,

$$v_{\pi^*}(s) = \max_{\pi} q_{\pi^*}(s, a) \quad (3.11)$$

The action value function has the recursive form of the Bellman equation,

$$q_\pi(s, a) = \sum_{s' \in \mathcal{S}} p(s, a, s') \left[R(s, a, s') + \gamma v_\pi(s') \right] \quad (3.12)$$

To estimate the action value function in the state-action pairs, the online policy action value function strategy and offline action value function updating strategy could be considered. The typical online strategy is called state-action-reward-state-action (SARSA) strategy, which evaluates the policy used to decide the action with a specific state. The classical offline policy is

called Q-learning strategy, which evaluates the process of action value updating without using the current policy.

The algorithm of SARSA could be explained by the following,

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each step)
  Initialize  $s$ 
  Choose  $a$  from  $s$  using the policy derived from  $Q$  (i.e.  $\epsilon$ -greedy)
  Repeat (for each step):
    Take action  $a$ , observe  $r, s'$ 
    Choose  $a'$  from  $s'$  using the policy derived from  $Q$  (i.e.  $\epsilon$ -greedy)
     $Q(s, a) = Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'; a \leftarrow a'$ 
  Until  $s$  is terminal

```

At this point, the action value function $q_{\pi}(s, a)$ is updated via obtained experience. The updating equation is given by,

$$q_{\pi}^{i+1}(s, a) = q_{\pi}^i(s, a) + \alpha[R(s, a, s') + q_{\pi}^{i+1}(s', a') - q_{\pi}^i(s, a)] \quad (3.13)$$

where the selection of action a obeys the ϵ -greedy strategy described in detail in section 3.3. The computation of goal value Q follows the next step action a' , so it's an online learning strategy. In (3.13), α represents the learning rate in this updating process, which also determines the contribution of newly acquired information for updating action value function. If $\alpha = 0$, the agent would not learn anything from the agent. If $\alpha = 1$, the agent would only consider the newly acquired information [22].

The Q-learning algorithm could be explained as following,

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each step)
  Initialize  $s$ 
  Repeat (for each step)
    Choose  $a$  from  $s$  using the policy derived from  $Q$  (i.e.  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) = Q(s, a) + \alpha[r + \gamma \max Q(s', a) - Q(s, a)]$ 
     $s \leftarrow s'$ 
  Until  $s$  is terminal

```

At this point, the action value function $q_{\pi}(s, a)$ is not updated by the obtained knowledge. The action value function updating equation in Q-learning is given by,

$$q_{\pi}^{i+1}(s, a) = q_{\pi}^i(s, a) + \alpha[R(s, a, s') + \max q_{\pi}^{i+1}(s', a) - q_{\pi}^i(s, a)] \quad (3.14)$$

where the action selection obeys the Q network and ϵ -greedy strategy. The target Q value computation follows the next step action which could generate biggest action value, but it's not necessary to execute this optimal action. Therefore, it's the offline policy that updates the action value.

3.2 MDP simulation

To explain how a package can be transferred within the BLE mesh network, an experiment for transferring a single package within the network is conducted. In addition, an illustrative example provides the context to explain how the MDP could be used for helping the data packet find the lowest delay path in the mesh network.

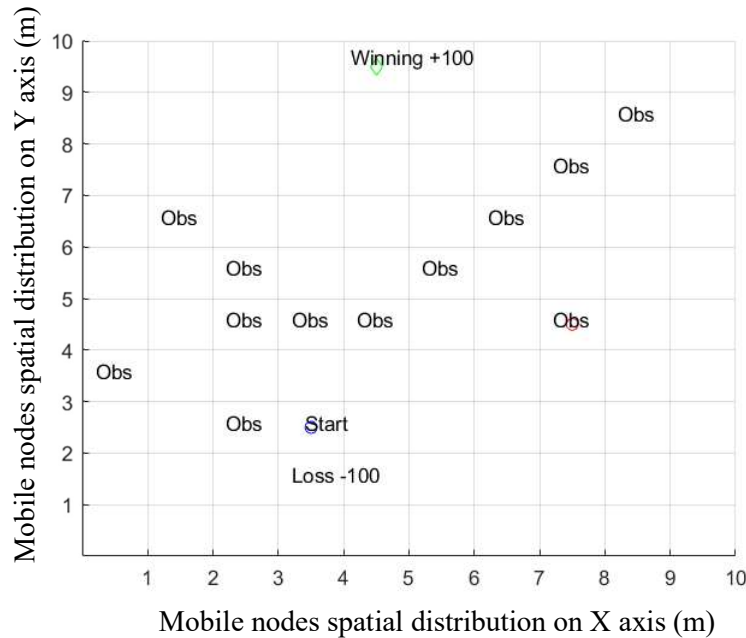


Figure 3.1 Map of package transferring within the mesh network

As we can see from figure 3.1, the starting point denotes the source of the package generated within the network. Obstacles (“Obs” in the grid) represent the area without an available connection channel for transferring this package. The loss metric is set to indicate the scenario that the package was lost in the transfer process. The Winning point means the destination of this package transmission process. Therefore, we intend to find the shortest route in this packet’s transferring map with MDP methodology.

The reward of getting in the winning point is +100 in the cumulative state value. Similarly, to avoid the obstacle and loss point in the grid, the reward for them is set -100 and 0. For each step of movement on this grid, the immediate reward is -1. The value iteration process time was set as 10000 with convergence criteria 10^{-9} . The discount factor of calculating the future reward return

is 1, which indicates the future steps state value has the same influence on the current action selection. We can see the simulation results from figure 3.2 and figure 3.3.



Figure 3.2 State value and step direction choosing in each of the square

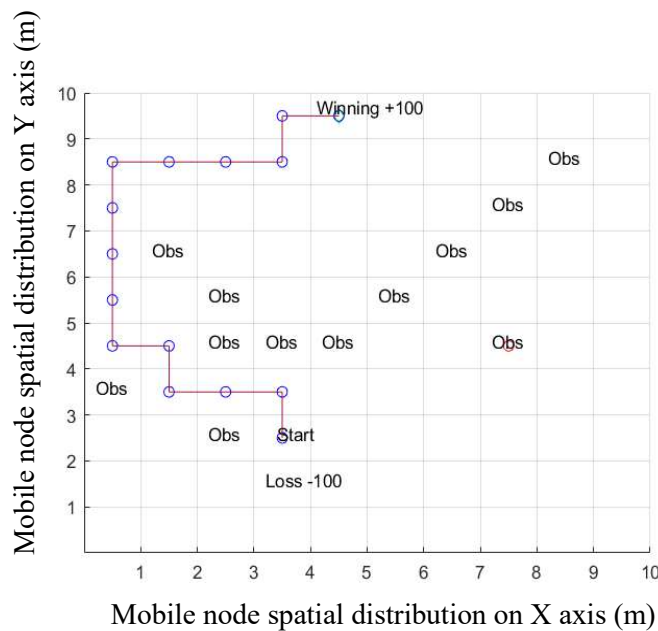


Figure 3.3 Shortest route from starting point to destination

From (3.11) (3.12), the state value function could be calculated by the reverse direction value iteration. The state value function calculation starts from the starting point. What we can see from figure 3.2, figure 3.3 is, the state value for the chosen action choosing is larger when the grid (node) is closed to the destination winning point.

We defined 4 cardinal movement directions in this action choosing, which includes N (north), S(south), W(west), E (east). Also, the action value for the selection of each direction would be different. The figure 3.2 shows the optimized action value choosing for single step c which intends to maximize the action and state cumulative value.

3.3 The exploration Algorithm

The exploration algorithm plays an essential role in reinforcement learning. It is utilized to find the balance between exploration and exploitation and maximize the cumulative rewards. The exploitation mode can be defined as using the current available knowledge to select the best policy to be used. On the other hand, exploration is known as investigating new policies in the hope of getting a policy that is better than the current best one [22].

- The ϵ -greedy algorithm

This exploration algorithm uses the exploration probability ϵ to find a balancing point between exploration and exploitation modes. This parameter changes the mode based on its value at each time slot.

In this algorithm, the current best action is selected with probability $1 - \epsilon$. On the other hand, a random non-greedy action is selected with probability ϵ . The ϵ can be either fixed, or with adaptive value during the learning time [22]. In the case of adaptive ϵ -greedy, ϵ takes values that change with time. For example, ϵ is set to $1/i$, where i is the time slot number. In this case, at the beginning of the session, the exploration probability ϵ has larger value which increases the possibility of exploration. As time goes by, the probability of exploration decrease and the exploitation probability increases. However, most of the policies have been explored and it is referred to exploit the best current policy. In our single agent model, we applied fixed ϵ value as 0.5 in the value iteration since the three nodes point-relay-point communication system does not need much exploration.

Chapter 4

Optimization of Energy Harvesting Mobile Network

4.1 Energy Harvesting Node Model

4.1.1 MDP parameter setting and assumption

In this chapter, a point to point communication system with a middle relaying agent will be discussed. The system consists of a source agent (SR), a relaying agent (RE) a destination agent (DE). As shown in image 4.1, agents SR, RE, and DE are equipped with infinite data buffers to store data. All three agents are capable of harvesting renewable energy and store it in a finite battery. A time-slotted system with time slots of equal length would be considered. Each time slot consists of two equal sub-slots. The first sub-slot is used to transmit data, receive packets from the other agents, or buffer the packets. The second sub-slot is used to harvest renewable energy. Image 4.2 illustrates how this process is accomplished.

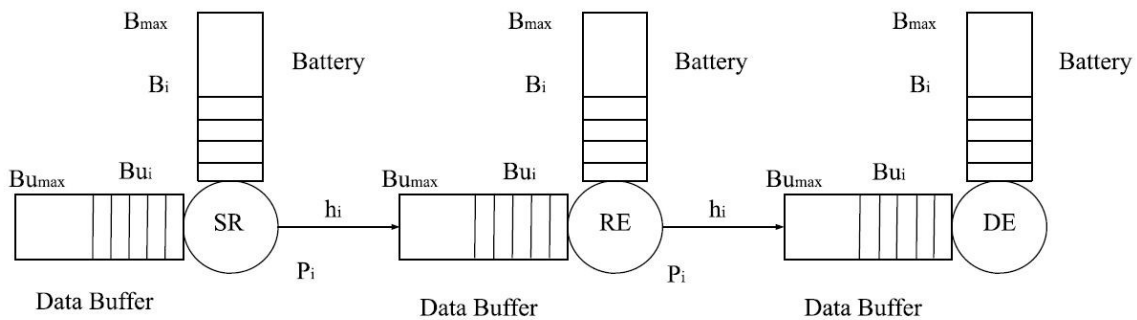


Image 4.1 Source-Relay-Destination communication system model with finite battery space and data buffer

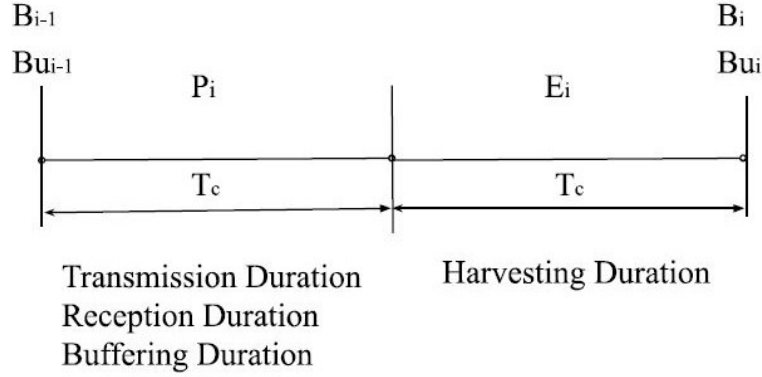


Figure 4.2 Slotted system model

In this chapter, the energy consumption and storage mechanism, such as renewable energy harvesting, storage, and consumption of data transmission are quantized in an integer multiple of a fundamental energy unit. The renewable energy harvesting process will be triggered after initializing of a data sending or receiving process. Also, comparing the amount of energy cost for data transmission and reception with the data buffering and storage, we assume the data buffering and the stored process would not consume energy. The parameter in the node model description will be listed as follows.

- 1) The battery has a limited storage capacity of B_{max} . Let B_i denote the battery charge level of SR, RE, and DE at the beginning of time slot i , where

$$B_i \in B = \{b_1, b_2, \dots, b_{N_B}\}, b_{N_B} = B_{max}, \text{ and } N_B \text{ is the number of elements in } B.$$

- 2) During time slot i , the amount of package receiving is denoted by Re_i , where

$$Re_i \in RE = \{Re_1, Re_2, \dots, Re_{N_E}\}, \text{ and } N_E \text{ represents the number of elements in } RE. \text{ For the received data packets, the transmission probability from state } Re_i \text{ to } Re_j \text{ during time slot } i \text{ is given by } P_{RE}(Re_i, Re_j).$$

- 3) The agent buffer state during time slot i is given by Bu_i , where

$$Bu_i \in BU = \{bu_1, bu_2, \dots, bu_{N_B}\}, \text{ and } N_B \text{ denotes the number of elements in } BU. \text{ The}$$

buffer state transmission probability from state bu_i to state bu_j during one time slot is given by $P_{BU}(bu_i, bu_j)$.

4) The harvested energy of the agent during time slot i , is denoted by E_i , where

$E_i \in E = \{e_1, e_2, \dots, e_{N_E}\}$, and N_E represents the number of elements in E . For harvested energy, the transmission probability from state e_i to state e_j during one time slot is given by $P_E(e_i, e_j)$.

5) The transmission and receiving channel state of the agent at time slot i is given by H_i ,

where $H_i \in H = \{h_1, h_2, \dots, h_{N_H}\}$, and N_H denotes the number of elements in H . The channel transmission probability from state h_i to state h_j is given by $P_H(h_i, h_j)$.

Let the data transmission and receiving power during time slot i be denoted by P_i^T , where

$P_i \in P = \{p_1, p_2, \dots, p_{N_p}\}$, and N_p is the number of elements in P . Let T_c be the transmission and reception duration, which is the fixed value of 1 sec during all time slots.

For the harvested energy node model, each state s_j of the node consists of 3 elements, which include battery level of agent, data buffer level, and channel gain. It can be represented as $s_j = \{b_j, bu_j, h_j\}$. In this context, the state satisfies the Markov property, where the future state depends only on the current state, which is independent of the previous state in the other time slots.

Based on the current battery level, the agent could select the action, include receiving, sending, or caching, that maximizes the sum rate (throughput) and channel gain of a single agent, and minimizes the level of variation in the battery. Therefore, the immediate reward for the agent state transforming during time slot i is given by

$$R_i = \log_2 \left(1 + \frac{|h_i|^2 b_i}{bu_i} \right) \quad (4.1)$$

In this model, energy consumption is considered only in connection with data transmission and receiving, and it does not take into account any other energy consumption, such as processing, circuitry, storage, etc.

4.1.2 Channel random access model

As we defined in the previous chapter about the MDP model for a single agent, we need to consider the case of multiple channel access of a single node, which means that channel collision would happen when multiple devices are intended to access one node for receiving messages. Assume the converter system includes a wireless access point (master node) and N slave devices, where each device has finite energy and infinite data storage, as shown in image 4.3. Time is slotted and the duration of each time slot is T .

Random channel access model

Define s_i as the state of device i at time t , which includes the level data buffer, battery level, and channel gain. At each slot, each device makes the access decision according to its state. The device is able to access the network if there is enough energy.

The successful connection between devices does not only depend on the action of the sender and receiver, but also on the other available nodes within the transmission range, especially when more than one device is eager to access the master node simultaneously.

The probability of successful access can be formulated as follows

$$P_s = a_i \prod_{j \neq i}^N (1 - a_j) \quad (4.2)$$

where a_i is the successfully access probability of device i . According to (4.2), the device can only access successfully if all other devices are denied access by the master device. In the scenerio of one channel, the other devices would possibly lose the data packets. At each time slot, the access attempt of these devices may fail, but would still consume the energy. In the random access model we defined here, all the devices are not cooperative, so they should adopt the corresponding strategy based on considering its state and the influence of other devices, which intends to maximize the power consumption of all nodes in the network [21].

4.1.3 Parameter Setting

In this section, the setting of the parameter would be investigated for the ϵ -greedy exploration algorithm in SARSA method and Q-learning. In the numerical experiment, it is assumed that each time slot consists of two equal sub-slots, each of the lasting 1 sec duration. During the first sub-slot, the agent transmits its package to the other receivers or receives the package from the other agents. The available bandwidth for BLE is 2.4GHz for signal and noise. The discount factor γ is 0.5, which indicates the returning rate of rewards is 0.25 of the original rewards. The learning rate is set to 0.1. All results are derived from 1000 iteration cycles to find the convergent value of the agent state.

In the experiment, SR, RE, DE are all equipped with solar panels with an area of 100 cm^2 with 10% harvesting efficiency. Where an outdoor solar panel can get benefits of 10 mW/cm^2 , solar irradiance under standard environments with harvesting efficiency is between 5%-30%, which depends on the used material in the panel [22]. Therefore, we assume the fundamental energy cost unit for energy harvesting, stored, and consumption is 30 mJ.

In the simulations, the energy harvesting state is $E = \{0,1\}$. This means the harvested energy in half of the time slot time would only be able to transmit the fundamental energy unit with probabilities $P_E(e_i, e_j) = 0.8$. Assume the interval of channel gain of agent be

$H = (0.02 * 10^{-7}, 6.0 * 10^{-7})$ with transition probabilities $P_H(h_i, h_j) = 0.95$. The equipped battery size accommodates a maximum capacity of 2 units.

4.2 Analysis of training method

In this part, the evaluation for the performance of point-relay-point wireless communication system model will be discussed under the SARSA learning algorithm and Q-learning algorithm using ϵ – *greedy* based exploration algorithm, where it would be compared with the optimal scenario. The optimal policy is using the trained optimal policy in the chosen action with a specific state. This yields the upper bond of performance of the agent. However, this scenario requires prior statistical knowledge of the environment, which is not applicable for the agent used in this example.

In the SARSA experiment, the fixed ϵ value is adopted in ϵ – *greedy* algorithm. In this algorithm, a different fixed ϵ value is applied. Image 4.3 explained the training process of a point-relay-point communication system. The stopping criteria for training is that the cumulative value reaches the set averaging reward value. The episode reward denotes the action value cumulative value with the increasing of simulation step. The training process involves 200 episodes, which includes 50 steps in per episode.

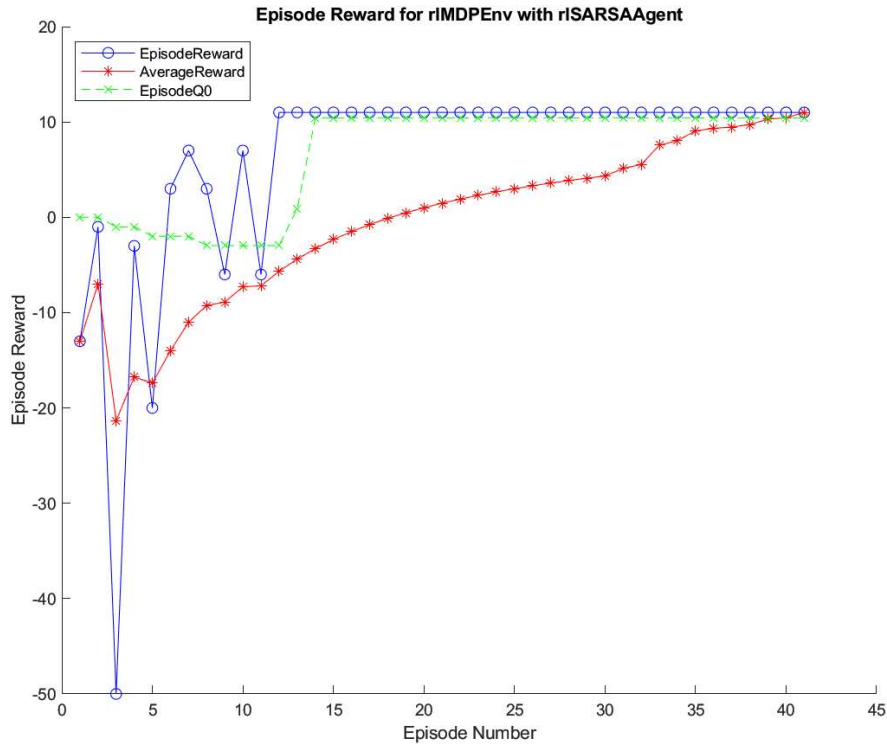


Figure 4.3 Action value with SARSA training

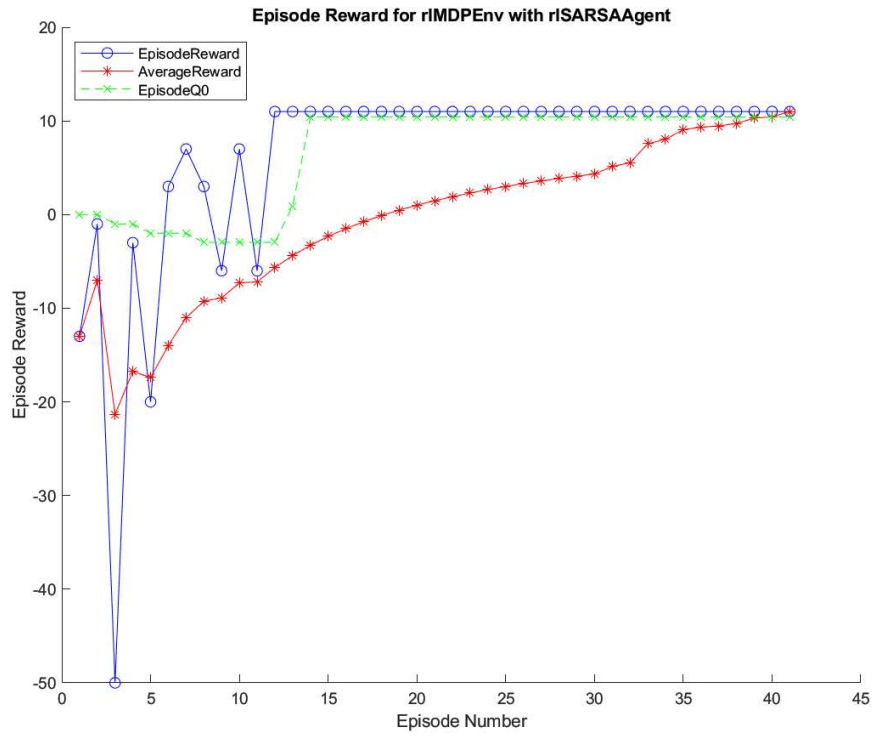


Figure 4.4 Action value with SARSA training ($\epsilon=0.04$)

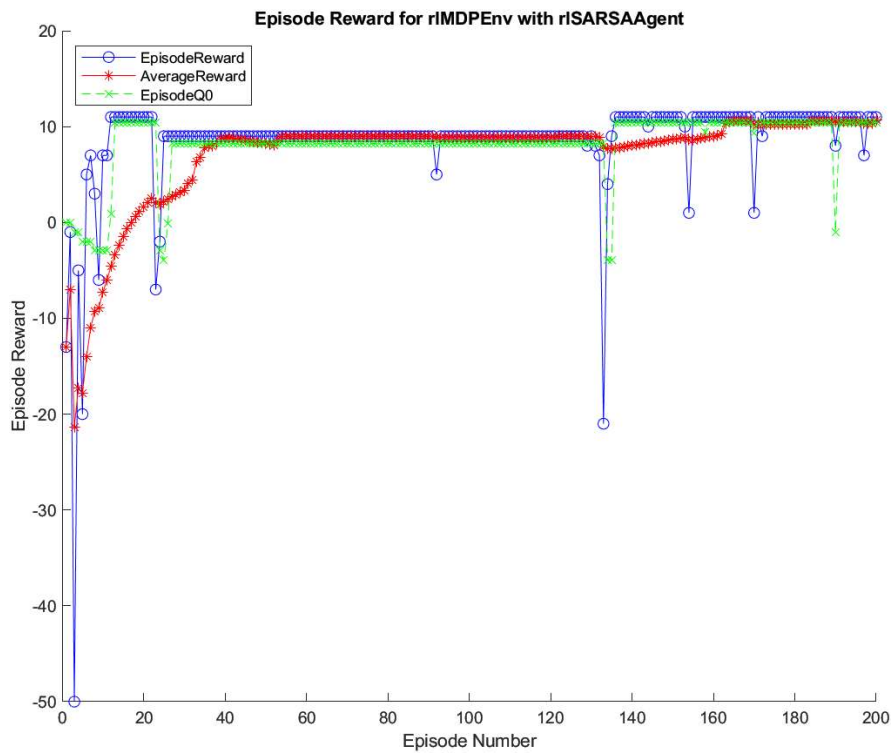


Figure 4.5 Action value with SARSA training ($\epsilon=1$)

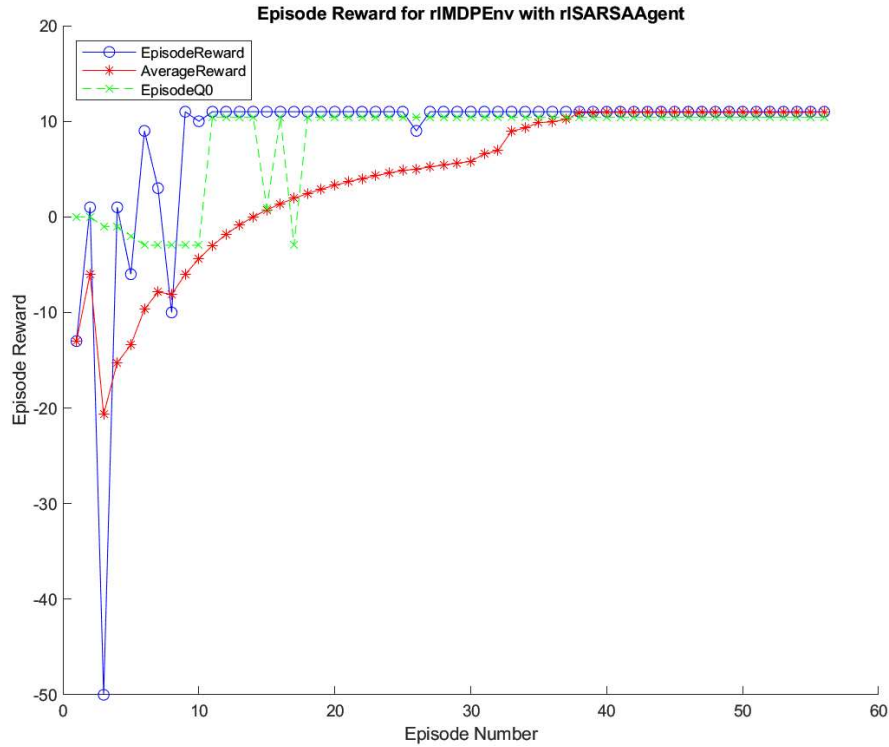


Figure 4.6 Action value with SARSA training ($\epsilon=0.01$)

As shown in figure 4.4, figure 4.5, figure 4.6, the action value reward with different ϵ parameter value settings for the ϵ -greedy algorithm has different training convergence speed. At the beginning of the session, it can be noticed that the episode reward value reaches -50 with all three different ϵ settings. This indicates that the agent is currently under the exploration for an optimized iteration policy. After that, the episode reward with $\epsilon=0.04$ reaches the constant rewards value in 13 episodes training. Reward value with $\epsilon=1$ reaches a constant value in almost 30 episodes training. Reward value with $\epsilon=0.01$ reaches a constant value in 10 episodes of training. We can see from this that the ϵ parameter decides the probability of exploration in the SARSA ϵ -greedy algorithm. When $\epsilon=1$, the agent has larger probability values in exploring the optimal policy to execute the action under the current state. This could have the more benefits in a complex system when it's not easy to find a balanced and optimized strategy. However, in our point-relay-point wireless communication system, we would choose the small ϵ value to reach faster action value reward convergence.

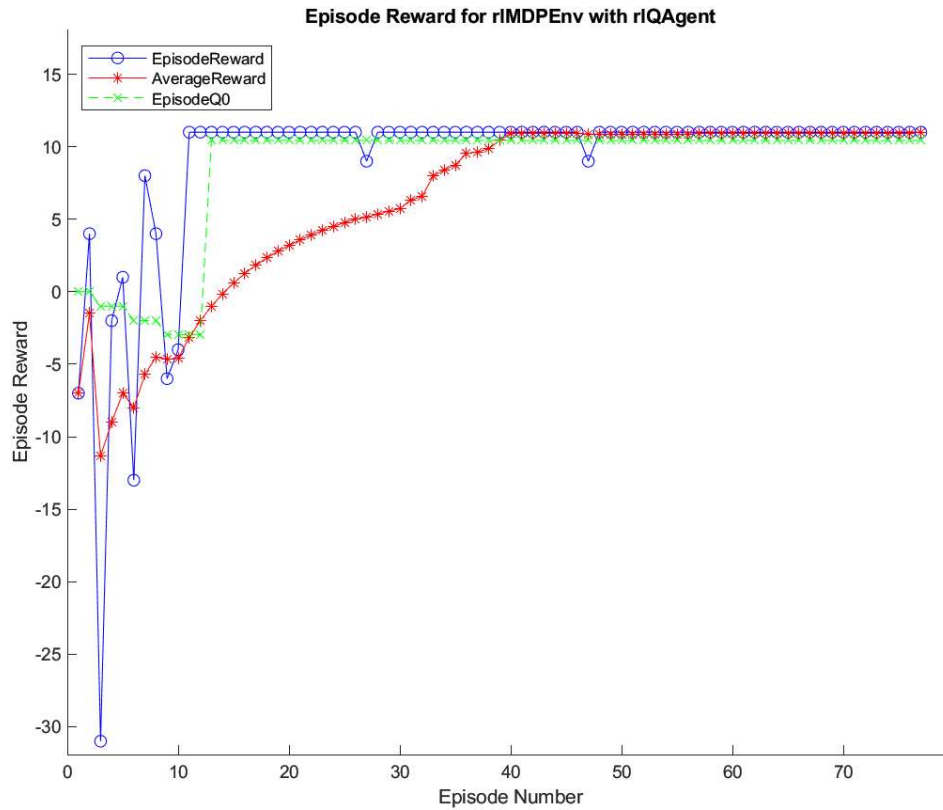


Figure 4.7 Action value with Q-learning training ($\epsilon=0.04$)

In the Q-learning experiment, the fixed ϵ value is adopted in the ϵ -greedy algorithm. In this algorithm, a different fixed ϵ value is applied. The training process set has 200 episodes, which includes 50 steps per episode. The stopping criteria for stop training is that the cumulative value reaches the set averaging reward value of 11.

We can see from figure 4.7 and figure 4.3, the SARSA and Q-learning method both reach the constant action value in 13 episodes, which indicates the same speed of convergence with same ϵ value setting. However, the Q-learning training takes more episodes to reach the stop training criteria average reward, which suggests the SARSA could finish the training process faster.

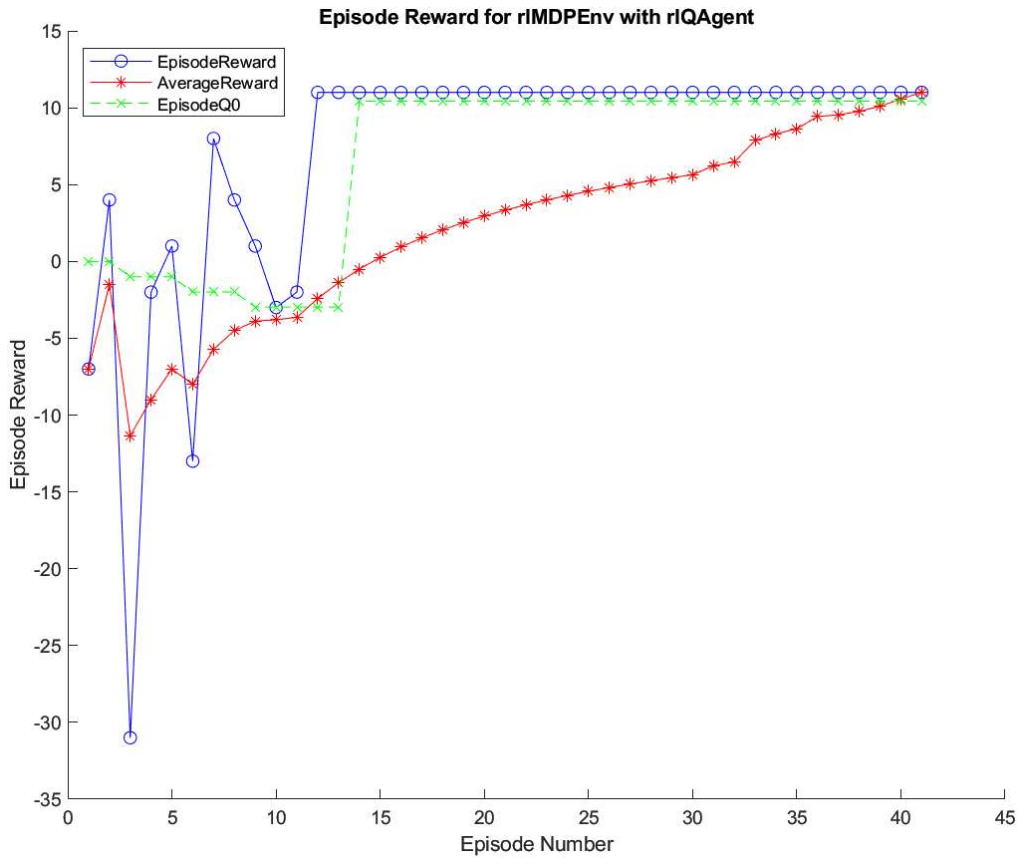


Figure 4.8 Action value with Q-learning training ($\epsilon=0.01$)

Similarly, since the Q-learning method takes ϵ -greedy algorithm to balance the exploration and exploitation process in finding the optimal policy for action choosing, bigger ϵ value could help the agent reach the constant reward faster, which suggests that the policy could be further optimized.

Chapter 5

Simulation results

5.1 Scalable network analysis

In this section, we set up two simulations to test the throughput and collision rate of the EH mobile node within distributed converter network. In here, we assume the estimated ideal throughput of device node is 450 data packets per second, in which each packet contains 8 bits of power data. In here, the throughput of the mobile node suggests the average throughput of the whole network. The average throughput calculated by computing the variance of data buffer state in the source-relaying-destination model, which represents the average data package throughput of a single node within the network.

The channel sharing for two or multiple competitive node uses channel random access model in section 4.2. The packets transmission collision rate in the simulation suggests the probability of sharing channel collision when multi-client nodes are intended to access the single channel, which cause the package lost in the process of data transmission. To calculate the collision rate, we use the channel as the tool. The low channel gain set as the channel sharing happen in the same time, which cause the package transmission collision or package lost. The high channel gain set as the channel sharing of multi-client devices happen in different time interval during the transmission process. In the following simulation, the product of the collision rate and the number of nodes denotes packets dropping unit in the 100 packets transferring process.

In the test of average throughput and collision rate, the Q-learning parameter learning rate α set as 0.1 with discount value γ 0.95, and the ε value in greedy algorithm is 0.04, which balance the exploration and exploitation of the learning process.

As shown in figure 5.1, the average throughput keeps the value nearly ideal throughput of single node under optimized policy when the number of mobile device node is less than 8. However, the throughput drops instantaneously when the number of node exceeds 8 nodes and it holds the throughput around 50 packets in the rest of node increasing process. A few possible reasons to

explain this situation are the low value of ε constrains the ability of the network node to explore new policy for data buffering and transmission, especially in the end of training process with the adding of new nodes into the network.

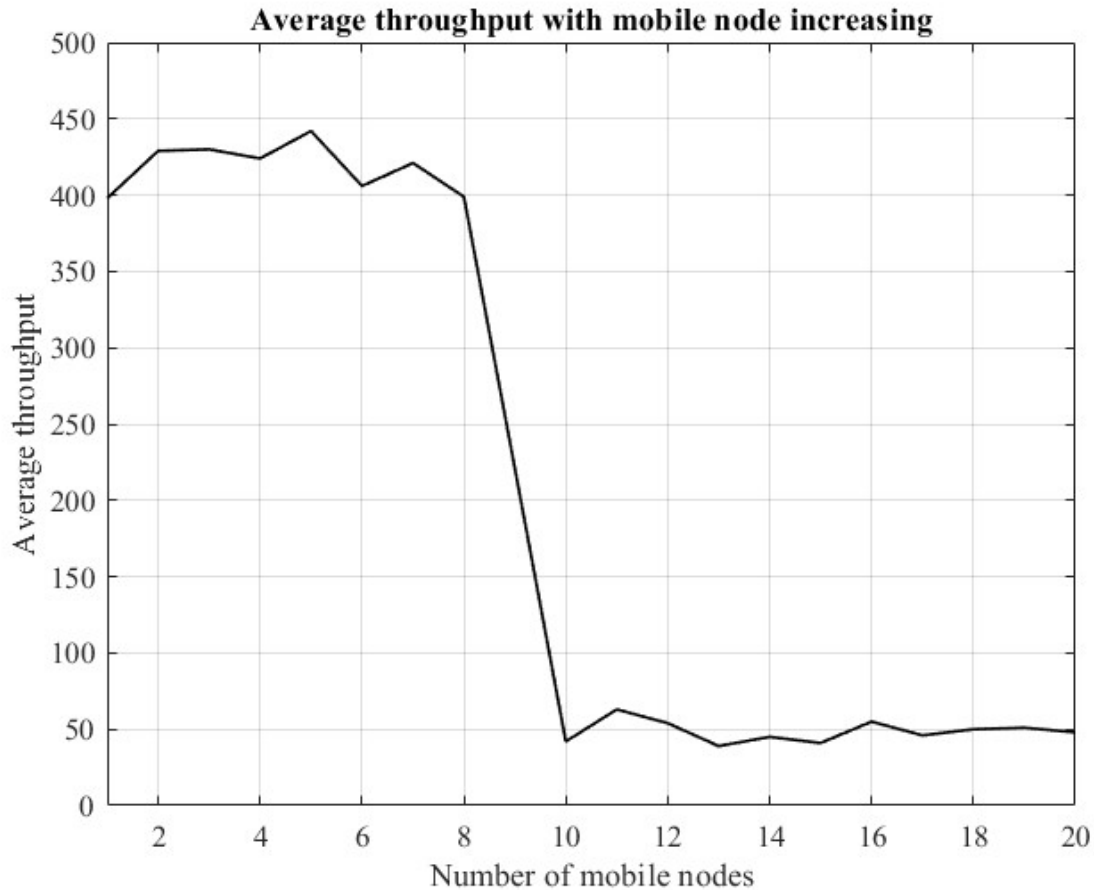


Figure 5.1 Average throughput dynamics

As we can see from figure 5.2, the collision rate drops when the number of nodes is 2, which also reaches the lowest value in the whole process of node increasing since there is no collision exists for two node transmission (sender and receiver pair). In the following, with the increasing of network nodes, the collision rate remains around 0.05. However, since of the increasing of mobile nodes number, the number of packets dropping is increased. The collision rate of the first 20 nodes shows the increasing collision rate tendency. Therefore, we can estimate that the number of packets dropping is increasing with the adding of the mobile nodes within the network.

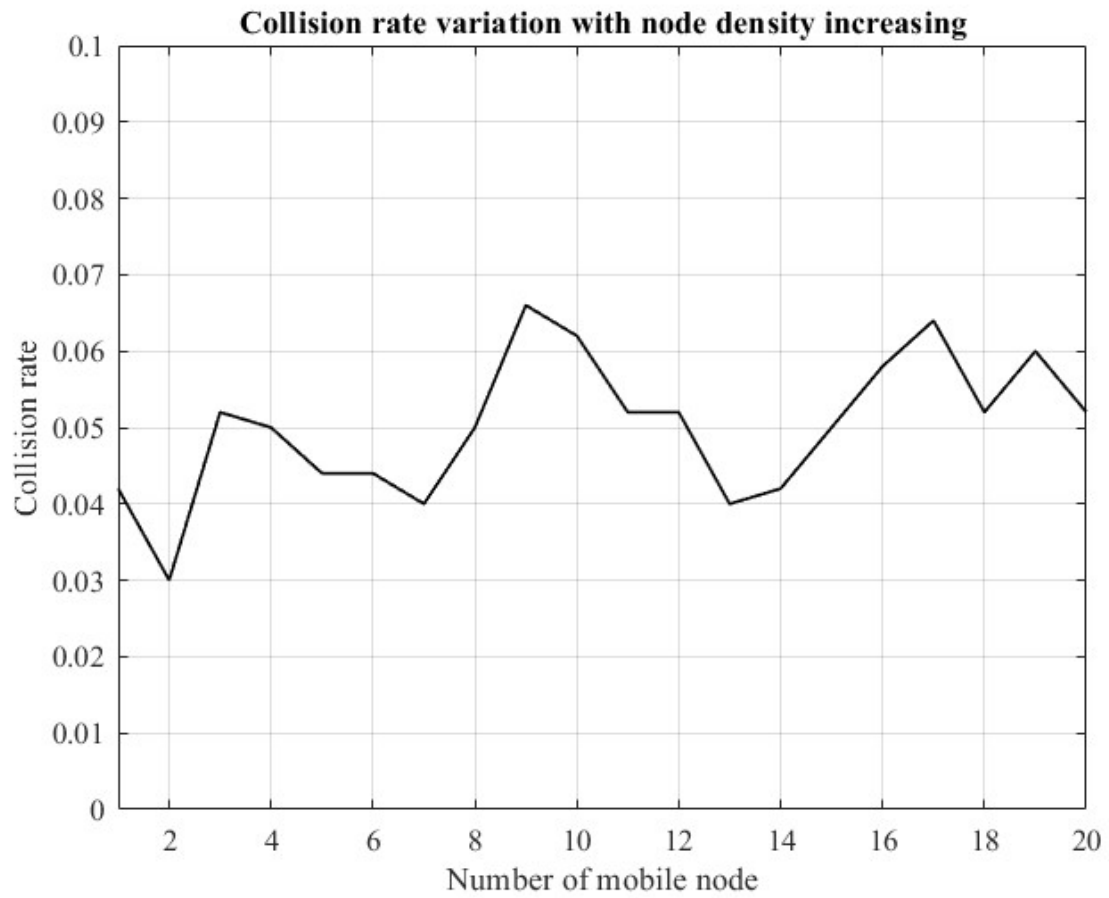
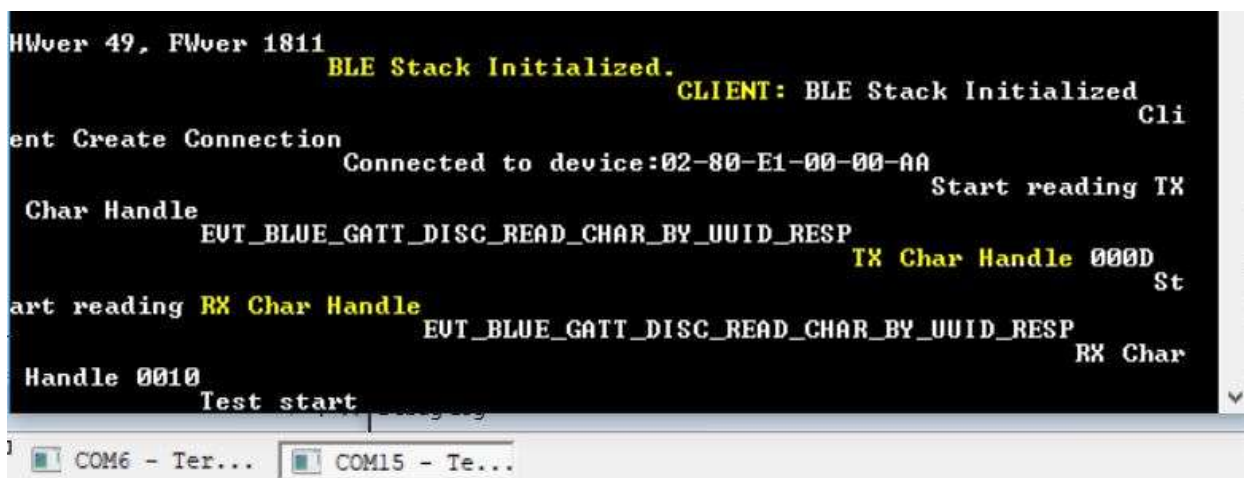


Figure 5.2 Average collision rate dynamics

5.2 Test on embedded system

In this session, the BLE communication hardware development testing results will be discussed. Figure 5.3 and figure 5.4 demonstrates the process and functions of the client(slave) and server(master) board from BLE network configuring process to data packages exchanging process.

From figure 5.3, we can see that BlueNRG version of the Bluetooth shield board on client side, and the connected board information when the client board established the connection successfully. After that, the client board starts to execute a transmission function on the EVT_BLUE_GATT_DISC_READ_CHAR_BY_UUID_RESP profile and then start receiving data packages from another board. This process repeats until 500 packages are transferred.



```
HWver 49, FWver 1811
BLE Stack Initialized.
CLIENT: BLE Stack Initialized
ent Create Connection
Connected to device:02-80-E1-00-00-AA
Char Handle
EVT_BLUE_GATT_DISC_READ_CHAR_BY_UUID_RESP
TX Char Handle 000D
Start reading TX
RX Char Handle
EVT_BLUE_GATT_DISC_READ_CHAR_BY_UUID_RESP
Handle 0010
Test start
```

Figure 5.3 Information of data sending on client (slave) side

From the server(master) side, we can check the throughput of this application and the data exactly transmitted from the client side. We can also calculate the transmission speed in this procedure by quantifying throughput and elapsed transmission time. Specially, the data received here is not the original message received in the 500 packages. The separated receiving data is used for showing the random number generated by STM32F7 board.

The screenshot shows a Tera Term window titled "COM6 - Tera Term VT". The window contains a series of data reception statistics. Each line of statistics follows a similar pattern: "Data received is: 536871932", "500 packets. Elapsed time: 382 ms. App throughput: 209 kbps.", and "Data received is: 536871932". The statistics are repeated multiple times, with some lines having a yellow highlight. The window also has a menu bar with "File", "Edit", "Setup", "Control", "Window", and "Help". At the bottom of the window, there are two tabs: "COM6 - Ter..." and "COM15 - Te...".

```
VT COM6 - Tera Term VT
File Edit Setup Control Window Help
Data received is: 536871932
500 packets. Elapsed time: 382 ms. App throughput: 209
kbps.
Data received is: 536871932
500 packets. Elapsed time: 382 ms. App th
roughput: 209 kbps.
Data received is: 536871932
500 packets. Elapsed time: 3
82 ms. App throughput: 209 kbps.
Data received is: 536871932
500 packets. El
apsed time: 381 ms. App throughput: 209 kbps.
Data received is: 536871932
500 packets. El
apsed time: 382 ms. App throughput: 209 kbps.
Data received is
: 536871932
500 packets. Elapsed time: 382 ms. App throughput: 209 kbps.
Data
a received is: 536871932
500 packets. Elapsed time: 382 ms. App throughput:
209 kbps.
Data received is: 536871932
500 packets. Elapsed time: 382 ms. Ap
p throughput: 209 kbps.
Data received is: 536871932
500 packets. Elapsed tim
e: 381 ms. App throughput: 209 kbps.
Data received is: 536871932
500 packets
. Elapsed time: 382 ms. App throughput: 209 kbps.
Data received is: 5368719
32
500 packets. Elapsed time: 382 ms. App throughput: 209 kbps.
Data receive
d is: 536871932
```

Figure 5.4 Information of data receiving on the server (master) side

Chapter 6

Conclusion

In this thesis, for designing a scalable distributed converter network with the device nodes having renewable harvesting ability, a Markov decision process (MDP) model using ϵ -greedy based SARSA and Q-learning method is developed and simulated. To explore the relation between Markov decision process and energy harvesting scalable network, Markov property and Markov rewards property has been introduced for the memoryless and policy iteration characteristic of MDP. A single package transmission process in the network with obstacles has been simulated. Then a online learning SARSA and offline learning Q-learning method has been introduced and applied to the following training process analysis of energy harvesting node. In the end, a simulation based on Q-learning approach has been conducted. The results show the Q learning method could improve the performance of the distributed network when the number of nodes is less than 10. In addition, the STM32 hardware development has been conducted to realize the point-relaying-point communication system, where the test results from the serial port show applicable results.

Reference

- [1] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in Proc. Int. Conf. Machine Learning, New Brunswick, July 1994, pp. 157-163
- [2] Vincent J. Winstead, "Universal and Scalable Smart Grid Power Converter", Jun. 2016.
- [3] Y. Xiao, M. Peng, J. Gibson, G. G. Xie, D. Du and A. V. Vasilakos, "Tight Performance Bounds of Multihop Fair Access for MAC Protocols in Wireless Sensor Networks and Underwater Sensor Networks," in *IEEE Transactions on Mobile Computing*, vol. 11, no. 10, pp. 1538-1554, Oct. 2012.
- [4] P. Deshpande and M. S. Madankar, "Techniques improving throughput of wireless sensor network: A survey," *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, Nagercoil, 2015, pp. 1-5.
- [5] Jian Li and P. Mohapatra, "An analytical model for the energy hole problem in many-to-one sensor networks," *VTC-2005-Fall. 2005 IEEE 62nd Vehicular Technology Conference, 2005.*, Dallas, TX, USA, 2005, pp. 2721-2725.
- [6] J. Lin and M. A. Ingram, "SCT-MAC: A scheduling duty cycle MAC protocol for cooperative wireless sensor network," *2012 IEEE International Conference on Communications (ICC)*, Ottawa, ON, 2012, pp. 345-349.
- [7] V. Sharma, U. Mukherji, V. Joseph and S. Gupta, "Optimal energy management policies for energy harvesting sensor nodes," in *IEEE Transactions on Wireless Communications*, vol. 9, no. 4, pp. 1326-1336, April 2010.
- [8] Ahmad Almadhor, 2018. "Feedback-Oriented Intelligent Monitoring of a Storage-Based Solar Photovoltaic (PV)-Powered Microgrid with Mesh Networks," *Energies*, MDPI, Open Access Journal, vol. 11(6), pages 1-18, June.
- [9] Ortiz, Andrea & Al-Shatri, Hussein & Weber, Tobias & Klein, Anja. (2017). Multi-Agent Reinforcement Learning for Energy Harvesting Two-Hop Communications with Full Cooperation.

- [10] S. Luo, R. Zhang and T. J. Lim, "Optimal Save-Then-Transmit Protocol for Energy Harvesting Wireless Transmitters," in *IEEE Transactions on Wireless Communications*, vol. 12, no. 3, pp. 1196-1207, March 2013.
- [11] J. Yang and S. Ulukus, "Optimal Packet Scheduling in an Energy Harvesting Communication System," in *IEEE Transactions on Communications*, vol. 60, no. 1, pp. 220-230, January 2012.
- [12] P. Blasco, D. Gunduz and M. Dohler, "A Learning Theoretic Approach to Energy Harvesting Communication System Optimization," in *IEEE Transactions on Wireless Communications*, vol. 12, no. 4, pp. 1872-1882, April 2013.
- [13] B. Devillers and D. Gündüz, "A general framework for the optimization of energy harvesting communication systems with battery imperfections," in *Journal of Communications and Networks*, vol. 14, no. 2, pp. 130-139, April 2012.
- [14] M. Miozzo, L. Giupponi, M. Rossi and P. Dini, "Distributed Q-learning for energy harvesting Heterogeneous Networks," *2015 IEEE International Conference on Communication Workshop (ICCW)*, London, 2015, pp. 2006-2011.
- [15] M. Miozzo, L. Giupponi, M. Rossi and P. Dini, "Distributed Q-learning for energy harvesting Heterogeneous Networks," *2015 IEEE International Conference on Communication Workshop (ICCW)*, London, 2015, pp. 2006-2011.
- [16] A. Ortiz, H. Al-Shatri, X. Li, T. Weber and A. Klein, "Reinforcement learning for energy harvesting point-to-point communications," *2016 IEEE International Conference on Communications (ICC)*, Kuala Lumpur, 2016, pp. 1-6.
- [17] A. Masadeh, Z. Wang and A. E. Kamal, "Reinforcement Learning Exploration Algorithms for Energy Harvesting Communications Systems," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, 2018, pp.
- [18] A. Ortiz, H. Al-Shatri, Xiang Li, T. Weber and A. Klein, "Throughput maximization in two-hop energy harvesting communications," *2015 International Symposium on Wireless Communication Systems (ISWCS)*, Brussels, 2015, pp. 291-295.

- [19] A. Ortiz, H. Al-Shatri, X. Li, T. Weber and A. Klein, "Reinforcement Learning for Energy Harvesting Decode-and-Forward Two-Hop Communications," in *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 3, pp. 309-319, Sept. 2017.
- [20] M. Miozzo, L. Giupponi, M. Rossi and P. Dini, "Switch-On/Off Policies for Energy Harvesting Small Cells through Distributed Q-Learning," *2017 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, San Francisco, CA, 2017, pp. 1-6.
- [21] M. Chu, H. Li, X. Liao and S. Cui, "Reinforcement Learning-Based Multiaccess Control and Battery Prediction With Energy Harvesting in IoT Systems," in *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2009-2020, April 2019.
- [22] L. Huang, "Fast-convergent learning-aided control in energy harvesting networks," *2015 54th IEEE Conference on Decision and Control (CDC)*, Osaka, 2015, pp. 5518-5525.

Appendix

1. Single package transmission within a network with obstacles

```
%DEFINE THE 2-D R ARRAY
MAX_X=10;
MAX_Y=10;

Ra = -3; %reward in non-terminal states (used to initialise
r[][])

%This array stores the coordinates of the R and the
%Objects in each coordinate
R=Ra*ones(MAX_X,MAX_Y);
Pi=ones(MAX_X,MAX_Y);

% Obtain Obstacle, Target and Robot Position
% Initialize the R
% Obstacle=-1,Target = 0,Robot=1,Space=2
axis([0 MAX_X 0 MAX_Y])
%set(gca,'color',[1 1 0]);
%set(gca,'color','b');

set(gca,'XTick',[1:MAX_X])
set(gca,'YTick',[1:MAX_Y])

grid on;
hold on;

% Determine Terminals, Obstacles, Start Locations

%Terminals
%Winning point
xWin=5;%X Coordinate of the Winning point
yWin=10;%Y Coordinate of the Winning point
R(xWin,yWin)=100;%Reward = 100
Pi(xWin,yWin)='+';%Policy
plot(xWin-.5,yWin-.5,'gd');
text(xWin-.9,yWin-.3,'Winning +100')

%Loss point
xLos=4;%X Coordinate of the Loss point
yLos=2;%Y Coordinate of the Loss point
R(xLos,yLos)=-100;%Reward = -100
```

```

Pi (xLos, yLos)='-';%Policy
%plot(xLos-.5,yLos-.5,'rd');
text(xLos-.8,yLos-.4,'Loss -100')

%Obstacles
xObs=8;%X Coordinate of the First Obstacle
yObs=5;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')

%Obstacles
xObs=3;%X Coordinate of the First Obstacle
yObs=3;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')

%Obstacles
xObs=4;%X Coordinate of the First Obstacle
yObs=5;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')

%Obstacles
xObs=3;%X Coordinate of the First Obstacle
yObs=5;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')

%Obstacles
xObs=2;%X Coordinate of the First Obstacle
yObs=7;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')

```

```
%Obstacles
xObs=1;%X Coordinate of the First Obstacle
yObs=4;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')
```

```
%Obstacles
xObs=5;%X Coordinate of the First Obstacle
yObs=5;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')
```

```
%Obstacles
xObs=3;%X Coordinate of the First Obstacle
yObs=6;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')
```

```
%Obstacles
xObs=6;%X Coordinate of the First Obstacle
yObs=6;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')
```

```
%Obstacles
xObs=7;%X Coordinate of the First Obstacle
yObs=7;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
Pi(xObs,yObs)='#';%Policy
%plot(xObs-.5,yObs-.5,'ro');
text(xObs-.8,yObs-.4,'Obs')
```

```
%Obstacles
xObs=8;%X Coordinate of the First Obstacle
yObs=8;%Y Coordinate of the First Obstacle
R(xObs,yObs)=0;%Reward = 0
```

```

Pi (xObs, yObs)='#';%Policy
%plot (xObs-.5,yObs-.5,'ro');
text (xObs-.8,yObs-.4, 'Obs')

%Obstacles
xObs=9;%X Coordinate of the First Obstacle
yObs=9;%Y Coordinate of the First Obstacle
R (xObs, yObs)=0;%Reward = 0
Pi (xObs, yObs)='#';%Policy
%plot (xObs-.5,yObs-.5,'ro');
text (xObs-.8,yObs-.4, 'Obs')

%Start
xStart=4;%X Coordinate of the Start
yStart=3;%Y Coordinate of the Start
plot (xStart-.5,yStart-.5, 'bo');
text (xStart-.6,yStart-.4, 'Start')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ALGORITHM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N = 10000; %max number of iterations of Value Iteration

deltaMin = 1e-9; %convergence criterion for iteration
delta = 0;

POINTS_COUNT= MAX_X * MAX_Y;
POINTS=[POINTS_COUNT,6];

%Put all the points in list with their rewards and initial
UP and U
%POINTS
%LIST |X val |Y val |Reward |Uprime |Utility |Policy

k=1;%Dummy counter
for i=1:MAX_X
    for j=1:MAX_Y
        POINTS(k,1)=i;
        POINTS(k,2)=j;
        POINTS(k,3)=R(i,j);
        POINTS(k,4)=0;
        POINTS(k,5)=0;
    
```

```

        POINTS(k, 6)=Pi(i, j);

        k=k+1;
    end
end

R=POINTS(:, 3);%instantaneous reward
Up=POINTS(:, 4);%UPrime, used in updates
U=POINTS(:, 5);%long-term utility
Pi=POINTS(:, 6);%policy

n=0;

%while((delta < deltaMin) && (n < N))
while 1

    POINTS(:, 5)=POINTS(:, 4);%U=Up
    U=Up;

    n=n+1;

    delta = 0;

    for i=1:MAX_X
        for j=1:MAX_Y
            upPi=updateUPrimePi(i, j, POINTS, Ra);
            Up=upPi(:, 1);
            Pi=upPi(:, 2);
            POINTS(:, 4)=Up;
            POINTS(:, 6)=Pi;

            k=find(POINTS(:, 1)==i & POINTS(:, 2)==j);
            %k=k(1);

            diff=abs(Up(k)-U(k));

            if diff > delta
                delta = diff;
            end

        end
    end

    if (delta < deltaMin || n > N)

```

```

        break;
    end

end

charPi=char(Pi);

for i=1:MAX_X
    for j=1:MAX_Y
        k=find(POINTS(:,1)==i & POINTS(:,2)==j);
        text(i-.7,j-.2,num2str(U(k)), 'color', 'g')
        text(i-.5,j-.8,charPi(k), 'color', 'b')
    end
end

path=[];

i=1;
path(1,1)=xStart;
path(1,2)=yStart;

newX=xStart;
newY=yStart;

while 1
k=find(POINTS(:,1)==newX & POINTS(:,2)==newY);
if (charPi(k)~='+' && charPi(k)~='-' && charPi(k)~='#')
    i=i+1;
    if charPi(k)=='N'
        path(i,1)=newX;
        path(i,2)=newY+1;
        newX=newX;
        newY=newY+1;
    else if charPi(k)=='S'
        path(i,1)=newX;
        path(i,2)=newY-1;
        newX=newX;
        newY=newY-1;
    else if charPi(k)=='W'
        path(i,1)=newX-1;
        path(i,2)=newY;
        newX=newX-1;
        newY=newY;
    else %'E'

```

```

        path(i,1)=newX+1;
        path(i,2)=newY;
        newX=newX+1;
        newY=newY;
    end

    end

end
else
    break;
end

i=size(path,1);
%Plot the Path!
p=plot(path(i,1)-.5,path(i,2)-.5,'bo');
plot(R,n);

for i=1:size(path,1)
    pause(.25);
    set(p,'XData',path(i,1)-.5,'YData',path(i,2)-.5);
    drawnow ;
end;
plot(path(:,1)-.5,path(:,2)-.5);
end

```

2.Q-learning for calculating average throughput

```

a=zeros(1,10000);
gamma=0.95;
state_new=1;
new=zeros(1,20);
m=1;
ibs = 0.04;
s=zeros(1,5);
afa=0.1; %learning parameter
q=zeros(4,5);
s1=[0 0 1 1 1];
s2=[0 0 0 1 1];
s3=[0 0 1 0 1];
s4=[0 0 1 1 0];
s5=[s1;s2;s3;s4];
state=1;
k=1;

```



```

for u=1:10000

    max=q(state,1);
    for i=2:5
        if max<q(state,i)
            max=q(state,i);
            m=i;
        end
    end
    if m==1
        e=unidrnd (20);
        if mod(e,6)==0
            f=unidrnd(4);
            switch f
                case 1
                    s=s5(state,1:5)&[1 0 1 1 1];

                case 2
                    s=s5(state,1:5)&[1 1 0 1 1];

                case 3
                    s=s5(state,1:5)&[1 1 1 0 1];

                case 4
                    s=s5(state,1:5)&[1 1 1 1 0];

            end
        else
            s=s5(state,1:5)&[0 1 1 1 1];

        end
    end
end

if m==2
    e=unidrnd (20);
    if mod(e,6)==0
        f=unidrnd(4);
        switch f
            case 1
                s=s5(state,1:5)&[0 1 1 1 1];

            case 2
                s=s5(state,1:5)&[1 1 0 1 1];
        end
    end
end

```

```

        case 3
            s=s5(state,1:5)&[1 1 1 0 1];

        case 4
            s=s5(state,1:5)&[1 1 1 1 0];

        end

    else
        s=s5(state,1:5)&[1 0 1 1 1];

    end
end
if m==3
    e=unidrnd (20);
    if mod(e,6)==0
        f=unidrnd(4);
        switch f
            case 1
                s=s5(state,1:5)&[0 1 1 1 1];

            case 2
                s=s5(state,1:5)&[1 0 1 1 1];

            case 3
                s=s5(state,1:5)&[1 1 1 0 1];

            case 4
                s=s5(state,1:5)&[1 1 1 1 0];

        end

    else
        s=s5(state,1:5)&[1 1 0 1 1];

    end
end
if m==4
    e=unidrnd (20);
    if mod(e,6)==0
        f=unidrnd(4);
        switch f
            case 1

```

```

        s=s5(state,1:5)&[0 1 1 1 1];

    case 2
        s=s5(state,1:5)&[1 0 1 1 1];

    case 3
        s=s5(state,1:5)&[1 1 0 1 1];

    case 4
        s=s5(state,1:5)&[1 1 1 1 0];

    end

else
        s=s5(state,1:5)&[1 1 1 0 1];

end
end

if m==5
    e=unidrnd (20);
    if mod(e,6)==0
        f=unidrnd(4);
        switch f
            case 1
                s=s5(state,1:5)&[0 1 1 1 1];

            case 2
                s=s5(state,1:5)&[1 0 1 1 1];

            case 3
                s=s5(state,1:5)&[1 1 0 1 1];

            case 4
                s=s5(state,1:5)&[1 1 1 0 1];

        end

    else
        s=s5(state,1:5)&[1 1 1 1 0];

    end
end
end

```

```

        if s== s5(1,1:5)
            state_new = 1;
        elseif s==s5(2,1:5)
            state_new = 2;
        elseif s==s5(3,1:5)
            state_new = 3;
        elseif s==s5(4,1:5)
            state_new = 4;
        else
            state_new = 1;
        end
    if state_new == state
        a(k)=1;
        r=-5;
    else
        a(k)=0;
        r=1;
    end
    new(k)=state_new;
    k=k+1;

    max_2=q(state_new,1);
    for i=2:5
        if max_2<q(state_new,i)
            max_2=q(state_new,i);
        end
    end
    q(state,m)=(1-afa)*q(state,m)+afa*(r+gamma*max_2);
    state=state_new;
end

sum=0;
b=zeros(1,20);
for n=1:20
    for i=((n-1)*500+1):n*500
        sum=sum+a(i);
    end
    b(n)=sum;
    sum=0;
end

```

```

end
n=1:20;
plot(n,b(n));
axis([1 20 0 500]);

```

3. Q-learning for generating the average collision rate

```

format short
format compact
a=zeros(1,10000);
gamma=0.95;
state_new=1;
new=zeros(1,20);
m=1;
s=zeros(1,5);
afa=0.1; %learning parameter
    q=rand(7,5);
    s1=[0 0 1 1 1];
    s2=[0 0 0 1 1];
    s3=[0 0 1 0 1];
    s4=[0 0 1 1 0];
    s5=[0 0 0 0 1];
    s6=[0 0 0 1 0];
    s7=[0 0 1 0 0];
    s8=[s1;s2;s3;s4;s5;s6;s7];
    state=1;
    k=1;

for u=1:10000

    max=q(state,1);
    for i=2:5
        if max<q(state,i)
            max=q(state,i);
            m=i;
        end
    end
    end
    if m==1
        e=unidrnd(10);
        if mod(e,6)==0

```

```

f=unidrnd(4);
switch f
    case 1
        s=s8(state,1:5)&[1 0 1 1 1];

    case 2
        s=s8(state,1:5)&[1 1 0 1 1];

    case 3
        s=s8(state,1:5)&[1 1 1 0 1];

    case 4
        s=s8(state,1:5)&[1 1 1 1 0];

    end
else
    s=s8(state,1:5)&[0 1 1 1 1];

end
end

if m==2
    e=unidrnd(10);
    if mod(e,6)==0
        f=unidrnd(4);
        switch f
            case 1
                s=s8(state,1:5)&[0 1 1 1 1];

            case 2
                s=s8(state,1:5)&[1 1 0 1 1];

            case 3
                s=s8(state,1:5)&[1 1 1 0 1];

            case 4
                s=s8(state,1:5)&[1 1 1 1 0];

            end

        else
            s=s8(state,1:5)&[1 0 1 1 1];

        end
    end
end

```

```

end
if m==3
    e=unidrnd (10);
    if mod(e,6)==0
        f=unidrnd(4);
        switch f
            case 1
                s=s8(state,1:5)&[0 1 1 1 1];

            case 2
                s=s8(state,1:5)&[1 0 1 1 1];

            case 3
                s=s8(state,1:5)&[1 1 1 0 1];

            case 4
                s=s8(state,1:5)&[1 1 1 1 0];

        end

    else
        s=s8(state,1:5)&[1 1 0 1 1];

    end
end

if m==4
    e=unidrnd (10);
    if mod(e,6)==0
        f=unidrnd(4);
        switch f
            case 1
                s=s8(state,1:5)&[0 1 1 1 1];

            case 2
                s=s8(state,1:5)&[1 0 1 1 1];

            case 3
                s=s8(state,1:5)&[1 1 0 1 1];

            case 4
                s=s8(state,1:5)&[1 1 1 1 0];

        end

    end
end

```

```

        else
            s=s8(state,1:5)&[1 1 1 0 1];
        end
    end
end

if m==5
    e=unidrnd (10);
    if mod(e,6)==0
        f=unidrnd(4);
        switch f
            case 1
                s=s8(state,1:5)&[0 1 1 1 1];

            case 2
                s=s8(state,1:5)&[1 0 1 1 1];

            case 3
                s=s8(state,1:5)&[1 1 0 1 1];

            case 4
                s=s8(state,1:5)&[1 1 1 0 1];

        end

    else
        s=s8(state,1:5)&[1 1 1 1 0];
    end
end
end

```

```

if s== s8(1,1:5)
    state_new = 1;
elseif s==s8(2,1:5)
    state_new = 2;
elseif s==s8(3,1:5)
    state_new = 3;
elseif s==s8(4,1:5)
    state_new = 4;
elseif s==s8(5,1:5)
    state_new = 5;
end

```



```

elseif s==s8(6,1:5)
    state_new = 6;
elseif s==s8(7,1:5)
    state_new = 7;
else
    state_new = 1;
end
if state_new == state
    a(k)=1;
    r=-5;
else
    a(k)=0;
    r=1;
end
new(k)=state_new;
k=k+1;

max_2=q(state_new,1);
for i=2:5
    if max_2<q(state_new,i)
        max_2=q(state_new,i);
    end
end
q(state,m)=(1-afa)*q(state,m)+afa*(r+gamma*max_2);
state=state_new;
end

sum=0;
b=zeros(1,20);
for n=1:20
    for i=((n-1)*500+1):n*500
        sum=(sum+a(i))/10;
    end
    b(n)=sum;
    sum=0;
end
n=1:20;
plot(n,b(n))
axis([1 20 0 1])

```