ABSTRACT

Title of Dissertation: Resource and Environment Aware Sensor Communications: Framework, Optimization, and Applications

Charles Pandana, Doctor of Philosophy, 2005

Dissertation directed by: Professor K. J. Ray Liu
Department of Electrical and Computer Engineering

Recent advances in low power integrated circuit devices, micro-electro-mechanical system (MEMS) technologies, and communications technologies have made possible the deployment of low-cost, low power sensors that can be integrated to form wireless sensor networks (WSN). These wireless sensor networks have vast important applications, i.e.: from battlefield surveillance system to modern highway and industry monitoring system; from the emergency rescue system to early forest fire detection and the very sophisticated earthquake early detection system. Having the broad range of applications, the sensor network is becoming an integral part of human lives. However, the success of sensor networks deployment depends on the reliability of the network itself. There are many challenging problems to make the deployed network more reliable. These problems include but not limited to extending network lifetime, increasing each sensor node throughput, efficient collection of

information, enforcing nodes to collaboratively accomplish certain network tasks, etc. One important aspect in designing the algorithm is that the algorithm should be completely distributed and scalable. This aspect has posed a tremendous challenge in designing optimal algorithm in sensor networks.

This thesis addresses various challenging issues encountered in wireless sensor networks. The most important characteristic in sensor networks is to prolong the network lifetime. However, due to the stringent energy requirement, the network requires highly energy efficient resource allocation. This highly energy-efficient resource allocation requires the application of an energy awareness system. In fact, we envision a broader resource and environment aware optimization in the sensor networks. This framework reconfigures the parameters from different communication layers according to its environment and resource. We first investigate the application of online reinforcement learning in solving the modulation and transmit power selection. We analyze the effectiveness of the learning algorithm by comparing the effective good throughput that is successfully delivered per unit energy as a metric. This metric shows how efficient the energy usage in sensor communication is. In many practical sensor scenarios, maximizing the energy efficient in a single sensor node may not be sufficient. Therefore, we continue to work on the routing problem to maximize the number of delivered packet before the network becomes useless. The useless network is characterized by the disintegrated remaining network. We design a class of energy efficient routing algorithms that explicitly takes the connectivity condition of the remaining network in to account. We also present the distributed asynchronous routing implementation based on reinforcement learning algorithm. This work can be viewed as distributed connectivity-aware energy efficient routing. We then explore the advantages

obtained by doing cooperative routing for network lifetime maximization. We propose a power allocation in the cooperative routing called the maximum lifetime power allocation. The proposed allocation takes into account the residual energy in the nodes when doing the cooperation. In fact, our criterion lets the nodes with more energy to help more compared to the nodes with less energy. We continue to look at the problem of cooperation enforcement in ad-hoc network. We show that by combining the repeated game and self learning algorithm, a better cooperation point can be obtained. Finally, we demonstrate an example of channel-aware application for multimedia communication. In all case studies, we employ optimization scheme that is equipped with the resource and environment awareness. We hope that the proposed resource and environment aware optimization framework will serve as the first step towards the realization of intelligent sensor communications.

RESOURCE AND ENVIRONMENT AWARE SENSOR COMMUNICATIONS: FRAMEWORK, OPTIMIZATION, AND APPLICATIONS

by

Charles Pandana

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Committee:

Professor K. J. Ray Liu, Chairman/Advisor
Professor Ramalingam Chellappa
Professor Adrian Papamarcou
Professor Sennur Ulukus
Professor Lawrence C. Washington

DEDICATION

To my parents and my brothers

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor, Prof. K. J. Ray Liu, for his guidance and support during my study in University of Maryland. He always encourages me to pursue my goal and work hard to achieve the excellence. I especially appreciate his effort to help his students and give them advice whenever they need. He has played a significant role in both my professional and personal development in Maryland, and his vision, energy and desire for excellent goal achievement have influenced me with lifetime benefits.

I would like to take this chance to thank members in the CSPL group for their friendship, encouragement and help. I always feel lucky to be in such an energetic and excellent group, and their accompanying during my stay in Maryland has helped me to survive my Ph.D. study. Special thanks to my seniors: Yan Sun, Zhu Han, Hong Zhao. And also my friends Johannes Thorsteinsson, Guan-Ming Su, Yinian Mao, Thanongsak Himsoon, Wipawee Pam Siriwongpairat, Ahmed Sadek, Karim Seddik, Wei Yu, Zhu Ji. Thank you for the time spent together, and the happy time in our office will be always in my memory.

I am grateful to Herman Pandana, my brother, for his companion and support during our study in University of Maryland. It is because of his help that my life in the past years is much more joyful and colorful. I would also like to thank my parents and my elder brother for their constant support and countless sacrifices. Without them, I could never accomplish so much and reach this milestone in my life. I dedicate this thesis to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Recent advancement in low power integrated circuit devices, micro-electro-mechanical system (MEMS) technologies, and wireless communications has made possible the large scale deployment of low cost, low power, and multi-functional sensor nodes that are small in size and are able to communicate untethered in short distances. Depending on the applications, these tiny nodes typically contain three major components, data sensing component, data processing component, and the communication component. These tiny sensor nodes altogether form a micro-sensors network. There are several features of sensor networks that make them unique. These features are the deployment position of the sensor nodes need not be engineered or predesigned and the sensor nodes cooperatively accomplish some predetermined complex tasks. The first feature implies that the sensor network can be deployed randomly in some inaccessible terrain. This feature also indicates that the protocols and algorithms employed in the sensor network should have self-configuring and self-organizing capabilities. The second feature implies that each node in sensor network should be intelligent enough to collaboratively accomplish the predefined task in an efficient manner. For instance, instead of sending the raw

data, nodes locally carry out simple computations so that the overall transmission is as efficient as possible.

The above features enable a vast range of applications for sensor networks. In military, the rapid deployment, self-organization, and fault tolerance characteristics of sensor networks make them a very promising sensing technique for military command, control, communications, computing, intelligence, surveillance, reconnaissance, and targeting systems. In health, sensor networks can be deployed to monitor patients and assist disabled patients. In emergency rescue system, sensor nodes can be deployed to perform early detection of earthquake, fire detection, etc. Some other commercial applications include managing inventory, monitoring product quality, habitat and environment monitoring, and monitoring modern highway system. Due to this broad applications of sensor networks, the sensor network is becoming an integral part of human lives. In September 1999 [5], Business Week "21 ideas for the 21st century" pointed out that micro-sensor technology is a key technology for the 21st century. Recently, the MIT Technological Review [4] ranked the wireless sensor network as the number one emerging technology.

A more recent application of sensor network in industry is the joint research between British Petroleum (BP) and Intel. In this project, the sensor network is used to support preventive maintenance on board an oil tanker in the North Sea. BP wanted to determine if the sensor network could operate in a shipboard environment, where it would have to withstand temperature extremes, substantial vibration, and significant radio frequency noise in certain parts of the ship. A sensor network was installed onboard the ship and operated successfully for over four months. During this trial deployment, the system gathered data reliably and recovered from errors when they occurred. The project was recognized by

InfoWorld as one of the top 100 IT projects in 2004, an award given to "innovative new projects that highlight the resourcefulness of the IT community." BP is now exploring the use of sensor network technology throughout the company, in shipping, manufacturing and refining operations.

Typical configuration of sensor network (shown in Figure 1.1) consists of task manager/user, satellite and internet backbone, sinks, gateways, and sensor nodes. The sensor nodes are scattered in the sensor field, where nodes collect the data of interest and route back to the sinks through gateways. The data are routed back to the sinks by a multihop infrastructureless architecture. The sinks receive queries from and report back the collected data to the task manager via the satellite or internet backbone

Several factors that influence the successful deployment of the large scale wireless sensor networks (WSNs) are listed as follows [6, 30]

1. Fault tolerance: Due to the random deployment in some severe and harsh environment, some of the deployed sensor nodes may be failed or dead because of the exhausted energy. The failure of these small number of nodes in the network should not affect the overall functionality of the sensor networks.

2. Production cost: Since a large number of micro-sensors are deployed in the sensor network, it is crucial that each single one micro-sensor has very low cost to justify the overall network deployment cost. It is envisioned in [6], the cost of a micro-sensor should be much lower than US$1. Therefore, it is very important to develop low power computer aided design (CAD) tools to reduce the production cost, yet meet the requirement of each sensor. Another important aspect is that the designed algorithm should be simple enough to be implemented in the micro-sensor node, yet effective in such a

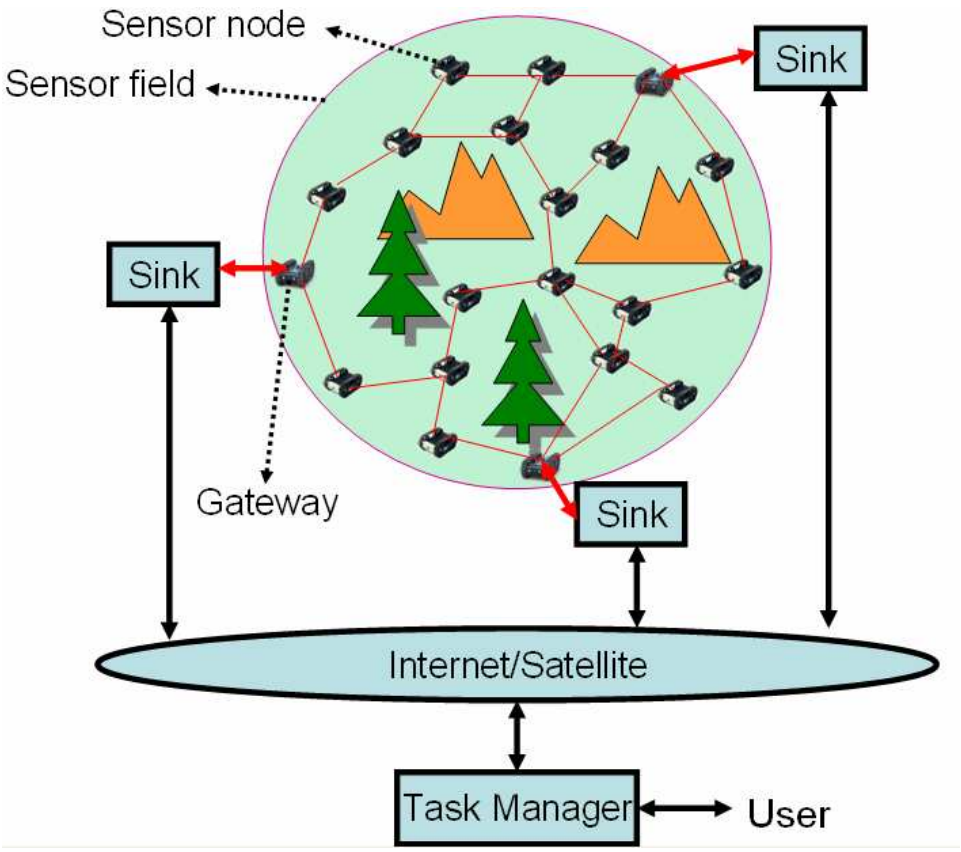


Figure 1.1: Typical sensor network configuration

harsh environment.

3. Hardware constraints: As stated in previous paragraph, typical sensor node is composed of three major components, the sensing component, data processing component, and communication component. In addition, depending on the application, the node may also have power generator, location finding, and mobilizer components. Different from the traditional sensor, the sensor nodes in the sensor network should have self-configuring and self-organizing capabilities. These requirements complicate the design of hardware in terms of size, computation power, and power consumption. All the three components should be fitted into a single small sized module. Therefore, the computational power/capability in each of the sensor may be limited. Finally, it is very important to integrate all these components with extremely low power design.

4. Severe environment: Since many applications of sensor networks are for emergency rescue, habitat monitoring, and military applications. The sensor network may be deployed in some inaccessible area and the deployed sensor nodes face tremendous severe environment. Hence, it is of paramount important that the sensor nodes should be adaptive to the severe environment.

5. Transmission media: The transmission link in the sensor network basically can be in many forms from radio, infra-red, and optical link. The latter two media require a line of sight (LOS) between the transmitter and the receiver. The radio link enables the global operation of this network. Many of the current solutions for sensor network is based on radio transmission, such as μAMPS, Wireless Integrated Sensor Network (WINS) architecture, etc. The

SmartDust mote uses optical medium for communication. Sensor node may also support two or more transmission interfaces.

6. Power consumption: The sensor nodes, usually battery powered, have limited energy. In many of the applications, such as battlefield monitoring, emergency rescue system, and habitat monitoring, the replenishment of the power/energy resources may not be possible at all. And the micro-sensor node lifetime is highly dependent on its battery lifetime. The node lifetime turns out to affect the overall lifetime of the network. Therefore, it is very important that all designed algorithms in the sensor network are as energy efficient as possible. This last factor influences major differences in the design of protocols and algorithms used in the sensor network.

Having described the possible application and the successful design factors, we are ready to discuss the challenges to build a successful sensor network. In the next two sections, we will explain the envisioned protocol stack and the requirement of resource and environment aware resource allocation.

1.1 Cross Layer Design

Traditional communication systems are designed based on layers. The Open System Interconnection (OSI) reference model defined seven layers from top to bottom as application, presentation, session, transport, network, data link, and physical layer. Each layer is designed for a specific purpose and optimized to achieve its own goal within each layer. The main purpose of the OSI reference model is to simplify the implementation. The downside of the layered implementation is the overhead between layers. Moreover, the solution of separated optimization in each

layer may be far from efficient compared to the cross layer design. And a better solution can be obtained when one considers the optimization across several communication layers. Due to the limited resources (limited bandwidth and power), there is an increasing need to perform cross layer optimization.

The cross layer design has received tremendous attentions from many researchers. This is especially important when designing protocols and algorithms for sensor networks. There are several differences between the cross-layer design employed in traditional communication compared to the sensor network. These differences root from the different design objective in typical traditional communication systems and the sensor network. The cross-layer approaches are used to maximize the quality of services (QoS), minimize delay, maximize throughput, etc. in traditional communication systems. In contrast, the cross-layer design in sensor networks focuses on energy minimization, efficient utilization of the energy, and aggressive network lifetime maximization. It can be easily understood, since power/energy is the most precious resource in sensor communication systems. This can be justified from the efforts of many researchers to apply cross-layer approaches to meet the stringent energy requirement in the sensor communication [6, 30, 36, 45, 80, 94].

In [6], they outline the suitable protocol stack for sensor network shown in Figure 1.2. This protocol stack consists of application layer, transport layer, network layer, data link layer, and physical layer. There are three modules that do the optimization and control across different communication layers. Those modules are power management module, mobility management module, and the task management module. The information obtained from different communication layers is transparent to the management module. These management modules optimize and adjust the parameters in several communication layers to achieve the energy

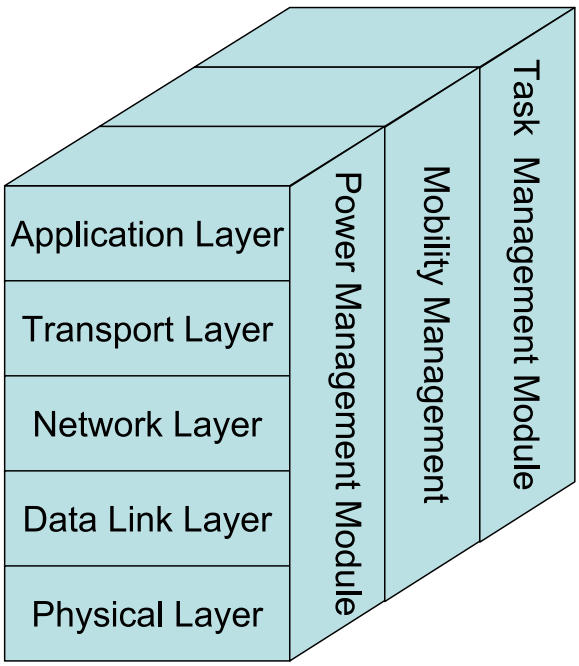


Figure 1.2: Sensor network protocol stack

efficient communication. The power management module configures a sensor node to use its power in an optimized way. For instance, the sensor node may go to sleep mode when it is not in the sensing mode and communicating mode. Also, when the residual energy in the sensor node is low, it can broadcast to its neighbors and avoid participating in relaying packets. The mobility management module acts to keep track who their neighboring sensor nodes are. Moreover, it can also act as location finder module. Finally, since not all the sensors are required to perform the sensing task in all region, the task management balances and schedules the sensing tasks in a specific region. These modules are required so that sensor nodes are able to work together in the most power/energy efficient way.

The above protocol stack motivates us to generalize the communication design in sensor network. In fact, to make the sensor node as intelligent as possible, the sensor node should be able to obtain and process as much information as possible.

This information in a narrow sense includes parameters obtained from each communication layers, but in a broader sense includes several behavioral information in the network. Therefore, we propose a resource and environment aware sensor communication framework in the next section. The resource may include the information about the residual energy, computing power, adaptive modulation and power level supported in the nodes, etc. In contrast, the environment may include the channel condition in the sensor communication, the connectivity of the neighboring nodes, the topology of the network, etc. Based on the proposed framework, many algorithms that have channel, power, residual energy, residual connectivity awareness are resulted. This framework can be thought as the first step towards the realization of intelligent sensor communication systems.

Even though the cross-layer optimization may result in better solution, there are challenges in the cross layer design. The most obvious problem is the increase in optimization complexity. When each communication layer is optimized separately, the number of variables in each communication layer optimization is tractable. However, when many communication layers are jointly optimized, the optimization problem definitely grows exponentially. This will require a very complex data processing unit in each of the sensor and result in high power consumption. Hence, the first challenge in cross layer design for sensor networks is to develop a simple optimization algorithm that can capture the relationship of parameters across different communication layers. The resulting algorithm should be simple to implement, even more importantly, it should be energy efficient. We refer the explosion in the optimization variables in cross layer design to as the *curse of dimensionality*.

The second challenge is the lack of good model to describe the complex rela-

tionships between parameters from different communication layers and the performance objective of interest. Researchers in communication theory community have been doing the optimization within each communication layer for a long time. They have developed many good models within each of the communication layers. However, these models may not be suitable for describing the relationships between parameters from different communication layers. This lack of good model makes the optimization even more challenging. Thereafter, this lack of good model is referred to as *curse of model*.

The next but not last challenge in the cross-layer approaches in sensor communication design is that the optimization should be done online and distributed manner. Online optimization implies that the optimization should be able to adapt to the information changes obtained in various communication layers. Distributed optimization is always required as there may be no centralized node to coordinate the optimization. All the activities in the network are done in an ad-hoc manner. After listing many important design challenges, we first describe the general resource and environment aware optimization in the next section. Moreover, we will outline the proposed method that can fully or partly solve the curse of dimensionality, curse of model, online and distributed challenges in designing optimization algorithms for sensor networks.

1.2 Resource and Environment Aware Optimization Framework

The resource and environment aware optimization framework is a general optimization framework that takes the resource and the environment condition into

account when doing the optimization. Generally speaking, the resource may include the bandwidth, the residual energy in the nodes, energy consumption, computing power, adaptive modulation and power level supported in the nodes, etc. The environment aware includes the channel-aware, topology-aware, remaining connectivity-aware, and location-aware optimization. All of this information can be obtained from different layers of communication. In particular, we focus on the following optimization framework shown in Figure 1.3. In this framework, each sensor node detects the local information and resource information. Based on this local information, the adaptive learning algorithm adjusts the parameters in network layer, data link layer, and physical layer. This adjustment will be evaluated by the local system performance evaluation, which informs the adaptive system of how well the parameter adjustment performs. The adaptation of the parameters also effects the local information and resource. Using this framework, the design of sensor communication has the awareness of the resource and environment when doing the online optimization through adaptive learning algorithm.

In general, we still face the problems of curse of dimensionality, curse of model, and the requirement of simple yet effective online distributed optimization in the resource and environment aware optimization framework. Due to above challenges and limitations, we propose the use of stochastic approximation technique [34, 57, 93] in solving the online optimization. The stochastic approximation (SA) is suitable for the online optimization problems encountered in sensor networks due to the following reasons

1. SA is categorized as one of the random search methods. This method has been shown to be effective in solving the Markov Decision Process problem which also has the problem of *curse of dimensionality* [18, 97]. This method

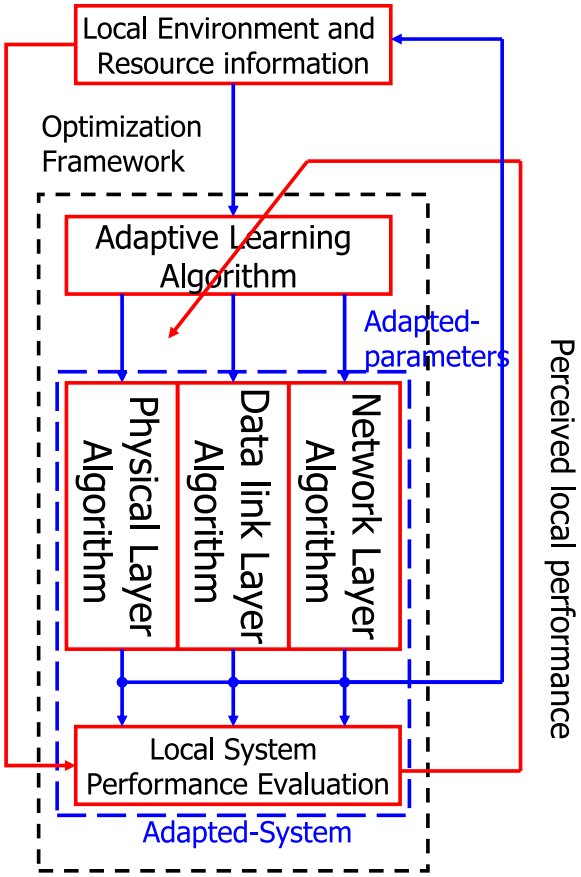


Figure 1.3: Resource and Environment Aware Optimization Framework

includes (but not limited to) the reinforcement learning algorithms.

2. SA does not require the knowledge of the complete function to be optimized. The only requirement is the noisy sample of the function at any arguments of the function. Due to this characteristic, the SA algorithm is suitable to tackle the problem of curse of model. In many practical scenarios, the complete objective function to be optimized may not be available at the time the optimization is done. However, it is typical that the sensor node can observe and evaluate the function of interest after deciding a set of parameters to use. For instance, upon the selection of modulation and power level, the nodes can measure the total consumed power when employing the decided modulation and power level.

3. Using the small constant step-size in the SA algorithm, the algorithm is robust to track non-stationarity in the function to be optimized. Therefore, the SA algorithm is robust and is able to adapt to the resource and environment variations.

4. The SA iterations usually involve very simple additions and multiplications. Due to its computation simplicity, the SA iteration is envisioned to be implementable in low cost sensor.

1.3 Organization of Dissertation

The rest of the dissertation is organized as follows. Chapter 2 gives the detailed mathematical framework for the optimization, as well as a brief review of the related computational approaches. Chapter 3 gives the application of the online decision making problem, where sensor nodes have to select the suitable modulation

level and transmit power such as to maximize the throughput per unit energy usage. This chapter can be viewed as the cross layer application that combines the data link layer, namely modulation selection and the physical layer, namely transmit power selection. The modulation and power selection are formulated as discrete optimization, which better reflects the practical situation.

Chapter 4 suggests one application of topology aware energy efficient routing. In this chapter, we suggest the time until the remaining network becomes disconnected as the definition of the overall network lifetime. Using this definition, we propose to embed the connectivity weights that reflect the importance of sensor nodes in the routing metric. The importance of a node is characterized as how severe the remaining network becomes when that particular node dies. In this way, the routing decision will select route that always keeps the remaining network connected.

Chapter 5 studies the effect of cooperative routing in maximizing the network lifetime. This chapter is one implementation of cross layer design between the network layer and physical layer. It provides a residual energy-aware cooperative routing scheme. In particular, we proposed a different power allocation called maximum lifetime power allocation, instead of using the traditional minimum power allocation in the cooperative routing. Our power allocation scheme jointly considers the channel effect and the residual energy in each sensor nodes. Using the maximum lifetime power allocation, the nodes that have more residual energy will help more compared to the nodes that have less residual energy. In this way, the overused of nodes that have low residual energy will be avoided. Therefore, our proposed method prolongs the overall network lifetime.

Chapter 6 studies how to enforce cooperation among nodes in the ad-hoc net-

work. We propose a self-learning repeated game framework to enforce cooperation and obtain good cooperation point. In practice, the distributed nodes with only local information may not know how to cooperate, even though they are willing to cooperate. This motivates us to propose self learning algorithm to search for good cooperation point. The proposed scheme consists of two parts; in the first part, an adaptive repeated game scheme is designed to ensure the cooperation among nodes for the current cooperative packet forwarding probabilities. In the second part, self-learning algorithms are employed to find the better cooperation probabilities that are feasible and beneficial to all nodes. Starting from noncooperation, the above two steps are employed iteratively, so that a better cooperating point can be achieved and maintained in each iteration.

Chapter 7 provides an application of cross layer optimization between the application layer and physical layer. This chapter serves as the example of channel-aware wireless optimization. In particular, we propose a channel aware priority transmission scheme for OFDM system. The scheme jointly considers the effects of different channel estimation algorithms and the property of multimedia stream to allocate the data in the most efficient way. We observe that OFDM subchannels experience different average bit error rate (BER) due to channel estimation inaccuracy. The leakage effect in FFT based channel estimation method or the model mismatch in polynomial based channel estimation method results in a variation on the decoded BER across different OFDM subchannels. Motivated by this fact, the proposed priority transmission utilizes the bit error rate variation across different OFDM subchannels and provides unequal error protection (UEP) for multimedia transmission. The proposed scheme achieves significant gain in peak-signal-to-noise ratio (PSNR) of the reconstructed images for different channel estimation

methods.

Finally, chapter 8 concludes the dissertation with some remarks as well as a discussion on the contributions of the dissertation and potential future research directions.

Chapter 2

Mathematical Framework

In this chapter, we summarize the mathematical framework and the related computational methods to find solutions of the stated problem. We also provide several definitions and terminologies. We note that we only state many of the theorems required to justify the computational method. The detailed proofs of the theorems can be found in the related literatures, highlighted before the theorem.

2.1 Markov Decision Process

A Markov Decision Process (MDP) [16, 23, 84] is defined as a $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R})$ tuple where $\mathbf{S}$ is the state space that contains all possible states of the system, $\mathbf{A}$ is the set of all possible control actions at each state, $\mathbf{P}$ is a transition function $\mathbf{S} \times \mathbf{A} \times \mathbf{S} \rightarrow [0, 1]$, and $\mathbf{R}$ is a reward function $\mathbf{S} \times \mathbf{A} \rightarrow \mathbf{R}$. The transition function defines a probability distribution over the next state as a function of the current state and the agent's action, i.e. $[\mathbf{P}]_{s_k, s_{k+1}}(a_k) = P_{s_k, s_{k+1}}(a_k)$ specifies the transition probability from state $s_k \in \mathbf{S}$ to $s_{k+1} \in \mathbf{S}$ under the control action $a_k \in \mathbf{A}$. Here, the notation $[\mathbf{A}]_{i,j}$ denotes the element on the i^{th} row and the j^{th}

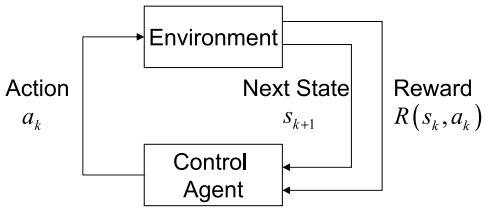


Figure 2.1: Interaction between agent and environment in MDP

column of matrix $\mathbf{A}$. The transition probability function $\mathbf{P}$ describes the dynamic of the environment as the response to the agent current decision. The reward function specifies the reward incurred at state $s_k \in \mathbf{S}$ under control action $a_k \in \mathbf{A}$. The interaction between the agent and environment in MDP is illustrated in Figure 2.1. At time k, the control agent detects $s_k \in \mathbf{S}$ and decides an action $a_k \in \mathbf{A}$. The decision a_k causes the state to evolve from s_k to s_{k+1} according to probability $P_{s_k,s_{k+1}}(a_k)$, and some reward $R(s_k, a_k)$ is obtained. From this figure, the control agent makes an action choice at a series of discrete time steps and experiences through a series of states. The state evolution and the reward obtained at each step are dependent on the actions chosen.

A policy is a rule which the control agent employs to choose an action at each time step. In general, the policy may depend on the time, state and the history of actions taken and states visited. A stationary policy is a policy for which the current choice of action depends only on the current state. It has been shown in [9, 42, 84], the stationary policy is general enough to get the optimal solution of the corresponding MDP. Therefore, for the rest of this chapter, we will focus only on the stationary policy. Given a fixed stationary decision/action policy $a_k \in \mathbf{A}$, the corresponding MDP reduces to a Markov Chain. The solution of the MDP consists of finding the decision policy $\pi : \mathbf{S} \rightarrow \mathbf{A}$ that maximizes some objective functions. Several typical objective functions are expected discounted reward, expected total

reward and average reward per stage [16, 84]. In the next two subsections, we will define the discounted MDP and the average cost MDP.

2.1.1 Discounted Markov Decision Process

The solution of the discounted cost MDP is to find the decision/action policy that maximizes

$$J^{\pi}(s_0) = \lim_{n\to\infty} E_{\pi}\left[\sum_{k=0}^{n-1} \alpha^k R\Big(s_k, \pi(s_k)\Big)\right], s_k \in \mathbf{S}, \pi(s_k) \in \mathbf{A}, \tag{2.1}$$

where $J^{\pi}(s_0)$ is the discounted reward obtained using decision policy π when the initial state is s_0, α is the discount factor. $E_{\pi}(\cdot)$ denotes the expectation value when a policy π is used. The solution of the discounted MDP is characterized by the following Bellman's equation [16, 84].

Theorem 1 *The maximum expected discounted future reward starting from state $s_0 = s$ is given by the solution to the following Bellman equation*

$$V(s) = \max_{a\in A(s)} \left[R(s,a) + \alpha \sum_{s'\in\mathbf{S}} P_{s,s'}(a) V(s')\right], \text{ for each } s \in \mathbf{S}, \tag{2.2}$$

where $\alpha \in (0,1)$ is the discount factor. This solution exists and is unique, and an optimal deterministic stationary policy exists. The optimal decision policy is obtained by the maximizing action policy in this equation.

2.1.2 Average Cost Markov Decision Process

In many practical optimization, the average cost is a more relevant objective function compared to the discounted cost MDP. The solution of average cost MDP is to find the decision/action policy that maximizes

$$\rho^{\pi}(s_0) = \lim_{n\to\infty} \frac{1}{n} E_{\pi}\left[\sum_{k=0}^{n-1} R\Big(s_k, \pi(s_k)\Big)\right], s_k \in \mathbf{S}, \pi(s_k) \in \mathbf{A}, \tag{2.3}$$

where $\rho^{\pi}(s_0)$ is the average reward obtained using decision policy π when the initial state is s_0. We note that the expectation operation in (2.3) is the conditional expectation given one particular policy. The optimal policy is the decision rule that maximizes the average reward per stage $\rho^{\pi}(s_k)$ over all possible policies π.

When the Markov chain resulting from applying *every* stationary policy is recurrent or ergodic, it is well-known that the optimal average reward per stage is independent of the initial state s_0 [16,84]. Moreover, the optimal stationary policy is characterized by the following theorem [16, 84].

Theorem 2 *The solution of average cost MDP is given by the solution to the following Bellman equation*

$$\rho^* + h^*(s) = \max_{a \in A(s)} \left[R(s,a) + \sum_{s'=1}^{|S|} P_{s,s'}(a) h^*(s') \right], \tag{2.4}$$

where ρ^* is the optimal average reward per stage and $h^*(s)$ is known as optimal relative state value function for each state s.

In the next section, we summarize many computational tools for finding the solution of Bellman equation (2.2),(2.4).

2.2 Solutions of Markov Decision Process

The solution of MDP can be obtained using either dynamic programming or linear programming. Each of the methods has its own advantages and disadvantages. Typically, the dynamic programming (DP) approach has lower computational complexity, however, the linear programming (LP) formulation suggests different interpretation and provides randomized stationary solution as shown in dual LP formulation.

2.2.1 Dynamic Programming

The traditional approaches for solving the MDP are collectively termed by dynamic programming approaches. There are two ways to find the solution of Bellman equation using the dynamic programming namely, the value iteration method and policy iteration method. In the following, we summarize the value iteration and policy iteration methods for both the discounted and average cost MDP.

Value Iteration for discounted MDP

The value iteration method for discounted MDP relies on the following operator $T : \mathbb{R}^{|\mathbf{S}|} \rightarrow \mathbb{R}^{|\mathbf{S}|}$

$$(T \circ V)(s) = \max_{a \in A(s)} \left[R(s,a) + \alpha \sum_{s' \in \mathbf{S}} P_{s,s'}(a) V(s') \right], \text{ for each} s \in \mathbf{S}, \tag{2.5}$$

where $A(s)$ denotes the set of actions when current state is in s. It can be shown [16] that operator T is a contraction mapping with respect to the supremum norm. Moreover, [16] shows that the iteration $T^n(V_0)$ converges uniformly to the unique solution to Bellman's equation (2.2), for any bounded initial condition $V_0 \in \mathbb{R}^{|\mathbf{S}|}$. The value iteration algorithm finds a stationary ε-optimal policy as follows

1. Select $V_0(s)$ $s = 1 \cdots, |\mathbf{S}|$, set iteration $n = 0$ and specify $\varepsilon > 0$.

2. For each $s \in \mathbf{S}$, compute $V_{n+1}(s)$ as

$$V_{n+1}(s) = \max_{a \in A(s)} \left[R(s,a) + \alpha \sum_{s' \in \mathbf{S}} P_{s,s'}(a) V_n(s') \right] \tag{2.6}$$

3. If $||V_{n+1} - V_n||_\infty < \varepsilon(\frac{1-\alpha}{2\alpha})$, then go to step 4. Otherwise, increment n by 1 and return to step 2. We note that $||V||_\infty$ is the supremum norm (the maximum absolute element in the vector V).

4. For each $s \in \mathbf{S}$, find the ε-optimal policy as

$$a^*(s) = \arg\max_{a \in A(s)} \left[R(s,a) + \alpha \sum_{s' \in \mathbf{S}} P_{s,s'}(a) V_{n+1}(s') \right] \tag{2.7}$$

Obviously, the number of iterations in the Value iteration method depends on how accurate ε the solution is required to be. Generally, it requires an infinite number of iterations to find the optimal value function exactly.

Policy Iteration for discounted MDP

The policy iteration method resembles the newton-like optimization for solving the nonlinear equation. The detailed steps in policy iteration are summarized as follows

1. Select initial policy π_0 arbitrarily, set iteration $n = 0$.

2. (Policy Evaluation): solve V_n^π from the following set of equations.

$$V^{\pi_n}(s) = R(s, \pi_n(s)) + \alpha \sum_{s' \in \mathbf{S}} P_{s,s'}(\pi_n(s)) V^{\pi_n}(s'), \text{ for each } s \in \mathbf{S}. \tag{2.8}$$

3. (Policy Improvement): Update the policy as

$$\pi_{n+1}(s) = \arg\max_{\pi} \left[R(s,\pi) + \alpha \sum_{s' \in \mathbf{S}} P_{s,s'}(\pi) V^{\pi_n}(s') \right] \tag{2.9}$$

4. Stopping criteria: When $\pi_{n+1} = \pi_n$.

Unlike the value iteration method, the policy iteration will converge to the optimal solution in a finite number of iterations. The following theorem characterizes the optimality and the finite iteration property in the policy iteration method [16].

Theorem 3 *For the policy iteration algorithm, $V^{\pi_{n+1}}(s) \geq V^{\pi_n}(s)$ for all $s \in \mathbf{S}$, with the equality at $s \in \mathbf{S}$ if and only if $\pi_n(s)$ is optimal. Therefore, the algorithm*

will converge in a finite number of iterations due to the finite number of states and actions in each states.

After summarizing the value iteration and policy iteration for solving the discounted MDP, we briefly explain the value iteration and policy iteration algorithm for solving the average cost MDP in the following two subsections.

Value Iteration for average cost MDP

The following value iteration algorithm finds a stationary ε-optimal policy for the unichain average cost MDP [16]

1. Select $Th_0(s) = h_0(s),\ s = 1 \cdots, |\mathbf{S}|$, set iteration $n = 0$, specify $\varepsilon > 0$, and determine the reference state $\dot{s}$ arbitrarily.

2. For each $s \in \mathbf{S}$, compute $Th_{n+1}(s)$ as

$$Th_{n+1}(s) = \max_{a \in A(s)} \left[R(s,a) + \sum_{s' \in \mathbf{S}} P_{s,s'}(a) h_n(s') \right] \tag{2.10}$$

3. Set $c^+ = \max_{s' \in \mathbf{S}}(Th_{n+1}(s) - Th_n(s))$ and $c^- = \min_{s' \in \mathbf{S}}(Th_{n+1}(s) - Th_n(s))$. If $|c^+ - c^-| < \varepsilon$, then go to step 4. Otherwise, set $h_{n+1}(s) = Th_{n+1}(s) - Th_{n+1}(\dot{s}),\ s = 1 \cdots, |\mathbf{S}|$, increment n by 1, and return to step 2.

4. For each $s \in \mathbf{S}$, find the ε-optimal policy as

$$a^*(s) = \arg \max_{a \in A(s)} \left[R(s,a) + \sum_{s' \in \mathbf{S}} P_{s,s'}(a) h_n(s') \right] \tag{2.11}$$

Similar to the value iteration algorithm for discounted MDP, the value iteration algorithm for average cost MDP finds an ε-optimal stationary policy.

Policy Iteration for average cost MDP

Finally, the policy iteration counterpart for solving average cost MDP is listed as follow [16]

1. Select initial policy π_0 arbitrarily, set iteration $n = 0$, and determine the reference state $\dot{s}$.

2. (Policy Evaluation): solve ρ^{π_n} and $h^{\pi_n}(s)$ $s \in (S)$ from the following set of equations

$$\rho^{\pi_n} + h^{\pi_n}(s) = \left[R(s, \pi_n) + \sum_{s'=1}^{|S|} P_{s,s'}(a) h^*(s')\right], \tag{2.12}$$

$$h(\dot{s}) = 0. \tag{2.13}$$

3. (Policy Improvement): Update the policy as

$$\pi_{n+1}(s) = \arg\max_{\pi} \left[R(s, \pi) + \sum_{s' \in \mathbf{S}} P_{s,s'}(\pi) h^{\pi_n}(s')\right]. \tag{2.14}$$

4. Stopping criteria: When $\pi_{n+1} = \pi_n$.

The similar argument on finiteness of number of states and actions in each states implies that the algorithm finds the optimal solution in a finite number of iterations.

2.2.2 Linear Programming

The solution of the MDP can also be characterized using linear programming. In this subsection, we summarize the linear programming solution for MDP problem. The detailed treatment of linear programming formulation for MDP can be found in [16, 42, 84]

Primal Linear Program for discounted MDP

The primal linear programming for discounted MDP is listed as follow

$$\begin{aligned} &\text{Minimize} \quad \textstyle\sum_{s\in\mathbf{S}} \frac{1}{|\mathbf{S}|} V(s) \\ &\text{subject to} \quad V(s) \geq R(s,a) + \alpha \textstyle\sum_{s'\in\mathbf{S}} P_{s,s'}(a) V(s') \text{ for } a \in A(s), s \in \mathbf{S}. \end{aligned} \tag{2.15}$$

The coefficient $\frac{1}{N}$ in the objective function of primal linear programming has the interpretation that the process begins from any state $s \in \mathbf{S}$ with equal probability. In fact, the objective function can be replaced by

$$\sum_{s\in\mathbf{S}} \gamma(s) V(s), \tag{2.16}$$

where $\gamma(s) > 0$ and $\sum_{s\in\mathbf{S}} \gamma(s) = 1$. $\gamma(s)$ represents the initial distribution of states. The replacement will not modify the optimal policy obtained from the linear programming [84].

Dual Linear Program for discounted MDP

The associated dual representation of the above primal linear program is presented as follow

$$\begin{aligned} &\text{Maximize} \quad \textstyle\sum_{s\in\mathbf{S}} \sum_{a\in A(s)} f(s,a) R(s,a) \\ &\text{subject to} \quad \textstyle\sum_{a\in A(s')} f(s',a) - \alpha \sum_{s\in\mathbf{S}} \sum_{a\in A(s)} P_{s,s'}(a) f(s,a) = \frac{1}{|\mathbf{S}|} \text{ for } s' \in \mathbf{S} \\ &\qquad\qquad f(s,a) \geq 0 \text{ for } a \in A(s), s \in \mathbf{S}. \end{aligned} \tag{2.17}$$

We note that the right hand side (RHS) of the first constraint can be replaced with $\gamma(s')$, the initial distribution of states if it is known.

The solution of dual linear program can represent both randomized stationary policy and deterministic stationary policy. We note that the deterministic stationary policy assigns one action with probability one whenever the control agent is in

one particular state. In other word, the action the control agent chooses at state s is a function of the state s, i.e. $a : |\mathbf{S}| \rightarrow |\mathbf{A}|$. In contrast, the randomized stationary policy allows several actions to be chosen with some probabilities whenever the control agent is in state s. In the solution of linear programming shown above, the probability of taking action a whenever the control agent is in state s can be represented as follow

$$P_r\{a_n = a | s_n = s\} = \frac{f(s,a)}{\sum_{a' \in A(s)} f(s,a')}. \tag{2.18}$$

The primal and dual linear program for average cost MDP are presented as follows.

Primal Linear Program for average cost MDP

The primal linear program counterpart for average cost MDP is presented as follow

$$\begin{aligned} &\text{Minimize} \quad \rho \\ &\text{subject to} \quad \rho + h(s) \geq R(s,a) + \textstyle\sum_{s' \in \mathbf{S}} P_{s,s'}(a) h(s') \;\; a \in A(s) \;\; s \in \mathbf{S}. \end{aligned} \tag{2.19}$$

with ρ and $h(s)$ unconstrained.

Dual Linear Program for average cost MDP

The dual linear program for average cost MDP is represented as

$$\begin{aligned} &\text{Maximize} \quad \textstyle\sum_{s \in \mathbf{S}} \sum_{a \in A(s)} f(s,a) R(s,a) \\ &\text{subject to} \quad \textstyle\sum_{a \in A(s')} f(s',a) = \sum_{s \in \mathbf{S}} \sum_{a \in A(s)} P_{s,s'}(a) f(s,a), \quad \text{for } s' \in \mathbf{S} \\ &\qquad\qquad f(s,a) \geq 0 \text{ for } a \in A(s), s \in \mathbf{S} \\ &\qquad\qquad \textstyle\sum_{s \in \mathbf{S}} \sum_{a \in A(s)} f(s,a) = 1 \end{aligned} \tag{2.20}$$

Similar to the discounted MDP, the $f(s,a)$ has the interpretation of the joint probability of state s and action a. In particular, the resulting stationary policy

can be described as

$$P_r\{a_n = a|s_n = s\} = \frac{f(s,a)}{\sum_{a' \in A(s)} f(s,a')}. \tag{2.21}$$

From computational perspective, the LP formulation has higher dimension compared to the dynamic programming approach. In particular, the dual program has $|\mathbf{S}|$ rows and $\sum_{a \in \mathbf{S}} |A(s)|$ columns. However, the policy obtained from the dynamic programming is more likely to be a stationary deterministic policy, which implies that at any particular state, one action will be applied with probability one. In contrast, the LP solution includes the stationary randomized solution. This can be easily seen from (2.20) and (2.21), where (2.21) may range from $(0, 1)$. It is well known [9] that the optimal policy for unconstrained MDP is non-randomized stationary policy (pure/deterministic stationary policy). In the constrained MDP case, the solution is more likely to be randomized stationary policy. For this reason, it is more convenient to work on linear programming formulation in the case of constrained Markov Decision Process [9].

2.3 Reinforcement Learning

The computational methods described in previous subsections for solving the MDP require an explicit model of the cost function and the transition probabilities in the system. In many practical problem, however such a model is not available when the optimization is done, but instead the system and the cost function can be sampled at any time. This implies that the control agent knows the state space and the control space, but the control agent does not have the cost function and the system state transition probability. And it obtains the noisy cost function accordingly whenever it takes some action in some states. The reinforcement learning algorithm

is a popular paradigm for solving learning-control MDP [18, 97]. In RL, a control agent learns to make optimal decisions by experiencing the reward received, as the result of its action. The control agent does not require the explicit model of the environment. Hence, the reinforcement learning approach is very useful when the control agent has little knowledge of the environment and the exact function to be optimized.

An excellent tutorial on the RL algorithms can be found in [18, 97]. The RL algorithms can be used to solve different types of objective function in MDP, including the discounted MDP and average cost MDP. In this subsection, we will focus on the RL algorithm to solve the average cost MDP. In particular, we will describe a class of RL algorithm called adaptive actor critic algorithm. This class of algorithm can be used to solve both the discounted and average cost MDP, however, we will merely focus on adaptive actor critic algorithm to solve the average cost MDP.

The essence of adaptive actor critic algorithm used to solve average cost MDP is to update the relative state value function $h(s)$ and ρ in (2.4) using incremental averaging. In the following, we explain step-by-step the development of update equations in adaptive actor critic algorithm and show their connection with Bellman's equation. Define the operator $B(h^\pi(s)) = R(s, \pi(s)) + \sum_{s'=1}^{|S|} P_{s,s'}(\pi(s))h^\pi(s')$. Recall that given any stationary policy π, the corresponding average reward ρ^π and relative state-value $h^\pi(s)$ satisfy the following relation for all $s \in \mathbf{S}$

$$\rho^\pi + h^\pi(s) = \left[R(s, \pi(s)) + \sum_{s'=1}^{|S|} P_{s,s'}(\pi(s))h^\pi(s') \right]. \tag{2.22}$$

The above relation (2.22) can then be expressed as

$$\begin{aligned} h^{\pi}_{k+1}(s_k) &= B(h^{\pi}(s_k)) - \rho^{\pi}_k \\ \rho^{\pi}_{k+1} &= B(h^{\pi}(s_k)) - h^{\pi}_k(s_k). \end{aligned} \tag{2.23}$$

The RL algorithm eliminates the need for state transition probability by replacing the operator $B(\cdot)$ with $B'(h^{\pi}(s)) = R(s, \pi(s)) + h^{\pi}(s')$, where s' is the next state occurring in the sample path. Obviously, the next state s' occurs according to the probability $P_{s,s'}(\pi(s))$. The RL algorithm learns the state-value function as

$$\begin{aligned} h^{\pi}_{k+1}(s_k) &= (1-\alpha_k)h^{\pi}_k(s_k) + \alpha_k h^{\pi}_{k+1}(s_k) \\ &= (1-\alpha_k)h^{\pi}_k(s_k) + \alpha_k(B'(h^{\pi}_k(s_k)) - \rho^{\pi}_k) \\ &= h^{\pi}_k(s_k) + \alpha_k[R(s_k, \pi(s_k)) + h^{\pi}_k(s_{k+1}) - h^{\pi}_k(s_k) - \rho^{\pi}_k] \end{aligned} \tag{2.24}$$

Similarly, the average reward ρ is updated as

$$\begin{aligned} \rho^{\pi}_{k+1} &= (1-\beta_k)\rho^{\pi}_k + \beta_k \rho^{\pi}_{k+1} \\ &= (1-\beta_k)\rho^{\pi}_k + \beta_k(B'(h^{\pi}_k(s_k)) - h^{\pi}_k(s_k)) \\ \rho^{\pi}_{k+1} &= \rho^{\pi}_k + \beta_k[R(s_k, \pi(s_k)) + h^{\pi}_k(s_{k+1}) - h^{\pi}_k(s_k) - \rho^{\pi}_k], \end{aligned} \tag{2.25}$$

We note that α_k and β_k determine the weighting of the current and future estimate of the state value function and the average reward. The term $(R(s_k, \pi(s_k)) + h^{\pi}_k(s_{k+1}) - h^{\pi}_k(s_k) - \rho^{\pi}_k)$ is often referred to as the temporal difference [97] or the error between the current and next estimate. This temporal difference (error) guides the learning process. α_k and β_k determine the learning rate for the differential state value function and the average reward.

Since the Bellman equation chooses the action that optimizes the right hand side (RHS) of (2.4), there should be some function related to the decision made in each iteration. The adaptive actor critic algorithm chooses the decision/action

according to the Gibbs softmax method [97], i.e., action a_k is chosen in state s_k according to probability

$$Pr\{a_k = a | s_k = s\} = \frac{e^{p(s,a)}}{\sum_{b \in A(s)} e^{p(s,b)}}. \tag{2.26}$$

Whenever an action a_k is chosen at state s_k, the preference metric $p(s_k, a_k)$ is updated according to

$$p(s_k, a_k) = p(s_k, a_k) + \epsilon_k [R(s_k, a_k) + h_k(s_{k+1}) - h_k(s_k) - \rho_k], \tag{2.27}$$

where the ϵ_k determines the learning rate for the preference metric. The preference metric update equation has the following interpretation. The algorithm typically is initialized using $p(s, a) = 0, \forall s \in \mathbf{S}, \forall a \in \mathbf{A}$. This implies that initially the algorithm chooses every action uniformly at any particular state s. As the iteration proceeds, the action that results in increasing relative state value function $h_{k+1}(s_k)$ is prioritized by increasing the preference metric of choosing that particular action (2.27). In contrast, the action that results in smaller relative state value function (the temporal difference is negative) is penalized by reducing its preference metric. In this sense, the equations (2.24)-(2.27) choose the action that maximizes the RHS of (2.4). Hence, the adaptive actor critic algorithm resembles the Bellman optimality equation. The application of adaptive actor critic algorithm in joint adaptive modulation and power level selection will be given in Chapter 3.

2.4 Constrained Markov Decision Process

The Markov Decision Process explained in previous sections are unconstrained MDP. In many engineering problems, there are many inherent physical limitations that constrain the optimization problem. Due to this inherent physical

limitations, it is important to consider optimization framework that can handle constraints. In the next subsections, we describe the structure of discounted constrained Markov Decision Process (CMDP), the characterization of the solution of discounted CMDP, the structure of average cost CMDP, and the solution of average cost CMDP. An extensive discussion on the CMDP can be found in monograph [9]. The solutions of the CMDP is characterized by the constrained linear programs similar to the cases in Section 2.2.2.

2.4.1 Discounted Constrained Markov Decision Process

The structure of discounted constrained Markov decision process is given as

$$\begin{aligned} &\text{Maximize} && \lim_{n\to\infty} E_{\pi}^{s_0}\big[\sum_{k=1}^{n} \alpha^{k-1} R(s_k, \pi(s_k))\big] \\ &\text{s. t.} && \lim_{n\to\infty} E_{\pi}^{s_0}\big[\sum_{k=1}^{n} \alpha^{k-1} C_l(s_k, \pi(s_k))\big] \geq B_l \text{ for } l = 1, \cdots, L, \end{aligned} \tag{2.28}$$

where $R(s_k, a_k)$ is the reward function when the control agent takes action a_k at the state s_k. We note that the subscript in the variables indicates the time. $C_l(s_k, a_k)$ represents the constrained function obtained by the agent when it takes action a_k and at state s_k. L is the number of constraints. The $E_{\pi}^{s_0}$ is the conditional expectation when policy π is used and the initial state is s_0. We note that similar to the unconstrained MDP case, the state evolution in the system is governed by the state transition probability $P_{s,s'}(a)$. The equation (2.28) can be interpreted as maximize the discounted sum of the reward function, while maintaining the discounted sum of the constraints feasible. The optimal solution for the problem is to find decision/action policy such that the objective function is minimized while satisfying the constraints.

2.4.2 Average Cost Constrained Markov Decision Process

Similarly, the structure of average cost constrained Markov decision process can be represented as follow

$$\begin{aligned} &\text{Maximize} \quad \lim_{n\to\infty} \tfrac{1}{n} E_\pi^{s_0}\big[\textstyle\sum_{k=1}^{n} R(s_k, \pi(s_k))\big] \\ &\text{s. t.} \qquad \lim_{n\to\infty} E_\pi^{s_0}\big[\textstyle\sum_{k=1}^{n} C_l(s_k, \pi(s_k))\big] \geq B_l \text{ for } l = 1, \cdots, L. \end{aligned} \tag{2.29}$$

The optimal solution of the above problem is to find the decision/action policy that minimize the average reward of the system while maintaining the feasibility of the time average constrained functions. We note the structure of CMDP is very general. For example, in the telecommunication network optimization, the designer always wants to guarantee the quality of service (QoS). This can be represented as the average time of the instantaneous signal to noise ratio (SNR) should be larger than the minimum SNR, i.e. $\lim_{n\to\infty} E_\pi^{s_0}\big[\sum_{k=1}^{n} \gamma(s_k, \pi(s_k))\big] \geq \gamma_{min}$, where $\gamma(s_k, a_k)$ is the instantaneous SNR when the system is in state s_k and the control agent takes action a_k. γ_{min} is the minimum SNR.

2.5 Solutions of Constrained Markov Decision Process

Similar to the linear programming formulation for the unconstrained MDP, we have primal and dual linear program for each of discounted and average cost problem. The characterization of the solution of discounted CMDP can be obtained by solving the following primal linear program

$$\begin{aligned} &\min_{V,\lambda} \quad \textstyle\sum_{s\in\mathbf{S}} \gamma(s) V(s) - \sum_{l=1}^{L} \lambda_l B_l \\ &\text{s. t.} \qquad V(s) \geq R(s,a) + \textstyle\sum_{l=1}^{L} \lambda_l C_l(s,a) + \alpha \sum_{s'\in\mathbf{S}} P_{s,s'}(a) V(s') \text{ for } a \in A(s),\ s \in \mathbf{S}, \end{aligned} \tag{2.30}$$

where $\gamma(s)$ is the initial distribution of the state, which satisfies $\sum_{s\in\mathbf{S}}\gamma(s)=1$. λ_l is the corresponding Lagrange multiplier for the l-th constraint, $A(s)$ is the set of action when the system is in state s.

The dual representation of the above linear program can be written as

$$\begin{aligned}
&\text{Maximize} \quad \textstyle\sum_{s\in\mathbf{S}}\sum_{a\in A(s)} f(s,a)R(s,a)\\
&\text{subject to} \quad \textstyle\sum_{a\in A(s')} f(s',a)=\alpha\sum_{s\in\mathbf{S}}\sum_{a\in A(s)} P_{s,s'}(a)f(s,a) \text{ for } s'\in\mathbf{S}\\
&\qquad\qquad f(s,a)\geq 0 \text{ for } a\in A(s), s\in\mathbf{S}\\
&\qquad\qquad \textstyle\sum_{s\in\mathbf{S}}\sum_{a\in A(s)} f(s,a)C_l(s,a)\geq B_l \text{ for } l=1,\cdots,L.
\end{aligned} \tag{2.31}$$

We note that $f(s,a)$ has the interpretation of the joint probability of state s and action a. In particular, the resulting stationary policy can be described as $P_r\{a_n=a|s_n=s\}=\frac{f(s,a)}{\sum_{a'\in\mathbf{A(s)}}f(s,a')}$.

The primal program for solving average cost CMDP is listed as follow

$$\begin{aligned}
&\min_{\rho,h,\lambda} \quad \rho-\textstyle\sum_{l=1}^{L}\lambda_l B_l\\
&\text{s. t.} \qquad \rho+h(s)\geq R(s,a)+\textstyle\sum_{l=1}^{L}\lambda_l C_l(s,a)+\alpha\sum_{s'\in\mathbf{S}}P_{s,s'}(a)h(s')\\
&\qquad\qquad a\in A(s),\ s\in\mathbf{S},
\end{aligned} \tag{2.32}$$

The corresponding dual linear program can be represented as

$$\begin{aligned}
&\text{Maximize} \quad \textstyle\sum_{s\in\mathbf{S}}\sum_{a\in A(s)} f(s,a)R(s,a)\\
&\text{subject to} \quad \textstyle\sum_{a\in A(s')} f(s',a)=\alpha\sum_{s\in\mathbf{S}}\sum_{a\in A(s)} P_{s,s'}(a)f(s,a) \text{ for } s'\in\mathbf{S}\\
&\qquad\qquad f(s,a)\geq 0 \text{ for } a\in A(s), s\in\mathbf{S}\\
&\qquad\qquad \textstyle\sum_{s\in\mathbf{S}}\sum_{a\in A(s)} f(s,a)C_l(s,a)\geq B_l \text{ for } l=1,\cdots,L\\
&\qquad\qquad \textstyle\sum_{s\in\mathbf{S}}\sum_{a\in A(s)} f(s,a)=1
\end{aligned} \tag{2.33}$$

Suppose under a stationary policy π, we say that there are $m(s,\pi)$ randomizations in state s if there are exactly $m+1$ actions in $A(s)$ for which $Pr(a|s)>0$. Moreover, let's denote the number of randomizations under stationary policy π as

$\sum_{s\in\mathbf{S}} m(s,\pi)$. The following theorem [9] is very useful to determine the bound on the number of randomizations in solving CMDP.

Theorem 4 *If the CMDP is feasible then there exists an optimal stationary policy such that the total number of randomization* $\sum_{s\in\mathbf{S}} m(s,\pi)$ *that it uses is at most* L*, where* L *is the number of constraints.*

2.6 Stochastic Approximation

The subject on stochastic approximation (SA) was introduced as an iterative method to find the zeros of a function when only a perturbed value of the function at the current estimate of the zero is known [88]. However, SA has found many application in modern disciplines, such as adaptive algorithm, neural networks, stochastic optimization, reinforcement learning [15, 34, 57, 93]. The stochastic approximation iteration typically involves algorithm of the form

$$\theta_{n+1} = \theta_n + \lambda_{n+1} F_{n+1}(\theta_n). \tag{2.34}$$

There are two common ways to prove the convergence of stochastic approximation algorithm. First, the proof of convergence is based on supermartingale/martingal convergence theorem [91]. The second way is based on ordinary differential equation (ODE) approach [57]. The latter approach shows that the interpolated process from stochastic approximation iteration asymptotically converges to the following ODE

$$\dot{\theta} = F(\theta). \tag{2.35}$$

Therefore, the solution θ will asymptotically converges to the limit point of the ODE. The detailed treatment of the SA convergence proof can be found in [15, 34, 57].

Chapter 3

Near-Optimal Modulation and Power Selection using Reinforcement Learning

This chapter[1] gives an application of maximizing the efficiency of one unit energy. In particular, we consider the problem of average throughput maximization per total consumed energy in packetized sensor communications. Our study results in a near-optimal transmission strategy that chooses the optimal modulation level and transmit power while adapting to the incoming traffic rate, buffer condition, and the channel condition. We investigate two scenario, the point-to-point and multi-node communication. Many solutions of the previous works require the state transition probability, which may be hard to obtain in a practical situation. Therefore, we are motivated to propose and utilize a class of reinforcement learning (RL) algorithms (called adaptive actor critic algorithm) to obtain the near-optimal policy

[1]Material in this chapter has been published in IEEE Journal on Selected Area in Communications [80]

in point-to-point communication and a good transmission strategy in multi-node scenarios. For comparison purpose, we develop the stochastic models to obtain the optimal strategy in the point-to-point communication. We show that the learned policy is close to the optimal policy. We further extend the algorithm to solve the optimization problem in a multi-node scenario by *independent* learning. We compare the learned policy to a simple policy, where the agent chooses the highest possible modulation and selects the transmit power that achieves a predefined signal to interference ratio (SIR) given one particular modulation. The proposed learning algorithm achieves more than twice the throughput per energy compared to the simple policy, particularly in high packet arrival regime. Beside the good performance, the RL algorithm results in a simple, systematic, self-organized, and distributed way to decide the transmission strategy.

The contributions of this chapter are as follows; we propose an optimization framework that generally captures several parameters from different communication layers and develop practical algorithms based on the RL algorithm to learn the near-optimal control policy in the point-to-point communication and a good transmission strategy in multi-node scenarios. The proposed optimization scheme is simple, inherently distributed and self-organized. This chapter is organized as follows. We first give the motivation and summary of the previous works. Then, we formulate the throughput maximization per total consumed energy in a point-to-point communication scenario. We extend the formulation of the point to point communication to the multi-node scenario. Finally, the applicability of the algorithms to wireless sensor networks is discussed.

3.1 Motivation

Several attempts to design the resource allocation protocol for WSN are based on the existing wireless resource allocation methods. We first briefly outline the existing wireless resource management approaches, which are closer to our method. In [11], a power control scheme for wireless packet networks is formulated using dynamic programming (DP). The extension of this work to multi-modal power control is also investigated in [12]. In these two schemes, the power control follows some threshold policy that balances between the buffer content and the channel interference. The DP formulation for power control with imperfect channel estimation is addressed in [51]. They show that the DP solution is better than the fixed signal to interference ratio (SIR) approach. Jointly optimized bit-rate and delay control for packet wireless networks has also been studied within the DP framework [86]. Most of the literature assumes the knowledge of the exact probability model and obtain the optimal solution by solving Bellman's optimality equation [16]. In practice, the probability models may not be available when the optimization is being done. This motivates us to develop and investigate an optimization scheme that learns the optimal policy without knowing the probability model.

We focus on the average throughput maximization per total consumed energy in packetized wireless sensor communications from an optimal control point of view. We consider the point-to-point communication and the multi-node scenarios. In both cases, we assume that an intelligent control agent resides in the transmitter and decides the right action in the right situation. We propose to utilize the reinforcement learning algorithm to solve the online optimization problem. In point-to-point communication, the communication takes place between one trans-

mitter and one receiver. Before the transmission, the transmitter observes the number of packets in its buffer and the channel gain from the previous transmission. Based on this knowledge, the objective of the intelligent control agent is to find the best modulation level and transmit power to maximize the long-term average throughput per total consumed energy. The long-term average throughput per total consumed energy is obtained by averaging the throughput per energy at every transmission. The total consumed energy at every transmission consists of the transmission energy and buffer processing cost. Clearly, the buffer in the transmitter is affected by the agent's decision. In this scenario, we compare the optimal policy with the policy learned by RL algorithm and show that the RL algorithm obtains the near-optimal control policy. Moreover, we also compare the learned policy with a policy, where the control agent chooses the highest possible modulation and uses the transmit power that achieves a predefined signal to interference ratio (SIR) given one particular modulation. We refer this policy as the *simple* policy. We demonstrate that the proposed learned policy obtains more than twice throughput per energy compared to the simple policy, especially in the high mean packet arrival region.

In contrast to the point-to-point communication, we consider N transmitters simultaneously communicate to one receiver in multi-node scenario. The channel link experienced by one node depends on the transmission power (decision) employed by other nodes in multi-node scenarios. Hence, the optimal (equilibrium) solution generally depends on the policy employed by the other nodes. We extend the RL algorithm to solve the multi-node problem. We propose to let every node *independently* learn its transmission strategy based on its buffer condition and the measured channel interference. Similarly, we compare the independent learned

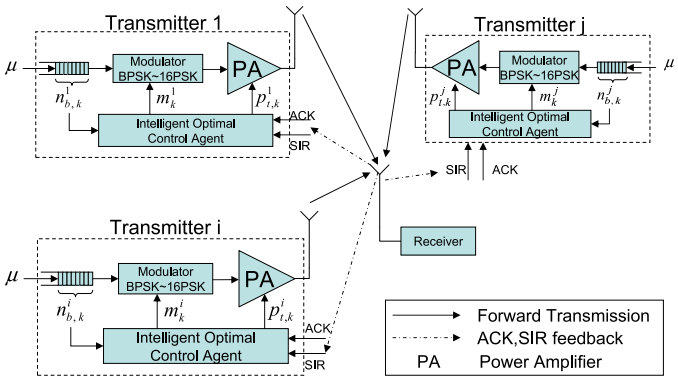


Figure 3.1: Interaction of nodes in distributed control agent

policy to the simple policy where each node chooses the highest modulation level and selects the transmit power level to achieve a predefined SIR at given modulation. The proposed modified RL algorithm provides a significant improvement in the average throughput per total consumed energy.

3.2 Throughput maximization in point-to-point communication

The interaction between the communicating nodes for both scenarios can generally be illustrated in Figure 3.1. The point-to-point communication can be considered as the special case where only one transmitter and receiver are participating in the communication. We will refer to this illustration, when explaining the interaction between the optimal control agent and the environment. We study the average throughput maximization per total consumed energy considering the parameters of the channel condition, the transmitter buffer, the modulation and the transmit power. In the following, we first present the reward function and the adaptive actor critic (AC) algorithm used to learn the near-optimal policy. In order to com-

pare the learned strategy with the optimal solution, we present the models that constitute the MDP in point-to-point communication. These models are required in solving the Bellman optimality equation. We note that the proposed optimal control framework does not depend on any particular model used in our formulation. Hence, more accurate models, if further discovered, can be employed without changing the optimization framework.

3.2.1 Reward function

Several utility functions or reward functions have been used in the context of power control schemes. In [11] [12], the transmit power and cost incurred in the buffer are used as the objective functions to be minimized. In [90], the number of information bits successfully transmitted per Joule is used as the objective function. In the application of wireless sensor networks, the energy consumption, the throughput and the delay are all very critical parameters. We certainly do not want to minimize the energy consumption with an unacceptable throughput or infinite delay. Hence, we employ the number of successfully transmitted packets per total consumed energy as our objective function. To enforce the bounded delay transmission, we incorporate the buffer processing cost/energy into the total energy, which is the summation of the transmission energy and the buffer processing cost/energy. Including the buffer processing cost/energy minimizes the possibility of buffer overflow, which can be interpreted as enforcing the Quality of Service (QoS).

Suppose the transmitter sends a packet consisting L_b information bits, and let the number of bits in one packet after adding error decoding code be L bits. The transmission rate is R bits/s. Figure 3.1 illustrates this scenario, where only one

transmitter and receiver pair is communicating. We assume the receiver feeds back its current received channel gain γ to the transmitter before the next transmission. Let m and p_t denote the modulation level and the transmission power. Also let $S(\Gamma(\gamma, p_t), m)$ denote the probability of successful packet reception, where $\Gamma(\gamma, p_t)$ is the targeted signal to interference ratio. Let K denote the number of retransmissions required to successfully transmit a packet. Assuming each transmission is statistically independent, K is a geometric random variable with mass function

$$P_K(k) = S(\Gamma(\gamma, p_t), m)[1 - S(\Gamma(\gamma, p_t), m)]^{k-1}. \tag{3.1}$$

The time duration for each transmission is L/Rm seconds and total retransmission time becomes KL/Rm seconds. When the transmitted power is p_t watts, the energy consumed per packet transmission is $E[K]p_tL/Rm$, where $E[\cdot]$ is the expectation. In one packet, the useful information portion is only L_b/L. Hence, the utility function becomes

$$\frac{GoodPacket}{TransmitEnergy} = \frac{L_b}{L} \cdot \frac{R \cdot m \cdot S(\Gamma(\gamma, p_t), m)}{L \cdot p_t} \tag{3.2}$$

where the unit for the utility function is packet per Joule. Let p_b denote buffer processing cost/energy. The buffer processing cost is assumed to be a monotonically increasing function with respect to the number of packets in the buffer n_b, i.e. $p_b = f(n_b)$. Thus, the reward function is expressed as

$$R((n_b, \gamma), (m, p_t)) = \begin{cases} \frac{L_b}{L} \cdot \frac{R \cdot m \cdot S(\Gamma(\gamma, p_t), m)}{L \cdot (p_t + f(n_b))} \times 10^{-3} & \text{if } n_b \neq 0 \text{ and } p_t \neq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.3}$$

where (n_b,γ) is the aggregate state and (m,p_t) is the action that can be taken by the control agent. The reward function is interpreted as the number of good

received packets per total energy consumed. We note that the reward function is equal to zero if there is no packet in the buffer or no transmission occurs (transmit power is zero). Also by adding the buffer processing cost, the control agent will gradually become more aggressive as the buffer increases. Hence, the probability of buffer overflow will be decreased.

3.2.2 Near-Optimal Solution using Actor-Critic Algorithm

In this section, we present the complete Actor-Critic (AC) algorithm developed in Section 2.3 to solve MDP with average reward per stage. The architecture of an Actor-Critic algorithm is shown in Figure 3.2. As we can infer from its name, the AC algorithm consists of two major parts, the actor and the critic. The policy structure is known as the actor, because it decides the action, and the estimated state value function is known as the critic, since it generates temporal difference (error) that criticizes the actions made by the actor. The complete AC algorithm is shown in Table 3.1 . The AC algorithm uses the state value function update and the average reward update as in (2.24) and (2.25). The actor selects the decision according to the Gibbs softmax method [97]. In parallel with the discussion in Section 2.3, the Gibbs softmax method (2.27) acts as the actor and the temporal difference serves as the critic (2.24-2.25).

In the Gibbs softmax method, the actor chooses the action with the highest conditional probability of state-action $\pi(s_k, a_k) = Pr(a_k|s_k)$. The higher $\pi(s_k, a_k)$ is, the more likely a_k will be chosen. The algorithm starts with equal $\pi(s_k, a_k)$ for every action. Therefore, the actor has equal probability to choose any available actions at the initial stage. This stage is often referred to as an exploration stage. As in all RL algorithms, the AC algorithm needs to balance the exploration and

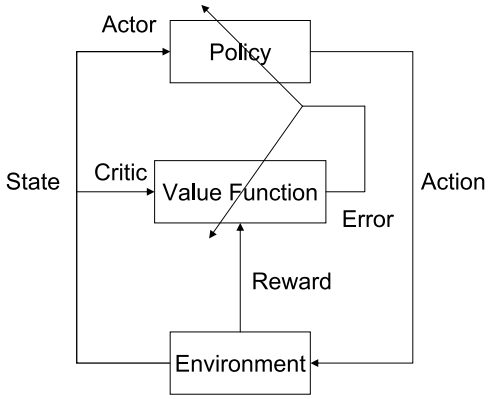


Figure 3.2: Actor-Critic architecture

exploitation step in the learning process. The exploitation step is used to search for the average reward maximizing decision and the exploration step is used to try out all possible best decisions [97].

3.2.3 The Optimal Dynamic Programming Solution

As described in Section 2.2, the solution of the Bellman optimality equation requires knowledge of the state transition probability. Before describing the optimal solution, we present the models of each of the components that constitute the MDP system in a point-to-point scenario as follows.

Finite state Markov channel (FSMC)

In point-to-point communication, the wireless channel dynamic can be modelled using a Finite State Markov Channel (FSMC). The approach in the Finite State Markov Channel (FSMC) for wireless channels [101] [105] is to partition the received signal-to-noise ratio (SNR) or the equivalent channel gain into a finite number of intervals. Suppose that the channel gain is partitioned into K intervals, $0 = \Gamma_0 < \Gamma_1 \ldots < \Gamma_K$. The channel gain is said to be in state k if it is between

Table 3.1: Actor-Critic Algorithm

Actor-Critic Algorithm

Initialize α, β, ϵ, $k = 0$, $h(s_k) = 0$ for all $s_k \in \mathbf{S}$, and $\rho_k = 0$.

Set preference function $p(s, a) = 0, \quad \forall s \in \mathbf{S}, \forall a \in \mathbf{A}(s)$.

Set s_0 arbitrarily.

Loop for $k = 0, 1, 2, \ldots$

1. *Choose a_k in s_k according to Gibbs softmax method*:

   $\pi(s_k, a_k) = Pr(a_k = a | s_k = s) = e^{p(s,a)} / \sum_b e^{p(s,b)}$

2. *Get Reward from current decision and observe next state s_{k+1}*:

   $r = R(s_k, a_k)$

3. *Evaluation of temporal difference (error)*

   $\delta = r + h(s_{k+1}) - h(s_k) - \rho_k$

4. *Update relative state value function and average reward per state*

   $h(s_k) = h(s_k) + \alpha\delta$

   $\rho_{k+1} = \rho_k + \beta\delta$

5. *Update actor preference*

   $p(s, a) = p(s, a) + \epsilon\delta$

End Loop.

Γ_{k-1} to Γ_k. In the packet transmission system, the channel transition occurs at the time slot boundary, and the channel gain is constant during one time slot of transmission. Furthermore, the channel transition only occurs from a given state to its two adjacent states as in Figure 3.3. The state transition probability completely specifies the dynamics of the channel and is determined as follows [101] [105]

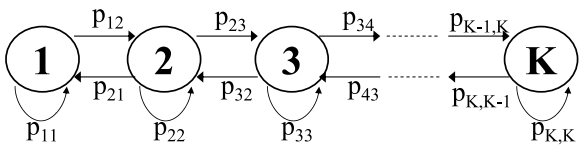


Figure 3.3: FSMC with K-state

1. Steady-state probabilities:

$$\zeta_k = \int_{\Gamma_{k-1}}^{\Gamma_k} p(\gamma) d\gamma, \qquad k = 1, \ldots, K. \tag{3.4}$$

In a Rayleigh fading channel, γ is exponentially distributed with probability density function as $p(\gamma) = 1/\gamma_0 \exp(-\gamma/\gamma_0)$, where γ_0 is the average channel gain.

2. State transition probabilities:

$$\begin{aligned} p_c(k, k+1) &= N(\Gamma_{k+1}) T_p/\pi_k \qquad k = 1, \ldots, K-1 \\ p_c(k, k-1) &= N(\Gamma_k) T_p/\pi_k \qquad k = 2, \ldots, K. \end{aligned} \tag{3.5}$$

where $N(.)$ is the level crossing function given by $N(\Gamma) = \sqrt{2\pi\Gamma/\gamma_0} f_d \exp(-\Gamma/\gamma_0)$, T_p is the packet transmission time and f_d is the maximum doppler frequency.

State transition probability construction

We construct the system state as the aggregate of the number of packets in the buffer, n_b and the channel gain, γ, that is $s \equiv (n_b, \gamma)$. The control space consists of the modulation level and transmit power, i.e. $a \equiv (m, p_t)$. The state transition probability maps $(n_b, \gamma) \times (n_b, \gamma) \times (m, p_t) \rightarrow [0, 1]$. In particular, the state transition probability depends on the probability of packet arrival, the channel transition probability and the successful packet transmission probability.

We model the packet arrival process as a Poisson process with mean packet arrival rate, μ. The channel is modelled as FSMC and the channel gain state transition probability is calculated according to Section 3.2.3. The successful packet transmission probability, $S(\Gamma(\gamma, p_t), m)$, depends on the targeted SIR, $\Gamma(\gamma, p_t)$, which is represented as

$$\Gamma(\gamma, p_t) = \gamma \times \frac{W}{R} \frac{p_t \times A_t}{\sigma^2}, \tag{3.6}$$

where γ, modelled as the FSMC, is the channel gain variation between the transmitter and receiver, W denotes the total bandwidth of the transmission, R is the transmission rate, (W/R is also known as the processing gain in CDMA literature), $A_t \propto 1/d^4$ is the attenuation factor resulting from the path loss, d is the distance between the transmitter and receiver, and σ^2 is the variance of the thermal noise.

Denote the number of packet arrivals as $n_a = 0, 1, \ldots$, the probability of packet arrival as $p_a(n_a)$, the successful packet transmission probability as $S(\Gamma(\gamma, p_t), m)$, and the channel transition probability as $p_c(\gamma_k, \gamma_{k+1})$. Here, $p_c(\gamma_k, \gamma_{k+1})$ indicates the transition probability of the channel gain from state γ_k at time instant k to γ_{k+1} at the next time instant. Suppose the current state is $s_k = (n_{b,k}, \gamma_k)$, where $n_{b,k}$ is the number of packets in the buffer at time k, and γ_k is the channel gain that is fed back. The action taken at time k is $a_k = (m_k, p_{t,k})$, where m_k and $p_{t,k}$ denote the modulation level and the transmit power employed at time k. Assuming that the events of packet arrival, successful transmission and channel transition are all mutually independent, the corresponding system state transition probability is determined as

1. Transmission failure:

$$\begin{aligned} s_{k+1} &= (n_{b,k} + n_a, \gamma_{k+1}) \\ P_{s_k, s_{k+1}}(a_k) &= p_a(n_a)(1 - S(\Gamma, m_k)) p_c(\gamma_k, \gamma_{k+1}) \end{aligned} \tag{3.7}$$

2. Successful transmission:

$$s_{k+1}=(n_{b,k}+n_a-m_k,\gamma_{k+1})$$
$$P_{s_k,s_{k+1}}(a_k)=p_a(n_a)S(\Gamma,m_k)p_c(\gamma_k,\gamma_{k+1}) \tag{3.8}$$

The formulation of MDP has the following interpretation. Before a packet transmission, the transmitter is in some state (obtained from the previous history of transmission, i.e. buffer content and channel condition, see Figure 3.1). The transmitter uses this information to determine what modulation and transmit power should be used to maximize the average throughput per total consumed energy. At the end of a packet transmission, the transmitter obtains feedback information from the receiver containing the quantized channel gain and ACK/NACK. The quantized channel gain is used to track the channel evolution γ_k. The ACK/NACK is used to update the buffer content. When an ACK signal is received, the transmitter will send the following packet at the next transmission time. Otherwise, it retransmits the packet. The number of successful transmitted packets per the energy consumed in one transmission time is recorded as the reward, $R(s_k,a_k)=R((n_{b,k},\gamma_k),(m_k,p_{t,k}))$.

3.2.4 Numerical Results

In this section, we construct the simulation using parameters shown in Table 3.2. We note that the total number of states are 72 and total number of actions are 44. Given the MDP state transition probability, the optimal solution of the posed MDP problem is solved numerically using the policy iteration method [16]. We compare the optimal solution to the policy learned by the Actor Critic (AC) algorithm. The AC algorithm parameters are $\alpha=0.01$, $\beta=0.0001$ and $\epsilon=0.01$. For comparison

Table 3.2: Single node Simulation parameters

Packet size	$L_b = 64$, $L = 80$
System Parameters	$W = 10$Mhz, $R = 100$kbits/s, $T_p = 0.8$ms $\sigma^2 = 5 \times 10^{-15}$W
Channel Gain	$f_D = 50$Hz, $\gamma \in= [-8, -6, ..., 8]$ dB $A_t = 1.916 \times 10^{-14}$
Buffer Cost	$f(n_b) = 0.05(n_b + 4)$ if $n_b \neq \max(n_b)$ $\max(n_b) = 7$, $f(\max(n_b)) = 3$
modulation level	m=1,2,3,4 (BPSK,QPSK,8PSK,16PSK),
Packet success probability	$S(\Gamma(\gamma, p_t), m)=(1 - P(\Gamma(\gamma, p_t), m))^L$ $P(\Gamma, m)=erfc(\sqrt{\Gamma} * \sin(\frac{\pi}{2^m}))$
Transmit power	$p_t = [0, 0.2, ..., 2]$ Watt
SNR range	$\Gamma = [0, 1, ..., 24]$ dB

purposes, we also simulate the *simple policy*, where the transmitter tries to transmit at the highest throughput (modulation) possible, while maintaining a predefined link SNR given a particular modulation. Specifically, The transmitter chooses BPSK to transmit when there is only one packet in the buffer, and it chooses QPSK, 8PSK and 16PSK to transmit when there are two packets, three packets, and more than 4 packets in the queue, respectively. For each modulation, the transmitter selects the transmit power to achieve a fixed predefined SNR. We use (6, 10, 15, 20) dB as the predefined link SNR for BPSK to 16PSK, respectively. These predefined SNRs can achieve more than 80% packet correct reception probability.

Figure 3.4 shows the average throughput learned by the AC algorithm and the optimal throughput when $\mu = 2.0$. It is obvious that the learned throughput is asymptotically very close to the optimal one. Moreover, due to the selection of the small, *constant* learning parameter α, β and ϵ (as opposed to the decreasing magnitude of learning parameters with time) the AC algorithm has the ability to

track the variation in the governing probability as demonstrated in Figure 3.4. In this figure, the mean packet arrival rate varies as μ =(0.5, 1.0, 1.5, 2.0, 1.0). Based on the sample realization, the AC algorithm adjusts the learned policy adapting to different packet arrival rates. The capability of the AC algorithm to obtain the near-optimal policy and track the variation in the governing probability is due to the fact that the algorithm explores all the possible decisions and selects the throughput maximizing policy. This exploration is achieved by the initial stage of the Gibbs softmax method used in the actor part of the algorithm.

The corresponding optimal and learned policies for $\mu = 2.0$ are shown in Figure 3.5. In these figures, the channel is better when the channel gain is larger and the buffer content indicates number of packets in the buffer. For the same buffer content, the optimal policy tends to use higher modulation levels when the channel is good and lower modulation when the channel is bad. The agent also tends to select higher power levels when the channel is bad to guarantee acceptable throughput. At the same channel gain, as more packets are queued in the buffer, the agent becomes more aggressive and attempts a higher modulation and power level. This effect is due to including the buffer processing cost in the reward function, causing the agent to try to balance the transmission energy and buffer processing cost/energy to obtain the maximum average throughput per total expended energy. Moreover, both the optimal DP and the near-optimal AC solution jointly decide the best modulation and transmit power to maximize the average throughput per expended energy.

Figure 3.6 shows the throughput that is achieved for the optimal policy, AC learned policy and the *simple policy* described in the previous paragraph for various packet arrival rate. It is obvious that the policy learned by the AC algorithm is

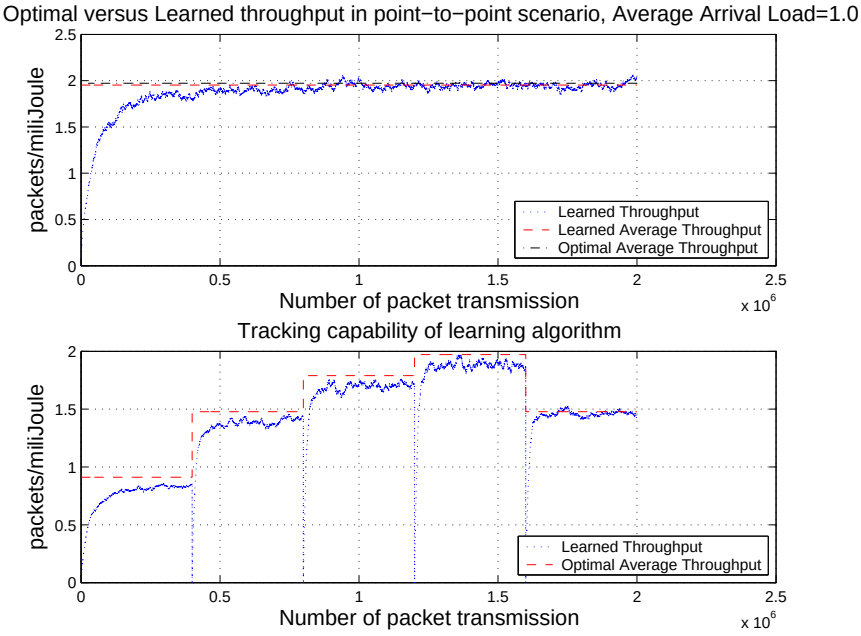


Figure 3.4: Performance of the learning algorithm

very close to the optimal policy. Compared to the *simple policy*, the AC algorithm obtains twice to three times throughput per total expended energy. Hence it is a higher energy efficiency scheme. It is important to point out that the optimal solution may not be feasible in practical applications, since the optimal solution requires the knowledge of channel transition probability and packet arrival probability. The AC algorithm and *simple policy* algorithm do not require any knowledge of governing probability, but the AC algorithm is still able to obtain a near-optimal average throughput.

3.3 Multi-node Energy-Aware Optimization

In this section, we extend the throughput maximization per total consumed energy in point-to-point communication to the multi-node scenario as shown in Figure 3.1,

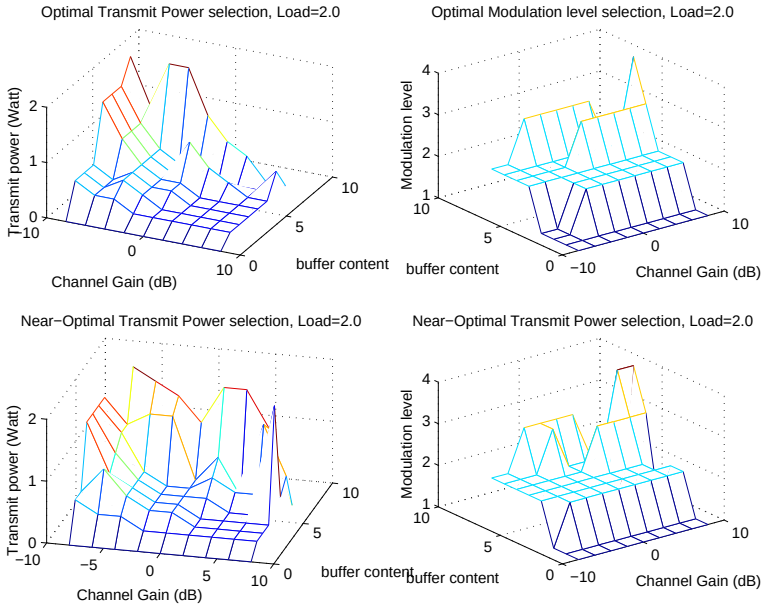


Figure 3.5: Learned and optimal policies, packet arrival load $\mu = 2.0$

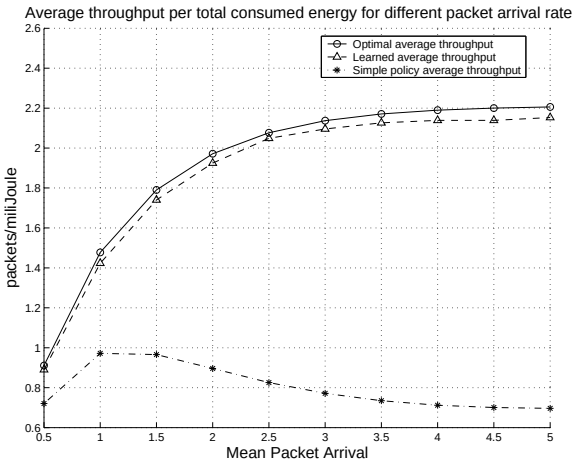


Figure 3.6: Average throughput corresponding to different packet arrival load $\mu = 2.0$

where multiple transmitters send packets to one receiver. The main difference between these two scenarios is the channel model. In point-to-point communication, the channel gain evolves according to the FSMC model and it is unresponsive to the transmitter power selection. The transmitter exercises all the available actions (modulation level and transmission power) to obtain the highest throughput per energy, while adapting to the channel variation, packet arrival rate and the buffer condition . In a multi-node scenario, the interference in one link depends on the power transmitted from other nodes. In fact, the channel experienced by one particular node depends on its previous decision and other nodes' decisions. When one node increases its power level, it will increase the interference experienced by the other nodes. This event may trigger the other nodes to increase their transmission power and may result in the increasing of the channel interference experienced by the original node.

In the following, we first describe the channel model and problem formulation in multi-node communication. We develop an extension of the AC algorithm for the multi-node problem and evaluate the performance by simulations. Similarly to the point-to-point communication, we compare the learned policy with the *simple policy*, where the each transmitter chooses the highest modulation possible, while maintaining a predefined signal-to-interference ratio (SIR) of the link for a particular modulation.

3.3.1 Channel model for multi-node communication and Problem Formulation

The channel model in the multi-node scenario captures the interaction dynamics between each node. This interaction is described in terms of the received SIR of

each node. Suppose there are N nodes that want to simultaneously communicate with the receiver. The SIR of link i can be expressed as [46]

$$\Gamma^i(p_t^1, \cdots, p_t^N) = \frac{W A_t^i p_t^i}{R(\sum_{j \neq i} A_t^j p_t^j + \sigma^2)}, \tag{3.9}$$

where W and R are the system bandwidth and transmission rate, p_t^i is the transmission power employed by node i, A_t^i is the path loss corresponding to link i, and σ^2 is the variance of the thermal noise. The path loss A_t^i depends on the distance between the transmitter i and the receiver, that is $A_t^i = c/(d^i)^4$, where d^i is the distance between the transmitter i and the receiver. Equivalently, (3.9) can be written in dB as

$$\Gamma^i(p_t^1, \cdots, p_t^N)dB = 10 \log_{10} \left(\frac{W p_t^i}{R} \right) - \eta^i, \tag{3.10}$$

where $\eta^i = 10 \log_{10} \left(\frac{\sum_{j \neq i} A_t^j p_t^j + \sigma^2}{A_t^i} \right)$ is the equivalent interference of link i. Other nodes' power transmission influences the link quality of node i through the relation (3.10).

The MDP formulation of multi-node scenarios is similar to the formulation of the point-to-point communication case. We point out the differences as follows. At time instant k, the system state at node i is $s_k^i \equiv (n_{b,k}^i, \eta_k^i)$, where $n_{b,k}^i$ and η_k^i are the number of packets in the transmitter's buffer and the quantized equivalent link interference experienced by node i, and the action space is $a_k^i \equiv (m_k^i, p_{t,k}^i)$, where m_k^i and $p_{t,k}^i$ denote the modulation level and transmission power employed by node i at time instant k, respectively. We assume that the transmitting node i receives the quantized estimation of its link quality from the receiver through the error free channel, hence the transmitting node knows the history of the quantized interference of its link, η^i. This implies that the control agent can fully observe its corresponding state, $s^i \equiv (n_b^i, \eta^i)$. Having observed its system state $s_k^i \equiv (n_{b,k}^i, \eta_k^i)$

at time instant k, every node exercises all the possible actions $a_k^i \equiv (m_k^i, p_{t,k}^i)$ to maximize its average throughput per unit energy while adapting to the incoming traffic, buffer condition and the link quality. The reward obtained is represented as

$$R^i((n_b^i, \eta^i), (m^i, p_t^i)) = \begin{cases} \frac{L_b \cdot R \cdot m^i \cdot S^i(\Gamma^i, m^i)}{L^2 \cdot (p_t^i + f^i(n_b^i))} \times 10^{-3} & \text{if } n_b^i \neq 0 \text{ and } p_t^i \neq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.11}$$

where the superscript i denotes the node i. The actions taken by every node in current transmission affect the next transmission's link quality through (3.9). The objective of every node is to maximize its average throughput per total consumed energy.

3.3.2 Extension of Reinforcement Learning

To solve the posed multi-node problem, we propose to extend the single-agent AC algorithm in Table 3.1 to learn *independently* policy in each agent. In this independent learning, each agent learns its transmission strategy by assuming that the agent itself is the only agent that influences the evolution of its state. Each agent uses only its local state information to do the decision, that is the agent does not take into account the state, action and reward involved in other agents' decision making processes. Although the proposed independent AC learning may not be optimal, it has several advantages. First, since no global information (the state and the decision of other agents) is used in the learning process, less control handshaking (each node doesn't need to exchange its state information and the decision employed) is required. Second, the extension of the AC algorithm has the same computational complexity as the single-agent scenario. Each agent requires only

to update the relative state value function, average reward and actor preference function of the state and action it experiences. (Table 3.1).

In the extension of single-agent RL, the actor-critic algorithm is applied directly to each node in multi-node scenarios. Before the transmission of a packet, every node uses the Gibbs softmax method to make the decision based on its current local state. At the end of the packet transmission, each node observes the ACK/NACK signal and receives the feedback information from the receiver containing the channel link quality in previous transmission. Also, each node observes the packet arrival and it records the reward (packet good throughput per unit energy). The agent uses this information to update its state, state value function and learned average reward as in Table 3.1. The above procedure is repeated throughout the transmission. We note that the resulting RL algorithm is inherently distributed, since every node applies the AC algorithm and makes its decision based on its local information. For comparison purposes, we simulate a policy where each node tries to transmit as high throughput as possible with a predefined SIR for one particular modulation. As before, we refer this policy as the *simple policy*.

3.3.3 Simulation Results: Multi-node scenario

In this section, we assess the performance of the *independent* AC algorithm in multi-node scenarios. We simulate the multi-node system with 3 nodes communicating with one receiver. The locations of the transmitting nodes are 340 meters, 460 meters and 570 meters away from the receiver. Node 1 is the nearest node to the receiver and node 3 is the farthest node from the receiver. Most of the simulation parameters are similar to Table 3.2 except those shown in the Table

3.3. The total number of states in each agent are 496 and total number of actions are 44.

Table 3.3: Simulation parameters

Channel Model	d=[320,460,570] m, $A_t^i = 0.097/(d^i)^4$
Buffer Cost	$f(n_b) = 0.05(n_b + 4)$ if $n_b \neq \max(n_b)$ $\max(n_b) = 15,\ f(\max(n_b)) = 3$
modulation level	m=1,2,3,4 (BPSK,QPSK,8PSK,16PSK),
Transmit power	$p_t = [0, 0.2, ..., 2]$ Watt
SIR range	$\Gamma = [0, 1, ..., 24]$ dB
Quantized Interference	$\eta = [-16, -15, ..., 14]$ dB

The AC algorithm is initialized with $\alpha = 0.05$, $\beta = 0.0005$ and $\epsilon = 0.01$. Figure 3.7 shows the average throughput learned by the AC algorithm and the *simple policy* for packet arrival rate $\mu = 2.0$. It is obvious that in both policies, the node nearer to the receiver will effectively have higher packet throughput per energy, since it requires less energy for achieving the same throughput. Figure 3.8 shows the throughput that is achieved for the AC learned policy and the *simple policy* for various packet arrival rates. From this figure, both policies achieve similar throughput when the packet arrival rate is low ($\mu \leq 1$). But when the packet arrival rate becomes large, the AC algorithm achieves higher throughput. In particular, the AC algorithm achieves 1.5 more throughput for node 1 when $\mu = 3.0$ and it achieves 6.3 and 7.1 times throughput for node 2 when $\mu = 3.0$ and node 3 when $\mu = 2.0$, respectively. We note that in the simple policy, the node 3 is not able to transmit anything for the packet arrival rate beyond $\mu = 2.0$. In this situation, the energy in node 3 is completely wasted without the ability to transmit anything. Using the AC algorithm, each node in the network is able to

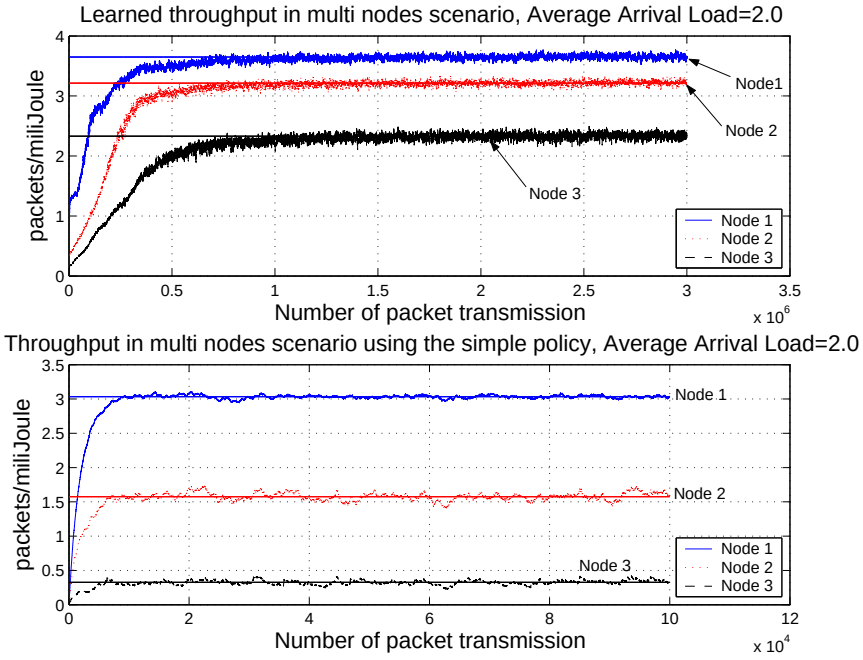


Figure 3.7: Learned and the simple policy throughput, packet arrival load $\mu = 2.0$

achieve higher throughput per unit energy for a broad set of packet arrival rates. This is due to the fact that the AC algorithm has the ability to explore policies other than the greedy policy adapting to the channel condition and packet arrival rate. The greedy policy will obviously result in a total breakdown of the network.

3.4 Discussions on the applicability of the RL algorithm to WSN

One of the major advantages of the RL algorithm is its capability to learn the environment with very little information. This property is very suitable for the WSN application, where each nodes may not have exact knowledge of its envi-

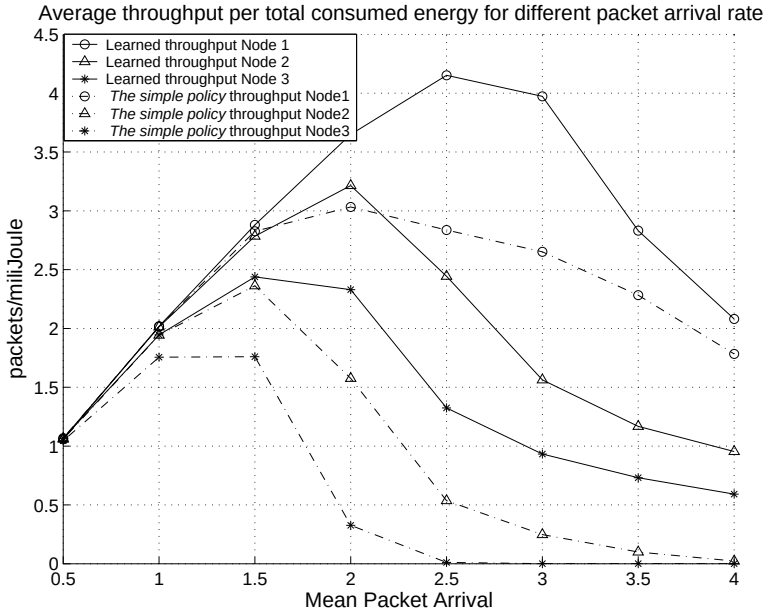


Figure 3.8: Average throughput corresponding to different packet arrival load $\mu = 2.0$

ronment. Moreover, the wireless environment in WSN tends to be varying due to many practical reasons. Having the learning capability, the algorithm decides the best transmission mode by adapting to the variation in the environment.

Since very little information is required in the learning process, the algorithm needs to explore all the possible actions/decisions and determines the best action according to the current environment/state. From (2.24)-(2.27), one observes that the learning algorithm uses the iterative averaging method to learn/estimate the value function h, average reward ρ and the preference metric. Intuitively, as the number of actions and states become larger, more data are required to refine the accuracy of the estimates. As the consequence, more time is required to experience and explore all possible decisions. In short, the convergence time of the algorithm is highly dependent on the number of actions and states in the learning system.

The larger the state and action space are, the more time will be required for the learning.

As discussed in the previous paragraph, the number of states and actions affects the time required for the learning process. In general, the number of states and actions in the learning system is closely related to the type of the applications. In particular, when the state is the sample of the physical quantity such as interference, a larger number of states results in a more accurate solution. On the other hand, the number of actions in the system reflects the degree of reconfiguration in the system. Therefore, the states and actions in the system, on one hand should be chosen carefully to accurately model the physical situation. On the other hand, the excessively large number of states and actions makes the learning algorithms slow. In our problem, the aggregate of the number of packets queued in the buffer and the interference level constitutes the state space and the action space consists of the power and modulation. The determination of these parameters may be dictated by the accuracy of the model and the cost for deploying the sensors. One obvious way to keep the number of states and actions small is to use small buffer length, small transmit power range and limited modulation levels. In this way, the resulting number of states and actions can be kept small enough to make the convergence fast, as will be demonstrated below.

To demonstrate that a smaller number of states and actions can actually have shorter learning stage, we perform another simulation where 7 nodes are simultaneously communicating with one receiver. The distance of nodes are (320, 460, 570, 660, 740, 810, 880) meters from the receiver. Two modulation levels, BPSK and QPSK are available and the transmit power levels are [0, 0.5, 1] Watt. Buffer length is equal to 4, the quantized interference has 8 levels. In this simulation,

each agent has 6 actions and 40 states. The resulting learned throughput per unit energy and the throughput obtained from *simple policy* are shown in Figure 3.9. Obviously, by reducing the number of states and actions, the learning time of the algorithm is also reduced. The learned policy outperforms the simple policy by (1.001, 1.1100, 1.1324, 1.7565, 2.6799, 3.2403, 3.0946) times of the achievable throughput for node 1 to 7, respectively.

Another important property of the learning algorithm is that the transient learning period only occurs once in the initial warming up stage of the sensor. Moreover, the algorithms efficiently capture the history information when learning the state value and average reward function, hence, it is not necessary for each node to record all the history of the transmission. After the learning stage, the algorithm is able to use the history efficiently to obtain good decisions.

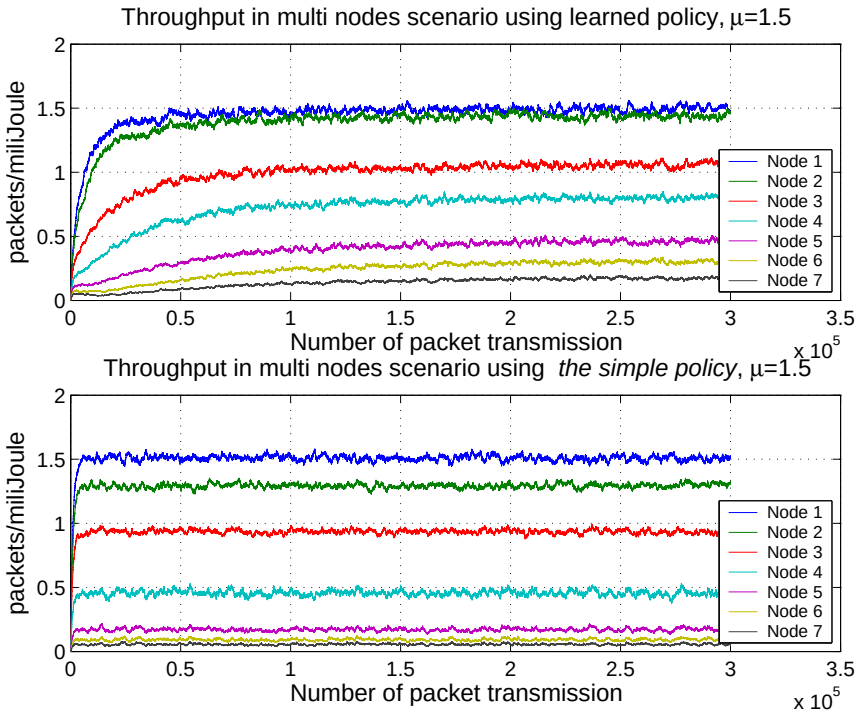


Figure 3.9: Learned and the simple policy throughput per unit energy for packet arrival load $\mu = 1.5$. The learned policy achieves (1.001, 1.1100, 1.1324, 1.7565, 2.6799, 3.2403, 3.0946) times throughput per energy compared to the *simple policy*, for the node 1 to 7 respectively.

Chapter 4

Robust Maximum Connectivity Energy-aware Routing

This chapter[1] demonstrates the topology-aware (connectivity-aware) energy-efficient routing for wireless network. In particular, we consider the energy-aware routing algorithm that explicitly takes into account the connectivity of the sensor networks. In typical sensor network deployments, some nodes may be more important than other nodes because the death of these nodes cause the network disintegration, which in turn causes early termination of information delivery. To overcome this problem, we propose a class of routing algorithms called *keep connect* algorithms, that explicitly consider the connectivity of the network while making the routing decision. The algorithm can be used along with the existing routing algorithms. When making the routing decision, the *keep connect* algorithm embeds the importance of the nodes in the routing cost. The importance of a node is quantified by the connectivity of the remaining network when that particular node dies. In

[1]Material in this chapter has been submitted to Transactions on Networking [79]

particular, we propose two criteria for describing the connectivity of the remaining network. First, the importance of a node is quantified by how severe the remaining network becomes disconnected/disintegrated when that particular node dies. Second, the connectivity of the remaining network is quantified by the Fiedler value of the graph when that particular node is removed. In other word, the importance of a node is characterized by the algebraic connectivity of the remaining graph. We prove some characteristics of our proposed routing algorithm. The proposed algorithm achieves on average $20\% \sim 50\%$ better network lifetime and total delivered packets when it is used on top of minimum total energy(MTE) routing algorithm. The proposed algorithm also achieves around 20% improvement compared to the flow augmentation (FA) algorithm. We also present the distributed implementation of our proposed algorithm. The MTE based distributed implementation achieves more than two times more total delivered packets before the network becomes disconnected, compared to flow augmentation based algorithm.

The organization of this chapter is as follows. First, we briefly give the motivation, system description and problem formulation. We review several important facts from spectral graph theory that will be used in the rest of the sections. Then, we explain our proposed *keep connect* algorithm and analyze some properties of the proposed routing metrics. Finally, we outline the distributed implementation and evaluate the effectiveness of our proposed algorithm through extensive simulations.

4.1 Motivation

There are many important characteristics of sensor networks. First, the sensor nodes are typically deployed in an area with high redundancy and each of the sensor nodes has very limited energy, therefore is prone to failure. In order to be

useful, the sensor nodes are required to collaboratively accomplish a special task. Second, the nodes in the sensor networks typically stay in their original deployed places for the entire of their lifetime. Hence, it is very important to always keep the remaining network connected, since the disintegrated clusters of nodes are useless for information gathering. Moreover, due to the immobility of the nodes, it may not be possible to reorganize the remaining nodes to create a new connected network.

Due to the above characteristics of the sensor networks, the design of the routing algorithm becomes very different from the typical ad-hoc networks in the following aspect. Instead of minimizing the hop count and delivery delay in the network, the routing algorithm in the sensor networks focuses more on extending the scarce battery lifetime of the nodes. There are many existing literatures focusing on the routing design of sensor networks. The minimum total energy routing (MTE) [99] algorithm selects the route that minimizes the total transmission energy along the route. The max-min residual energy (MMRE) algorithm [92] tries to maximize the residual battery among the minimum residual battery routes. This algorithm avoids the overuse of nodes along the minimum total energy route. In [99], the conditional min-max battery cost routing (CMMBCR) is proposed. This algorithm uses MTE algorithm when all nodes in some possible routes between source and destination have sufficient remaining battery capacity. However, when the batteries of nodes in the routes fall below certain threshold, the algorithm uses MMRE algorithm to select the route. Similarly, the max-min zP_{min} algorithm [63] controls the trade-off of MTE and MMRE using the variable z. When $z = 1$ the algorithm corresponds to MTE and when $z = \infty$, it corresponds to the MMRE. All the above algorithms share the same characteristic that is to trade-off between MTE

and MMRE algorithms. In [25,27], they propose a heuristic called flow augmentation (FA) algorithm that gradually makes transition from MTE to MMRE. They show that their algorithm performs better than CMMBCR and zP_{min} algorithms. However, all the existing algorithms do not explicitly consider the connectivity of the network in their routing decision.

Unlike most of the previous works which use the time until the first node in the network dies as the definition of network lifetime. In this chapter, we argue that the definition of network lifetime should be defined as the time until there is no route from any source to any destination instead of the time until the first node in the network dies. In other words, the network lifetime should be defined as the time until the network becomes disconnected. Using this definition as the network lifetime, the network connectivity becomes a very important criterion to be considered in designing the routing algorithm, especially when the information generation is not known *a priori*. Here, the information generation indicates the source and destination pairs during transmission in the network. To be precise, we employ the notion of algebraic connectivity of a graph in the spectral graph theory to quantify the importance of the node. In particular, we propose two criteria for describing the connectivity of the remaining network. First, the importance of a node is quantified by how severe the remaining network becomes disconnected/disintegrated when that particular node dies. We define the importance of a node as how many disconnected clusters will be resulted if that particular node becomes dead. The larger the number of disconnected clusters, the more important that node is. Second, the connectivity of the remaining network is quantified by the Fiedler value of the graph when that particular node is removed. One property of the Fiedler value is that the Fiedler of a graph is higher when the graph is more

connected. By considering the nodes' importance in the routing design, the node with higher importance will be retained in the network, therefore the connectivity of the remaining network is always maintained. By embedding the nodes' importance in the routing cost, we propose a class of algorithms called *keep connect* to solve the posed problem. Our proposed algorithm is very flexible and can be used along with other existing algorithms such as MTE and FA algorithm. Moreover, we show the effectiveness of our proposed method by extensive simulations.

4.2 System Model and Problem Formulation

In this section, we present the network model. Several definitions for the sensor network lifetime are also reviewed. We also give the problem formulation and the related work on power aware routing.

4.2.1 Network Model

A wireless sensor network is modelled as an undirected simple finite graph $G(V, E)$, where $V = \{v_i, \cdots, v_n\}$ is the set of nodes in the network, E is the set of all links/edges, n is the number of vertices in the graph, and $|E| = m$ is the number of edges in the graph. The undirected graph implies that all the links in the network are bidirectional, i.e. node v_i is able to reach node v_j implies the vice versa. The simple graph implies that there are no self-loops in each node and there are no multiple edges connecting two nodes. And the finite graph implies the cardinality of the nodes and edges are finite. The link (v_i,v_j) implies that node $v_j \in S_{v_i}$ can be directly reached by node v_i with a certain transmit power level in the predefined dynamic range, where S_{v_i} is the set of nodes that can be directly

reached by node v_i. We assume that every node has the initial battery energy of $\mathcal{E}_i$ for $\forall i \in N$. Every packet transmission consumes energy. The energy expenditure for transmission from node v_i to v_j is proportional to d_{ij}^{α}, where d_{ij} is the distance between node v_i and v_j, α ranges from 2 to 4. The path loss exponent, α depends on the transmission environment [85]. In this chapter, we assume $\alpha = 2$ for free space propagation. When the energy in one node is exhausted, we say that the node is dead for the remaining of the network lifetime.

4.2.2 Definitions of Network lifetime

Depending on the application in the wireless sensor network, there are many definitions of the network lifetime. In [25, 27, 99], the network lifetime is defined as the time until the first node/sensor in the network dies. In contrast, in [24], the network lifetime is defined as the time until all nodes die. A more general definition on the network lifetime is given in [19]. In [19], Blough and Santi defined the lifetime of sensor networks as the $\min\{t_1, t_2, t_3\}$, where t_1 is the time it takes for the cardinality of the largest connected components to drop below $c_1 \cdot n(t)$, where $n(t)$ is the number of alive nodes at time t, t_2 is the time it takes for $n(t)$ to drop below $c_2 \cdot n(0)$, and t_3 is the time it takes for the area covered to drop below $c_3 \cdot A$, where A is the area covered by the initial deployment of the sensors. In above definition, c_1, c_2, and c_3 are the pre-defined constants between zero and one. It is well-known that the network connectivity is very important to ensure the maximal delivery of the collected information in both the ad-hoc and sensor networks, therefore it should be taken into account in the network lifetime definition. In sensor network applications, the time until the first node/sensor dies may not serve as a good definition of the network lifetime. Since, the death of the first node/sensor

does not imply the breakdown of information delivery. Moreover, network disintegration typically causes severe impact in the information delivery. Therefore, we argue that it is crucial to consider the network connectivity in designing the energy-aware routing algorithm. In this chapter, we employ the time until the remaining network becomes disconnected as our network lifetime definition.

4.2.3 Problem Formulation

The problem of maximizing the minimum residual energy of nodes in the network has been studied in [25–27]. The time until the first node in the network dies can be found using the following linear program

$$\begin{aligned} &\text{Maximize} \quad T \\ &\text{s.t.} \qquad f(i,j)^{(c)} \geq 0,\ \forall i \in N,\ \forall j \in S_i,\ \forall c \in C, \\ &\qquad \textstyle\sum_{j \in S_i} e(i,j) \sum_{c \in C} f(i,j)^{(c)} \leq \mathcal{E}_i,\ \forall i \in N, \\ &\qquad \textstyle\sum_{j:i \in S_j} f(j,i)^{(c)} + TQ_i^{(c)} = \sum_{j \in S_i} f(i,j)^{(c)},\ \forall i \in N \setminus D^{(c)}, \forall c \in C, \end{aligned} \tag{4.1}$$

where $f(i,j)^{(c)}$ is the flow rate or the number of packets of commodity c information that is transmitted from node v_i to its neighbor node $v_j \in S_i$, S_i is the set of i's neighboring nodes, the commodity $c \in C$ indicates different source nodes $O^{(c)}$ and destination nodes $D^{(c)}$, where $O^{(c)}$ being the origin/source of commodity c and $D^{(c)}$ being the destination of commodity c, $e(i,j)$ is the energy required to guarantee successful transmission from node v_i to node v_j, and $Q_i^{(c)}$ denotes the information-generation rates at node v_i of commodity c. The second constraint implies the total energy used for packet transmission from node v_i should be less than the total energy in node v_i. The last constraint indicates the flow conservation at each nodes in the network, we note that $Q_i^{(c)}$ is equal to zero if no information is

generated in node v_i and the notation $j : i \in S_j$ denotes the summation of the flow from all nodes v_j whose neighbor is node v_i. The flow conservation simply states that the total flow coming into node v_i is equal to the total flow going out from node v_i plus any information rate generated from node v_i.

We note that the above formulation has two problems, the first problem is that the formulation requires the knowledge of all commodities when performing the optimization. This implies that the information on sources and destinations in all commodities during the whole duration of network lifetime is required to solve the linear program. Moreover, the information generation rates for all commodities should be specified in the linear programming formulation. The second problem is that the above formulation does not reflect the sequences of the commodities. Therefore, the above formulation is only suitable for off-line optimization.

The qualitative performance comparison of online and off-line algorithm for routing algorithm is given in [63]. They show that *there is no online algorithm for message routing that has a constant competitive ratio in terms of network lifetime*, where the competitive ratio is defined as the ratio of solution of online algorithm with respect to the optimal off-line solution. This implies that the online algorithm will be much worse compared to the off-line algorithm, however, the off-line algorithm is infeasible in practical situation. For this reason, we focus on designing robust online algorithm by taking into account the connectivity of the remaining network in making the routing decision. The robustness of our proposed scheme comes from the fact that when the information generation is not known a priori and the routing decision is made on the fly, employing the connectivity weight in the routing decision avoids the early termination of the information delivery. Therefore, more future traffic can still be delivered because of the higher degree of

connectivity of the remaining network.

4.2.4 Related work

In this subsection, we briefly review the related work. Depending on the objective of the routing algorithm, we have different routing algorithms for sensor network. Most of the existing algorithms can be categorized as either minimizing the sum of cost function along some routes or minimizing the maximum cost on some routes. Algorithms that minimizing the sum of cost function along routes between source and destination are minimum total energy (MTE) routing [99] and the flow augmentation (FA) algorithm [27]. Such routes can be computed using Dijkstra's shortest path algorithm, where the path cost is replaced by the amount of energy required to do transmission in that path. Let's consider a route $r = \{v_0, \cdots, v_d\}$, where v_0 is the source node and v_d is the destination node, and the energy consumed in transmitting a packet over the hop (v_i, v_j) as $e(v_i, v_j)$, then the total expended energy in that route is

$$P_{MTE}(r) = \sum_{i=0}^{d-1} e(v_i, v_{i+1}), \tag{4.2}$$

where $P_{MTE}(r)$ is the total transmit energy in route r. We note that the energy consumed in transmitting a packet over the hop (v_i, v_j) can be represented as $e(v_i, v_j) = K \cdot d(i,j)^{\alpha}$, where K is some transmission constant, $d(i,j)$ is the distance between node v_i and node v_j, α is the attenuation coefficient. The MTE routing selects the route among all routes that minimizes the total expended energy in the route, i.e.

$$r^*_{MTE} = \arg\min_{r \in R(v_0, v_d)} P_{MTE}(r), \tag{4.3}$$

where $R(v_0, v_d)$ is the set of routes from source node v_0 to destination node v_d. The flow augmentation (FA) algorithm is similar to the MTE routing algorithm,

except it weights the energy consumed over one hop by the normalized residual energy. In particular, the FA algorithm employs $e(v_i, v_j)^{x_1} \underline{\mathcal{E}}_{v_i}^{-x_2} \mathcal{E}_{v_i}^{x_3}$ as the energy metric over the hop (v_i, v_j), where $\underline{\mathcal{E}}_{v_i}$ is the residual energy of node v_i at current time, $\mathcal{E}_{v_i}$ is the initial energy of node v_i. The coefficients x_1, x_2, and x_3 control the effects of the transmit energy, residual energy, and the initial energy to the total routing cost. In the rest of this chapter, we will employ $x_1 = 1$, $x_2 = 5$, $x_3 = 5$. Hence, the total weighted energy expended in a route r is

$$P_{FA}(r) = \sum_{i=0}^{d-1} e(v_i, v_{i+1})^{x_1} \underline{\mathcal{E}}_{v_i}^{-x_2} \mathcal{E}_{v_i}^{x_3}. \tag{4.4}$$

And the algorithm selects the route that minimizes the total weighted energy, $r_{FA}^* = \arg\min_{r \in R(v_0, v_d)} P_{FA}(r)$.

Algorithms that minimize the maximum cost on some routes use algorithm to find the minimum of the maximum or the maximum of the minimum cost along the route. This max-min route can be computed using the modified Dijkstra algorithm shown in Table 4.1. The maximin residual energy (MMRE) [92], conditional maximin battery capacity (CMMBC) [99], and the lifetime maximization heuristic [73] can be implemented using the modified Dijkstra's algorithm in Table 4.1. The MMRE ensures that no node will be overused in the routing. Let's define the residual energy cost along route j as $R_j = \min_{i \in r_j} \mathcal{E}_{v_i}(t)$, where $\mathcal{E}_{v_i}(t)$ is the residual energy of node i at time t and the set of all routes between source s and destination d as $R(s, d)$, then the MMRE selects the route that maximizes R_j, that is

$$r_{MMRE}^* = \max\{R_j | j \in R(s, d)\}. \tag{4.5}$$

The maximin residual energy (MMRE) route can be implemented using the algorithm in Table 4.1 by replacing $c(i, j) = \underline{\mathcal{E}}_{v_i}$. Similarly, the lifetime maximization

heuristic in [73] can be implemented using the algorithm in Table 4.1 by using $c(i,j) = \frac{\mathcal{E}_{v_i}}{e(v_i,v_j)}$.

Table 4.1: Modified Dijkstra's algorithm for Max-min cost along the route

Input: a network $G = (V, E)$, source node s and destination node d.

Output: a path with the max-min cost along all routes from s to d.

1. Initialization: for each node v in G, set $parent[v] = 0$ and $Cost[v] = -\infty$.
2. Set: $Cost[s] = \infty$ and initialize priority queue F.
3. For each neighbor, w of s do

   $parent[w] = s$, $Cost[w] = c(s, w)$, and add w to F.
4. Repeat

   Extract node u with maximum Cost from F.

   for each neighbor w of u do

      if $Cost[w] = -\infty$,

        Set $parent[w] = u$, $Cost[w] = min(Cost[u], c(u, w))$,

        and add w to F.

      else if $w \in F$ and $Cost[w] < min(Cost[u], c(u, w))$

        Set $parent[w] = u$, $Cost[w] = min(Cost[u], c(u, w))$.

   Stop when $Cost[t] \neq -\infty$ and t not in F.

The conditional minimax battery cost routing (CMMBCR) [99] combines the objectives of minimizing the total expended energy in routes and maximizing the minimum residual energy in the route. The algorithm uses the MTE routing whenever there are routes from the source to the destination with the residual energy cost, $R_j > \gamma$, where γ is some threshold. Otherwise, the MMRE will be employed. When comparing the CMMBCR and the FA algorithm, we observe

that both algorithms make trade-off between the minimum total energy routing and the max-min residual energy routing. The CMMBCR explicitly switches the routing criterion based on whether $R_j > \gamma$. The FA algorithm will initially use the MTE route since the normalized residual energy is equal to one. However, as the residual energy in nodes along some routes in the network becomes depleted, the FA algorithm avoids using the less residual energy route.

4.3 Facts from Spectral Graph Theory

Before we describe our proposed solution, we briefly summarize some important facts from spectral graph theory [32, 40, 41]. These lemmas provide insights for understanding the proposed scheme and will be used to prove properties of the proposed scheme.

4.3.1 Eigenvalues of Laplacian Matrix

In this subsection, we briefly discuss the definition of Laplacian Matrix, its eigenvalues and the relationship between the eigenvalues and the connectivity of the associated graph. For simplicity and practicality in our problem, we are only interested in a simple graph (graph that does not contain loops and multiple edges between two nodes). Precisely, the simple graph does not have edge from node v_i to node v_i, moreover, there is only one edge connecting node v_i and node v_j for $v_i \neq v_j$. The following notations will be used throughout the chapter: $G = (V, E)$ is the graph with set of vertices V and set of edges E. We denote the number of vertices as $|V| = n$ and the number of edges as $|E| = m$. The Laplacian matrix associated with a graph is defined as follow.

Definition 1 (Laplacian matrix associated with a graph) *In a graph $G = (V, E)$, let d_v denote the degree of vertex $v \in V$. The Laplacian matrix associated with a graph,* $\mathbf{L}$ *is an n by n matrix defined as follows:*

$$L(u,v) = \begin{cases} d_v & \text{if } u = v, \\ -1 & \text{if } u \text{ and } v \text{ are adjacent or } (u,v) \in E, \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Equivalently, the Laplacian matrix $\mathbf{L}$ *can be expressed as:*

$$\mathbf{L} = T - A, \tag{4.7}$$

where T is an n by n diagonal matrix with the (v,v)-th entry having value d_v, and A is the n by n adjacent matrix.

Definition 2 (Normalized Laplacian matrix associated with a graph) *A normalized Laplacian matrix $\mathcal{L}$ associated with a graph, $G = (V, E)$ is defined as [32]:*

$$\mathcal{L}(u,v) = \begin{cases} 1 & \text{if } u = v \text{ and } d_v \neq 0 \ , \\ -\frac{1}{\sqrt{d_u d_v}} & \text{if } u \text{ and } v \text{ are adjacent or } (u,v) \in E, \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

Equivalently,

$$\mathcal{L} = T^{-1/2} L T^{-1/2}, \tag{4.9}$$

where T denotes the n by n diagonal matrix with the (v,v)-th entry having value d_v with the convention $T^{-1}(v,v) = 0$ for $d_v = 0$.

The eigenvalues of the Laplacian matrix, $\mathbf{L}$, $(\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1})$ are usually called the spectrum of the graph. The following lemma describes the relationship between the eigenvalue and the connectivity of a graph.

Lemma 1 *Let's denote* $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1}$ *as the eigenvalues of the Laplacian matrix* $\mathbf{L}$ *of a graph* G*. If* G *is connected, then* $\lambda_1 > 0$*. Moreover, if* $\lambda_i = 0$ *and* $\lambda_{i+1} \neq 0$*, then* G *has exactly* $i+1$ *disjoint connected components.*

Proof 1 *The above lemma follows from the fact that the union of two disjoint graphs has its spectrum as the union of the spectra of the original graphs. Generally, any disjoint connected components has its first eigenvalue as zero, then the union of* $i+1$ *disjoint connected components will have zero eigenvalue with* $i+1$ *multiplicity.*

Since the Laplacian matrix $\mathbf{L}$ has all zero row sums, the $\mathbf{L}$ has an eigenvalue 0 and the corresponding eigenvector $(1, \cdots, 1)^T$. Moreover, since $\mathbf{L}$ is real, symmetric and positive semi-definite, thus all the eigenvalues of $\mathbf{L}$ should be real and larger or equal to zero. Therefore, we conclude that the smallest eigenvalue of Laplacian matrix $\mathbf{L}$ is zero. Similarly, the smallest eigenvalue of the normalized Laplacian matrix $\mathcal{L}$, is also zero, and the corresponding eigenvector is $(\sqrt{d_1}, \cdots, \sqrt{d_n})^T$. The above lemma indicates that if G is strongly connected (there exists a simple path from any initial node i to the terminal node j, where $i \neq j$) then the Laplacian matrix $\mathbf{L}$ has simple eigenvalue 0 (the eigenvalue 0 has multiplicity of 1). Moreover, if the eigenvalue 0 of the Laplacian matrix $\mathbf{L}$ has multiplicity n, then there are n connected components. For the rest of this chapter, we will focus on the eigenvalues and eigenvectors of the Laplacian matrix, $\mathbf{L}$.

4.3.2 Fiedler value and vector

Let's denote the eigenvalues of the Laplacian matrix, $\mathbf{L}$ associated with $G = (V, E)$ as $\lambda_0(G), \cdots, \lambda_{n-1}(G)$ and the corresponding eigenvectors as $\nu_0(G), \cdots, \nu_{n-1}(G)$. Obviously, $\nu_0 = e = (1, \cdots, 1)^T$. Suppose that the graph $G = (V, E)$ is strongly

connected (the second smallest eigenvalue is larger than zero, $\lambda_1 > 0$). This second smallest eigenvalue can be represented as (Courant-Fisher Theorem)

$$\lambda_1 = \min_{x^T x=1, x^T \nu_0=0} x^T \mathbf{L} x \tag{4.10}$$

The second smallest eigenvalue of the Laplacian matrix is always referred to as the algebraic connectivity of the graph G [40]. It is also called as *Fiedler value* of a graph. The reason for calling the second smallest eigenvalue as the algebraic connectivity of a graph G comes from the following lemma.

Lemma 2 *If G_1 and G_2 are edge-disjoint graphs with the same vertices, then $\lambda_1(G_1) + \lambda_1(G_2) \leq \lambda_1(G_1 \cup G_2)$.*

Proof 2 *Since $\mathbf{L}(G_1 \cup G_2) = \mathbf{L}(G_1) + \mathbf{L}(G_2)$. Then*

$$\begin{aligned} \lambda_1(G_1 \cup G_2) &= \min_{x^T x=1, x^T e=0} \left(x^T \mathbf{L}(G_1) x + x^T \mathbf{L}(G_2) x \right) \\ &\geq \min_{x^T x=1, x^T e=0} \left(x^T \mathbf{L}(G_1) x \right) + \min_{x^T x=1, x^T e=0} \left(x^T \mathbf{L}(G_2) x \right) \\ &\geq \lambda_1(G_1) + \lambda_1(G_2). \end{aligned} \tag{4.11}$$

Lemma 3 *The Fiedler value λ_1 is non-decreasing for graphs with the same set of vertices, i.e. $\lambda_1(G_1) \leq \lambda_1(G)$, where $G_1 = (V, E_1)$, $G = (V, E)$, and $E_1 \subseteq E$. The $\lambda_1(G)$ denotes the second smallest eigenvalue of the normalized Laplacian matrix associated with graph G.*

Proof 3 *Direct result from Lemma 2.*

We observe that G and G_1 have the same number of vertices. Since G_1 has less number of edges compared to G and $E_1 \subseteq E$, this implies that G_1 is less connected compared to G. From Lemma 3, we have the Fiedler value corresponding to G_1

is smaller than G, $\lambda_1(G_1) \leq \lambda_1(G)$. It is in this sense that the Fiedler value represents the degree of connectivity in a graph. Finally, the relation of Fiedler value for graph obtained from removing a vertex and all its adjacent edges is given by the following lemma.

Lemma 4 *Let G_1 be a graph obtained from removing 1 vertex from G and all the adjacent edges. Then $\lambda_1(G_1) \geq \lambda_1(G) - 1$.*

Proof 4 *Suppose graph G has n vertices and graph G_1 is obtained from G by removing one of its vertex (let denote this vertex as v_n). Let also denote a graph $\bar{G}$ which is obtained from G by completing the edges connecting vertex v_n. Then*

$$\mathbf{L}(\bar{G}) = \begin{bmatrix} \mathbf{L}(G_1) + I & -e \\ -e^T & n-1 \end{bmatrix}. \tag{4.12}$$

Let ν be the eigenvector of $\mathbf{L}(\bar{G})$ corresponding to $\lambda_1(\bar{G})$. Since

$$\mathbf{L}(\bar{G}) \begin{bmatrix} \nu \\ 0 \end{bmatrix} = (\lambda_1(\bar{G_1}) + 1) \begin{bmatrix} \nu \\ 0 \end{bmatrix}, \tag{4.13}$$

then, $\lambda_1(\bar{G_1}) + 1$ is the eigenvalue of $\mathbf{L}(\bar{G})$. Since $\lambda_1(\bar{G_1}) + 1$ is larger than zero, then $\lambda_1(\bar{G}) \leq \lambda_1(\bar{G_1}) + 1$. Form Lemma 3, we have $\lambda_1(G) \leq \lambda_1(\bar{G}) \leq \lambda_1(\bar{G_1}) + 1$.

Finally, the following two lemmas give some upper and lower bounds for the Fiedler value.

Lemma 5 *Let $G = (V, E)$, d_{v_i} be the degree of node v_i, then*

$$\lambda_1(G) \leq \left[\frac{n}{n-1}\right] \min_{v_i} d_{v_i}. \tag{4.14}$$

Lemma 6 *Let $\varepsilon(G)$ is the edge connectivity of the graph G (the minimal number of edges whose removal would result in losing connectivity of the graph G). Then, we have*

$$\lambda_1(G) \geq 2\varepsilon(G)\left[1 - cos\left(\frac{\pi}{n}\right)\right], \tag{4.15}$$

where n is the number of vertices $|V| = n$.

4.4 Proposed Solutions

In this section, we use the properties described in previous section to develop heuristics that drive the routing algorithm to maximize the network lifetime with the connectivity consideration. The proposed solution can be employed along with different existing routing algorithms. The key idea of our proposed method is that when there is no *a priori* knowledge about the information generation, the best we can do is to design the routing algorithm that does it best to keep the remaining nodes as a connected graph. This objective is obvious, since the disintegrated network causes severe performance degradation in terms of the amount of delivered information as discussed in Section 4.2. In particular, we propose a class of algorithms that makes use of the graph connectivity condition in performing the routing decision. We proposed 5 algorithms namely, maximin remaining connectivity routing (MMRCR), maximin the remaining energy with connectivity condition (MMREKC), minimum hop while considering the connectivity condition (MHKC), minimum total energy while considering the connectivity condition (MTEKC), and the flow augmentation with the connectivity condition (FAKC). The first two algorithms use the modified Dijkstra's algorithm in Table 4.1, while the latter three use the shortest path algorithms with modified path cost.

Before describing the detail algorithm, let's first consider how to quantify the

connectivity of the remaining graph as the routing cost. We propose to use 2 different criteria that reflect the connectivity condition on the remaining of the graph. The first criterion is based on checking the eigenvalue 0 multiplicity of the (normalized) Laplacian matrix of the graph. Each node is weighted according to how that node affects the connectivity of the remaining graph when that particular node dies; that is how many connected components will result as that particular node dies. The weight of the node can be thought as the importance of the node in the sense that the most important node is the node that results in a large number of disconnected components as it dies. The procedure for computing the node importance, *keep connect* algorithm is listed as in Table 4.2. We note that the

Table 4.2: Keep Connect Algorithm 1

Let $G(V, E)$ be the original graph. Let's define graph $G_{-v_i}(\{V - v_i\}, E_{-v_i})$ as the graph obtained after omitting node v_i.

1. Initialization:

   Set nodes' weights as zeros $W(v_i) = 0, \forall v_i \in V$

2. For each node v_i:

   2a. Form the Laplacian matrix $\mathbf{L}_{-v_i}$ of graph $G_{-v_i}(\{V - v_i\}, E_{-v_i})$ as (4.6).

   2b. Find the multiplicity of eigenvalue 0 of matrix $\mathbf{L}_{-v_i}$. Let's denote this value as l.

   2c. Set the weight of node as: $W(v_i) = l$.

   End for

keep connect algorithm can also be extended further by looking ahead to the case

when there are more than one node die. The KC algorithm shown in Table 4.2 only considers the connectivity of the remaining graph after one node dies. Since it is very common that the death of one node causes the other nodes to become more important, we extend the KC algorithm to consider the case when more than one node die. The *keep connect 2* as shown in Table 4.3 can be used to capture the connectivity of the remaining graph after two nodes die. Ideally, this lookahead can be applied to the case when more than 2 nodes die. However, due to the complexity of the algorithm, we only consider up to 2 nodes lookahead.

Table 4.3: Keep Connect Algorithm 2

Let $G(V, E)$ be the original graph. Let's define graph $G_{-v_i}(\{V - v_i\}, E_{-v_i})$ as the graph obtained after omitting node v_i. Moreover, let $G_{(-v_i,-v_j)}$ be the graph obtained by omitting node v_i and then node v_j, where $v_j \in \{V - v_i\}$.

1. For each node v_i:
   a. Form the $G_{(-v_i,-v_j)}$, for $v_j \in \{V - v_i\}$ and find the corresponding Laplacian matrix as (4.6). Let l_{-v_j} be the multiplicity of eigenvalue 0 of the associated Laplacian matrix corresponding to $G_{(-v_i,-v_j)}$.
   b. Set the weight of node v_i as $W(v_i) = \prod_j^{\{V-v_i\}} l_{-v_j}$.

   End for

The second criteria representing the connectivity of the remaining graph is

Table 4.4: Keep Connect using Fiedler value

Let $G(V, E)$ be the original graph. Let's define graph $G_{-v_i}(\{V - v_i\}, E_{-v_i})$ as the graph obtained after removing node v_i.

1. Initialization:

   Set nodes' weights as zeros $W(v_i) = 0, \forall v_i \in V$

2. For each node i:

   2a. Form the Laplacian matrix $\mathbf{L}(G_{-v_i})$ of graph $G_{-v_i}(\{V - v_i\}, E_{-v_i})$ as (4.6).

   2b. Find the Fiedler value and let denote the Fiedler value as $\lambda_1(G_{-v_i})$

   2c. Set the weight of node as: $W(v_i) = 1/\lambda_1(G_{-v_i})$.

   End for

based on the Fiedler value. Recall from Section 4.3.2, the Fiedler value represents the connectivity of a graph in the sense that the larger the Fiedler value is the more connected the graph is. The degree of connectivity of the remaining graph can be quantified by the Fiedler value of the graph resulted by removing that particular node and all the edges connected to that node from the original graph. We design the weight of each node by setting the weight of node v_i as $1/\lambda_1(G_{-v_i})$. In this way, the node that caused severe reduction in the remaining network connectivity will be avoided when doing the routing decision. The routing algorithm that uses the keep connect algorithm tries to avoid the nodes that are more important to keep the remaining network connected.

The modification of the existing routing algorithm by incorporating the con-

nectivity condition is straightforward. Using the node importance, the edge cost for edge connecting node v_i and node v_j is weighted by $W(v_i)^y$. In the following subsections, we describe the maximin remaining connectivity (MMRC) routing , maximin the remaining energy while keeping connectivity (MMREKC), minimum hop while keeping connectivity (MHKC), minimum total energy while keeping connectivity (MTEKC), and flow augmentation while keeping connectivity (FAKC).

4.4.1 Maximin remaining connectivity (MMRC) routing

The basic idea of MMRC algorithm is to maximize the minimum weight of nodes along the routes between the source and the destination. In this way, for a fixed source and destination, the node with lowest weight will be avoided. This MMRE algorithm can be calculated using the modified Dijkstra's algorithm in 4.1 with the cost between node u and w as

$$c(u, w) = \frac{1}{W(u)}. \tag{4.16}$$

The weight of each node can be found using the algorithm keep connect algorithm 1 (Table 4.2), keep connect algorithm 2 (Table 4.3), and keep connect algorithm using fiedler value (Table 4.4).

4.4.2 Maximin the remaining energy while keeping connectivity (MMREKC(y)) routing

Although, the MMRC algorithm avoids to use the most important node (node that cause huge degradation in connectivity of the remaining graph), it does not consider the remaining energy in the node as the criteria to select the routing decision. To jointly select the routing decision based on the remaining energy

in nodes and the connectivity criterion, we propose the MMREKC(y) routing algorithm. The MMREKC(y) algorithm avoids the nodes with low residual energy while considering the connectivity of the remaining graph. The MMREKC(y) can be computed using the modified Dijkstra's algorithm in Table 4.1 with the edge cost function defined as

$$c(u,w) = \frac{\mathcal{E}_u}{W(u)^y} \tag{4.17}$$

4.4.3 Minimum hop while keeping connectivity (MHKC) routing

The minimum hop routing is usually used in ad-hoc or wire-line network to minimize the delay in the packet delivery. The keep connect algorithm can also be used along with the minimum hop routing. The MHKC algorithm can be used to achieve the purpose of minimizing the number of hops between the source and destination while the connectivity of the remaining graph relatively high. By setting the edge cost function as $c(u,w) = \frac{1}{W(u)}$ and using any shortest path algorithm, the MHKC can be calculated. Suppose the route between source and destination is described as $r = \{v_0, \cdots, v_d\}$, where v_0 is the source and v_d is the destination. Then, the MHKC(y) selects route that minimizes $r^* = \arg\min_r \sum_{i=0}^{d-1} \frac{1}{W(v_i)}$.

4.4.4 Minimum total energy while keeping connectivity (MTEKC) routing

The MTEKC(y) algorithm is obtained by embedding the connectivity weight to the original minimum total energy (MTE) [99] algorithm. The modified algorithm uses the edge cost between node u and node w as $c(u,w) = e(u,w) \cdot W(u)^y$. The complete algorithm for MTEKC is shown in Table 4.5. Similar to previous

algorithms, the weight in the MTEKC(y) algorithm can be calculated either using keep connect algorithm 1 (Table 4.2), keep connect algorithm 2 (Table 4.3), or keep connect algorithm using Fiedler value (Table 4.4). We note that in the MTEKC algorithm, the parameter y determines how important the connectivity weight should influence the weighted minimum total energy. When y is very large, we expect the performance of the MTEKC(y) will be near to the performance of MHKC algorithm, which may be far from the energy efficient route. We will determine this parameter in the following section.

Table 4.5: MTEKC(y)

1. For any source-destination pairs, find the minimum total energy path with edge cost as: $e(v_i, v_j) \cdot W(v_i)^y$ for $v_i \in V$, $v_j \in S_{v_i}$, where $e(v_i, v_j)$ is the transmission energy from node v_i to v_j when v_j is the neighbor of v_i, S_{v_i}; $W(v_i)$ is the weight of node v_i.
2. If node dies, recompute the remaining nodes' weight using *Keep Connect* algorithm. Recompute the minimum total energy path.

4.4.5 Flow Augmentation while keeping connectivity (FAKC(y)) routing

Our last algorithm FAKC(y) is obtained by including the connectivity weight to the flow augmentation algorithm [25, 27]. In particular, the edge cost is modified as $c(u, w) = e(u, w)^{x_1} \underline{\mathcal{E}}_u^{-x_2} \mathcal{E}_u^{x_3} \cdot W(u)^y$. The detailed algorithm for FAKC(y) is listed in Table 4.6. The FA-KC algorithm searches not only for the route with the

maximum residual energy, but it also tries to avoid the nodes with high importance.

Table 4.6: FAKC(x_1, x_2, x_3,y)

In every update time:
1. For any source-destination pairs, find the minimum total energy path with edge cost as: $e(v_i, v_j)^{x_1} \cdot \underline{\mathcal{E}}_{v_i}^{-x_2} \cdot \mathcal{E}_{v_i}^{x_3} \cdot W(v_i)^y$ for $v_i \in V$, $v_j \in S_{v_i}$, where $e(v_i, v_j)$ is the transmission energy from node v_i to v_j when v_j is the neighbor of v_i, S_{v_i}. $\mathcal{E}_{v_i}$ is initial energy of node v_i, $\underline{\mathcal{E}}_{v_i}$ is the remaining energy of node v_i and $W(v_i)$ is the weight of node v_i.
2. If node dies, recompute the remaining nodes' weight using *Keep Connect* algorithm. Recompute the minimum total energy path using FA algorithm.

4.4.6 Illustrative Example

Now let us give an illustrative example of how the *keep connect* (KC) algorithm can really improve the performance of the existing routing algorithm. Consider the network shown in Figure 4.1(a), suppose that there are 10 packets from node 1 to node 3 and another 10 packets from node 3 to node 7, respectively in that order. Also, assume that initially all of the nodes have 10 unit energy and one packet transmission requires one unit energy. Here, we also assume that packet reception energy is negligible. Figure 4.1 shows the routing results by using the FA algorithm. After the first 10 packet transmission, the remaining energy of each nodes are shown in Figure 4.1 (b). The FA algorithm will do the load balancing

between route $1-2-3$ and route $1-4-3$. However, because of this transmission order, only 5 packet transmission in the second 10 packet transmission will be delivered. Hence, the total throughput is 15 packets. Similar to the FA algorithm, the MTE algorithm will choose one of the route with equal probability. In the worst case the route $1-4-3$ is chosen and the resulting throughput is 10 packets and in the best case the route $1-2-3$ is chosen first, and the resulting throughput is 20 packets. In contrast, Figure 4.2 illustrates the routing results using the joint MTE-KC algorithm. The weights of each node (calculated using KC1 algorithm in Table 4.2) are tabulated in the right hand side of the figure. In the first 10 packet transmission, the proposed algorithm use mainly route $1-2-3$, since node 4 is more important and the death of node 4 causes the remaining network becomes disconnected. Hence, the resulting throughput is 20 packets. This solution is the same with the optimal throughput obtained by solving (4.1). In summary, when the traffic generation is not known *a priori*, it is better to keep the remaining nodes in the network connected.

4.5 Properties of the proposed solution

In this section, we give some properties and quantitative analyses on the class of our proposed algorithms. We will make use of bounds of the Fiedler value known in the spectral graph theory [32, 40, 41, 74]. The lower and upper bound of Fiedler value are stated in Lemma 6 and Lemma 5. In the following, we give some simple lemmas on the lower bound and upper bound of the metric used in keep connect algorithm.

Lemma 7 (Lower bound of MTEKC metric using Fiedler value) *For each route, the MTEKC(y) employing the Fiedler value metric has the following prop-*

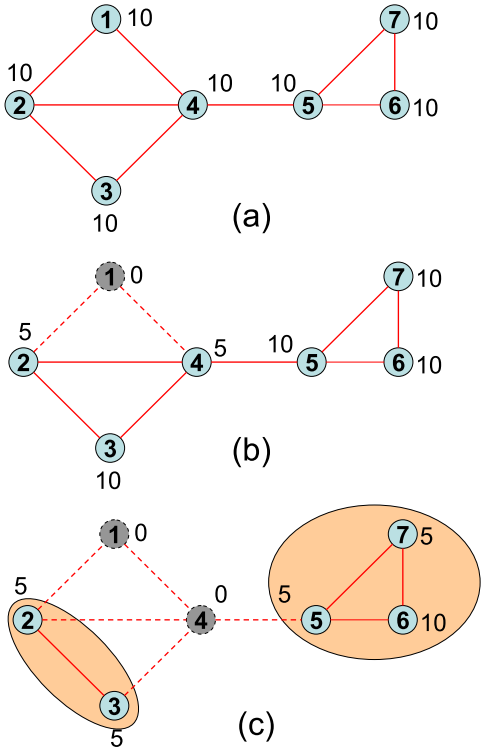


Figure 4.1: Illustration 1

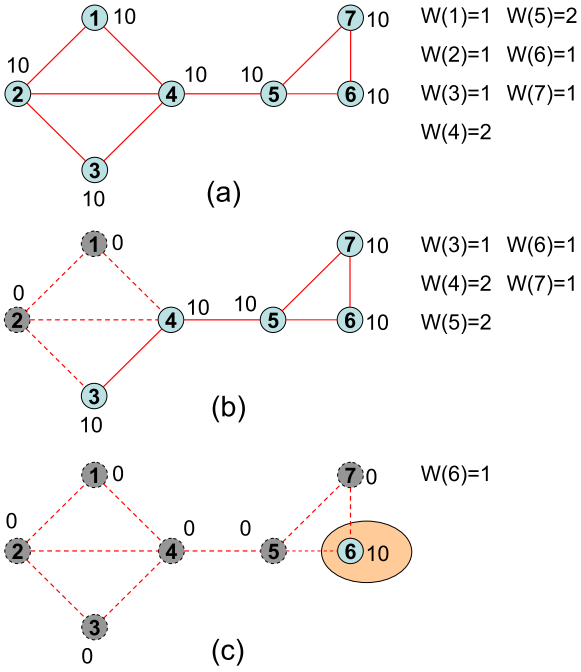


Figure 4.2: Illustration 2

erty

$$\begin{aligned}\sum_{i=0}^{d-1} e(v_i, v_{i+1}) \cdot W(v_i)^y &\geq \left(\sum_{i=0}^{d-1} e(v_i, v_{i+1})\right)\left(\frac{n-2}{n-1}\frac{1}{\min_i d_{v_i}}\right)^y \\ &\geq \left(\sum_{i=0}^{d-1} e(v_i, v_{i+1})\right)\left(\frac{(n-2)n}{2(n-1)m}\right)^y, \end{aligned} \tag{4.18}$$

where d_{v_i} is the degree of node v_i in the graph, n is the number of vertices in the graph, and m is the number of edges in the graph.

Proof 5 *To prove these inequalities, we require the upper bound of the Fiedler value as follow (stated in Lemma 5). Consider $G = (V, E)$ and let d_{v_i} be the degree of node v_i. Then, $\lambda_1(G) \leq \frac{n}{n-1}\min_i d_{v_i}$. Now, consider the graph G_{-v_i} obtained from graph G by removing node v_i and all edges connecting to node v_i. Obviously, $\lambda_1(G) \leq \frac{n}{n-1}\min_i d_{v_i}$ implies that $\lambda_1(G_{-v_i}) \leq \frac{n-1}{n-2}\min_i d_{v_i}$, since the minimum degree of graph G_{-v_i} is smaller or equal to minimum degree of graph G, $\min_i d_{v_i}$. Now, using keep connect algorithm with Fiedler value, we have*

$$\begin{aligned}\sum_{i=0}^{d-1} e(v_i, v_{i+1}) \cdot W(v_i)^y &= \sum_{i=0}^{d-1} \frac{e(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^y} \\ &\geq \sum_{i=0}^{d-1} e(v_i, v_{i+1})\left(\frac{n-2}{(n-1)\min_i d_{v_i}}\right)^y. \end{aligned} \tag{4.19}$$

Since $n \cdot \min_i d_{v_i} \leq \sum_i d_{v_i} = 2m$, we have $\left(\frac{1}{\min_i d_{v_i}}\right)^y \geq \left(\frac{n}{2m}\right)^y$. Then, we obtain the second inequality

$$\sum_{i=0}^{d-1} e(v_i, v_{i+1})\left(\frac{n-2}{(n-1)\min_i d_{v_i}}\right)^y \geq \sum_{i=0}^{d-1} e(v_i, v_{i+1})\left(\frac{(n-2)n}{2(n-1)m}\right)^y.$$

Lemma 8 (Upper bound of MTEKC metric using Fiedler value) *For each route, the MTEKC(y) employing the Fiedler value metric has the following property*

$$\sum_{i=0}^{d-1} e(v_i, v_{i+1}) \cdot W(v_i)^y \leq \left(\sum_{i=0}^{d-1} e(v_i, v_{i+1})\right)\left(\frac{1}{2\big(\varepsilon(G)-1\big)\big(1-cos\big(\frac{\pi}{n-1}\big)\big)}\right)^y, \tag{4.20}$$

where $\varepsilon(G)$ is the edge-cut or edge connectivity of the graph. The edge-cut/edge connectivity is defined as the minimal number of edges whose removal would result in disconnected graph. n is the number of vertices in the graph.

Proof 6 *Similar to Lemma 7, we use the lower bound of Fiedler value in Lemma 6. Consider $G = (V, E)$ and let $\varepsilon(G)$ be the edge-cut of the graph. Then, $\lambda_1(G) \geq 2\varepsilon(G)[1 - cos(\pi/n)]$. Now, consider the graph G_{-v_i} obtained from graph G by removing node v_i and all edges connecting to node v_i. From upper bound of Fiedler value, we have $\lambda_1(G_{-v_i}) \geq 2\varepsilon(G_{-v_i})[1 - cos(\pi/(n-1))]$. Moreover, we have $\varepsilon(G_{-v_i}) \geq \varepsilon(G) - 1$. Hence, $\lambda_1(G_{-v_i}) \geq 2[\varepsilon(G) - 1][1 - cos(\pi/(n-1))]$. Follow the similar proof in Lemma 7, we can obtain*

$$\begin{aligned}\sum_{i=0}^{d-1} e(v_i, v_{i+1}) \cdot W(v_i)^y &= \sum_{i=0}^{d-1} \frac{e(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^y} \\ &\leq \sum_{i=0}^{d-1} e(v_i, v_{i+1}) \left(\frac{1}{2[\varepsilon(G) - 1][1 - cos(\frac{\pi}{n-1})]} \right)^y.\end{aligned}$$

Now we are ready to develop the upper bound on the energy consumed in the MTEKC algorithm with Fiedler value. Let us first define the following notations, before we state the theorem on the upper bound on the energy consumed using MTEKC algorithm with Fiedler value. Suppose r^* is the minimum total energy route connecting any fixed source node v_0 and destination node v_d, then MTE route satisfies $r^* = \arg\min_{r \in R(v_0, v_d)} \sum_{i=1}^{d(r)-1} e(v_i, v_{i+1})$, where $R(v_s, v_d)$ is the set of all routes connecting source node v_s and destination node v_d, $d(r)$ is the number of hops between in the route. We note that $d(r)$ is obviously a function of the route r. Let's denote $r^\dagger$ as the MTEKC route obtained using Fiedler value, then this route satisfies $r^\dagger = \arg\min_{r \in R(v_0, v_d)} \sum_{i=1}^{d(r)-1} \frac{e(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^y}$. The following theorem gives the upper bound on the consumed energy

Theorem 5 *The energy consumed in the MTEKC using Fiedler value satisfies the following upper bound*

$$\sum_{i=1}^{d(r^{\dagger})-1} e(v_i, v_{i+1}) \leq \sum_{i=1}^{d(r^{*})-1} e(v_i, v_{i+1}) \cdot \left[\frac{(n-1)m}{n(n-1)(\varepsilon(G)-1)(1-cos(\pi/(n-1)))}\right]^{y} \tag{4.21}$$

Proof 7 *From the definitions, we have the following inequalities*

$$\sum_{i=1}^{d(r^{*})-1} e(v_i, v_{i+1}) \leq \sum_{i=1}^{d(r^{\dagger})-1} e(v_i, v_{i+1}) \tag{4.22}$$

$$\sum_{i=1}^{d(r^{\dagger})-1} \frac{e(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^{y}} \leq \sum_{i=1}^{d(r^{*})-1} \frac{e(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^{y}} \tag{4.23}$$

From inequality (4.23), Lemma 7 and 8, we have

$$\left(\sum_{i=1}^{d(r^{\dagger})-1} e(v_i, v_{i+1})\right) \cdot \left(\frac{(n-2)n}{2(n-1)m}\right)^{y} \leq \sum_{i=1}^{d(r^{\dagger})-1} \frac{e(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^{y}} \tag{4.24}$$

$$\sum_{i=1}^{d(r^{*})-1} \frac{e(v_i, v_{i+1})}{\lambda_1(G_{-v_i})^{y}} \leq \left(\sum_{i=1}^{d(r^{*})-1} e(v_i, v_{i+1})\right) \cdot \left(\frac{1}{2[\varepsilon(G)-1][1-cos(\frac{\pi}{n-1})]}\right)^{y} \tag{4.25}$$

Combining the above two inequalities and (4.23), we have

$$\sum_{i=1}^{d(r^{\dagger})-1} e(v_i, v_{i+1}) \leq \left(\sum_{i=1}^{d(r^{*})-1} e(v_i, v_{i+1})\right) \cdot \left(\frac{m(n-1)}{n(n-2)(\varepsilon(G)-1)(1-cos(\frac{\pi}{n-1}))}\right)^{y} \tag{4.26}$$

The above theorem gives the upper bound on the energy consumed in MTEKC route compared to the minimum total energy for routing the packet. The following theorem gives the bound on the ratio of energy consumed by MTEKC using Fiedler value when the number of nodes becomes large.

Theorem 6 *Suppose that the network generated satisfies* $m = a_1 \cdot n$ *and* $\varepsilon(G) - 1 = a_2$*, where* a_1 *and* a_2 *are some constants. Then the upper bound on the ratio of energy consumed can be presented as follow*

$$\frac{\sum_{i=1}^{d(r^{\dagger})-1} e(v_i, v_{i+1})}{\sum_{i=1}^{d(r^{*})-1} e(v_i, v_{i+1})} = O((n^2)^{y}) \tag{4.27}$$

Proof 8 *Using the assumption of the theorem, we have*

$$\begin{aligned}\frac{\sum_{i=1}^{d(r^\dagger)-1} e(v_i, v_{i+1})}{\sum_{i=1}^{d(r^*)-1} e(v_i, v_{i+1})} &\leq \left(\frac{a_1 n(n-1)}{a_2 n(n-2)(1-\cos(\frac{\pi}{n-1}))}\right)^y \\ \frac{\sum_{i=1}^{d(r^\dagger)-1} e(v_i, v_{i+1})}{\sum_{i=1}^{d(r^*)-1} e(v_i, v_{i+1})} &\leq C\left(\frac{n(n-1)(1+\cos(\frac{\pi}{n-1}))}{n(n-2)\sin^2(\frac{\pi}{n-1})}\right)^y,\end{aligned} \tag{4.28}$$

where $C = (a_1/a_2)^y$. *As* $n \to \infty$, *we have*

$$\frac{\sum_{i=1}^{d(r^\dagger)-1} e(v_i, v_{i+1})}{\sum_{i=1}^{d(r^*)-1} e(v_i, v_{i+1})} \leq C\left(\frac{(n-1)^2}{\pi^2}\right)^y, \tag{4.29}$$

where we have used small angle approximation in sinusoidal function, $sin(\theta) \approx \theta$, *as* $\theta \ll 1$. *Hence, we have* $\frac{\sum_{i=1}^{d(r^\dagger)-1} e(v_i, v_{i+1})}{\sum_{i=1}^{d(r^*)-1} e(v_i, v_{i+1})} = O((n^2)^y)$.

From this theorem, we see that the ratio of energy will increase in quadratic function of number of nodes, compared to the minimum energy used to route a packet when the network is very large. This ratio of energy can be easily controlled by the parameter y, for instance if $y = 1/2$, then the ratio of consumed energy increases in a linear function of number of nodes in the network. In the extreme case, setting $y = O(1/n)$ makes the proposed algorithm approaching to MTE as $n \to \infty$.

4.6 Distributed Implementation and Learning Algorithm

In this section, we outline the distributed implementation of the proposed maximum connectivity routing algorithm. The method is based on the distributed reinforcement learning routing algorithm [68]. The resulting algorithm can be characterized as a version of distributed Bellman-Ford algorithm that performs

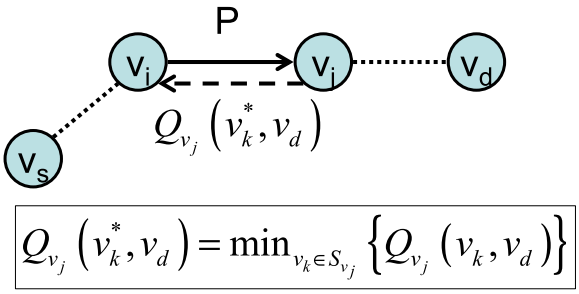


Figure 4.3: Exchange and Update Q-value

its path relaxation step asynchronously and online with the edge cost defined as weighted energy required to transmit packet in that hop. The routing decision is learned by all nodes in the network. Each node maintains the best packet delivery cost to all the destinations. In particular, each node v_i maintains a table of Q-values $Q_{v_i}(v_j, v_d)$, for $v_j \in S_{v_i}$, where v_j is in the set of node v_i neighbors, S_{v_i}, and node v_d is the destination. The $Q_{v_i}(v_j, v_d)$ has the interpretation of node v_i's best estimated cost that a packet would incur to reach its destination node v_d from node v_i when the packet is sent via node v_i's neighbor node v_j.

The value in the Q-table will be exchanged between node v_i and v_j, whenever there is a packet is sent from node v_i and v_j, and vice versa. The exchange mechanism is illustrated as in Figure 4.3. Whenever node v_i transmits a packet P to node v_j, node v_j feedbacks $Q_{v_j}(v_k^*, v_d) = \min_{v_k \in S_{v_j}} Q_{v_j}(v_k, v_d)$ to node v_i as shown in the figure. We note that $Q_{v_j}(v_k^*, v_d)$ represents the best estimated cost that a packet would be sent to the destination node v_d from node v_j. Node v_j reports the best cost that a packet will incur if it is sent from v_j to destination v_d. The node i uses this value to update its own Q-value as follow

$$Q_{v_i}(v_j, v_d) = (1 - \delta)Q_{v_i}(v_j, v_d) + \delta[Q_{v_j}(v_k^*, v_d) + c(v_i, v_j)], \tag{4.30}$$

where $c(v_i, v_j)$ is the cost for sending packet from node v_i to node v_j, and $\delta \in [0, 1]$

is the learning rate for the algorithm.

Since in this chapter, the routing algorithms are driven for the purpose of maximizing the network lifetime, then the cost of sending a packet between node v_i and node v_j is related to the energy consumption for sending the packet. In particular, the cost of sending packet for MTE and MTEKC routing algorithms are

$$\text{[MTE]:} \qquad c(v_i, v_j) = e(v_i, v_j), \tag{4.31}$$

$$\text{[MTEKC(y)]:} \qquad c(v_i, v_j) = e(v_i, v_j) \cdot W(v_i)^y, \tag{4.32}$$

For MTE, $Q_{v_i}(v_j, v_d)$ represents the total energy consumption used to delivery a packet from node v_i to node v_d via node v_i's neighbor node v_j. In contrast, $Q_{v_i}(v_j, v_d)$ in MTEKC represents the total energy consumption in delivering a packet from node v_i to v_d via v_j, while considering the connectivity of the remaining network. The procedure for implementing the MTEKC is summarized in Table 4.7. We note that when $\delta = 1.0$, the algorithm becomes the distributed Bellman-Ford iterations [17].

Intuitively, the FA and FAKC algorithms can also be implemented similar to the Table 4.7 with the cost for sending the packet as follows

$$\text{[MTE]:} \qquad c(v_i, v_j) = e(v_i, v_j)^{x_1} \cdot \underline{\mathcal{E}}_{v_i}^{-x_2} \cdot \mathcal{E}_{v_i}^{x_3}, \tag{4.33}$$

$$\text{[MTEKC(y)]:} \qquad c(v_i, v_j) = e(v_i, v_j)^{x_1} \cdot \underline{\mathcal{E}}_{v_i}^{-x_2} \cdot \mathcal{E}_{v_i}^{x_3} \cdot W(v_i)^y, \tag{4.34}$$

However, careful observation on the cost for sending the packet, we notice that the cost for sending the packet is continuously changing whenever a packet is transmitted. Even though the network topology is not changing, the edge cost of sending a packet depends on the residual energy of the node. In typical distributed algorithm, each node requires some time to learn the correct Q-table when the edge

Table 4.7: Distributed Asynchronous MTEKC(y)

1. Network Initialization: Flooding the adjacent neighbors information to all nodes in the network
2. Each node computes its own connectivity weight
3. Whenever node v_i sends a packet to node v_j:
   a. Node v_j informs v_i its minimum cost transmitting a packet to the destination v_d, $Q_{v_j}(v_k^*, v_d)$ (Figure 4.3)
   b. Node v_i updates its metric as:
   $$Q_{v_i}(v_j, v_d) = (1-\delta)Q_{v_i}(v_j, v_d) + \delta[Q_{v_j}(v_k^*, v_d) + c(v_i, v_j)]$$
   c. Node v_i leaves other estimates unchanged.
4. When a node dies, informs nodes in the network through flooding and repeat step 2

cost is fixed. This problem also appears in distributed implementation of routing algorithms that make use of nodes' residual energy in determining the routing path. This causes the trade-offs between the accuracy of residual energy estimation and the stability of the learning algorithm. Although, the FA algorithm seems to be amenable to distributed implementation, but due to the rapid changing in the residual energy of nodes in the network, the problem of whether the steady state is achievable is still in question.

4.7 Simulation Results

We simulate the routing algorithms in a discrete-event simulator. The simulator initially deploys nodes in the network. Every events are timestamped and queued. The most current event will be dequeued and some task is performed according

to the type of the event. There are three types of events; the packet arrival event, the reporting event, and the sending event. The packet arrival event injects packets to the network from some source node to some destination node. We assume the packet arrival follows the Poisson arrival process with mean μ. The report event occurs periodically to retrieve the simulation parameters such as the average delivery delay per packet, average hops per packet transmission, energy consumed per one delivered packet, and the number of packets delivered in this report interval. All events that are neither the packet arrival events nor reporting events are the sending events. In the sending events, a packet is sent to its next hop. The next hop is determined based on the routing algorithm used. Whenever a packet arrives at a node, it is queued in the node's buffer and will be sent in the next transmission time. Whenever a packet reaches its destination, the number of delivered packets is incremented and the event associated with that packet is freed. In the following, we first investigate the centralized solution, where the system knows all the edges' costs and the residual energy of all nodes. Hence, the optimal routing path can be evaluated before the packet is being sent in a centralized manner. In the second path, we consider the performance of several routing algorithms when there is only limited information exchange. Finally, a fully distributed solution outlined in Section 4.6 is evaluated.

4.7.1 Centralized solution

We generate 10 connected random network in the area of 100m by 100m with 36 nodes. The networks have average edges per node from 4.4 to 7.5. The transmission energy between two nodes is quantified as $6.4 \times 10^{-4} d^2$, where d is the distance between the two nodes. All nodes in the network initially have 500 unit energy.

The traffic follows the Poisson distribution with mean μ (this is also referred to as the traffic/network load). The source and destination are selected uniformly from the alive nodes. We use the network lifetime (time before the remaining network becomes disconnected), packet delivery time, average transmit energy per packet, and total delivered packet before the remaining network becomes disconnected as our performance metrics. We first compare the performance of the maximin residual connectivity (MMRC) routing, the minimum hop (MH) routing, minimum hop while keeping connectivity (MHKC) routing, and the minimum transmit energy (MTE) routing. All of these algorithms recalculate the routing path at the beginning of the simulation and whenever a node dies. In MMRC and MHKC, the nodes' weights are calculated using the Fiedler value. We normalized the performance metrics to the performance metrics achieved by the MTE algorithm. Figure 4.4 shows the normalized metrics achieved by MMRC, MH, and MHKC algorithms. All the algorithms have lower network lifetime compared to the MTE algorithm. This is obvious since all the algorithms do not use the transmit energy to guide the routing decision, hence the route is not energy efficient at all. This can be observed from the figure that all the algorithms consume 20% to 90% more energy per packet. However, the MH and MHKC take only 50% less time to deliver one packet, this is clear because MH based algorithms select route that has the minimum hops from source to destination, hence the resulting packet delivery delay is the smallest. In fact, the MMRC algorithm also requires less time to delivery a packet. In term of network lifetime and total delivered packets, the MH performs a little bit better compared to the MMRC; in contrast, the MHKC achieves about 5% more in terms of network lifetime and total delivered packets compared to MMRC. Compared to the MH, MHKC achieves on average 4.72% and 4.67% more network

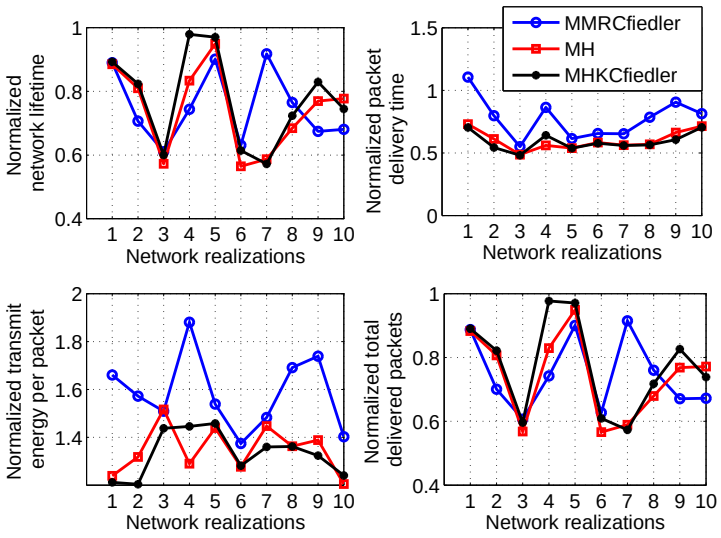


Figure 4.4: Comparison of normalized metric for different algorithms w.r.t. MTE algorithm, when the packet arrival follows the Poisson process with mean $\mu = 1.0$.

lifetime and total delivered packets, respectively, while both MH and MHKC have comparable packet delivery time and energy consumed per packet transmission.

Next, we evaluate the performance of employing the connectivity condition in the MTE algorithm. Figure 4.5 shows the performance metrics when MTE algorithm is employed when keeping the remaining network connected. We plot the normalized network lifetime, normalized packet delivery time, normalized transmit energy per packet, and normalized total delivery packets. The normalization is done with respect to the performances of MTE algorithm. From the figure, the $MTEKC1(y), y = 1$ deviates the least from the MTE algorithm in terms of all the performance metrics, this is obvious since most of the deployed network are more than k-connected, where $k = 1$. The $MTEKC1(y), y = 1$ algorithm on average achieves around 5.67% and 5.71% more network lifetime and total delivered packet, while it is only 1% less energy efficient compared to MTE. The

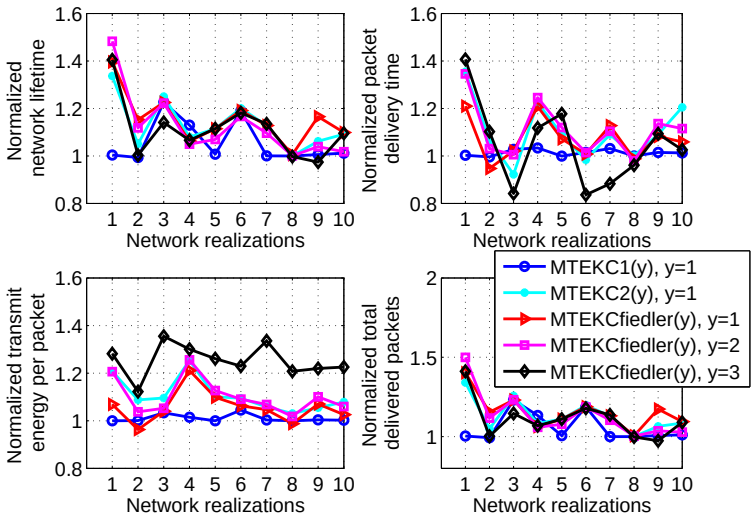


Figure 4.5: Comparison of normalized metric for different MTEKC algorithms w.r.t. MTE algorithm, when the packet arrival follows the Poisson process with mean $\mu = 1.0$.

$MTEKC2(y), y = 1$, $MTEKCfiedler(y), y = 1$, $MTEKCfiedler(y), y = 2$, and $MTEKCfiedler(y), y = 3$ achieve on average 13.13% and 13.18%, 15.39% and 15.52%, 12.62% and 13.42%, 11.12% and 11.15% more network lifetime and total delivered packet, respectively, while they are around 10.61%, 5.74%, 10.15% and 25.39% less energy efficient, respectively. For the proposed algorithm using the Fiedler value, the value of y should only be chosen around 1 to 2. The reason for this is that when y is too large, the algorithm performs more like MHKC algorithm, which is less energy efficient as seen in Figure 4.4. This can be observed from $MTEKCfiedler(y), y = 3$, this algorithm achieves slightly larger network lifetime, but consumes much more energy per packet compared to the case when lower y is employed.

Now, we are about to compare algorithms that use the residual energy in nodes

to determine the routing decision. We note that all the algorithms that use residual energy to guide the routing decision such as MMRE, MMREKC, FA algorithms recompute the routing path every 20 simulation time and whenever a node dies. Figure 4.6 shows the performance of FA algorithm with connectivity condition, MTE, and the maximin residual energy (MMRE). In the figure, the energy efficiency of FA algorithms with/without connectivity condition are in between the energy efficiency of MMRE and MTE. This is obvious, since the MMRE does not take into account the transmit energy in their routing metric, therefore the MMRE results in worst energy efficiency. In terms of network lifetime and total delivery packets, the FA algorithm is better than MTE and MMRE algorithms. In fact, adding the connectivity condition does not improve the FA algorithm much in the low packet arrival rate. This can be verified by exploring the impact of employing network connectivity weight to these routing algorithms when the packet arrival rate is high. Specifically, we study 3 of the 10 networks that we generated. These networks are shown in Figure 4.7. Figure 4.8 to 4.10 show the network lifetime and total delivered packets achieved by the MTE and FA algorithm with/without connectivity condition. In Figure 4.8, the $MTEKC2(1)$ achieves 33.67% to 42.84% more network lifetime from the low load to high load, and it achieves longer than 34% more successful delivered packets compared to the MTE algorithm. The MTEKC using Fielder value with $y = 1$ achieves $40 \sim 46\%$ longer network lifetime and it achieves more than 41% more packet delivery. Finally, the MTEKC using Fiedler value with $y = 2$ achieves $46 \sim 51\%$ longer lifetime and more than 50% more packet delivery. Compared to the FA algorithm, the $FAKC2(1)$ achieves around $0 \sim 15\%$ longer lifetime and $0 \sim 20\%$ more successfully delivered packets. Similar to the $FAKC2(1)$, the FAKC with Fiedler value for $y = 1, 2$ outperforms

FA algorithm by more than 10% in terms of network lifetime and successfully delivered packets. Observing Figure 4.9, we found that in the high load ($\mu = 3.0$) MTEKC with Fiedler value for $y = 1, 2$ outperforms the FA algorithm. The performance gain by including the connectivity weight in calculating the routing decision for Figure 4.8 to 4.10 is summarized in Table 4.8 and 4.9. The main reason of this performance improvement is that by explicitly considering the connectivity of the remaining network, some more important nodes may not be overused in the high load. The more important nodes will not be used unless they are the nodes along the only path to the destination. The probability of the overused is higher in the high network load compared to the low network load. Hence, the performance improvement is more pronounced in the high network load. Finally, we remark that the FA based algorithms compared in this section require to recompute the routing path in every 20 simulation time. This implies a frequent information exchange of the residual energy of all nodes happens in the FA algorithm. Next, we consider the performance of FA algorithm when less frequent information exchange of residual energy of all nodes is performed.

4.7.2 Limited information exchange

We note that the comparison in the previous section is not completely fair, in terms of information exchange. The routing algorithms that employ the residual energy in each nodes to compute the routing path require the updated information regarding to the residual energy of all nodes every time before the routing path is recomputed. In contrast, the MTE and MTEKC only recalculate the routing path whenever there is a node dies. As we know, the information dissemination in the sensor network can be done through flooding, and for the sake of energy efficiency,

Table 4.8: Network lifetime and total delivery packets improvement for network 1, 2, and 3.

Load	Network	$\frac{MTEKC2}{MTE}$	$\frac{MTEKCfiedler(1)}{MTE}$	$\frac{MTEKCfiedler(2)}{MTE}$
1	1	$(33.67\%, 34.10\%)$	$(39.59\%, 40.27\%)$	$(48.30\%, 49.93\%)$
	2	$(4.98\%, 4.92\%)$	$(15.21\%, 15.41\%)$	$(11.81\%, 11.69\%)$
	3	$(6.21\%, 6.46\%)$	$(16.56\%, 17.32\%)$	$(3.9\%, 3.63\%)$
2	1	$(34.03\%, 34.12\%)$	$(39.83\%, 40.53\%)$	$(46.85\%, 47.78\%)$
	2	$(4.86\%, 4.45\%)$	$(15.33\%, 15.74\%)$	$(11.32\%, 11.78\%)$
	3	$(5.92\%, 6.13\%)$	$(17.11\%, 17.58\%)$	$(6.42\%, 6.09\%)$
3	1	$(42.84\%, 30.14\%)$	$(46.40\%, 41\%)$	$(50.59\%, 42.21\%)$
	2	$(5.01\%, 2.51\%)$	$(10.16\%, 17.69\%)$	$(11.58\%, 12.46\%)$
	3	$(0.1\%, 0.1\%)$	$(21.60\%, 38.55\%)$	$(-19.83\%, 59.86\%)$

Table 4.9: Network lifetime and total delivery packets improvement for network 1, 2, and 3.

Load	Network	$\frac{FAKC2}{FA}$	$\frac{FAKCfiedler(1)}{FA}$	$\frac{FAKCfiedler(2)}{FA}$
1	1	$(2.04\%, 1.96\%)$	$(1.14\%, 1.07\%)$	$(1.73\%, 1.76\%)$
	2	$(0.45\%, 0.5\%)$	$(-0.36\%, -0.29\%)$	$(0.93\%, 0.58\%)$
	3	$(0.03\%, 0.02\%)$	$(0.03\%, 0.02\%)$	$(0.03\%, 0.02\%)$
2	1	$(6.69\%, 6.84\%)$	$(8.35\%, 8.54\%)$	$(8.72\%, 8.98\%)$
	2	$(9.31\%, 9.64\%)$	$(9.14\%, 9.44\%)$	$(9.78\%, 10.16\%)$
	3	$(15.42\%, 15.20\%)$	$(14.83\%, 14.69\%)$	$(15.43\%, 15.32\%)$
3	1	$(6.13\%, 9.53\%)$	$(6.27\%, 8.10\%)$	$(7.18\%, 11.10\%)$
	2	$(8.89\%, 15.67\%)$	$(10.16\%, 19.01\%)$	$(7.82\%, 16.51\%)$
	3	$(10.55\%, 20.35\%)$	$(8.09\%, 18.31\%)$	$(10.09\%, 20.42\%)$

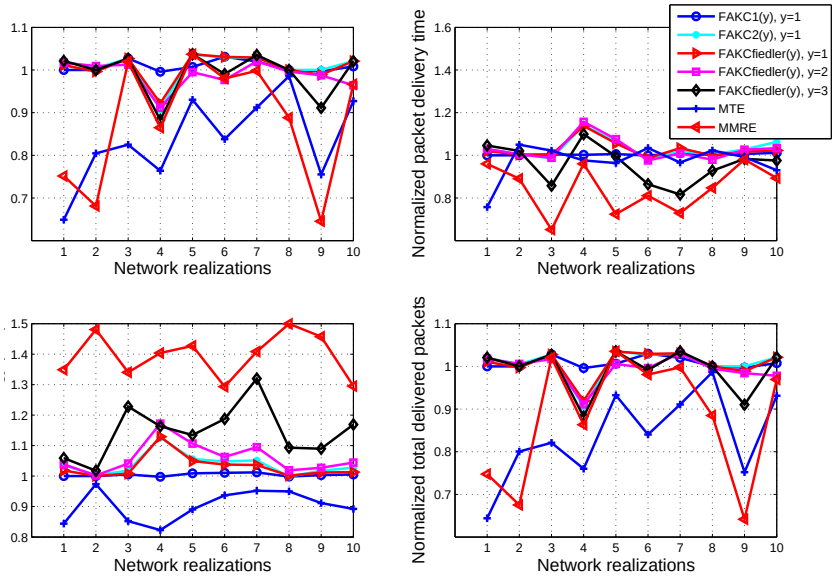
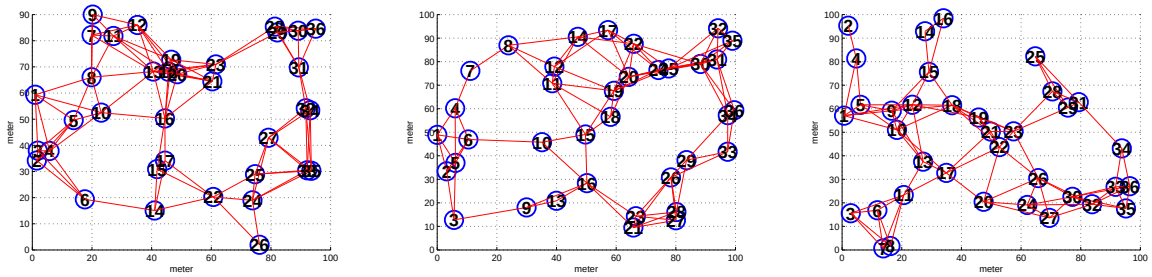


Figure 4.6: Comparison of normalized metric for different algorithms w.r.t. FA algorithm, when the packet arrival follows the Poisson process with mean $\mu = 1.0$.



(a) Network 1 (b) Network 2 (c) Network 3

Figure 4.7: Random networks

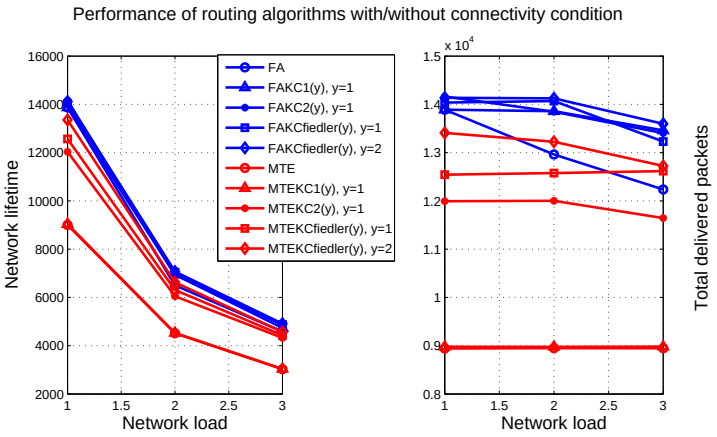


Figure 4.8: Comparison of normalized metric for different packet arrival rate in network 1.

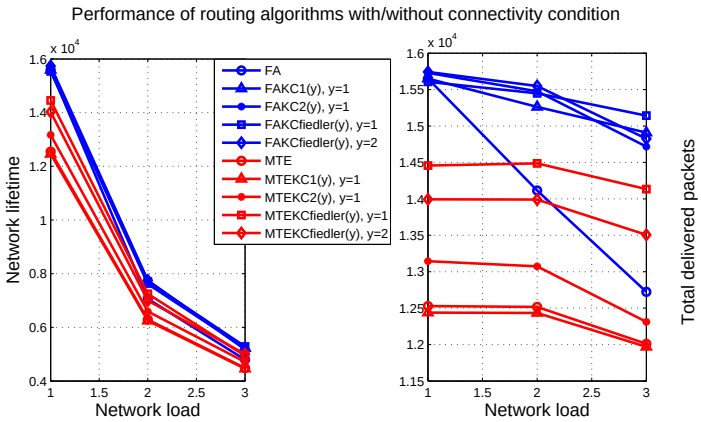


Figure 4.9: Comparison of normalized metric for different packet arrival rate in network 2.

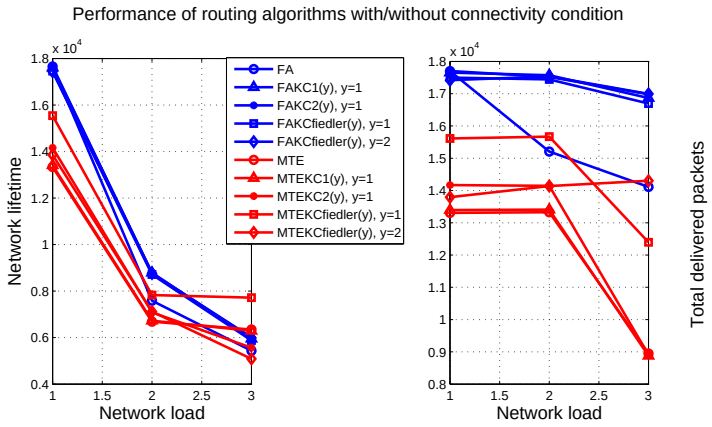


Figure 4.10: Comparison of normalized metric for different packet arrival rate in network 3.

the information dissemination should not be done very often. For fair comparison, we investigate the residual energy based algorithm with limited information exchange. In particular, the algorithms only do the information dissemination once in the beginning of the network setup and whenever a node dies. The FA algorithm recomputes the routing path at the beginning of the simulation and whenever a node dies. We refer to this algorithm as FA fair. Figure 4.11 to 4.13 show the normalized performance metrics for different network realizations, when the packet arrival rate is 1.0 to 3.0, respectively. From the figures, we observe that the MTE algorithm performs comparably with the FA fair algorithm in the low packet arrival case. However, the FA fair algorithm performs a little bit better than the MTE algorithm at high packet arrival rate. The $FAfairKC1(1)$ achieves on average $16.71\% \sim 22.24\%$ and $19.95\% \sim 22.51\%$ longer network lifetime and more total delivered packets, respectively, compared to the FA fair algorithm. The $FAfairKC2(1)$ achieves on average $19.43\% \sim 22.01\%$ and $20.50\% \sim 22.25\%$ longer network lifetime and more total delivered packets, respectively, compared to

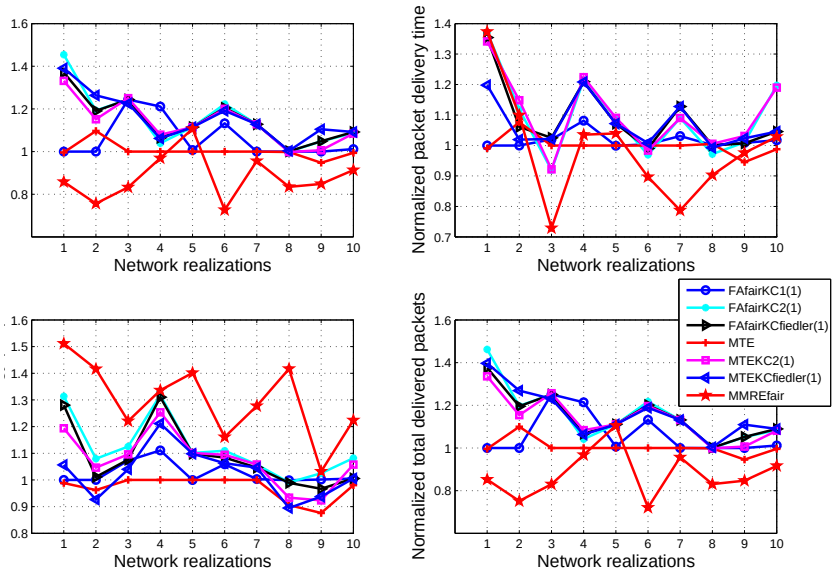


Figure 4.11: Comparison of normalized metrics for different routing algorithms w.r.t. FA fair in different network realization when packet arrival=1.0.

the FA fair algorithm. Similarly, the $FAfairKCfiedler(1)$ achieves $18\% \sim 21\%$ better lifetime and $19.15\% \sim 22\%$ more total delivered packets, compared to FA fair. From the simulation results, we can summarize that when the information exchange about the residual energy in sensor nodes are limited, explicitly considering the network connectivity improves the network lifetime and total delivered packets by around 22%. We note that the information exchange required to calculate the connectivity weight is minimal in the sense the information exchange only occurs whenever a node dies. This can be done through flooding.

4.7.3 Distributed implementation

In this subsection, we evaluate the distributed implementation outlined in Section 4.6. We run the distributed algorithm in Network 1 of Figure 4.7 with nodes

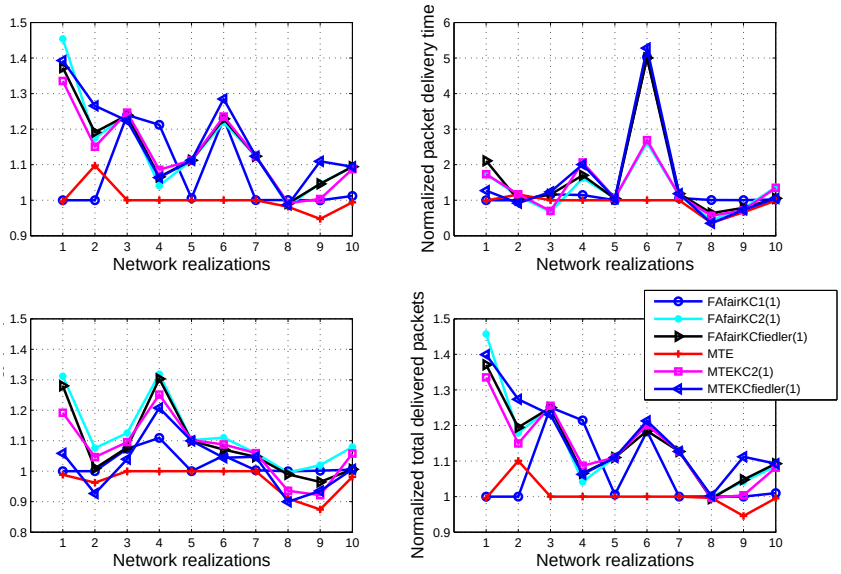


Figure 4.12: Comparison of normalized metrics for different routing algorithms w.r.t. FA fair in different network realization when packet arrival=2.0.

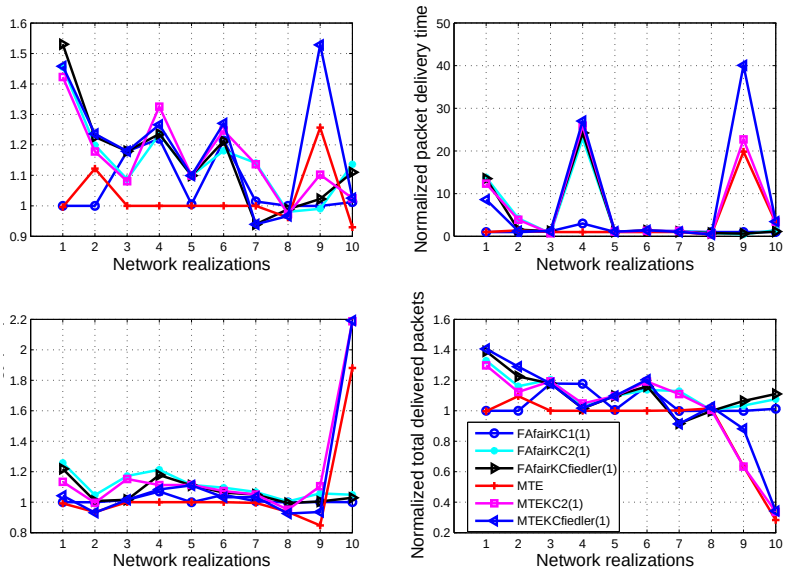


Figure 4.13: Comparison of normalized metrics for different routing algorithms w.r.t. FA fair in different network realization when packet arrival=3.0.

initial energy $\mathcal{E}_i = 1000, \forall i \in N$. Figure 4.14 and 4.15 show the average energy per packet, the number of delivered packets, average number of hops per packet, and average delivery time per packet of the distributed implementation for different routing algorithms. From Figure 4.14, the FA based algorithms have very low throughput compared to the MTE based algorithms. This is mainly due to the fact that the learned optimal routing path based on FA algorithm keeps changing as a packet reaches to its next hop. This changing of routing path is caused by the changing edge cost in the FA algorithm. Due to this changing edge cost, the optimal routing path based on FA algorithm is always changing and it never reaches a fixed stable route. Because of this reason, the learning algorithm may never reach the stable state. Hence, the FA based algorithms have low total delivered packets before the network becomes disconnected. In fact, the total delivered packet using FA and FAKCfiedler are 8863 and 8872, respectively. In contrast, the MTE and MTEKCfiedler deliver 18521 and 20128 packets. The MTE based algorithms can achieve higher throughput because the stable optimal routing path is achieved and the optimal routing path changes only whenever a node dies. This can be observed from the number of delivered packets, the jump in the graph corresponds to the relearning in the $Q - value$. The relearning happens only when a node dies. After the death of a node, the MTE and MTEKCfiedler algorithms have to relearn the new topology and refine the $Q - value$. From this figures, we summarize that the MTEKCfiedler consistently performs better than MTE. And both MTE and MTEKCfiedler achieve more than 2 times total delivered packets compared to the FA based algorithm. In Figure 4.15, we observe that the FA based algorithms always have increasing average delivery time. This increasing average delivery time shows the instability of the learned routing path.

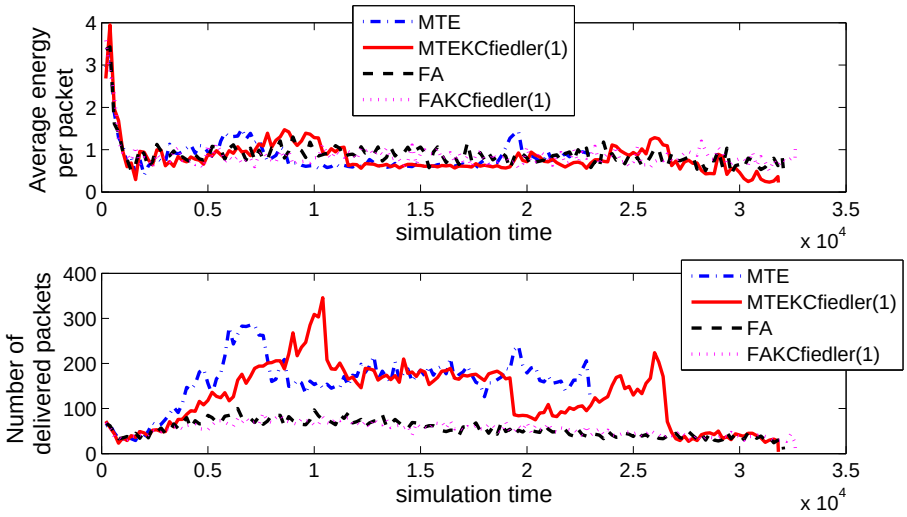


Figure 4.14: Comparison of average energy per packet and number of delivered packets for distributed implementation of different routing algorithms when the packet arrival=1.0

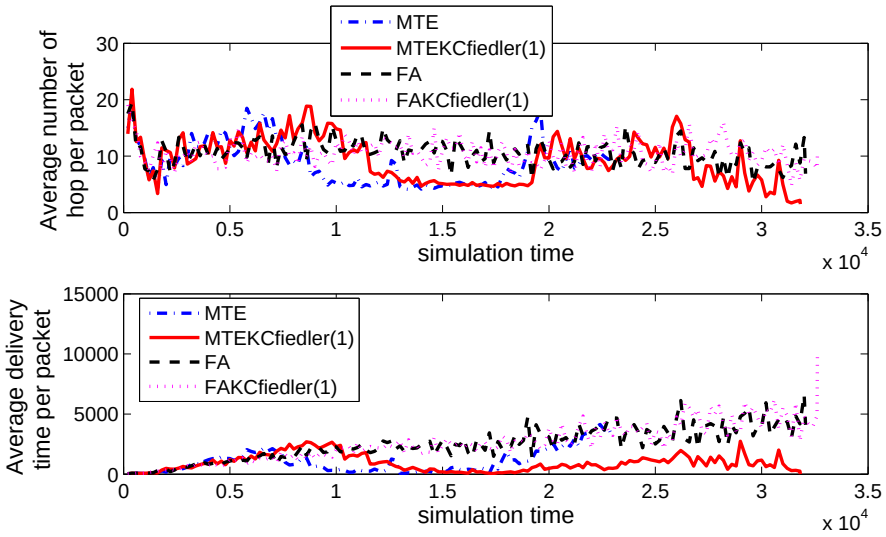


Figure 4.15: Comparison of average number of hops per packet and average delivery time per packet for distributed implementation of different routing algorithms when the packet arrival=1.0

Chapter 5

Cooperative Routing for Lifetime Maximization

This chapter studies the impact of cooperative routing for maximizing the network lifetime in sensor network applications. We We assume nodes in the network are equipped with a single omnidirectional antenna and they coordinate their transmission to achieve transmit diversity. We propose a joint cooperative transmission and energy aware routing algorithm to prolong the network lifetime. In contrast to the previous works, our approach uses the maximum lifetime power allocation, instead of minimum power allocation. Using the maximum lifetime power allocation, the cooperative nodes allocate their transmit power according to the channel condition and the residual energy in the nodes. Our maximum lifetime cooperative routing scheme combines the maximum lifetime power allocation and the energy aware routing to maximize the network lifetime. We study the performance of the cooperative routing in terms of network lifetime (defined as the time until the first node dies). We demonstrate that our proposed solution achieves $1 \sim 3.5$ and $1 \sim 2$ times longer network lifetime and total delivered packet compared to non-

cooperative routing, when it is used with MTE and FA algorithms, respectively. Furthermore, the maximum lifetime power allocation achieves $1 \sim 2$ times longer lifetime, compared to maximum power allocation in MTE and FA routing schemes. We also provide distributed implementation of the proposed algorithm.

This chapter is organized as follows. First, we give the motivation of our work, describe the system model, reviews the existing energy aware routing algorithm. Then, we formulate the link cost for the cooperative routing. We derive and explain the proposed maximum lifetime power allocation in cooperative routing. After the derivation, we present the joint cooperative transmission and energy aware routing scheme, furthermore, we outline the distributed implementation. Finally, the performance of our proposed algorithm is evaluated using extensive simulations.

5.1 Motivation

Advances in low power integrated circuit devices and communications technologies have enable the deployment of low-cost, low power sensors that can be integrated to form sensor networks. These networks have vast important applications, i.e.: from battlefield surveillance system to modern highway and industry monitoring system; from the emergency rescue system to early forest fire detection and the very sophisticated earthquake early detection system. Having the broad range of applications, the sensor network is becoming an integral part of human lives. Moreover, it has been identified as one of the most important technologies nowadays.

The deployment of the low cost and energy limited sensors implies that the energy efficient communication protocol is imperative to extend the lifetime of the network. The problem of energy efficient protocol can be approached from different

communication layers; from physical layer, data-link layer, MAC layer, network layer to the application layer. Moreover, the cross layer approach has been shown to be an effective energy saving method in the energy constrained communication [45, 80]. In ad hoc networking environment, the most energy consumption of the wireless network interface is due to the packet transmission [37]. Motivated by this fact, we focus on the cross layer approach by jointly design the energy efficient routing algorithm in network layer and the energy efficient signal combining in physical layer.

The energy efficient routing and transmitter side diversity have been studied separately in the literatures. The transmitter side diversity, pioneered by Alamouti's paper [7] shows the significant performance gains can be achieved in the multiple-input-multiple-output (MIMO) systems. However, multiple antennas in sensor node may be impractical due to the cost. To overcome this problem, the cooperative communication concept has been recently proposed [60]. This cooperative communication explores the broadcast nature of the wireless medium, where signal transmitted by a node will be received by all nodes within its transmission range. This property is usually referred to as the Wireless Broadcast Advantage. In the multi-hop transmission, nodes that have received the transmitted signal will cooperatively help relaying and form a virtual multi antenna system. This virtual multi antenna system achieves the significant performance gains as in the MIMO system.

There are many existing energy efficient routing algorithms in the literatures. The minimum total energy routing (MTE) [99] algorithm selects the route that minimizes the total transmission energy along the route. The min-max battery cost routing (MMBCR) algorithm [92] tries to minimize the battery cost among

the maximum battery cost routes. This algorithm avoids the overuse of nodes along the minimum total energy route. However, the MMBCR selects route that is far from the energy efficient route. To overcome the problem, the conditional min-max battery cost routing (CMMBCR) [99] is proposed to trade-off between MTE and MMBCR. In [27], they proposed a heuristic called flow augmentation (FA) algorithm that gradually makes transition from MTE to MMBCR. They show that their algorithm performs better than CMMBCR algorithm. However, all the existing energy efficient algorithms do not jointly utilize the cooperative transmission and energy efficient routing in performing routing decision. In [56], a joint cooperative communication and routing selection algorithm is proposed. They however, design the power allocation method is based on energy minimization.

In this chapter, we propose a cooperative routing algorithm to maximize the network lifetime. The proposed scheme combines the cooperative transmission and the energy aware routing algorithm. To maximize the network lifetime, we propose maximum lifetime power allocation in the cooperative transmission. The maximum lifetime power allocation allocates transmit power in each nodes according to the channel condition and the normalized remaining energy in the nodes. We argue and demonstrate that using this criterion along with any energy aware routing algorithm (such as MTE and FA algorithms) results in longer network lifetime. We show the effectiveness of our proposed method through extensive simulations.

5.2 System Model

We modeled the wireless network as an undirected simple finite graph $G(V, E)$, where V is the set of nodes in the network and E is the set of all links/edges. The link $(i, j) \in E$ implies that node $j \in S_i$ can be directly reached by node i with a

certain transmit power level in the predefined dynamic range, where S_i is the set of nodes that can be directly reached by node i. We assume that every node has the initial battery energy of $\mathcal{E}_i$ for $\forall i \in V$. Every packet transmission consumed energy. The energy expenditure for transmission from node i to j is proportional to $d(i,j)^\alpha$, where $d(i,j)$ is the distance between node i and j, α is between 2 and 4 and depends on the transmission environment [85]. In this chapter, we assume $\alpha = 2$ as the path loss exponent for free space propagation. In the rest of this section, we summarize the existing energy-aware routing algorithms and the link cost formulation in cooperative transmission.

5.2.1 Energy-aware routing

The minimum total energy (MTE) routing uses the simple energy metric representing the total energy consumed along the route. If we consider a route $r = \{n_0, \cdots, n_d\}$, where n_0 is the source node and n_d is the destination node. Let denote the energy consumed in transmitting packet over the hop (n_i, n_j) as $e(n_i, n_j)$, then the total expended energy in that route is $P(r) = \sum_{i=0}^{d-1} e(n_i, n_{i+1})$. The MTE routing selects the route among all routes that minimizes the total expended energy in the route

$$r^* = \arg\min_{r \in R(n_0,n_d)} P(r), \tag{5.1}$$

where $R(n_0, n_d)$ is the set of routes from source node n_0 to destination node n_d.

The flow augmentation (FA) algorithm is similar to the MTE routing algorithm, except it weights the energy consumed over one hop by the normalized residual energy. In particular, the FA algorithm employs $e(n_i, n_j)^{x_1} \underline{E}_{n_i}^{-x_2} E_{n_i}^{x_3}$ as the energy metric over the hop (n_i, n_j), where $\underline{E}_{n_i}$ is the residual energy of node n_i at current time and E_{n_i} is the initial energy of node n_i. Therefore, the to-

tal weighted energy expended in a route is $P(r) = \sum_{i=0}^{d-1} e(n_i, n_{i+1})^{x_1} \underline{E}_{n_i}^{-x_2} E_{n_i}^{x_3}$. And the algorithm selects the route that minimizes the total weighted energy, $r^* = \arg\min_{r \in R(n_0, n_d)} P(r)$. This metric gradually avoids the minimum energy path as the residual energy of nodes along the minimum energy path is low. We use $(x_1, x_2, x_3) = (1, 5, 5)$ throughout this chapter.

5.2.2 Link cost formulation

Employing the cooperative transmission changes the link cost for the routing algorithm. Let denote the transmitter set as $T_x = \{t_1, \cdots, t_n\}$ and the receiver set as $R_x = \{r_1, \cdots, r_m\}$. We summarize the link cost in the following 4 transmission modes [56].

1. *Point-to-point,* $T_x = \{t_1\}$, $R_x = \{r_1\}$: the received signal is represented as

$$r(t) = \beta\omega s(t) + \eta(t), \tag{5.2}$$

where β is the channel response, $s(t)$ is the unit-energy transmit signal, and $\eta(t)$ is the additive gaussian noise. The transmit power at the transmitter is represented as $P_T = |\omega|^2$ and the SNR at the receiver side is $\gamma = \frac{\beta^2 |\omega|^2}{P_\eta}$, where P_η is the noise variance. We assume the receiver can correctly decode the received signal if the receiver side SNR is above the minimum threshold value, γ_{min}. Therefore, the link cost in point-to-point is described as

$$L(t_1, r_1) = e(t_1, r_1) = \frac{\gamma_{min} P_\eta}{\beta^2}. \tag{5.3}$$

For free space propagation, $\beta = d(t_1, r_1)^{-\alpha/2}$, where $d(t_1, r_1)$ is the distance between node t_1 and r_1.

2. *Broadcast,* $T_x = \{t_1\}$, $R_x = \{r_1, \cdots, r_m\}$: using the wireless broadcast advantage, the cost for reaching the receiver set R_x is the cost to reach the farthest

receiver, i.e.,

$$L(t_1, R_x) = \max\{L(t_1, r_1), \cdots, L(t_1, r_m)\}. \tag{5.4}$$

3. *Cooperative,* $T_x = \{t_1, \cdots, t_n\}$, $R_x = \{r_1\}$: the signal at the receiver is

$$r(t) = \sum_{i=1}^{n} \beta_{i1} |\omega| s(t) + \eta(t). \tag{5.5}$$

Using simple calculation, the link cost/total power allocation can be obtained as

$$L(T_x, r_1) = \frac{1}{\sum_{i=1}^{n} \frac{\beta_{i1}^2}{\gamma_{min} P_\eta}}. \tag{5.6}$$

The energy consumption in each transmitter is presented as

$$e(t_1, r_1) = \frac{\beta_{i1}^2 \gamma_{min} P_\eta}{\left(\sum_{i=1}^{n} \beta_{i1}^2\right)^2}. \tag{5.7}$$

4. *Cooperative-Broadcast,* $T_x = \{t_1, \cdots, t_n\}$, $R_x = \{r_1, \cdots, r_m\}$: If we assume that the channel estimation is done in the receiver and the cooperative transmitted signal can be coherently decoded in the receiver, then we will have a multi-node to multi-node scenario. The link cost for this mode of transmission will be

$$L(T_x, R_x) = \max\left\{ \frac{1}{\sum_{i=1}^{n} \frac{\beta_{i1}^2}{\gamma_{min} P_\eta}}, \cdots, \frac{1}{\sum_{i=1}^{n} \frac{\beta_{im}^2}{\gamma_{min} P_\eta}} \right\}. \tag{5.8}$$

Similarly, the energy consumption in each of the transmitter is

$$e(t_1, R_x) = \frac{\beta_{im^*}^2 \gamma_{min} P_\eta}{\left(\sum_{i=1}^{n} \beta_{im^*}^2\right)^2}, \tag{5.9}$$

where

$$m^* = \arg\max \left\{ \frac{1}{\sum_{i=1}^{n} \frac{\beta_{i1}^2}{\gamma_{min} P_\eta}}, \cdots, \frac{1}{\sum_{i=1}^{n} \frac{\beta_{im}^2}{\gamma_{min} P_\eta}} \right\}. \tag{5.10}$$

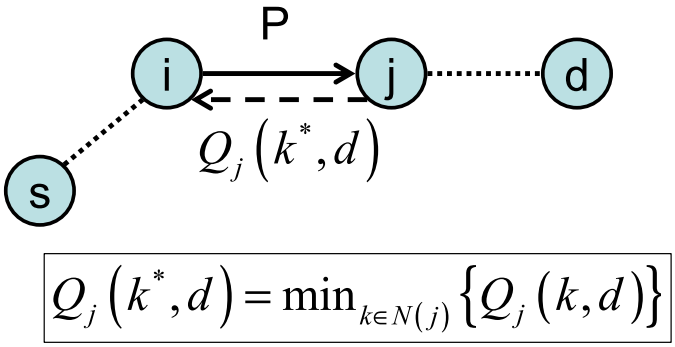


Figure 5.1: Exchange and Update Q-value

5.3 Proposed solution

5.3.1 Maximum lifetime power allocation

In this section, we propose a different power allocation that takes into account the goal of the routing algorithm; that is to maximize the network lifetime. We note that the flow augmentation routing algorithm minimizes the total transmit power in the route weighted by the normalized residual energy. By weighting the energy metric with the normalized residual energy, the route with extremely low residual node will be avoided. Based on this concept, we re-derive the power allocation problem in the cooperative transmission case and we referred the resulting power allocation to as the maximum lifetime power allocation. Specifically, we want to minimize

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \frac{E_i}{E_i} |\omega_i|^2 \\ \text{s.t.} \quad & \frac{|\sum_{i=1}^{n} \beta_{i1}\omega_i|^2}{P_\eta} \geq \gamma_{min}. \end{aligned} \tag{5.11}$$

The above optimization problem minimizes the weighted total power while ensuring the received SNR is larger than minimum required SNR. Using the Lagrange

multiplier method we have

$$L(\omega_1, \cdots, \omega_n, \lambda) = \sum_{i=1}^{n} \frac{E_i}{\underline{E}_i} |\omega_i|^2 - \lambda \left(\frac{|\sum_{i=1}^{n} \beta_{i1} \omega_i|^2}{P_\eta} - \gamma_{min} \right), \tag{5.12}$$

Taking the partial derivatives, we have

$$\frac{\partial L}{\partial \omega_i} = 2\frac{E_i}{\underline{E}_i}\omega_i - 2\frac{\lambda \beta_{i1} |\sum_{i=1}^{n} \beta_{i1} \omega_i|}{P_\eta} = 0, \ \forall i \tag{5.13}$$

$$\frac{\partial L}{\partial \lambda} = \frac{|\sum_{i=1}^{n} \beta_{i1} \omega_i|^2}{P_\eta} - \gamma_{min} = 0. \tag{5.14}$$

Equivalently, (5.13) can be represented as

$$\omega_i = \frac{\underline{E}_i}{E_i} \lambda \beta_{i1} \frac{\sqrt{P_\eta \gamma_{min}}}{P_\eta}. \tag{5.15}$$

We note that we have use $|\sum_{i=1}^{n} \beta_{i1} \omega_i|^2 = P_\eta \gamma_{min}$ to get (5.15). Substituting (5.15) to (5.14), we get

$$\sum_{i=1}^{n} \frac{\underline{E}_i}{E_i} \lambda \beta_{i1}^2 \frac{\sqrt{P_\eta \gamma_{min}}}{P_\eta} = \sqrt{P_\eta \gamma_{min}}, \tag{5.16}$$

hence, we have

$$\lambda = \frac{P_\eta}{\sum_{i=1}^{n} \left(\frac{\underline{E}_i}{E_i} \right) \beta_{i1}^2}. \tag{5.17}$$

This implies that (5.15) is equivalent to

$$\omega_i = \frac{\frac{\underline{E}_i}{E_i} \beta_{i1} \sqrt{P_\eta \gamma_{min}}}{\sum_{i=1}^{n} \left(\frac{\underline{E}_i}{E_i} \right) \beta_{i1}^2}, \tag{5.18}$$

and the energy consumption of cooperative transmission in each node using maximum lifetime criteria is

$$e(t_i, r_1) = |\omega_i|^2 = \frac{\left(\frac{\underline{E}_i}{E_i} \right)^2 \beta_{i1}^2 P_\eta \gamma_{min}}{\left[\sum_{i=1}^{n} \left(\frac{\underline{E}_i}{E_i} \right) \beta_{i1}^2 \right]^2}. \tag{5.19}$$

This power allocation criterion (5.19) has the interpretation that in addition to using the channel condition β_{i1} for power allocation, the node who has abundant

residual energy will help more on the cooperative transmission compared to the node with less residual energy. In initial deployment, when all the sensor nodes have abundant of residual energy, the criteria (5.19) practically reduces to (5.7). We will show by simulation in Section 5.5, that the joint maximum lifetime power allocation and the maximum lifetime routing algorithm can significantly extend the network lifetime.

5.3.2 Joint maximum lifetime routing and power allocation

The simplest way to jointly consider the maximum lifetime routing and the maximum lifetime power allocation is to select the cooperating nodes along the non-cooperative route [56]. The cooperative MTE and cooperative FA algorithm are summarized in Table 5.1 and Table 5.2, respectively.

Table 5.1: Centralized cooperative MTE-n

1. Find the minimum total energy route with edge cost as in (5.7)
2. Select the last n nodes in the MTE route to do cooperative transmission.
3a. For minimum energy allocation, deduct the amount of energy proportional to (5.7) from each of cooperating nodes.
3b. For maximum lifetime energy allocation, deduct the amount of energy proportional to (5.19) from each of cooperating nodes.

Table 5.2: Centralized cooperative FA(x_1,x_2,x_1)-n

1. In every update time: find the maximum lifetime route with edge cost between node i and j as $e(i,j)^{x_1}\underline{E}_{n_i}^{-x_2}E_{n_i}^{x_3}$. The optimal route is denoted as $r^* = \arg\min_{r\in R(s,d)}P(r)$, where $P(r) = \sum_{i=0}^{d-1} e(n_i, n_{i+1})^{x_1}\underline{E}_{n_i}^{-x_2}E_{n_i}^{x_3}$.
2. Select the last n nodes in the FA route to do cooperative transmission.
3a. For minimum energy allocation, deduct the amount of energy proportional to (5.7) from each of cooperating nodes.
3b. For maximum lifetime energy allocation, deduct the amount of energy proportional to (5.19) from each of cooperating nodes.

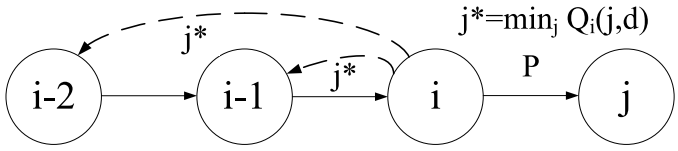


Figure 5.2: Cooperation transmission illustrated

5.4 Distributed cooperative routing and learning

In this section, we develop a distributed method to implement the maximum lifetime cooperative routing algorithm. The method is based on the distributed reinforcement learning routing algorithm [68]. The routing decision is learned by all nodes in the network. Each node maintains the best packet delivery cost to all the destinations. In particular, each node i maintains a table of Q-values $Q_i(j,d)$, for $j \in N(i)$, where j is in the set of node i neighbors, $N(i)$, and node d is the destination. The $Q_i(j,d)$ has the interpretation of node i's best estimated cost that a packet would incur to reach its destination node d from node i when the packet is sent via node i's neighbor node j.

The value in the Q-table will be exchanged between node i and j, whenever there is a packet is sent from node i and j, and vice versa. The exchange mechanism is illustrated as in Figure 5.1. Whenever node i transmits a packet P to node j, node j feedbacks $Q_j(k^*, d) = \min_{k \in N(j)} Q_j(k, d)$ to node i as shown in the figure. The node i uses this value to update its own Q-value as follow

$$Q_i(j, d) = (1 - \delta)Q_i(j, d) + \delta[Q_j(k^*, d) + c(i, j)], \tag{5.20}$$

where $c(i, j)$ is the cost for sending packet from node i to node j, and $\delta \in [0, 1]$ is the learning rate for the algorithm.

Since in this chapter, the routing algorithms are driven for the purpose of maximizing the network lifetime, then the cost of sending a packet between node i and node j is related to the energy consumption for sending the packet. In particular, the cost of sending packet for MTE and FA routing algorithms are

$$\text{[MTE]:} \qquad c(i, j) = e(i, j), \tag{5.21}$$

$$\text{[FA}(x_1,x_2,x_3]\text{:} \qquad c(i, j) = e(i, j)^{x_1} \underline{E}_{n_i}^{-x_2} E_{n_i}^{x_3}, \tag{5.22}$$

For MTE, $Q_i(j, d)$ represents the total energy consumption used to delivery a packet from node i to node d via node i's neighbor node j. In contrast, $Q_i(j, d)$ in FA represents the total energy consumption in delivering a packet from node i to d via j, weighted by the normalized residual energy of nodes along the route. We note that the entire route used for packet transmission is appended to the header when doing the learning. After making the next hop decision and before transmitting the packet, the node will inform its previous $n - 1$ nodes to do the cooperation based on the route in the packet header (where the packet comes from). Figure 5.2 illustrates this situation where node i has made the routing decision, but it has not transmitted the packet P yet. At this time node i informs the $n - 1$ nodes to

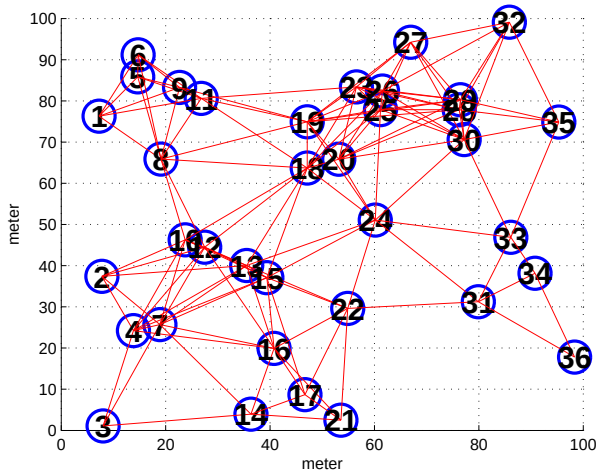


Figure 5.3: Random network with 36 nodes in 100 meter by 100 meter area

engage in cooperation ($n = 3$ is shown in the figure). In this way, the cooperative transmission also helps reducing the transmit energy expended during the learning period.

5.5 Simulation Results

We simulate the packet routing system as the discrete event system. The topology of the simulated network is shown in Figure 5.3. The network contains 36 nodes, which uniformly deployed in an 100 meter by 100 meter area. The traffic is generated from node 21 to node 6 and node 32 to node 3. The packet arrival follows the Poisson distribution and the number of packets introduced per unit simulation time step is referred to as packet arrival rate (traffic load), μ. Multiple packets generated at a node are stored in its unbounded first-in-first-out queue. At every time step, each node removes the packet in front of its queue and sends the packet to the next hop according to the routing decision. When a node receives a

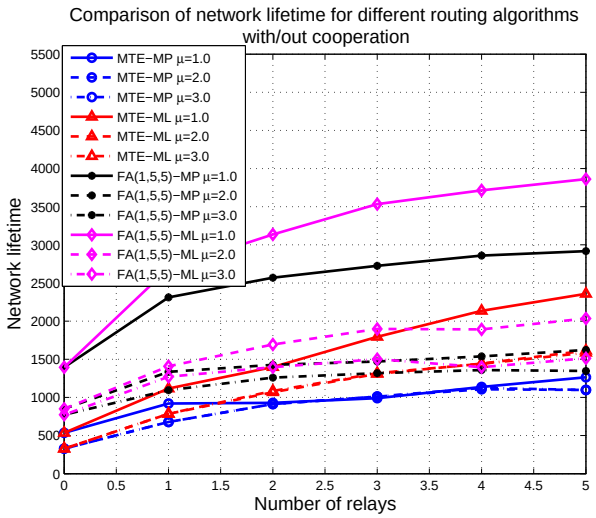


Figure 5.4: Network lifetime

packet, it either queues the received packet at the end of its queue or removes the packet from the network. The latter case happens when the packet arrives at its destination. All the nodes in the network initially have $\mathcal{E}_i = 100,\ \forall i \in V \setminus \{21, 32\}$ unit energy, except node 21 and node 32, which have $\mathcal{E}_{21} = 1000$ and $\mathcal{E}_{32} = 1000$.

We compare 2 routing algorithms, namely MTE and FA algorithm, each with 2 power allocation. We referred the MTE with the minimum power allocation criterion (5.7) to as MTE-MP and the MTE with the maximum lifetime power allocation (5.19) to as MTE-ML. Similar naming is given for FA algorithm. We compare network lifetime, average packet delivery time, average energy per packet, and total packet delivery. The network lifetime (measured in terms of simulation time step) is defined as the time before the first node dies. The average packet delivery time is defined as the time between packet introduction at the source and its removal time at the destination. The average energy per packet is the total energy consumed per delivered packets. Finally, the total delivered packet is the number of packets that are successfully delivered before the first node dies.

Figure 5.4 shows the lifetime of the network that is achieved by different *centralized* routing algorithm with minimum power allocation and maximum lifetime power allocation. We note that for the centralized routing algorithm, the minimum total energy route is only calculated at the beginning of the simulation. In contrast, the FA algorithm recomputes the route every 20 simulation time step using the most current residual energy on each node. The performance of each scheme is compared for different traffic load, namely low load ($\mu = 1.0$), medium load ($\mu = 2.0$), and high load ($\mu = 3.0$). The x-axis of the figure represents the number of relays used in the cooperative transmission. We recall that the n-relays are selected from the last n nodes along the noncooperative MTE and FA algorithm, correspondingly. In the figure, the cooperative MTE-MP achieves 71.77%, 73.45%, 85.06%, 112.30%, and 136.20%, longer network lifetime when $1, 2, \cdots, 5$ relays are used, respectively, compared to the noncooperative routing. However, if the maximum lifetime power allocation is used, the MTE-ML achieves 108.72%, 161.05%, 235.09%, 298.88%, and 340.54%, longer network lifetime when $1, 2, \cdots, 5$ relays are used. Obviously, we see that the maximum lifetime power allocation can achieves much higher network lifetime compared to the maximum power allocation. The MTE-ML algorithm can achieves from $1 \sim 3.5$ times higher network lifetime compared to noncooperation routing. Compared to MTE-MP, MTE-ML achieves around $1 \sim 2$ times better network lifetime. The reason for this phenomenon is that, the minimum power allocation allocates the power merely based on the channel condition (in our model (5.7) farther away nodes have lower channel gain, due to path loss, and therefore allocate less transmit power). Hence, in some popular route (minimum total energy route), some nodes will be overused in the minimum power allocation and the time until the first node dies (network lifetime)

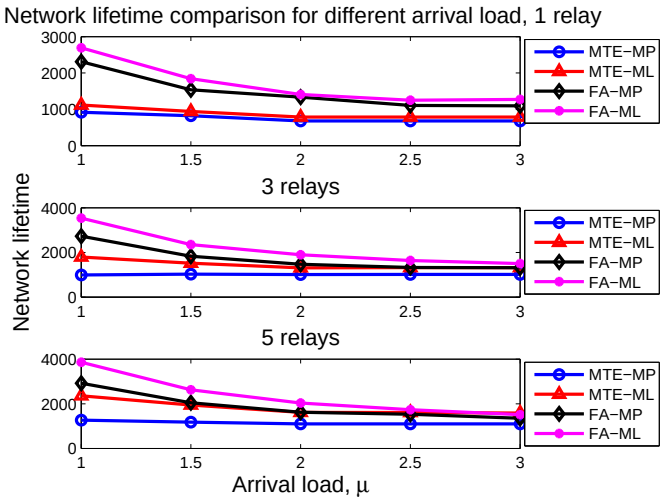


Figure 5.5: Network lifetime comparison for different routing algorithms when the number of relays is 1, 3, and 5

is shorter. In contrast, in maximum lifetime power allocation, the power allocation allocates the power according to the normalized residual energy in the nodes and the channel condition. Therefore, the situation that one particular node is overused will be minimized in cooperative transmission and MTE-ML results in longer network lifetime. Figure 5.4 also shows the network lifetime when the FA algorithm is used. Compared to the noncooperative routing, the FA-MP achieves 65.72%, 84.29%, 95.45%, 105.01%, and 109.18% longer lifetime, when $1, 2, \cdots, 5$ relays are used, respectively. In contrast, the FA-ML achieves 93.02%, 124.89%, 153.44%, 166.41%, and 176.96%, when $1, 2, \cdots, 5$ relays are used, respectively.

Figure 5.5 shows the sensitivity of the routing algorithm to the network load. From this figure, we observe that the MTE-MP is the least sensitive to the traffic load among all the algorithms. This can be understood since the MTE-MP only uses the minimum total transmit power as the routing selection and power allocation criterion. No matter how the traffic arrival load is, the route selection

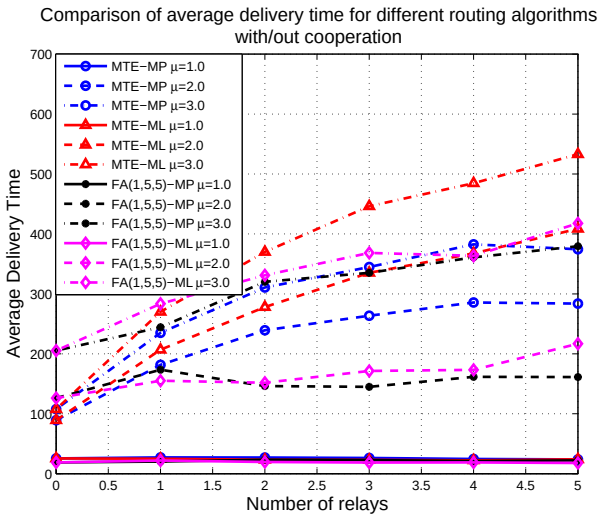


Figure 5.6: Average delivery time

and power allocation will not change. However, the MTE-ML, FA-MP, and FA-ML algorithms choose the route and power allocation according to the normalized residual energy in the nodes. When the traffic arrival load is small, the ML algorithm tries its best to balance the load to all the cooperative nodes. Similarly, the FA algorithm tries to balance the load to nodes in all possible routes between the source and destination. As the traffic load becomes large, the algorithms find less flexibility to distribute the load among either the cooperative nodes or the nodes in routes between source and destination. Hence, the algorithm based on normalized residual energy degrades as the arrival load becomes larger. In all cases, the FA-ML is superior to FA-MP, FA-MP outperforms MTE-ML, and MTE-ML is better than MTE-MP. But, the performance gains become smaller as the network arrival load is larger.

Figure 5.6-5.8 show the average delivery time, average consumed energy per packet and the total delivery packets before the first node dies. In Figure 5.6, all the algorithms have similar delivery time when the traffic load is low ($\mu = 1.0$).

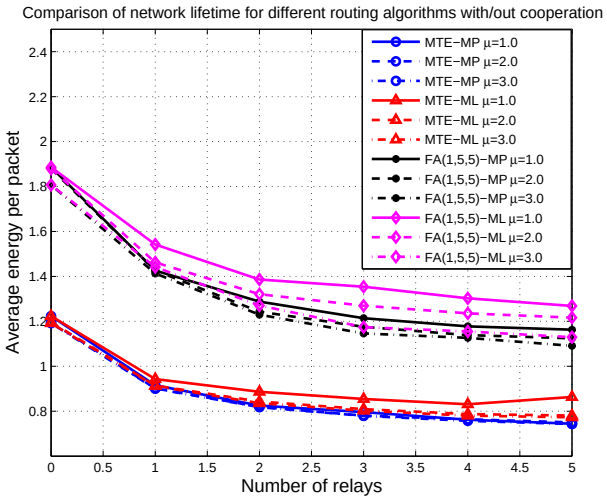


Figure 5.7: Average consumed energy per packet

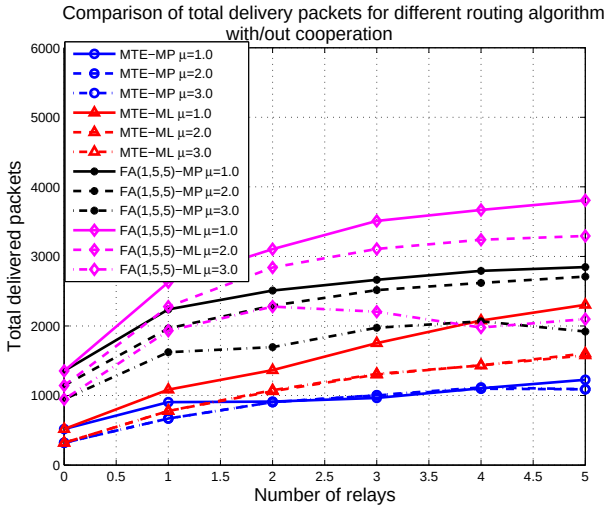


Figure 5.8: Total delivered packets

In the medium traffic load, the FA algorithms with MP and ML power allocation have lower delivery time compared to MTE algorithm. This is due to the fact that FA algorithm explores several routes besides the minimum energy route and distributes the traffic to queues among routes between source and destination. Therefore, the algorithm has practically lower delivery time. However, in the large network load, all the algorithms except MTE-ML have similar delivery time. The MTE-ML has the worst delivery time. Figure 5.7 shows the average energy per packet consumed by different routing algorithms. It is obvious that the MTE class of the algorithm has the lower energy consumed per packet. The FA algorithm has higher energy consumption because the algorithm selects less energy efficient route to balance the energy consumption among nodes. In both algorithms, the cooperative transmission is very effective in lowering the energy consumed per packet. Finally, Figure 5.8 shows the number of packets successfully delivered before the first node dies. The performance of different routing algorithms in this metric is very similar to the network lifetime. The FA-ML outperforms FA-MP, FA-MP outperforms MTE-ML, and MTE-ML is better than MTE-MP in low to high traffic load.

Figure 5.10 and 5.9 show the distributed reinforcement learning implementation for all routing algorithms according to Section 5.4. In this simulation, the learning parameter is chosen as $\delta = 0.85$. Figure 5.9 shows the learning curves in distributed implementation when the traffic load is 1.0 and 3 cooperative relay nodes are used. It is obvious that the distributed MTE-MP converges to the centralized solution within 100 simulation time. In contrast, the distributed FA algorithm converges to the centralized solution in rather longer time. We recall that the optimal route according to the FA algorithm changes as the residual energy in nodes changes. In

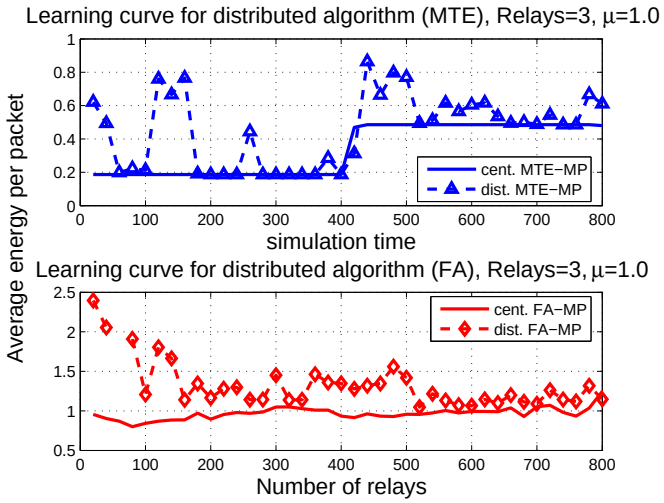


Figure 5.9: Learning curves in the distributed learning algorithm

this time varying route selection, the distributed implementation is able to achieve the centralized solution. Finally, Figure 5.10 shows the result of learning algorithm for different routing algorithms. In this Figure, the distributed implementation can achieves almost similar network lifetime as in the centralized solution. However, if we observe the total delivered packet, the distributed algorithm achieves much lower throughput compared to the centralized solution. The reason for this is that the distributed algorithms consumed some portion of the nodes' energy to explore and learn the good routing decision. During the exploration (learning) stage, loops in the route may appear and the algorithms practically deliver only a small portion of the traffic. From the figure, one can observe that the cooperative transmission can still improve the number of delivered packet.

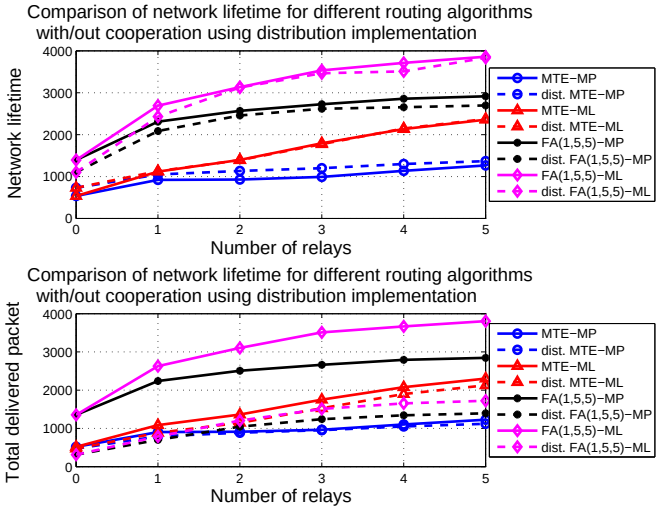


Figure 5.10: Comparison of network lifetime and total delivery packets for distributed reinforcement learning implementation, when the network load, $\mu = 1.0$

Chapter 6

Cooperation Enforcement and Learning for Optimizing Packet Forwarding Probability

In previous chapters, we focus on several techniques to efficiently use one unit energy in sensor communication. We also investigate several methods to prolong the network lifetime. The optimization is done under the assumption that all nodes are willing to cooperate. In wireless ad hoc networks, autonomous nodes would be reluctant to forward others' packets because of the nodes' limited energy. However, such selfishness and noncooperation would deteriorate both the system efficiency and nodes' performances. Moreover, the distributed nodes with only local information may not know the cooperation point, even if they are willing to cooperate. Hence, it is crucial to have a design goal of a distributed mechanism for enforcing and learning the cooperation among the greedy nodes for packet forwarding. In

this chapter[1], we propose a self-learning repeated game framework to overcome the problems and achieve the design goal. We employ the self transmission efficiency as the utility function of each autonomous node, where each node maximizes the ratio of the power for successful self-transmission over the total power used for self transmission and packet forwarding. Then, we propose a framework to search for good cooperation points and maintain the cooperation among selfish nodes. The framework has two steps: First, an adaptive repeated game scheme is designed to ensure the cooperation among nodes for the current cooperative packet forwarding probabilities. Second, self-learning algorithms are employed to find the better cooperation probabilities that are feasible and benefit all nodes. We propose three learning schemes for different information structures; namely, learning with perfect observability, learning through flooding, and learning through utility prediction. Starting from noncooperation, the above two steps are employed iteratively, so that a better cooperating point can be achieved and maintained in each iteration. From the simulations, the proposed framework is able to enforce cooperation among distributed selfish nodes and the proposed learning schemes achieve 70% to 98% performance efficiency compared to the optimal solution.

This chapter is organized as follows. First, we give the motivation, system model. We then propose and analyze the repeated game framework for packet forwarding under different information structures. We construct self-learning algorithms corresponding to different information structures in detail. Finally, we evaluate the performances of our proposed scheme using extensive simulations.

[1]Material in this chapter has been submitted to Transactions on Mobile Computing [78]

6.1 Motivation

Wireless ad-hoc networks consist of autonomous nodes without centralized control. In such autonomous networks, the nodes may not be willing to fully cooperate and accomplish the network task. Specifically for the packet forwarding problem, forwarding the others' packets consumes the node's limited battery resource. Therefore, it may not be of the node's best interest to forward others' arriving packets. However, rejection of forwarding other's packets non-cooperatively will severely affect the network functionality, and impair the nodes' own benefits if the other nodes also play non-cooperatively. Therefore, it is crucial to design a mechanism to enforce cooperation among greedy nodes. In addition, the randomly located nodes with local information may not know how to cooperate, even if they are willing to cooperate.

The packet forwarding problem in ad hoc networks has been extensively studied in the literature. The fact that nodes act selfishly to optimize their own performances has motivated many researchers to apply the game theory [43,77] in solving this problem. Broadly speaking, the approaches used in encouraging the packet forwarding task can be categorized into two methods. The first type of methods makes use of virtual payment. Virtual currency [21, 22], pricing [33], and credit based method [107] fall into this first type. The second type of approaches is related to personal and community enforcement to maintain the long-term relationship among nodes. Cooperation is sustained because defection against one node causes personal retaliation or sanction by others. This second approach includes the following works. Marti et. al. [70] propose mechanism called *watchdog* and *pathrater* to identify the misbehaving nodes and deflect the traffic around them. Buchegger et. al. [20] and Michiardi et. al. [71, 72] define protocols based on rep-

utation system. Felegyhazi et al. [38, 39] consider a model to show cooperation among participating nodes. Altman et. al. similarly [10] consider a less aggressive punishment policy. In [48], Han et. al. proposes leaning repeated game approaches to enforce cooperation and obtain better solution. Some other works using Game Theory in solving communication problems can be found in [59], [69], [47].

Since in some ad hoc networks, it might be difficult to implement the virtual payment system because of the practical implementation challenges such as enormous signallings. In this chapter, we concentrate on the second type of approaches and design the mechanism such that cooperation can be enforced in a distributed way. In addition, unlike the previous works which assume the nodes know the cooperation points or other nodes' behaviors, we argue that randomly deployed nodes with local information may not know how to cooperate even if they are willing to do so. Motivated by these facts, we propose a self-learning repeated game framework for cooperation enforcement and learning.

We quantify the node's utility as the self-transmission efficiency, where each node maximizes the ratio of the power for successful self-transmission over the total power used for self-transmission and packet forwarding. The goal of the node is to maximize the long-term average efficiency. Using this utility function, a distributed self-learning repeated game framework is proposed to ensure cooperation among nodes in ad-hoc networks. The framework has two major steps: First, the repeated game enforces cooperation in performing packet forwarding tasks. This first step in our framework ensures that any cooperation equilibrium that is more efficient than the Nash Equilibrium (NE) of the one stage game can be sustained. It is very important to emphasize that the inefficiency of NE is irrelevant to the utility function chosen; the inefficiency of NE is merely caused by the nature that

acting greedily to maximize node's own benefit results in low average efficiency in the network. The repeated game allows nodes to consider the history of actions/reactions of their opponents in making the decision. The cooperation can be enforced, since any deviation will cause the punishment from other nodes in the future. This results in lower long term efficiency.

The second step utilizes the learning algorithm to achieve the desired efficient cooperation equilibrium. We propose three learning algorithms for different information structures; namely, learning with perfect observability, learning through flooding, and learning through utility prediction. Starting from non-cooperation point, the two proposed steps are applied iteratively. A better cooperation is discovered and maintained in each iteration, until no more efficient cooperation point can be achieved. From the simulation results, our proposed framework is able to enforce cooperation among selfish nodes. Moreover, compared with the optimal solution obtained by a centralized system with global information, our proposed learning algorithms achieve similar performances in the symmetric network. Depending on learning algorithms and the information structures, our proposed schemes achieve 70% to 98% of the optimality in the random network.

6.2 System Model and Design Challenge

We consider a network with N nodes. Each node has a limited transmit power constraint. This implies that only nodes within the transmission range are neighbors. The packet delivery typically requires more than one hop. The source, the relays (intermediate nodes), and the destination constitute an active route. We assume an end-to-end mechanism that enables a source node to know if the packet is transmitted successfully. The source node can observe whether there is a packet

drop in one particular active path. However, the source node might not know where the packet is dropped. Finally, we assume that routing decision has already been done before optimizing the packet forwarding probabilities.

Let's denote the set of sources and destinations as $\{S_i, D_i\}$, for $i = 1, 2, \cdots, M$, where M represents the number of source-destination pairs that are active in the network. Suppose the shortest path for each source-destination pair has been discovered. Let's denote the route/path as $R_i = (S_i, f^1_{R_i}, f^2_{R_i}, \cdots, f^n_{R_i}, D_i)$, where S_i denotes the source node, D_i denotes the destination node, and $\{f^1_{R_i}, f^2_{R_i}, \cdots, f^n_{R_i}\}$ is the set of intermediate/relay nodes, thus, there are $n + 1$ hops from source node to the destination node. Let $V = \{R_i : i = 1, \cdots, M\}$ be the set of routes corresponding to all source-destination pairs. Let's denote further the set of routes where node j is the source as $V^s_j = \{R_i : S(R_i) = j, i = 1 \ldots M\}$, where $S(R_i)$ represents the source of route R_i. The power expended in node i for transmitting its own packet is

$$P^{(i)}_s = \sum_{r \in V^s_i} \mu_{S(r)} \cdot K \cdot d(S(r), n(S(r), r))^\gamma, \tag{6.1}$$

where $\mu_{S(r)}$ is the transmission rate of source node $S(r)$, K is the transmission constant, $d(i, j)$ is the distance between node i and node j, $n(i, r)$ denotes the neighbor of node i on route r, and γ is the transmission path-loss coefficient. For the link from node i to its next hop $n(i, r)$ on route r, $K \cdot d(i, n(i, r))^\gamma$ describes the reliable successful transmission power per bit transmission.

Clearly, probability of successful transmission from node i to its destination depends on the forwarding probabilities employed in the intermediate nodes and it can be represented as

$$P^i_{Tx,r} = \prod_{j \in (r \setminus \{S(r)=i, D(r)\})} \alpha_j, \tag{6.2}$$

where $D(r)$ is the destination of route i and $(r \backslash \{S(r) = i, D(r)\})$ is the set of nodes on route r excluding the source and destination. Define the *good power* consumed in transmission node i, $P^{(i)}_{s,good}$ as the product of the power used for transmitting node i's own packet and the probability of successful transmission from node i to its destination,

$$P^{(i)}_{s,good} = \sum_{r \in V^s_i} \mu_{S(r)} \cdot K \cdot d(S(r), n(S(r), r))^{\gamma} P^i_{Tx,r}. \tag{6.3}$$

Moreover, let the set of routes where node j is the forwarding node be W_j and α_i for $i = 1, \cdots, N$ be the packet forwarding probability for node i. The power used to forward others' packets is given by

$$P^{(i)}_f = \alpha_i \cdot K \cdot \sum_{r \in W_i} d(i, n(i, r))^{\gamma} \mu_{S(r)} P^i_{F,r}, \tag{6.4}$$

where $P^i_{F,r}$ is the probability that node i receives the forwarded packet in route r, and $\sum_{r \in W_i} \mu_{S(r)} \ P^i_{F,r}$ is the total rate that node i receives for packet forwarding. The probability that node i receives the forward packet in route r is represented as

$$P^i_{F,r} = \prod_{j \in \{f^1_r, f^2_r, \cdots, f^{m-1}_r\}} \alpha_j, \tag{6.5}$$

where $r = \{S(r), f^1_r, \cdots, f^{m-1}_r, f^m_r = i, \cdots, f^n_r, D(r)\}$ is the $n+1$ hops route from source $S(r)$ to destination $D(r)$, and the m^{th} forwarding node f^m_r is node i. $P^i_{F,r}$ depends on the packet forwarding probabilities of the nodes on the route r before node i.

We focus on maximizing the *self-transmission efficiency*, as the ratio of successful self-transmission power (good power) over the total power used for transmission and packet forwarding. Therefore, the stage utility function for node i can be represented as

$$U^{(i)}(\alpha_i, \alpha_{-i}) = \frac{P^{(i)}_{s,good}}{P^{(i)}_s + P^{(i)}_f}. \tag{6.6}$$

where α_i is the node i's packet forwarding probability, $\alpha_{-i} = (\alpha_1, \cdots, \alpha_{i-1}, \alpha_{i+1}, \cdots, \alpha_N)^T$ are the other nodes' forwarding probability. Putting (6.1), (6.4) and (6.3) into (6.6), we have

$$U^{(i)} = \frac{\sum_{r \in V_i^s} \mu_{S(r)} d(S(r), n(S(r), r))^\gamma \prod_{j \in (r \setminus \{S(r)=i, D(r)\})} \alpha_j}{\sum_{r \in V_i^s} \mu_{S(r)} d(S(r), n(S(r), r))^\gamma + \alpha_i \sum_{r \in W_i} d(i, n(i, r))^\gamma \mu_{S(r)} \prod_{j \in \{f_r^1, \cdots, f_r^{m-1}\}} \alpha_j}. \tag{6.7}$$

The problem in packet forwarding arises because the nodes in ad-hoc networks have their own authorities to decide whether to forward the incoming packets. Under this scenario, it is very natural to assume that each node selfishly optimizes its own utility function. In parallel to (6.7), node i selects α_i in order to maximize the transmission efficiency $U^{(i)}(\alpha_i, \alpha_{-i})$. In the game theory literatures [43, 77], Nash Equilibrium (NE) is a well-known concept, which states that in the equilibrium every node selects the best response strategy to the other nodes' strategies. The formal definition of NE is given as follow

Definition 3 *Define feasible range* Ω *as* $[0, 1]$. *Nash Equilibrium* $[\alpha_1^*, \cdots, \alpha_N^*]^T$ *is defined as:*

$$\begin{aligned} U^{(i)}(\alpha_i^*, \alpha_{-i}^*) &\geq U^{(i)}(\alpha_i, \alpha_{-i}^*), \forall i, \\ \forall \alpha_i &\in \Omega, \end{aligned} \tag{6.8}$$

i.e., given that all nodes play NE, no node can improve its utility by unilaterally changing its own packet forward probability. Here $\alpha_{-i}^* = (\alpha_1^*, \cdots, \alpha_{i-1}^*, \alpha_{i+1}^*, \cdots, \alpha_N^*)^T$.

Unfortunately, the NE for the packet forwarding game described in (6.7) is $\alpha_i^* = 0$, $\forall i$. This can be verified by finding the forwarding probability $\alpha_i \in [0, 1]$ such that $U^{(i)}$ is unilaterally maximized. To maximize the transmission efficiency of node i, the node can only make α_i as small as possible, since the successful probability of

its own packet transmission in (6.2) depends only on the other nodes' willingness to forward the packets. By greedily dropping its packet forwarding probability, node i reduces its total transmission power used for forwarding others' packets, therefore, increases its instantaneous efficiency. However, if all nodes play the same strategy, this causes zero efficiency in all nodes, i.e., $U^{(i)}(\alpha_1^* \dots \alpha_N^*) = 0, \ \forall i$. This implies the network breakdown. Hence, playing NE is inefficient. It is very important to emphasize that the inefficiency of NE is irrelevant to the utility function in (6.7). This inefficiency is merely the nature of greedy optimization unilaterally done by each of the nodes. To overcome the above problem, we propose a self-learning repeated game framework in the next two sections.

6.3 Repeated Game Framework and Punishment Analysis

As demonstrated in Section 6.2, the packet forwarding game has $\alpha_i^* = 0, \ \forall i$ as its unique Nash equilibrium if the game is only played once. This implies that all nodes in the network won't be cooperating in doing the packet forwarding. In practice, the packet forwarding game is played more than once and can be modelled as the infinitely repeated game (a game that is played in the infinite time horizon). In this chapter, we employ the average discounted utility with discounting factor δ. The average discounted utility of node i is given by:

$$\bar{U}_{\infty}^{(i)} = \lim_{t' \to \infty} \bar{U}_{t'}^{(i)} = (1-\delta) \sum_{t=1}^{\infty} \delta^{(t-1)} U^{(i)}(\vec{\alpha}(t)), \tag{6.9}$$

where $\vec{\alpha}(t) = (\alpha_1, \dots, \alpha_N)^T$, $U^{(i)}(\vec{\alpha}(t))$ is the utility of node i at each stage game (6.7) played at time t, and $\bar{U}_{t'}^{(i)}$ is the average discounted utility from time 1 to time t'. Unlike the one-time game, the repeated game allows a strategy to be contingent

on the past moves and results in the reputation and retribution effect, which give possibility for cooperation [43, 44, 55].

6.3.1 Design of Punishment Scheme under Perfect Observability

In this subsection, we analyze a class of punishment policy under the assumption of perfect observability. Perfect observability means that each node is able to observe actions taken by other nodes along the history of the game. This implies that node knows which node drops the packet and is aware of the identity of other nodes. This condition allows every node to detect any defection of other nodes and it also allows nodes to know if any node does not follow the game rule. The perfect observability is the ideal case and serves as a performance bound. In the next subsection, this assumption is relaxed to more practical situations where individual nodes only have limited local information.

Let's denote the NE in one stage forwarding game as $\vec{\alpha}^* = (\alpha_1^*, \cdots, \alpha_N^*)^T$, and the corresponding utility functions as $(v_1^*, \cdots, v_N^*)^T = (U^{(1)}(\vec{\alpha}^*), \cdots, U^{(N)}(\vec{\alpha}^*))^T$. We also denote

$$\begin{aligned}
\mathbf{U} &= \{(v_1, \cdots, v_N) | \ \exists \vec{\alpha} \in \Omega^N \text{ s.t.} \\
&\quad (v_1, \cdots, v_N) = (U^{(1)}(\vec{\alpha}), \cdots, U^{(N)}(\vec{\alpha}))\}, \\
\mathbf{V} &= \texttt{convex hull of } \mathbf{U}, \\
\mathbf{V}^\dagger &= \{(v_1, \cdots, v_N) \in \mathbf{V} | \ v_i > v_i^*, \ \forall i\}.
\end{aligned}$$

We note that $\mathbf{V}$ consists of all feasible utilities, and $\mathbf{V}^\dagger$ consists of feasible utilities that Pareto-dominate the one stage NE, this set is also known as the individually rational utility set. The Pareto-dominant utilities denote all utilities that are

strictly better than the one stage NE. From game theory literatures [43,44,55], the existence of equilibria that Pareto-dominate the one stage NE in repeated game is given by the Folk theorem [44].

Theorem 7 (Folk Theorem [44]) *Assume that the dimensionality of* $\mathbf{V}^{\dagger}$ *equals to* N*. Then, for any* $(v_1, \cdots, v_N)$ *in* $\mathbf{V}^{\dagger}$*, there exists* $\underline{\delta} \in (0,1)$ *such that for all* $\delta \in (\underline{\delta}, 1)$*, there exists an equilibrium of the infinitely repeated game with discounted factor* δ *in which player* i*'s average utility is* v_i*.*

Before we give the application of Folk theorem in the packet forwarding game, it is useful to recall the notion of dependency graph [39]. Given the routing algorithm and the source-destination pairs, the dependency graph is the directed graph that is constructed as follows. The number of nodes in the dependency graph is the same as the number of nodes in the network. When node i sends packets to node j via nodes $f^1, \cdots, f^n$, then there exist directed edges from node i to nodes $f^1, \cdots, f^n$. The resulting dependency graph is a directed graph, which describes the node dependency in performing the packet forwarding task. Let's define $deg_{in}(i)$ and $deg_{out}(i)$ as the number of edges going into node i and coming out from node i, respectively. Obviously, $deg_{in}(i)$ indicates of the number of nodes whose packets are forwarded by node i and $deg_{out}(i)$ is the number of nodes that forward node i's packets. Using the notation of the corresponding dependency graph, the application of Folk theorem in packet forwarding game is stated as follow:

Proposition 1 (Existence of Pareto-dominant forwarding equilibria under perfect observability) *Under the perfect observability condition and the corresponding dependency graph satisfies the following condition*

$$deg_{out}(i) > 0, \ \forall i. \tag{6.10}$$

Suppose that $\mathbf{V}^{\dagger}$ *has full dimensionality, then for any* $(v_1, \cdots, v_n) \in \mathbf{V}^{\dagger}$*, there exists* $\underline{\delta} \in (0,1)$*, such that for all* $\delta \in (\underline{\delta}, 1)$*, there exists an equilibrium of infinitely repeated game in which the node* i *average utility* v_i*.*

Proof 9 *Let* $\vec{\alpha} = (\alpha_1, \cdots, \alpha_N)^T$ *be the joint strategy results in* $(U^{(1)}(\vec{\alpha}), \cdots, U^{(N)}(\vec{\alpha}))$*. The full dimensionality condition ensures the set* $\big(U^{(1)}(\vec{\alpha}), \cdots, U^{(j-1)}(\vec{\alpha}), U^{(j)}(\vec{\alpha}) - \varepsilon, U^{(j+1)}(\vec{\alpha}), \cdots, U^{(N)}(\vec{\alpha})\big)$ *for any* $\varepsilon > 0$*, is in* $\mathbf{V}^{\dagger}$*. Let node* i*'s maximum utility be* $\overline{v}_i = \max_{\vec{\alpha}} U^{(i)}(\vec{\alpha}),\ \forall i$*. This maximum utility is obtained when all nodes try to maximize node* i*'s utility. Let the cooperating utility be* $v_i = U^{(i)}(\vec{\alpha}) \in \mathbf{V}^{\dagger}, \forall i$*. The cooperative utilities are obtained when all nodes play the agreed packet forwarding probabilities. Let the maximum utility node* i *can get when it is punished be* $\underline{v}_i = \max_{\alpha_i} \min_{\alpha_{-i}} U^{(i)}(\vec{\alpha})$*. This implies the best utility node* i *can get when all other nodes are punishing node* i*. Let's denote the node* j*'s utility when punishing node* i *as* w_j^i*. We note that from (6.7). Notice that* $\underline{v}_i$ *coincides with the one stage NE. If there exist* ϵ *and punishment period for node* i*,* T_i*, such that*

$$\frac{\overline{v}_i}{U^{(i)} - \epsilon} < (1 + T_i), \tag{6.11}$$

then the following rule of game ensures any individually rational utilities can be enforced.

1. Condition I: *All nodes play cooperation strategies if there is no deviation in the last stages. After any deviations go to Condition II (Suppose node* j *deviates).*

2. Condition II: *Nodes that can punish the deviating node (node* j*) play punishing strategies for punishment periods. The rest of the nodes keep playing cooperating strategies. If there is any deviation in Condition II, restart Condition II and punish the deviating node; if any punishing node does not play*

punishment in punishment period, the other nodes will punish that particular node for the punishment period; otherwise, after the end of the punishment, go to Condition III.

3. Condition III: *Play strategy that results in utility* $\left(U^{(1)}, \cdots, U^{(j-1)}, U^{(j)} - \varepsilon, U^{(j+1)}, \cdots, U^{(N)}\right)$. *If there is any deviation in Condition III, start Condition II and punish the deviating node.*

In the sequel, we show that under the proposition's assumptions:

- *the average efficiency gained by deviating node is smaller than the cooperating efficiency,*
- *the average efficiency gained by the punishing node that does not play punishment strategy in the punishment stage is worse than the efficiency gained by that node when it conforms to the punishing strategy.*

If node j *deviates in Condition I and then conforms, it receives at most* $\overline{v}_j$ *when it deviates,* $\underline{v}_j$ *for* T_j *periods when it is punished, and* $(U^{(j)} - \varepsilon)$ *after it conforms to the cooperative strategy. The average discounted deviation utility can be expressed as:*

$$\hat{U}_{\infty}^{(j)} = \overline{v}_j + \frac{\delta(1-\delta^{T_j})}{1-\delta}\underline{v}_j + \frac{\delta^{T_j+1}}{1-\delta}(U^{(j)} - \varepsilon). \tag{6.12}$$

Since if the node conforms throughout the game, it has the average discounted utility of $\frac{1}{1-\delta}U^{(j)}$. *So the gain of deviation is given by:*

$$\Delta U^{(j)} = \hat{U}_{\infty}^{(j)} - \frac{1}{1-\delta}U^{(j)} < \overline{v}_j + \frac{\delta(1-\delta^{T_j})}{1-\delta}\underline{v}_j - \frac{1-\delta^{T_j+1}}{1-\delta}(U^{(j)} - \varepsilon). \tag{6.13}$$

We note that $\underline{v}_j$ *coincides with the one stage NE, which is* $\underline{v}_j = 0, \forall j$. *As* $\delta \to 1$, $\frac{1-\delta^{T_j+1}}{1-\delta}$ *tends to* $1+T_j$. *Under the condition of (6.11), the deviation gain in (6.13) will be strictly less than zero. This indicates that the average cooperating utility is*

strictly *larger than deviation utility. Hence, rational nodes will not deviate from the cooperation point.*

If the punished node still deviates in the punishment period, the punishment period (Condition II) restarts and the punishment duration experienced by the punished node is lengthened. As the result, deviation in the punishment period postpones the punished node from receiving the strictly better utility $(U^{(j)} - \varepsilon)$ *in Condition III. Hence, it is better not to deviate in the punishment stage.*

On the other hand, if punishing node i *does not play punishing strategy during the punishment of node* j*, node* i *receives at most*

$$\hat{U}_{\infty}^{(i)} = \overline{v}_i + \frac{\delta(1-\delta^T)}{1-\delta}\underline{v}_i + \frac{\delta^{T+1}}{1-\delta}(U^{(i)} - \varepsilon). \tag{6.14}$$

However, if node i *conforms with the punishment strategy, it will receive at least*

$$\tilde{U}_{\infty}^{(i)} = \frac{(1-\delta^T)}{1-\delta}w_i^j + \frac{\delta^{T+1}}{1-\delta}U^{(i)}. \tag{6.15}$$

Here w_i^j *is the utility of node* i *to punish node* j*. Therefore, the node* i*'s reward for carrying out the punishment is (6.15) minus (6.14),*

$$\tilde{U}_{\infty}^{(i)} - \hat{U}_{\infty}^{(i)} = \frac{(1-\delta^T)}{1-\delta}(w_i^j - \delta\underline{v}_i) - \overline{v}_i + \frac{\delta^{T+1}\varepsilon}{1-\delta}. \tag{6.16}$$

Using $\underline{v}_i = 0, \forall i$ *and let* $\delta \to 1$*, the expression (6.16) is equivalent to*

$$\tilde{U}_{\infty}^{(i)} - \hat{U}_{\infty}^{(i)} = T \cdot w_i^j - \overline{v}_i + \frac{\varepsilon}{1-\delta}. \tag{6.17}$$

By selecting δ *close to one, this expression can be always larger than zero. In summary, punishing node is always conforming to the punishment strategy in the punishment stage.*

The same argument of no node deviating in Condition I can be used to show that no nodes deviates in Condition III. Therefore, we conclude that deviations in all Conditions are not profitable.

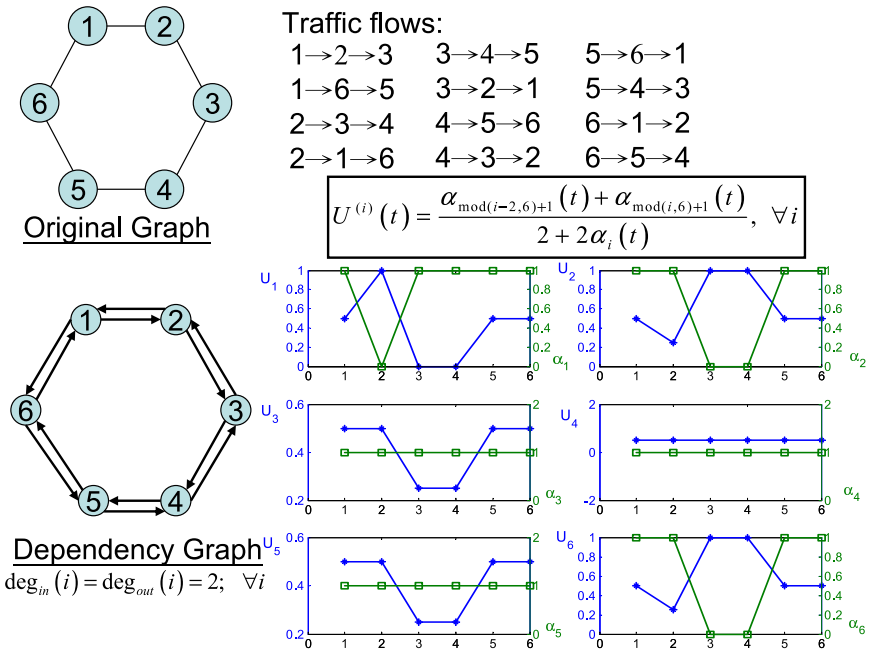


Figure 6.1: Example of punishment scheme under perfect observability

The proof above is based on two conditions: First, nodes are able to identify which node is defecting and which node does not carry out the punishment. This is guaranteed by the perfect observability assumption. Secondly, the proof assumes that there always exist nodes that can punish the deviating nodes, this is guaranteed by the assumption $\deg_{out}(i) > 0$ in the corresponding dependency graph.

Now let's consider the following example to understand the punishment behavior. In this example, we assume $\mu_{S(r)} = 1$, $K = 1$, and $d(i,j) = 1$, the resulting utilities are shown in Figure 6.1. Each node has the one stage utility given by:

$$U^{(i)} = \frac{\alpha_{\mathrm{mod}(i-2,6)+1} + \alpha_{\mathrm{mod}(i,6)+1}}{2 + 2\alpha_i}. \tag{6.18}$$

By selecting the discounted factor, $\delta = 0.9$ and $T = 2$ appropriately, all nodes are better-off when they are cooperating in packet forwarding by setting $\alpha_i = 1, \forall i$. If all nodes conform to the cooperative strategies, the 6-stage average discounted

utilities defined in (6.9) are given by $\bar{U}_6^{(i)} = 0.2343,\ \forall i$. In Figure 6.1, we plot the utility functions and forwarding probabilities of all nodes. The x-axis of the plot denotes the round of game, the left y-axis denotes the value of node's utility, and the right y-axis denotes the value of forwarding probability. The forwarding probability is denoted by the squared plot and the utility function is denoted by the plot with stars. In the figure, we show that node 1 is deviating in second round game by setting its forwarding probability to zero. At this time, node 1's utility changes from 0.5 to 1 as seen in the figure. As the consequence, node 2 and node 6 are punishing node 1 for the following $T = 2$ stages by setting their forwarding probabilities to zeros. In the third round of the game, node 1 has to return to cooperation; otherwise, the punishment from others restarts and consequently the average discounted utility will be further lowered. After the punishment, all nodes come back to the cooperative forwarding probabilities (as shown in the figure). The resulting 6-stage average utilities are as follows $\bar{U}_6^{(1)} = 0.2023$, $\bar{U}_6^{(2)} = \bar{U}_6^{(6)} = 0.2887$, $\bar{U}_6^{(3)} = \bar{U}_6^{(5)} = 0.1958$ and $\bar{U}_6^{(4)} = 0.2343$. So node 1 has less utility by deviation than by cooperation. Moreover, if both node 2 and node 6 fail to punish node 1, they will be punished by other nodes for the following T periods of game. And the resulting average utilities are $\bar{U}_6^{(1)} = 0.3485$, $\bar{U}_6^{(2)} = \bar{U}_6^{(6)} = 0.1425$, $\bar{U}_6^{(3)} = \bar{U}_6^{(5)} = 0.3035$ and $\bar{U}_6^{(4)} = 0.165$. Therefore, node 2 and 6 will carry out the punishment, since otherwise, they will in turn be punished and have less utility. We note that in this example the corresponding dependency graph has $deg_{in}(i) = deg_{out}(i) = 2, \forall i$. Therefore, there are always punishing nodes whenever any node deviates.

6.3.2 Design of Punishment Scheme under Imperfect Local Observability

We have shown that under the perfect observability assumption, the packet forwarding game along with the punishment scheme can achieve any Pareto-dominant efficiency. However, the perfect observability may be difficult to implement in ad-hoc networks, due to the enormous overheads and signallings. Therefore, we try to relax the condition of the perfect observability in this subsection. There are many difficulties in removing the perfect observability assumption. Suppose each node observes only its own history of stage utility function. In this situation, the node knows nothing about what has been going on in the rest of the network. The node only knows the deviation of nodes on which it relies on to do packet forwarding. And it cannot detect the deviation in other part of the network, even though it may be the one that can punish the deviating node. Therefore, it is impossible to implement the Folk Theorem (Theorem 7) in this information limited situation. Moreover, nodes may not know if the system is in punishment stage or not. As soon as one of the nodes sees the deviation, it starts the punishment period. This will quickly start another punishment stage by other nodes, since the nodes cannot differentiate if the stage efficiency change is caused by the punishment stage or the deviating node. As the result, the defection spreads like an epidemic and cooperation in the whole network breaks down. This is known as the *contagious equilibrium* [55]. Indeed, the only equilibrium in this situation is the one stage NE.

Nevertheless, from the game theory literatures [55], the cooperating equilibrium sometimes can be sustained based on the following argument. All nodes know whenever a defection is detected, the outcome will be the contagious equilibrium; therefore, it is better off for nodes to conform with the cooperating point.

However, this equilibrium is vulnerable to noise (such as observation errors for packet forwarding probability and deviation detection), in the sense that small amount of noise causes the cooperation breakdown in the whole network. Reasons to this vulnerability are that all the nodes have the *inconsistent* beliefs about the state of the system, they do not know whether the system is currently in the punishment state, the deviation state, or the end of punishment stage. Therefore, any mistake in invoking the punishment stage causes the contagious equilibrium.

The lack of the consistent knowledge of the system state can be mitigated using communications between nodes. Suppose each node observes only a subset of the other nodes' behavior. The communication is introduced by assuming that each node makes a public announcement about the behavior of the nodes it observes. This public announcement can be implemented by having the nodes exchange the behavior of nodes they observe. The intersection of these reports can be utilized to identify the deviating node. At the end of each stage game, the nodes report either no nodes deviate or the identity of the deviating node. The communicated reports can be used to obtain the stable equilibrium of the game. Since these reports can be exchanged in a relatively low frequency and only to the related nodes, the communication overheads are limited. Under this local observability assumption, the following theorem inspired by the Folk Theorem for private monitoring with communication [14] is proposed

Theorem 8 *Suppose* $\mathbf{V}^{\dagger}$ *has* N *dimensionality, where* N *is the number of nodes in the network. And if every node is monitored by at least two other nodes. This implies that the corresponding dependency graph satisfies the following condition*

$$deg_{in}(i) \geq 2, \ \forall i. \tag{6.19}$$

And also, there always exists a node that can punish the deviating node, i.e.,

$$deg_{out}(i) > 0, \ \forall i. \tag{6.20}$$

Moreover, the monitoring nodes can exchange the observations. Then, for every v in the interior of $\mathbf{V}^{\dagger}$, there exist $\underline{\delta} \in (0,1)$, such that for all $\delta \in (\underline{\delta}, 1)$, $v = (v_1, \cdots, v_N)$ is an equilibrium of an infinitely repeated game in which the node i's average utility is v_i.

Proof 10 *Suppose there exist ε, δ and punishment period T_i such that (6.11) holds and*

$$\sum_{t=0}^{\max_i\{T_i\}-1} \delta^t \max_i\{\max_{(\alpha,\alpha')} \big(v_i(\alpha) - v_i(\alpha')\big)\} < \sum_{t=\max_i\{T_i\}}^{\infty} \delta^t \varepsilon, \tag{6.21}$$

then the following rule of the game (Condition I to III) achieves the equilibrium when $deg_{in}(i) = 2, \ \forall i$.

Condition I: *If there is no announcement of the deviating nodes*

a. *If previous stage is in cooperating state, continue the cooperating state.*

b. *If nodes play the following strategy in the previous stage*

$$\big(U^{(1)}, \cdots, U^{(k-1)}, U^{(k)} - \varepsilon, U^{(k+1)}, \cdots, U^{(N)}\big)$$

for $k \in \{1, \cdots, N\}$, continue the previous state.

c. *If the previous stage is in punishing node k state and the punishment has not ended, then continue the punishing; otherwise, switch to strategy that results in*

$$\big(U^{(1)}, \cdots, U^{(k-1)}, U^{(k)} - \varepsilon, U^{(k+1)}, \cdots, U^{(N)}\big).$$

Condition II: *If node j is incriminated by both of its monitors j_1 and j_2*

a. *If previous stage's strategy is either in the following states: punishing node j state, in implementing $\left(U^{(1)}, \cdots, U^{(j-1)}, U^{(j)}-\varepsilon, U^{(j+1)}, \cdots, U^{(N)}\right)$, in implementing $\left(U^{(1)}, \cdots, U^{(j)}-\varepsilon, \cdots, U^{(l)}-\varepsilon, \cdots, U^{(N)}\right)$, for some $l \neq j$, or in implementing $\left(U^{(1)}, \cdots, U^{(l)}+\varepsilon, \cdots, U^{(j)}-\varepsilon, \cdots, U^{(N)}\right)$, for some $l \neq j$, then start the punishment state for punishing node j.*

b. *If previous stage's strategy is in punishing node j_1 state, then switch to strategy that results in $\left(U^{(1)}, \cdots, U^{(j_2)}+\varepsilon, \cdots, U^{(j)}-\varepsilon, \cdots, U^{(N)}\right)$. The similar argument is applied to increase node j_1's utility by ε when node j_2 is punished in the previous stage.*

Condition III: *If there is any inconsistent announcement by node j_1 and j_2. We note that the inconsistent announcement happens when there are at least two announcements of deviation node, but the deviation nodes in the announcements are different.*

a. *If the previous state is punishing node j_1 or node j_2, then restart the punishment state.*

b. *Otherwise, implement $\left(U^{(1)}, \cdots, U^{(j_1)}-\varepsilon, \cdots, U^{(j_2)}-\varepsilon, \cdots, U^{(N)}\right)$.*

In the above rules, we consider three different conditions, namely when no announcement of deviating node (Condition I), when the announcements are consistent (Condition II), and when the announcements are inconsistent (Condition III). Then we discuss the different strategies for different states within each Condition. We note that only the nodes whose packets are forwarded by node j have the potential ability of detecting the deviation of node j. The condition of $deg_{in} \geq 2$ on the corresponding dependency graph in Theorem 8 implies that there exist at least 2 flows that require node j to forward. And any deviation of node j can be

potentially detected by at least 2 sources. Similar to Theorem 1, $deg_{out}(i) > 0, \forall i$ implies that there always exists a node for punishing the deviating node.

If both the monitors of node j incriminate it, then node j is punished in a way similar to the punishment in Theorem 7. The deviator is punished for some period of time if the previous state is in one of the following states: punishing node i (this implies that the punishment stage will be restarted), finished punishing node i (i.e. in state $U^{(1)}, \cdots, U^{(j-1)}, U^{(j)} - \varepsilon, U^{(j+1)}, \cdots, U^{(N)}$), after penalizing nodes that make inconsistent announcements (i.e. in state $U^{(1)}, \cdots, U^{(j)} - \varepsilon, \cdots, U^{(l)} - \varepsilon, \cdots, U^{(N)}$), or in state $U^{(1)}, \cdots, U^{(l)} + \varepsilon, \cdots, U^{(j)} - \varepsilon, \cdots, U^{(N)}$. In all these states, the deviator will be punished for a certain period of time (Condition IIa). However, if the previous state is in punishing node j_1, then the system switches to strategy that results in $U^{(1)}, \cdots, U^{(j_2)} + \varepsilon, \cdots, U^{(j)} - \varepsilon, \cdots, U^{(N)}$ (Condition IIb). This strategy gives additional incentives ($U^{(j_2)} + \varepsilon$) for node j_2 to punish to node j. The condition IIb is used to avoid the situation where node j_2 lies on its announcement even though it observes that node j is deviating. This condition will become obvious as we discuss the condition III.

When there are incompatible announcements about node j (Condition III), nodes that make incompatible announcements will be penalized and they will receive utility $U^{(j_i)} - \varepsilon$ for $i = 1, 2$ (Condition IIIb). This strategy is sufficient to avoid lying in announcement, except in the case when node j_1 is being punished in previous stage. The Condition IIIa prevents node j_1 from falsely accusing node j. However, including the Condition IIIa creates the situation where node j_2 enjoy punishing node j_1. This means that when node j_1 is being punished and in the case node j has really deviated, node j_2 has the incentive to lie in its announcement and announces that no nodes is deviating. This problem is solved by Condition IIb that

gives additional reward for node j_2 to tell the truth and punish node j. Moreover (6.21) implies that this additional reward for node j_2 outweighs the benefit from punishing node j_1. (6.21) can be thought as incentives for the monitoring nodes to punish the deviating node when the announcements are inconsistent. Using the rule of game as in (Condition I to III), the infinite average discounted utility close to any $v = (v_1, \cdots, v_N)$ can be enforced.

In the general case when $deg_{in}(i) \geq 2$, $\forall i$, the only difference is for Condition III where there is inconsistence in the announcements. Under this condition, a similar approach to Condition IIIa and Condition IIIb can be obtained for more than two monitors.

Based on different information structures, Section 6.3.1 and Section 6.3.2 guarantee that any individually rational utilities can be enforced under some conditions. However, the individual distributed nodes need to know how to cooperate, i.e. what is the good packet forwarding probability. In the next section, we describe learning algorithm to achieve better utilities.

6.4 Self-Learning Algorithms

From Section 6.3, any Pareto dominant solutions better than one stage NE can be sustained. However, the analysis does not explicitly determine which cooperation point to be sustained. In fact, the system can be optimized to different cooperating points, depending on the system designer choices. For instance, the system can be designed to maximize the sum of the average infinitely repeated game's utilities as follow

$$\overline{U}_{sys} = \frac{1}{N} \sum_{i=1}^{N} \overline{U}_{\infty}^{(i)}. \tag{6.22}$$

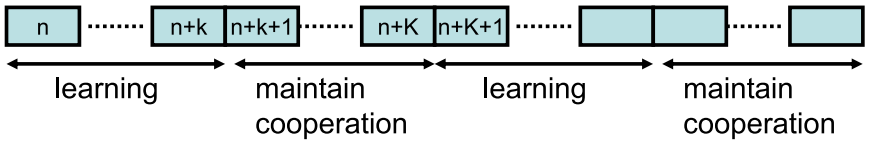


Figure 6.2: Time slotted transmission

The basic idea of the learning algorithm is to search iteratively the correct cooperating forwarding probability. Similar to the punishment design, we consider the learning schemes for different information availability; namely, the perfect observability and the local observability. We use (6.22) as an example, but we emphasize that any system objective function can be incorporated into the learning algorithm in a similar way.

The implementation of the learning algorithm and the repeated game maintaining the cooperation point can be illustrated as in Figure 6.2. In particular, we consider the time-slotted transmission that interleaves the learning mode and the cooperation maintenance mode. In the learning mode, the nodes search for better cooperating point. In the cooperation maintenance mode, nodes monitor the actions of other nodes and apply punishment if there is any deviation. In the learning mode, the nodes have no incentives to deviate since they do not know if they can get benefits. So they do not want to miss the chance of obtaining the better average discounted utilities in the learning mode. It is also worth mentioning that if a node deviates just before a learning period, it will still be punished in the following maintain cooperation period. So the infinite repeated game assumption is still valid in this time slotted transmission system.

Table 6.1: Self Learning Repeated Game Algorithm under Perfect Observability

For node i: Given $\vec{\alpha}_{-i}$, small increment β and minimum forwarding probability α_{min}
Iteration: $t = 1, 2, \cdots$
Calculate $\nabla \overline{U}_{sys}(\vec{\alpha}(t-1))$
Calculate $\vec{\alpha}(t) = \vec{\alpha}(t-1) - \beta \nabla \overline{U}_{sys}(\vec{\alpha}(t-1))$
Select $\alpha_i(t) = \min\{\max\{[\vec{\alpha}(t)]_i, \alpha_{min}\}, 1\}$

6.4.1 Self-learning under the perfect observability

Under the perfect observability information structure, every node is able to detect the deviation of the defecting node. And nodes observe which nodes help forwarding others' packets. This implies that every node is able to perfectly predict the average efficiencies of other nodes and optimize the cooperating point based on the system criterion (6.22). The basic idea of the learning algorithm is to use the steepest descent like iterations. All nodes predict the average efficiencies of others and the corresponding gradients. The detailed algorithm is listed as in Table 6.1. Learning with perfect observability assumes the perfect knowledge of utility functions of all nodes in the network, and represents the best solution that any learning algorithm can achieve.

Under perfect observability assumption, node does not want to defect in the learning stage for two reasons; first, they will not risk losing the opportunity to obtain better cooperation strategy. Secondly, since nodes know the utility functions of other nodes and the history of actions along the game, deviation in learning may also induce punishment in the next cooperation maintenance stage.

6.4.2 Self-learning under the local observability

In this subsection, we focus on the learning algorithm with the information structure available under local observability. Under this condition, the nodes may not have the complete information about the exact utility of others. Based on this information structure, we develop two learning algorithms; the first algorithm is called *learning through flooding*. The second algorithm makes prediction of the other nodes' stage efficiency based on flows that go through the predicting node. We called the second algorithm as *learning through utility prediction*.

Learning through Flooding

In this local observability assumption, we develop heuristics to learn the better cooperating point. The basic idea of the learning algorithm is as follow. Since the only information the node can observe is the effect of changing its forwarding probability onto its own utility function. The best way for the nodes to learn the packet forwarding probability is to gradually increase the probability and monitor if the utility function becomes better. If the utility becomes better, the new forwarding probability will be employed. Otherwise, the old forwarding probability will be kept. The algorithm lets all nodes change their packet forwarding probabilities simultaneously. This can be done by flooding the instruction for changing the packet forwarding probability. After changing the packet forwarding probability, the effect propagates throughout the network. All nodes wait for a period of time until the network becomes stable. At the end of this period, the nodes obtain their new utilities. If the utilities are better than the original ones, then the new packet forwarding probabilities are employed. Otherwise, the old ones are kept. We note that the packet forwarding probability increment is proportional

Table 6.2: Self Learning Repeated Game Algorithm (Flooding)

Initialization: $t = 0$
$\alpha_i^t = \alpha_{min}, \forall i$. Choose small increment ξ, η.
Iteration: $t = 1, 2, \cdots$
Calculate $U^{(i),t-1}(\alpha_i^{t-1})$ and $U^{(i),t-1}(\alpha_i^{t-1} + \xi)$,
Calculate $\Delta U^{(i),t-1} = U^{(i),t-1}(\alpha_i^{t-1} + \xi) - U^{(i),t-1}(\alpha_i^{t-1})$,
For each i such that $\Delta U^{(i),t-1} > 0$,
$\alpha_i^t = \alpha_i^{t-1} + \eta \frac{\Delta U^{(i),t-1}}{U^{(i),t-1}(\alpha_i^{t-1})}$,
$\alpha_i^t = \max(\min(\alpha_i^t, 1), \alpha_{min})$.
End when: No improvement and keep monitoring
the deviation start punishment scheme if there
is a deviation

to the increase in the utility function; nodes with higher increment in their utility functions increase their forwarding probability more compared to nodes with lower utility increment. Here, we introduce the normalization factor $U^{(i),t-1}(\alpha_i^{t-1})$ (the utility before changing the forwarding probability) in order to keep the updates in forwarding probability bounded. The forwarding probability increment depends on small increment constant η and the normalization factor. The above process is performed until no improvement can be made. The detailed algorithm is shown in Table 6.2.

Learning with Utility Prediction

In this second approach, we observe that some of the routing information can be used to learn the system optimal solution (6.22). We assume that the routing

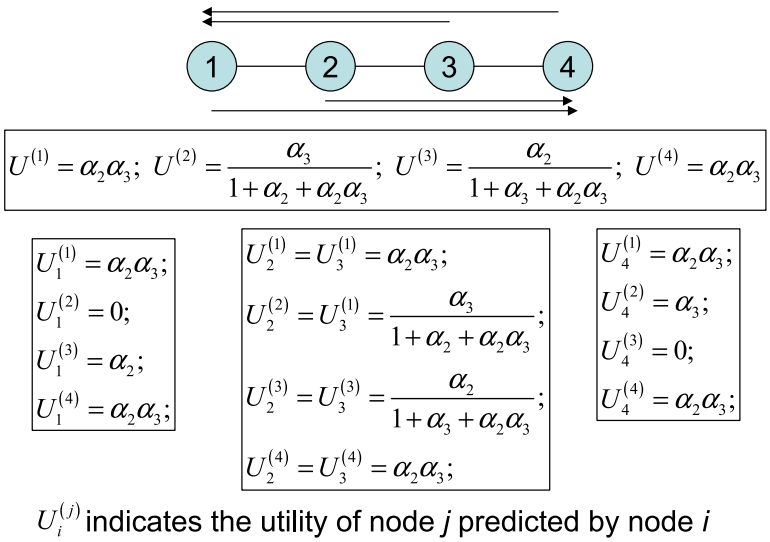


Figure 6.3: Example for learning with utility prediction

decision has been made before performing the packet forwarding task. For instance, in the route discovery using Dynamic Source Routing (DSR) [54] algorithm without route caching, the entire selected route is included in the packet header in the packet transmission. The intermediate nodes use the route (in packet header) to determine to whom the packet should be forwarded. Therefore, it is clear that the transmitting node knows where the packet goes through; the relaying nodes know where the packet comes from and heads to; and the receiving node knows where the packet comes from. The nodes use this information to predict the utilities of others' nodes. We note that because not all the nodes are involved in all of the flows in the network, the utility prediction may not be very accurate.

The utility prediction is illustrated using an example shown in Figure 6.3, assuming $\mu_{S(r)} = 1$, $K = 1$, and $d(i, j) = 1$. We denote $U_i^{(j)}$ as the utility of node j predicted by node i. From the figure, the node 1 receives flows from node 3 and 4 and node 4 receives flows from node 1 and 2. It is obvious that the flow from 2 to 4 is not perceived by node 1. Hence, the utilities of node 2 and 3 predicted by node 1

Table 6.3: Self Learning Repeated Game Algorithm with prediction

Initialization: $t = 0$
$\alpha_j^{(i),t} = \alpha_{min}, \forall i, j$. Choose small increment ζ.
Iteration: $t = 1, 2, \cdots$
For each node $j = 1, \cdots, N$
Calculate $[\nabla_j^{(1)}, \cdots, \nabla_j^{(N)}] = \left[\frac{\partial \sum_{n=1}^{N} U_j^{(n)}}{N \partial \hat{\alpha}_j^{(1),t}}, \cdots, \frac{\partial \sum_{n=1}^{N} U_j^{(n)}}{N \partial \hat{\alpha}_j^{(N),t}}\right]$
Calculate $\alpha_j^{(i),t} = \alpha_j^{(i),t-1} + \zeta \nabla_j^{(i)}$
Set $\alpha_j^{(i),t} = \max(\min(\alpha_j^{(i),t}, 1), \alpha_{min})$.
End when: No improvement and return $\alpha_j^{(i)} = \alpha_j^{(i),t}, \forall i, j$.
Keep monitoring the deviation, go to punishment
scheme whenever there is a deviation.

are not the accurate ones. Similarly, flow from node 3 to node 1 is not perceived by node 4, therefore, $U_4^{(2)}$ and $U_4^{(3)}$ are not accurate. The accuracy of the prediction depends on the flows. If all the flows involving node i pass through j then $U_j^{(i)}$ will be the accurate one and vice versa as is illustrated in Figure 6.3. However, as we show by simulations the inaccuracy in the prediction does not affect the result of optimization too much.

Since the objective of the optimization is to achieve the system optimal solution (6.22), the best node i can do is to find the solution that minimizes the total average predicted utility function, which is

$$\begin{aligned} \min \quad & \tfrac{1}{N} \textstyle\sum_{j=1}^{N} U_i^{(j)}(\hat{\alpha}_i^{(1)}, \cdots, \hat{\alpha}_i^{(N)}) \\ \text{s.t.} \quad & \alpha_{min} \leq \hat{\alpha}_i^{(j)} \leq 1, \forall j. \end{aligned} \tag{6.23}$$

where $\hat{\alpha}_i^{(j)}$ is the packet forwarding probability that node j should employ as predicted by node i. The detailed of the algorithm is presented as in Table 6.3.

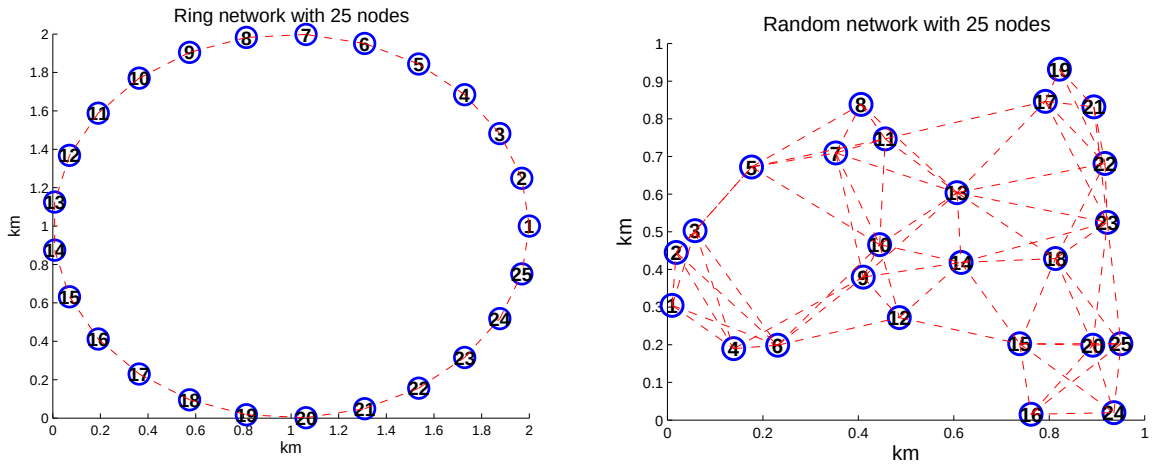


Figure 6.4: (a) Ring-25 network (b) Random-25 network

The algorithm in Table 6.3 imitates the steepest descent algorithm based on the predicted utility, every node finds the gradient of the predicted utility and optimizes the predicted system utility (6.23). After obtaining $\{\hat{\alpha}^{(i)}\}$, each node sets its own packet forwarding probability as $\alpha_i^t = \hat{\alpha}_i^{(i)}$. We note that the optimization problem (6.23) can be done in a distributed manner, since the optimization does not require the global knowledge of the utility function. Each node does the optimization based on its own prediction and sets its packet forwarding probability according to the optimized predicted average utility.

6.5 Simulation Results

To investigate the effectiveness of our proposed framework, we perform simulations with the following settings. We generate two networks with 25 nodes, the ring-25 network (Figure 6.4 (a)) and random-25 network (Figure 6.4 (b)). The ring-25 network consists of 25 nodes that are arranged in a circle with radius $1000m$. The random-25 network consists of 25 nodes that are uniformly distributed in the area

of $1000m \times 1000m$. We define the maximum distance d_{max}, such that two nodes are connected if the distance between two nodes is less than d_{max}. We select the maximum distance between two nodes to ensure the connectivity of the whole network. In the ring-N network, the angle separation between two neighboring nodes is $\frac{2\pi}{N}$. And, the distance between two neighboring nodes is $2r\sin(\frac{2\pi}{2N})$, where r is the radius of the circle. In particular, the maximum distance for the ring-25 network can be calculated as $2000\sin\left(\frac{2\pi}{50}\right)m = 250.7m$. In the random-25 network, the maximum distance between two nodes is chosen as $350m$ to ensure the connectivity of the whole network with high probability.

We also define the flows as source-destination (SD) pairs. We assume that the routing decision has been made before performing packet forwarding optimization. The shortest path routing is employed in the simulations. In random-25 network, we vary the number of SD pairs. When there are traffic flows from all nodes to all other nodes, we called this traffic as dense flow. This implies that each node has packets destined to the rest of nodes in the network. Obviously, the dense flow has $N \times (N-1)$ SD pairs in the N-node network. When the total flow is less than the dense flow, the SD pairs are determined randomly. In the ring-25 network, the number of SD pairs is defined in the following way. The $(K \cdot N)$ SD pairs are obtained when every node i sends packets to nodes $(\{mod(i+2, 25), \cdots, mod(i+K+1, 25)\}$. For instance, 25 SD pairs are obtained when every node i transmits packets to node $mod(i+2, 25)$, 50 SD pairs are obtained when every node i sends packets to nodes $\{mod(i+2, 25), mod(i+3, 25)\}$, etc. The rest of the simulation parameters are given as follows, transmission rate of source i as $\mu_i = 1, \ \forall i$, transmission constant $K = 1$, distant attenuation coefficient $\gamma = 4$. We compare three learning algorithms according to the information availability. The

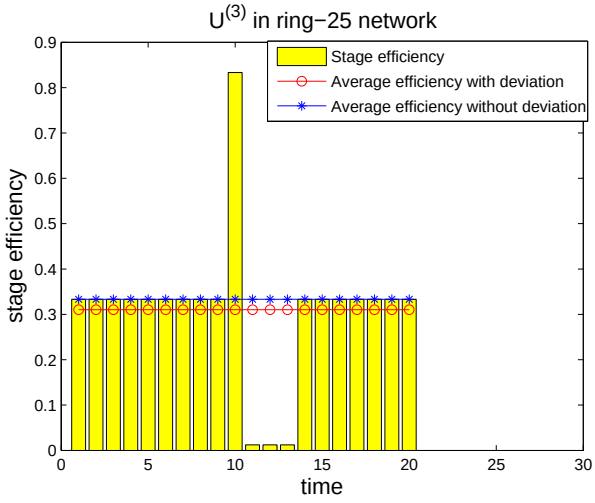


Figure 6.5: Punishment of Repeated Game in Ring Network

parameters for the learning algorithms are listed as follows $\beta = 0.05$, $\xi = 0.001$, $\eta = 1.0$, and $\zeta = 0.05$. The minimum forwarding probability is set to be $\alpha_{min} = 0.1$ and the maximum forwarding probability is set to be $\alpha_{max} = 1$. Finally, all the algorithms are initiated with $\alpha_i = \alpha_{min}, \forall i$.

Figure 6.5 shows the average efficiency of the deviation node in the ring-25 network when number of source-destination is 75 with the discounted factor $\delta = 0.9$. In the figure, the node 3 deviates at time instant 10. This deviation causes the stage efficiencies of node 1, 2 and 25 become lower. From the route, the node 1, 2 and 25 suspect that nodes in $\{2, 3, 4\}$, $\{3, 4, 5\}$ and $\{1, 2, 3\}$ are deviating, respectively. The nodes in the network know that node 3 is consistent to be incriminated for deviation and start the punishment stage (Here, the punishment period is set to 3). The punishment scheme results in lower average stage efficiency as described in Figure 6.5. From the figure, the average efficiency without deviation is better than the average efficiency with deviation. It is clear that it is better off for node 3 to conform to the previously agreed cooperation point. And no node wants

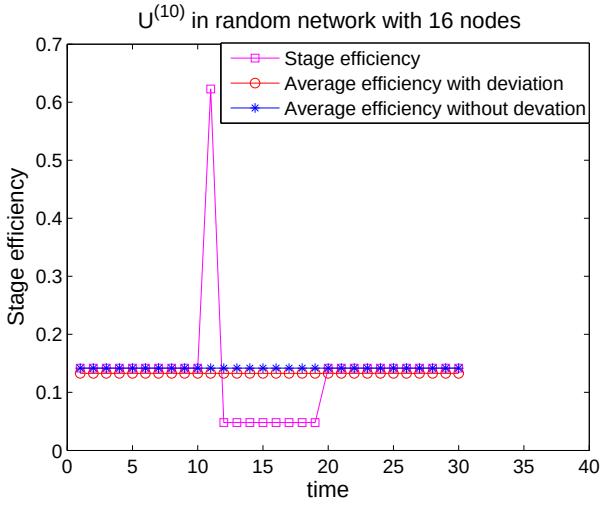


Figure 6.6: Punishment of Repeated Game in Random Network

to deviate, since deviation results in worse average efficiency. Similarly, Figure 6.6 shows the average utilities of deviating node and other nodes in random network with 16 nodes with the discounted factor 0.9. In time instant 11, node 10 in the network deviates. At the next time instant, all related nodes that detect deviation exchange the list of incriminated nodes. The consistent incriminated node (in this case node 10) is punished for some period of time (in this figure, 8 period of time). From the figure, it is clear that node 10 will have higher average efficiency when it conforms. So from Figure 6.5 and 6.6, the proposed repeated game can enforce the cooperation among autonomous greedy nodes in the networks.

Figure 6.7 and 6.8 show the learning curves for the proposed self-learning repeated game scheme for the ring-25 and random-25 network, respectively. In the figures, we compare the optimum solution, learning with perfect observability, learning with flooding, and learning with utility prediction. In Figure 6.7, all of the algorithms achieve the system optimal value when the source-destination pairs are 100, 200, and 275. From this figure, the learning with perfect observability

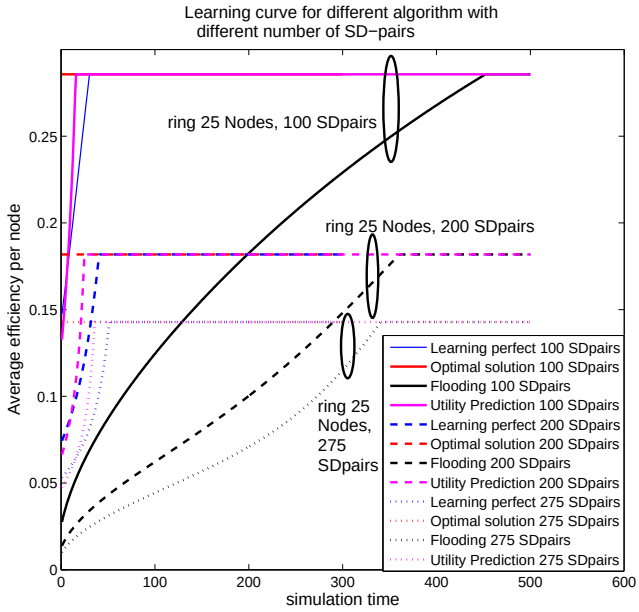


Figure 6.7: Learned average efficiency per node for different traffic loads in ring network

and utility prediction have approximately the same convergence speed. The learning with flooding converges slower, this can be understood since the learning with flooding does the trial-and-error to find the correct forwarding probability. This unguided optimization although requires minimal information has inferior convergence speed. Figure 6.8 shows the learning curves of the proposed algorithms for random-25 network with different source-destination pairs. One can observe that the learning with utility prediction achieves very close efficiency per node compared to the optimum solution and learning with perfect observation case. In contrast, the learning with flooding achieves inferior efficiency per node.

Figure 6.9 shows the learned average efficiency per node for various algorithms with different traffic flows in the ring-25 network. The efficiency becomes lower as the number of source-destination pairs become larger. This can be explained as

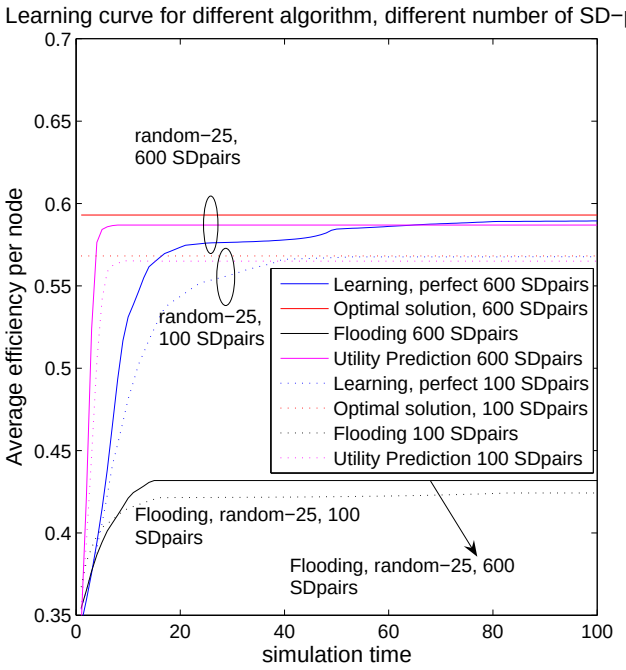


Figure 6.8: Learned average efficiency per node for different traffic loads in random network

follows. Because of the symmetric property of the utility functions, the local optimal forwarding probabilities for all the nodes are the same. It can be easily shown that the local optimal forwarding probabilities in the ring-25 network is 1 for all the nodes[2]. Therefore, the larger the number of source-destination pairs, the more packets a node needs to forward and the higher value of the denominator of the stage utility function (6.7). As the result, the average efficiency per node decreases as the number of source-destination increases. Using simple calculation, it can be shown that the average efficiency per node decays as $\frac{N_{sd}/N}{(N_{sd}/N+0.5*(N_{sd}/N+1)*(N_{sd}/N)}$, where N_{sd} is the number of source-destination pairs. In the figures, all the learning algorithms perform similarly for different numbers of source-destination pairs.

[2]This is not true in random network in general.

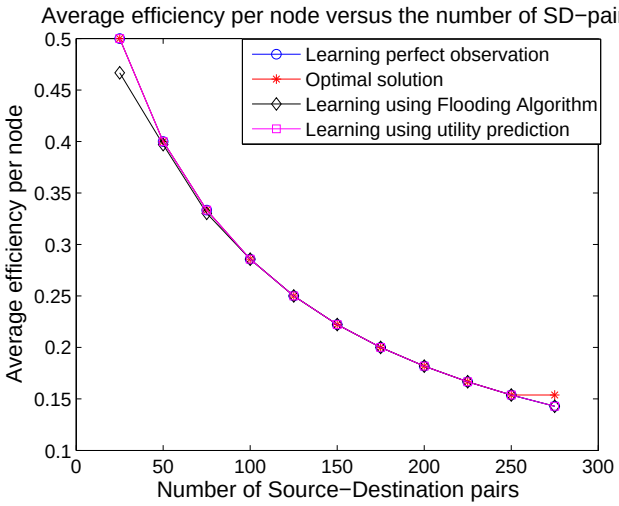


Figure 6.9: Average efficiency per node for different traffic loads in ring network

Figure 6.10 shows the achievable efficiency per node after the learning algorithms converge for different numbers of source-destination pairs in the random-25 network. Similarly, we observe that the learning with utility prediction achieves very close efficiency compared to the learning with perfect observation and optimum solution for broad source-destination pairs. The learning with flooding achieves lower efficiency per node, but it achieves much better efficiency compared to the Nash Equilibrium of the stage game. In average, the learning with utility prediction achieves around 99.2% of the efficiency achieved by the optimal solution. In contrast, the learning with flooding achieves more than 73.18% of the optimality.

Comparing Figure 6.9 and 6.10, we see that the learning with flooding performs well in the ring-25 network but inferior in random-25 network. The reason for this phenomenon is that in the ring-25 network, the utilities of every node are symmetric and optimizing the system criterion (6.22) results in the same average efficiency in each node. Since the learning with flooding tries to increase its node's

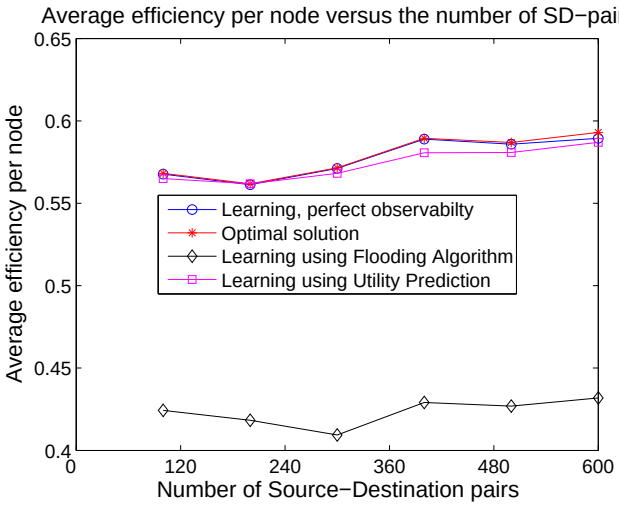


Figure 6.10: Average efficiency per node for different traffic loads in random network

efficiency by changing its own forwarding probability synchronously, this iteration will finally reach the point where all nodes' efficiencies are the same due to the symmetric structure of the network. This solution is coincidentally the same as the solution of the system criterion (6.22) optimization. In contrast to the ring-25 network, the utility functions for each nodes are highly asymmetry in the random-25 network. In this case, the node that firstly reaches a better solution will not change its forwarding probability, even though changing its forwarding probability results in slightly lower efficiency in that particular node but increases the other nodes' efficiencies quite a bit. Due to this greedy and unguided optimization, the learning with flooding achieves inferior average efficiency per node compared to the learning using utility prediction which obtains information from routing information and performs better learning.

In Figure 6.11, we investigate the performance of the learning algorithms in the dense flow with different number of nodes in random network. The maximum

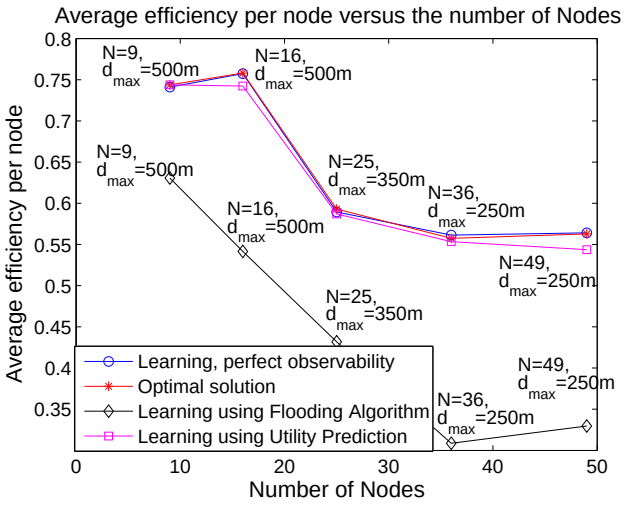


Figure 6.11: Average efficiency per node for different nodes in random network with dense traffic

distance and number of nodes are highlighted in the figure. Figure 6.11 shows a snapshot of average efficiency per node for different sizes of the network. In the figure, we observe that as the number of nodes increases, the number of average efficiency per node decreases. This is because the total power required for self transmission and forwarding increases much faster compared to the successful self-transmission power, as the number of nodes increases. Therefore, the stage utility for each node (6.7) decreases as the number of nodes increases in the dense flow. As the result, the average efficiency per node decreases as the node increases. We also observe that the learning with utility prediction achieves around 98.55% of that average efficiency per node achieved by optimum solution and learning with perfect observability. On the other hand, the learning with flooding achieves around 70% of the average efficiency obtained by optimal solution.

Chapter 7

Channel-Aware Priority Transmission

This chapter[1] presents an application of cross layer optimization between the application layer and physical layer. In particular, we propose a data loading technique that jointly considers the effect of channel estimation and the property of encoded multimedia data in Orthogonal Frequency Division Multiplexing (OFDM) systems. We observe that OFDM subchannels experience different average bit error rate (BER) due to channel estimation inaccuracy. The leakage effect in FFT based channel estimation method or the model mismatch in polynomial based channel estimation method results in a variation on the decoded BER across different OFDM subchannels. Thus, we are motivated to design the *Priority Transmission* (PT) scheme that utilizes this BER variation across different OFDM subchannels and provides unequal error protection (UEP) for multimedia transmission. In addition, since OFDM has been adopted in many multimedia transmission standards,

[1]Material in this chapter has been published in Transactions on Signal Processing [82]

we compare the different channel estimation techniques, which were compared only for generic data transmission before, in the context of multimedia transmission with the PT scheme. In particularly, we extend the polynomial based channel estimation that was previously designed for decision-directed scenario to pilot-symbol-assisted (PSA) channel estimation scenario. Then, we investigate the channel estimation mean square error (MSE) and BER performance of individual OFDM subchannels for both the FFT based and the polynomial based channel estimation. Furthermore, we design the PT scheme that achieves significant gain in peak-signal-to-noise ratio (PSNR) of the reconstructed images for both channel estimation methods. Finally, we compare different OFDM channel estimation techniques for multimedia transmission. It is shown that for generic data transmission, the polynomial based PSA channel estimation outperforms the FFT based method in realistic channel conditions, and both types of channel estimation have similar performance when using the proposed PT scheme for multimedia transmission.

The rest of this chapter is organized as follows. We first give the motivation of this work, introduce the system model. We then develop the polynomial based PSA channel estimation method and design the PT scheme based on the derived channel estimation MSE for individual subchannel. We continue with the design of the PT scheme for the FFT based channel estimation. Finally, we compare the Polynomial based and the FFT based channel estimation techniques for both generic data and multimedia transmission.

7.1 Motivation

Wireless multimedia services that require high data rate transmission have become a major driving force in the development of broadband wireless communi-

cations. Many high speed wireless transmission standards, such as digital audio broadcasting (DAB) [1], digital video broadcasting (DVB-T) [2], and broadband wireless LAN (IEEE 802.11a) [3], adopt Orthogonal Frequency Division Multiplexing (OFDM) modulation, which is known for its advantages of transforming frequency selective fading channels into a set of parallel flat fading subchannels and eliminating inter-symbol interference (ISI) [31, 58].

In OFDM systems, *channel estimation* is crucial for coherent demodulation and has a significant impact on overall performance [35, 65, 100]. Previous channel estimation techniques mainly concern the transmission of generic data, and focus on reducing the average estimation errors [35, 65]. Since multimedia data will contribute a large proportion of the traffic in high speed wireless communications, it is important to understand how the channel estimation can effect the multimedia transmission.

An important class of channel estimation techniques is pilot-symbol-assisted (PSA) channel estimation, which estimates OFDM channel based on a set of training symbols inserted into data streams and is suggested by many standards [2, 3]. Most PSA channel estimation schemes use Fast Fourier Transform (FFT) for reducing noise and estimate the subchannels that do not transmit training pilots, such as in [61, 64]. However, the FFT-based channel estimation suffers from the *leakage effect* when the delay paths are not separated by integer multiples of the system sampling period [35, 65, 100]. The main consequence of the leakage effect is that the OFDM subchannels experience non-uniform average estimation error. As a result, there exists a variation on decoded bit error rate (BER) across different subchannels. This BER variation is highly undesirable for generic data transmission because the worst subchannels dominate the error performance. For

the multimedia transmission, however, we can utilize the leakage effect to provide unequal error protection (UEP). In particular, we design a *Priority Transmission* (PT) scheme that loads multimedia data to OFDM subchannels according to the importance of the data and the channel estimation error with the decoding delay constraint [81,96]. The PT scheme is suitable for a variety of compressed multimedia data. In this chapter, we use SPIHT [89] encoded images to demonstrate the performance of the PT. We show that the PT scheme significantly improves the quality of the reconstructed images, compared to the schemes that do not exploit the channel estimation.

Another way to combat the leakage effect is to use polynomial based channel estimation techniques [28,102,103], which use polynomial basis functions to replace the exponential basis functions used in the FFT based methods. The polynomial based methods were originally proposed for decision-directed channel estimation schemes and do not suffer from the leakage effect [102]. In order to fully understand the effects of the PT scheme on different channel estimation methods, we develop the polynomial based PSA channel estimation, where we observe the variation of BER across subchannels. Therefore, the PT can also be applied to the polynomial based PSA methods. Moreover, we show that the FFT based method is effective in interpolating the sinusoidal like function, while the polynomial based method performs well as long as the channel varies smoothly in one interpolation window.

Previously, channel estimation techniques were compared for their average BER performance, which makes perfect sense for the data transmission, but not for multimedia. Therefore, the development of the PT scheme raises an interesting question on what channel estimation scheme is good for multimedia transmission. In this chapter, we first extend the development of polynomial based channel es-

timation in [103] to the polynomial based PSA channel estimation. We also show that the polynomial based PSA channel estimation outperforms the FFT based method for data transmission in most of the realistic channel conditions. Moreover, for multimedia transmission, the polynomial based PSA channel estimation is superior to the FFT based method when PT is not used, and both channel estimation schemes achieve similar good performance when the PT scheme is employed.

7.2 System Description

In this section, we introduce the transmission systems, channel model and the PSA channel estimation for OFDM systems. Also, we summarize the properties of the Set Partitioning in Hierarchical Trees (SPIHT) image codec to be used in our simulation.

7.2.1 OFDM System

Figure 7.1 illustrates a high level diagram of an OFDM system [102]. At the transmitter, input signals are arranged into blocks by a serial-to-parallel (S/P) converter and the data in each block are mapped into a set of complex constellation points, i.e. $\{X[0,k],\cdots,X[N-1,k]\}$. The mapped data block is often referred to as an OFDM block. Here, N is the total number of subchannels and k denotes the index of the OFDM blocks. After signal mapping, the modulation is implemented using inverse fast Fourier transform (IFFT). A cyclic prefix is then inserted to eliminate inter-symbol-interference (ISI). Finally, the modulated data block and the cyclic prefix are converted to an OFDM symbol by a parallel-to-serial (P/S) converter.

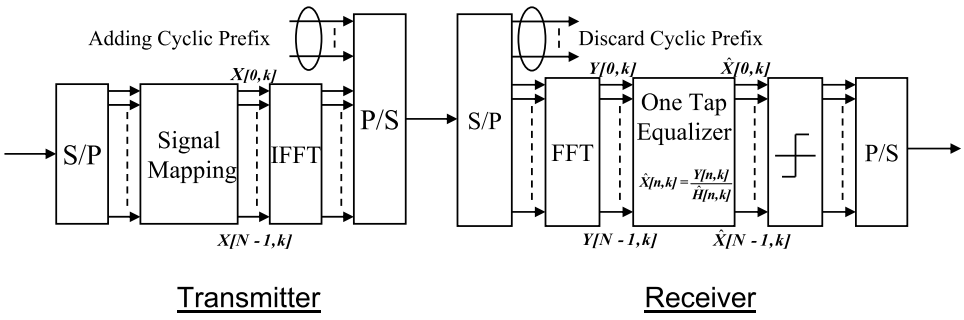


Figure 7.1: Typical OFDM Systems

At the receiver, the cyclic prefix is discarded and demodulation is performed by fast Fourier transform (FFT). When the length of the cyclic prefix is longer than the length of the channel impulse response, the interference between two consecutive OFDM symbols is eliminated. In this case, the channel can be viewed as a set of parallel independent subchannels and the received signal is represented as

$$Y[n,k] = H[n,k]X[n,k] + w[n,k], \qquad n = 0, \ldots, N-1, \tag{7.1}$$

where $Y[n,k]$ represents the received signal, $X[n,k]$ denotes the transmitted signal, $H[n,k]$ and $w[n,k]$ are the channel frequency response and the additive gaussian noise, respectively. Here, n is the index of subchannels and k is the index of OFDM blocks. The channel noise samples, $\{w[n,k]\}$, are modelled as Gaussian random variables with zero mean and variance σ^2, and are assumed to be independent for different n's or k's [65, 75, 103].

In addition, the receiver performs channel estimation and obtains the estimated channel frequency response, denoted by $\hat{H}[n,k]$. Finally, the receiver produces the estimated transmitted signal, denoted by $\hat{X}[n,k]$, using a one-tap equalizer as

$$\hat{X}[n,k] = \frac{\hat{H}^*[n,k]Y[n,k]}{|\hat{H}[n,k]|^2}. \tag{7.2}$$

7.2.2 Channel Model

In mobile wireless communication systems, signal transmission suffers from various impairments such as frequency-selective fading due to multipath delay [83]. As in [83, 95], the complex baseband representation of wireless channel impulse response is expressed as

$$h(t,\tau) = \sum_i \gamma_i(t)\delta(\tau - \tau_i), \tag{7.3}$$

where $\gamma_i(t)$ and τ_i are the gain and the delay of the i^{th} path, respectively. In Rayleigh fading, the sequence $\{\gamma_i(t)\}$ is modeled as zero-mean circular symmetric complex Gaussian random variable with variance σ_i^2, and is assumed to be independent for different paths [83, 95].

The channel frequency responses of OFDM subchannels can be approximated by the samples of the continuous channel frequency response [65], that is

$$H[n,k] \;=\; \int_{-\infty}^{\infty} h(t,\tau)e^{-j2\pi f\tau}d\tau\Big|_{f=n\Delta f,\ t=kT_f} = \sum_i \gamma_i(kT_f)e^{-j2\pi n\Delta f\tau_i}, \tag{7.4}$$

where T_f is the duration of an OFDM symbol, $\Delta f = B_d/N$ is the bandwidth of each subchannel, and B_d is the total bandwidth. This approximation does not consider the effect of the smoothing filter at the transmitter and the front-end filter at the receiver.

The correlation function of the channel frequency response is usually simplified as the multiplication of time correlation and frequency correlation [64, 65], i.e.

$$r_H[\Delta n, \Delta k] = r_f[\Delta n]r_t[\Delta k], \tag{7.5}$$

where the frequency correlation, $r_f[\Delta n]$, can be expressed as

$$r_f[\Delta n] = \sum_i \frac{\sigma_i^2}{\sum_j \sigma_j^2} e^{-j2\pi\Delta n\Delta f\tau_i}\ . \tag{7.6}$$

Based on Jakes' model [53], the time correlation, $r_t[\Delta k]$, can be expressed as

$$r_t[\Delta k] = J_0(2\pi T_f f_D \Delta k) \ , \tag{7.7}$$

where $J_0(x)$ is the zeroth-order Bessel function of the first kind, and $f_D = \frac{v f_c}{c}$ is the Doppler frequency calculated from vehicle speed v, carrier frequency f_c and the speed of light c.

7.2.3 Overview of Pilot-Symbol-Assisted (PSA) Channel Estimation

In PSA channel estimation, a set of predefined pilot symbols is inserted into the data streams to assist the channel estimation process [49,50,64,75]. Let n_p and k_p denote the locations of the subchannels and the OFDM blocks, respectively, where the pilot symbols are transmitted. The PSA channel estimation usually consists of two steps. First, the receiver estimates the channel frequency response at the pilot locations as

$$\widetilde{H}[n_p, k_p] \ = \ \frac{Y[n_p, k_p]}{X[n_p, k_p]} = H[n_p, k_p] + w'[n_p, k_p], \tag{7.8}$$

where $w'[n_p, k_p] = \frac{w[n_p, k_p]}{X[n_p, k_p]}$ is the noise term and $\tilde{H}[n_p, k_p]$ is often referred to as the *temporal estimate*. Second, the channel responses of all subchannels are calculated from the temporal estimates through interpolation or filtering [50, 64]. The interpolation is typically applied both across subchannels in one OFDM block and across different OFDM blocks [61]. In this chapter, we denote the pilot spacing along different subchannels and OFDM blocks as I_p and K_p, respectively. For instance, the pilot configuration shown in Figure 7.2 corresponds to $I_p = 4$ and $K_p = 4$.

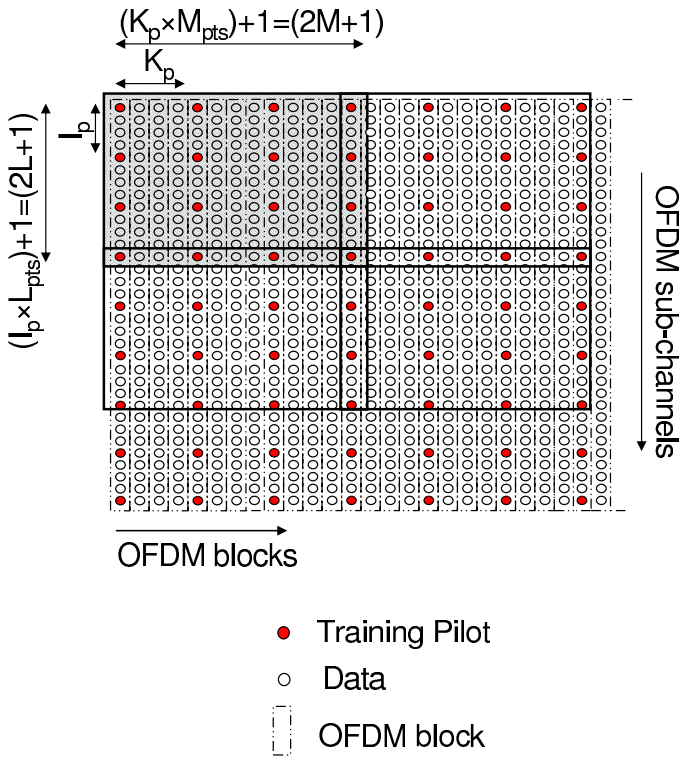


Figure 7.2: An example of pilot symbol configuration

7.2.4 Set Partitioning in Hierarchical Trees (SPIHT)

SPIHT [89] is a wavelet based image compression technique that uses set partitioning hierarchical trees encoding. In SPIHT, the wavelet coefficients of the image are encoded using bit planes, and two passes are performed on each bit-plane. The first pass is the *sorting pass*, which determines the sign values and implicit location information of significant wavelet coefficients. The second pass is the *refinement pass*, which refines bit values of the significant coefficients [89]. Several important properties of SPIHT are summarized as follows: First, the SPIHT has a good rate-distortion performance for still images with comparatively low complexity. Second, it is scalable or completely embeddable, that is, the decoding algorithm can be stopped at any received bit. This scalable property is very suitable for image transmission. The transmitted image can be decoded until the first irrecoverable error occurs, the more bits is received, the better quality the reconstructed image will have. Third, the encoded SPIHT bitstreams have the property that the later encoded bits are approximately less important than the earlier encoded bits. Due to this property, several unequal error protection (UEP) schemes [8,104] based on forward error correcting (FEC) codes have been proposed. Those approaches generally apply stronger FEC codes to the more important portions of the SPIHT bitstreams. The method proposed in this chapter utilizes this property in a different way. Instead of applying the stronger FEC to the more important bitstream, the PT scheme utilizes the best channel within the acceptable delay to transmit the most important data.

7.3 Priority Transmission for polynomial based channel estimation

Due to the advantages of the PSA methods in the fast fading environment, we first extend the derivation in [103] and develop the polynomial based PSA channel estimation scheme. We derive the channel estimation MSE and the decoded BER for different OFDM subchannels and finally, we propose the PT scheme to improve the reliability of multimedia transmission in OFDM systems using polynomial based PSA channel estimation.

7.3.1 PSA Polynomial Channel Estimation: Algorithm description

A time varying wireless channel response can be approximated by a set of piecewise polynomial basis functions [13]. Let (n,k) denote the n^{th} subchannel in the k^{th} OFDM block. In a time-frequency window that has dimension $(2L+1)\times(2M+1)$ and is centered at (n_0,k_0), the channel frequency response can be expressed as [13] [102]

$$\begin{aligned} H[n,k] &= \sum_{i=0}^{I_{deg}}\sum_{j=0}^{J_{deg}} C_{n_0,k_0}[i,j](n-n_0)^i(k-k_0)^j + S_{I_{deg}J_{deg}}[n,k]\ , \\ &\quad \text{for } n_0-L\le n\le n_0+L \quad \text{and} \quad k_0-M\le k\le k_0+M, \end{aligned} \tag{7.9}$$

where I_{deg} and J_{deg} are the orders of the polynomial basis at frequency and time domain, respectively, $C_{n_0,k_0}[i,j]$ are the polynomial coefficients, and $S_{I_{deg}J_{deg}}[n,k]$ are the model errors. When the model errors are small (negligible), the channel can be fully described by the polynomial coefficients, $C_{n_0,k_0}[i,j]$. Thus, the task of channel estimation is to obtain $C_{n_0,k_0}[i,j]$ from temporal estimation $\widetilde{H}[n_p,k_p]$ and

estimate $H[n,k]$ through the two dimensional polynomial interpolation as in (7.9).

In the following, we describe the polynomial based PSA channel estimation with rectangular pilot symbol arrangement as shown in Figure 7.2. Without loss generality, we focus on the channel responses in the window that has size $(2L+1)\times(2M+1)$ and is centered at (L,M). This window locates at the top-left corner of the pilot arrangement pattern in Figure 7.2. The number of pilot symbols inside this window are $L_{pts}\times M_{pts}$, where $L_{pts}=\lceil\frac{2L+1}{I_p}\rceil$ and $M_{pts}=\lceil\frac{2M+1}{K_p}\rceil$. Recall that I_p and K_p denote the pilot spacing at the frequency and time domain, respectively. In addition, these pilot symbols are located at positions $(a\cdot I_p, b\cdot K_p)$, where $a=0,1,\cdots,L_{pts}-1$ and $b=0,1,\cdots,M_{pts}-1$.

Using (7.9), the temporal estimate within this approximation window can be represented as

$$\widetilde{H}[a\cdot I_p, b\cdot K_p]=\sum_{i=0}^{I_{deg}}\sum_{j=0}^{J_{deg}} C_{n_0,k_0}[i,j](a\cdot I_p-n_0)^i(b\cdot K_p-k_0)^j+S'_{I_{deg}J_{deg}}[a\cdot I_p, b\cdot K_p] \quad (7.10)$$
$$\text{for } a=0,1,\cdots,L_{pts}-1 \text{ and } b=0,1,\cdots,M_{pts}-1,$$

where $(n_0,k_0)=(L,M)$ is the center of the approximation window. Since the temporal estimates, $\widetilde{H}[a\cdot I_p, b\cdot K_p]$, are noisy samples of the true channel frequency response, the residue term, $S'_{I_{deg}J_{deg}}[a\ \cdot I_p, b\cdot K_p]$, includes the model error as well as the noise. In order to determine $(I_{deg}+1)\times(J_{deg}+1)$ unknown polynomial coefficients from the temporal estimation, it is necessary to have at least $(I_{deg}+1)\times(J_{deg}+1)$ equations in (7.10). That is, $L_{pts}\geq I_{deg}+1$ and $M_{pts}\geq J_{deg}+1$.

Equation (7.10) can also be written in matrix format. Let $[A]_{i,j}$ denote the element on the i^{th} row and the j^{th} column of matrix A. We define the following

matrices as

$$\breve{\mathbf{H}}_{n_0,k_0} \text{ with size } L_{pts} \times M_{pts}, \text{ and } [\breve{\mathbf{H}}_{n_0,k_0}]_{i,j} = \widetilde{H}[i \cdot I_p, j \cdot K_p]$$

$$\mathbf{q}_{I_{deg}} \text{ with size } L_{pts} \times (I_{deg}+1), \text{ and } [\mathbf{q}_{I_{deg}}]_{i,j} = (i \cdot I_p - n_0)^j$$

$$\mathbf{C}_{I_{deg}J_{deg}} \text{ with size } (I_{deg}+1) \times (J_{deg}+1), \text{ and } [\mathbf{C}_{I_{deg}J_{deg}}]_{i,j} = C_{n_0,k_0}[i,j]$$

$$\mathbf{q}_{J_{deg}} \text{ with size } M_{pts} \times (J_{deg}+1), \text{ and } [\mathbf{q}_{J_{deg}}]_{i,j} = (i \cdot K_p - k_0)^j$$

$$\mathbf{S'}_{I_{deg}J_{deg}} \text{ with size } L_{pts} \times M_{pts}, \text{ and } [\mathbf{S'}_{I_{deg}J_{deg}}]_{i,j} = S'_{I_{deg}J_{deg}}[i \cdot I_p, j \cdot K_p].$$

We can show that (7.10) is equivalent to

$$\breve{\mathbf{H}}_{n_0,k_0} = \mathbf{q}_{I_{deg}} \mathbf{C}_{I_{deg}J_{deg}} \mathbf{q}^T_{J_{deg}} + \mathbf{S'}_{I_{deg}J_{deg}}. \tag{7.11}$$

Let $vec(\mathbf{A})$ denote the vector whose elements are taken column-wise from matrix $\mathbf{A}$, and let $\otimes$ denote the Kronecker product. The Kronecker product has the following property [52]

$$\mathbf{Y} = \mathbf{BXA}^T \Leftrightarrow vec(\mathbf{Y}) = (\mathbf{A} \otimes \mathbf{B}) vec(\mathbf{X}). \tag{7.12}$$

Using this property, (7.11) can be written as

$$\breve{\mathbf{h}}_{n_0,k_0} = (\mathbf{q}_{J_{deg}} \otimes \mathbf{q}_{I_{deg}}) \mathbf{c}_{I_{deg}J_{deg}} + \mathbf{s'}_{I_{deg}J_{deg}}, \tag{7.13}$$

where $\breve{\mathbf{h}}_{n_0,k_0} = vec(\breve{\mathbf{H}}_{n_0,k_0})$, $\mathbf{c}_{I_{deg}J_{deg}} = vec(\mathbf{C}_{I_{deg}J_{deg}})$ and $\mathbf{s'}_{I_{deg}J_{deg}} = vec(\mathbf{S'}_{I_{deg}J_{deg}})$.

In (7.13), $\breve{\mathbf{h}}_{n_0,k_0}$ contains the temporal estimates of the channel parameters that are obtained from training pilots using (7.8), $\mathbf{c}_{I_{deg}J_{deg}}$ contains the polynomial coefficients to be estimated, and $\mathbf{s'}_{I_{deg}J_{deg}}$ is the error term. Therefore, the least square solution of the polynomial coefficients, denoted by $\widehat{\mathbf{c}}_{I_{deg}J_{deg}}$, is calculated as

$$\widehat{\mathbf{c}}_{I_{deg}J_{deg}} = (\mathbf{q}_{J_{deg}} \otimes \mathbf{q}_{I_{deg}})^{\dagger} \breve{\mathbf{h}}_{n_0,k_0}, \tag{7.14}$$

where $\mathbf{A}^{\dagger}$ denotes the pseudo-inverse of matrix $\mathbf{A}$.

The next step is to compute the channel frequency response of all subchannels from the estimated polynomial coefficients. In the approximation window, the estimated channel parameters are represented by a $(2L+1) \times (2M+1)$ matrix $\widehat{\mathbf{H}}_{n_0,k_0}$, and $[\widehat{\mathbf{H}}_{n_0,k_0}]_{i,j} = \hat{H}[i,j]$. In addition, we denote $\widehat{\mathbf{h}}_{n_0,k_0} = vec(\widehat{\mathbf{H}}_{n_0,k_0})$ and define matrices

$$\mathbf{Q}_{I_{deg}} \text{ with size } (2L+1) \times (I_{deg}+1), \text{ and } [\mathbf{Q}_{I_{deg}}]_{i,j} = (i-L)^j,$$
$$\mathbf{Q}_{J_{deg}} \text{ with size } (2M+1) \times (J_{deg}+1), \text{ and } [\mathbf{Q}_{J_{deg}}]_{i,j} = (i-M)^j.$$

Then, the channel responses in the approximation window are estimated as

$$\widehat{\mathbf{h}}_{n_0,k_0} = (\mathbf{Q}_{J_{deg}} \otimes \mathbf{Q}_{I_{deg}})\widehat{\mathbf{c}}_{I_{deg}J_{deg}} \tag{7.15}$$
$$= (\mathbf{Q}_{J_{deg}} \otimes \mathbf{Q}_{I_{deg}})(\mathbf{q}_{J_{deg}} \otimes \mathbf{q}_{I_{deg}})^{\dagger}\breve{\mathbf{h}}_{n_0,k_0} \tag{7.16}$$
$$= (\mathbf{Q}_{J_{deg}}\mathbf{q}_{J_{deg}}^{\dagger} \otimes \mathbf{Q}_{I_{deg}}\mathbf{q}_{I_{deg}}^{\dagger})\breve{\mathbf{h}}_{n_0,k_0}. \tag{7.17}$$

Here, (7.15) is based on (7.9), (7.16) is derived from (7.14) and (7.15), and (7.17) is obtained using the properties of Kronecker product $(A \otimes B)(C \otimes D) = (AC \otimes BD)$ and $(A \otimes B)^{\dagger} = (A^{\dagger} \otimes B^{\dagger})$ [52].

Using (7.17) and (7.12), we can show that:

$$\widehat{\mathbf{H}}_{n_0,k_0} = (\mathbf{Q}_{I_{deg}}\mathbf{q}_{I_{deg}}^{\dagger})\breve{\mathbf{H}}_{n_0,k_0}(\mathbf{Q}_{J_{deg}}\mathbf{q}_{J_{deg}}^{\dagger})^T. \tag{7.18}$$

From implementation point of view, the term $(\mathbf{Q}_{I_{deg}}\mathbf{q}_{I_{deg}}^{\dagger})$ and $(\mathbf{Q}_{J_{deg}}\mathbf{q}_{J_{deg}}^{\dagger})$ can be computed off-line. Thus, the channel responses in one approximation window can be obtained from the temporal estimation by (i) a multiplication of a $(2L+1) \times L_{pts}$ matrix and an $L_{pts} \times M_{pts}$ matrix; and (ii) a multiplication of a $(2L+1) \times M_{pts}$ matrix and an $M_{pts} \times (2M+1)$ matrix. Based on the above discussion, the polynomial based PSA channel estimation can be performed using the following procedures:

Off-line Computation:

1. Determine the degrees of polynomial basis functions (I_{deg}, J_{deg}) and the window size (L, M) based on the training pattern and the channel conditions.
2. Calculate $(\mathbf{Q}_{I_{deg}}\mathbf{q}^{\dagger}_{I_{deg}})$ and $(\mathbf{Q}_{J_{deg}}\mathbf{q}^{\dagger}_{J_{deg}})$.

On-line Computation:

1. Compute the temporal estimation in $(2M+1)$ consecutive OFDM blocks.
2. Slide the approximation windows over these total $N \times (2M+1)$ subchannels, such that all subchannels are covered by at least one window. Then, compute the channel parameters in each window from the temporal estimation based on (7.18). Note that the matrix indexes should be adjusted according to the window centers when using (7.18).

The parameters in polynomial based PSA channel estimation including pilot spacing, polynomial degree, and window size, should be chosen to minimize the channel estimation error for given channel conditions. Unfortunately, there is no closed-form solution for such an optimization problem. In [102], the optimal parameters for decision-directed methods were obtained using exhaustive search for a given channel correlation function. In this chapter, we choose the parameters as $L_{pts} = I_{deg} + 1$, $M_{pts} = J_{deg} + 1$ and $I_{deg} = J_{deg} = 3$. Thus, the approximation window size only depends on the pilot spacing, as $(I_{deg}I_p + 1) \times (J_{deg}K_p + 1)$. We obtain these parameters by performing simulations over a broad range of channel conditions, and they demonstrate good performance in most channels. In particular, we performed simulation for Typical Urban (TU) and Hilly Terrain (HT) delay profiles [66, 95] for Doppler frequency from 40Hz to 200 Hz, and for channel SNR

from 5dB to 40dB. Here, Both of the delay profiles have 6 paths. The average path power and delay for the TU delay profile are $\sigma_i^2 = \{0.5, 1.0, 0.63, 0.26, 0.16, 0.1\}$ and $\tau_i = \{0, 0.2\mu s, 0.5\mu s, 1.6\mu s, 2.3\mu s, 5.0\mu s\}$, and the average path power and delay for the HT delay profile are $\sigma_i^2 = \{1.0, 0.64, 0.4, 0.2, 0.26, 0.06\}$ and $\tau_i = \{0, 0.2\mu s, 0.4\mu s, 0.6\mu s, 15.0\mu s, 17.2\mu s\}$.

7.3.2 Channel Estimation Error and Decoding BER

Instead of finding the channel estimation error averaged over all subchannels, we are more interested in calculating the channel estimation error of individual subchannel. Let $\varepsilon_{mse}^2[n,k]$ denote the mean square channel estimation error of the n^{th} subchannel in the k^{th} OFDM block, i.e. $\varepsilon_{mse}^2[n,k] = E[(H[n,k]-\hat{H}[n,k])(H[n,k]-\hat{H}[n,k])^*]$. Then, the MSE channel estimation of all subchannels in one estimation window can be described by a $(2L+1)(2M+1) \times 1$ vector as:

$$\varepsilon_{\mathbf{MSE}}^2 \equiv [\varepsilon_{mse}^2[0,0], \ldots, \varepsilon_{mse}^2[2L,0], \varepsilon_{mse}^2[0,1], \ldots, \varepsilon_{mse}^2[2L,1], \ldots, \varepsilon_{mse}^2[2L,2M]]^T.$$

Similar to Section 7.3.1, matrices $\mathbf{H}_{n_0,k_0}$ and $\widehat{\mathbf{H}}_{n_0,k_0}$ represent the true and estimated channel responses in one approximation window, respectively. Both matrices have size $(2L+1)\times(2M+1)$. For the window centered at (n_0,k_0), $[\mathbf{H}_{n_0,k_0}]_{i,j} = H[i-L+n_0, j-M+k_0]$ and $[\widehat{\mathbf{H}}_{n_0,k_0}]_{i,j} = \hat{H}[i-L+n_0, j-M+k_0]$. The vector representations of these two matrices are $\mathbf{h}_{n_0,k_0} = vec(\mathbf{H}_{n_0,k_0})$ and $\widehat{\mathbf{h}}_{n_0,k_0} = vec(\widehat{\mathbf{H}}_{n_0,k_0})$. Thus, (7.19) is equivalent to

$$\varepsilon_{\mathbf{MSE}}^2 = diag(E[(\mathbf{h}_{n_0,k_0} - \widehat{\mathbf{h}}_{n_0,k_0})(\mathbf{h}_{n_0,k_0} - \widehat{\mathbf{h}}_{n_0,k_0})^H]). \tag{7.19}$$

From (7.16) and (7.19), we obtain

$$\begin{aligned}\varepsilon_{\mathbf{MSE}}^2 = diag\{(E[(\mathbf{h}_{n_0,k_0} - (\mathbf{Q}_{J_{deg}} \otimes \mathbf{Q}_{I_{deg}})(\mathbf{q}_{J_{deg}} \otimes \mathbf{q}_{I_{deg}})^\dagger \widetilde{\mathbf{h}}_{n_0,k_0}) \\ (\mathbf{h}_{n_0,k_0} - (\mathbf{Q}_{J_{deg}} \otimes \mathbf{Q}_{I_{deg}})(\mathbf{q}_{J_{deg}} \otimes \mathbf{q}_{I_{deg}})^\dagger \widetilde{\mathbf{h}}_{n_0,k_0})^H]\}.\end{aligned} \tag{7.20}$$

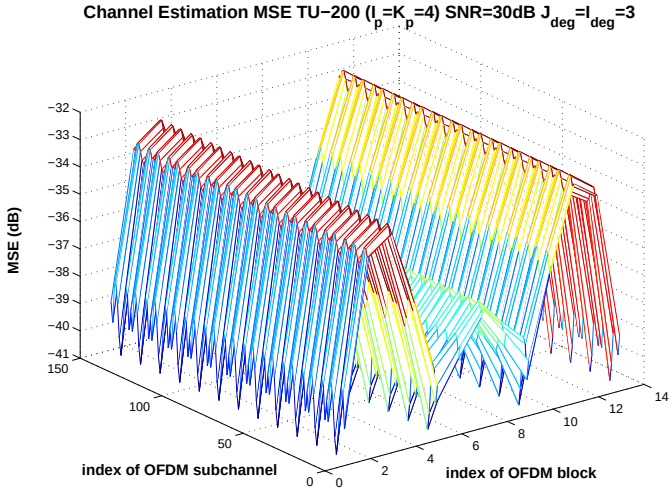
Using the channel delay profile, Doppler frequency and channel SNR, the channel estimation error can be calculated.

For M-QAM modulation, the effect of channel estimation on the BER has been discussed in [29, 98]. Particularly, [29] gave the close form expression of the BER in OFDM systems with imperfect channel estimation. Using the results in [29] and (7.20), we calculate the BER of different OFDM subchannels. Figure 7.3(a) shows the mean square channel estimation error calculated from (7.20) for the typical urban (TU) delay profile [66, 95] with Doppler frequency 200Hz and channel SNR 30dB. The pilot spacing, $I_p = K_p = 4$ is used. This implies that the approximation window size is 13 by 13, as discussed in Section 7.3.1. From Figure 7.3(a), we can see that the channel estimation error varies significantly for different subchannels and OFDM blocks. In addition, the corresponding BER for using 16-QAM modulation is shown in Figure 7.3(b). It is clear that the subchannels that have larger channel estimation error experience higher decoding BER.

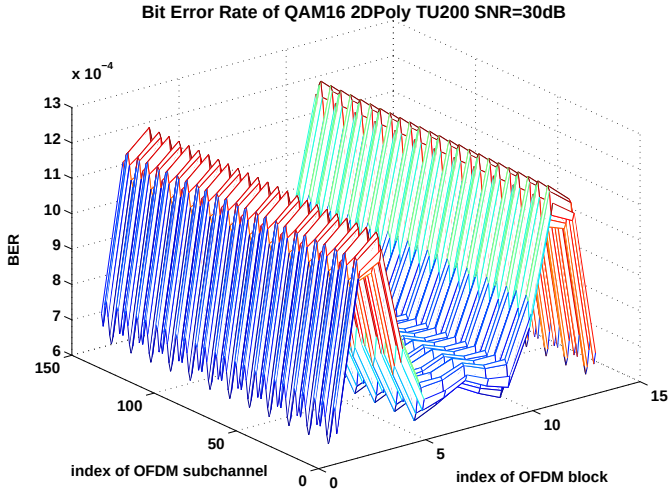
7.3.3 Priority Transmission Design for Two Dimensional Polynomial Channel Estimation

In the previous section, we have seen that there exists significant BER variation across different OFDM subchannels due to channel estimation inaccuracy. For multimedia data transmission, we can utilize this property to provide unequal error protection (UEP). In this section, we design the priority transmission (PT) scheme, which rearranges multimedia data in OFDM subchannels by jointly considering the effects of channel estimation and the importance of multimedia data. In particular, the PT scheme is applied in the following four steps.

Step 1. Calculate mean square channel estimation error in one approximation



(a) Channel Estimation MSE, TU-200, SNR=30dB



(b) BER, TU-200, SNR=30dB, QAM16

Figure 7.3: Channel Estimation MSE and decoding BER when using polynomial based channel estimation

window, based on channel estimation parameters (I_p, K_p, I_{deg}, J_{deg}, M_{pts}, N_{pts}) and the channel correlation matrix. Here, the channel correlation can be obtained through feedback from the receivers.

Step 2. Sort all subchannels within $D \times K_p \times J_{deg}$ OFDM blocks in the increasing order of BER. Here, D is the PT delay parameter and should be determined such that the maximum decoding delay allowed at the receiver is less than the time used to transmit $D \times K_p \times J_{deg}$ consecutive OFDM blocks.

Step 3. Rearrange the encoded multimedia bitstream in the decreasing order of importance.

Step 4. Match the rearranged multimedia data with the sorted subchannels, such that higher importance of the multimedia data are transmitted over the subchannels with lower BER.

The total decoding delay at the receiver depends on the parameter D as well as the approximation window size. Since the receiver must receive all the pilot symbols in one approximation window before performing channel estimation, the decoding delay caused by channel estimation is $K_p \times J_{deg}$ OFDM blocks. By applying PT, the receiver must obtain $D \times K_p \times J_{deg}$ consecutive OFDM blocks before rearranging the received data back into their original order. Thus, the decoding delay of the OFDM system with polynomial based PSA channel estimation and PT is $D \times K_p \times J_{deg}$ OFDM blocks. We note that the performance of the PT depends on the delay parameters D. When larger delay parameter D is allowed, the PT scheme will have more flexibility in arranging the transmission order of data, and will achieve better quality in reconstructed multimedia.

For multimedia transmission, UEP can also be achieved by applying forward error correction (FEC) codes with different rates to different portions of multimedia data stream. Compared to FEC based UEP methods, such as those in [8, 104], the PT scheme has the advantage of not introducing additional redundancy. Furthermore, the PT scheme can work together with FEC based methods when both the BER variation of channel estimation and the importance of multimedia data are taken into consideration for choosing the channel coding rates.

7.3.4 PT Scheme based on Polynomial Channel Estimation: Simulation Results

We simulate image transmission in an OFDM system with the following parameters to demonstrate the performance of the PT scheme. The transmitted data is a 512 by 512 Lena image, which is compressed to 0.5 bit per pixel (bpp) using SPIHT [89]. The compressed bitstream is packetized into 128 bit long packets. Each packet is appended with a 16 bit CRC code [67] [87] and then encoded using the shortened systematic RS(30,18) code, which is obtained by shortening RS(255,225) in GF(2^8) [2]. The encoded data are transmitted in an OFDM system, where the entire channel bandwidth is 800kHz, with 128 subchannels. In each OFDM block, four boundary subchannels at each end are used as guard tones [64] and the remaining 120 subchannels are used to transmit data. To eliminate ISI, a 32 symbol long cyclic prefix is inserted in each OFDM block [65]. All subchannels use QAM16 modulation. Rectangular pilot configuration with $I_p = K_p = 4$ is used in the TU delay profile and $I_p = 2, K_p = 4$ is used in the HT delay profile. At the receiver, error check is performed based on the CRC-16 code after RS decoding. If there are irrecoverable errors in a packet, this packet is dropped. The first

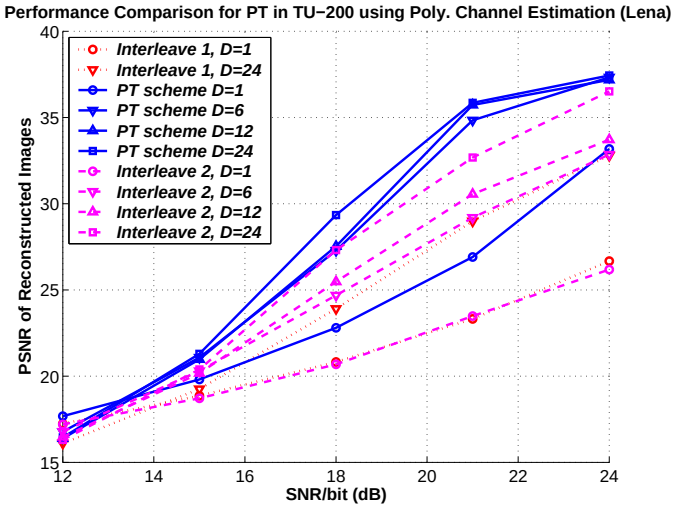
dropped packet stops the SPIHT decoder. We employ the peak-signal-to-noise ratio (PSNR) of the reconstructed image as our performance measure. The PSNR is defined as

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) dB, \tag{7.21}$$

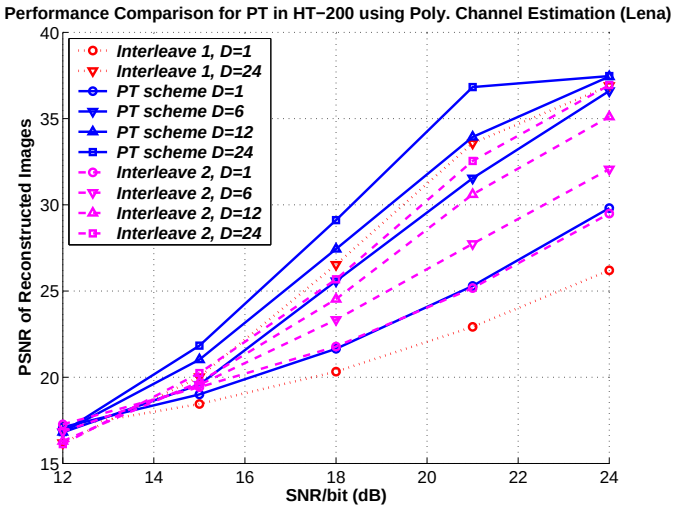
where *MSE* denotes the mean-square-error of the reconstructed image.

Three transmission strategies are compared. The first scheme, referred to as the *Interleaving 1*, transmits the encoded bitstream according to the following order $\{(n,k) = (0,0), (0, K_p \times J_{deg}), ..., (0, (D-1) \times K_p \times J_{deg}), (1,0), (1, K_p \times J_{deg}), ..., (1, (D-1) \times K_p \times J_{deg}), ..., (N-1, 0), (N-1, K_p \times J_{deg}), ..., (N-1, (D-1) \times K_p \times J_{deg}), (0,1), (0, K_p \times J_{deg}+1), ..., (0, (D-1) \times K_p \times J_{deg}+1), (1,1), ...\}$, where n is the index of OFDM subchannels and k denotes the index of OFDM blocks. We note that when $D = 1$, the *Interleaving 1* becomes the regular transmission that transmits the encoded image block by block according to the order $\{(n,k) = (0,0), (1,0), ..., (N-1,0), (0,1), ..., (N-1,1), ...\}$. The second scheme, referred to as the *Interleaving 2*, transmits data according to the order $\{(n,k) = (0,0), (0, K_p \times J_{deg}), ..., (0, (D-1) \times K_p \times J_{deg}), (0,1), (0, K_p \times J_{deg}+1), ..., (0, (D-1) \times K_p \times J_{deg}+1), ..., (0, D \times K_p \times J_{deg}-1), (1,0), (1, K_p \times J_{deg}), ..., (1, (D-1) \times K_p \times J_{deg}), ...\}$. When $D = 1$, the *Interleaving 2* scheme transmits according to order $\{(n,k) = (0,0), (0,1), ..., (0, K_p \times J_{deg}-1), (1,0), ..., (1, K_p \times J_{deg}-1), ..., (N-1,1), ...(N-1, K_p \times J_{deg}-1)\}$. The third scheme is the PT scheme, which rearranges the transmission order of the multimedia data within $D \times K_p \times J_{deg}$ OFDM blocks according to channel estimation errors. In the simulations, we assume the perfect estimation of channel correlation matrix (Step 1) in the PT scheme.

Figure 7.4 shows the average PSNR of reconstructed images in three transmission schemes. Figure 7.4(a) and 7.4(b) are for the TU and HT delay profiles, re-



(a) TU delay profile (f_D=200Hz, $I_p = K_p = 4$)



(b) HT delay profile (f_D=200Hz, $I_p = 2, K_p = 4$)

Figure 7.4: Comparison of the three transmission schemes using polynomial based channel estimation

spectively, with the maximum Doppler frequency 200Hz. The results are obtained by averaging 300 transmissions of Lena image for different fading and additive noise realizations. One can observe that the PT scheme performs better or at least as well as the regular transmission (*Interleave 1*, D=1) and *Interleave 2* with D=1. Moreover, the performance gain of the PT scheme is larger when the delay parameter D is larger. This is due to the fact that there are effectively larger number of good OFDM subchannels to transmit the more important SPIHT bitstreams when the delay parameter D is larger. Compared with the *Interleave 1* and *Interleave 2* schemes, the PT scheme achieves about 4 and 8 dB gain in reconstructed PSNR when the delay parameter $D = 24$ and the channel SNR is equal to 21 dB in TU delay profile with 200 Hz doppler frequency. With the same delay parameter D, all the three transmission schemes have similar interleaving benefit. Thus, the gain of the PT scheme is mainly from allocating the more important data to subchannels experiencing lower channel estimation error. Figure 7.5 shows the PSNR of individual reconstructed images. Here, the Doppler frequency is 200 Hz, and both TU and HT delay profiles are studied. When using the PT, the PSNR of the reconstructed images at different time instances does not change much, while two other transmission schemes do not have this advantage. Obviously, the PT scheme provides better and smoother performance over time.

7.4 Priority Transmission based on FFT based channel estimation

The FFT based channel estimation has been well studied for both the decision-directed [65, 100] and PSA scenarios [50, 61, 64, 75]. In this section, we design the

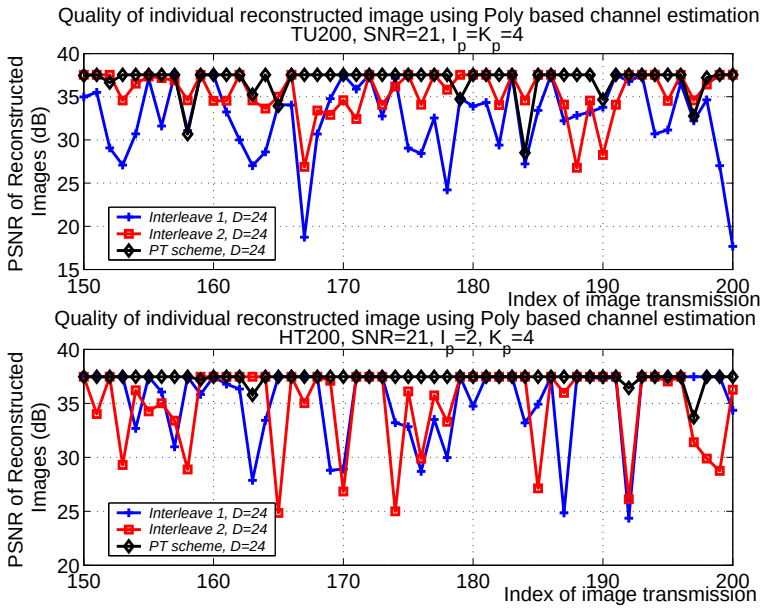


Figure 7.5: PSNR of individual reconstructed images of the three transmission schemes using polynomial channel estimation

PT scheme for the FFT based PSA channel estimation. As discussed in Section 7.3, the crucial idea behind the PT scheme is to evaluate the error performance of individual OFDM subchannels and to load multimedia data according to the quality of the subchannels. Thus, we first briefly summarize the FFT based channel estimation algorithm and design the PT scheme to improve the reliability of multimedia transmission.

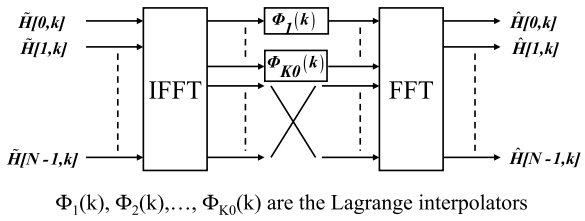


Figure 7.6: FFT based channel estimation scheme

7.4.1 FFT based Channel Estimation: Algorithm description

The structure of FFT based channel estimation in [61, 65] is illustrated in Figure 7.6. The input, $\tilde{H}(n,k)$, is obtained from the temporal estimation as

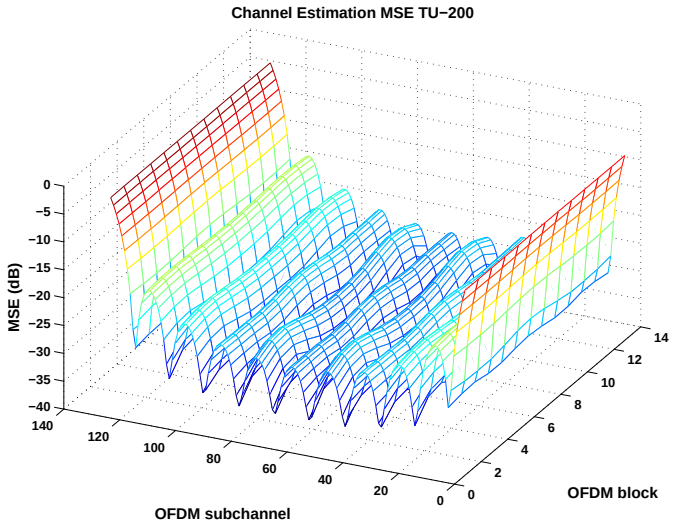
$$\tilde{H}(n,k) = \begin{cases} Y[n,k]/X[n,k], & \text{when } (n,k) \text{ are pilot positions} \\ 0, & \text{otherwise.} \end{cases}$$

The first K_0 outputs of the IFFT, representing low frequency components, are interpolated by the interpolation filters, denoted by $\phi_1, \phi_2, \cdots, \phi_{K_0}$. Here, K_0 is computed as $K_0 = (\lfloor B_d \times \tau_{max} \rfloor + 1)$ [65], where B_d is the total channel bandwidth and τ_{max} is the maximum delay spread. In [61], the Lagrange interpolators are chosen. The rest of the high frequency components after IFFT are set to zeros. The estimated channel parameters, denoted by $\hat{H}(n,k)$, are obtained after the FFT operation. In this channel estimation scheme, the frequency domain interpolation is performed through IFFT-FFT filtering, while the time domain interpolation is performed by the Lagrange interpolators. Consequently, the channel responses for all subchannels are estimated.

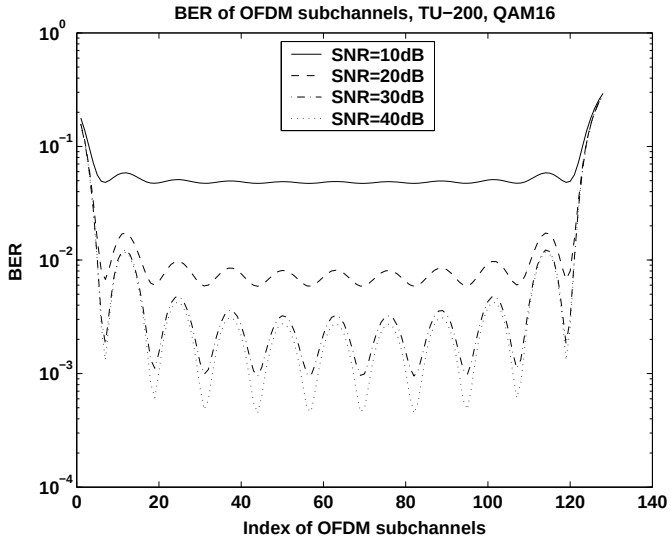
7.4.2 FFT based Channel Estimation: Channel Estimation Error and BER

FFT based PSA channel estimation can be applied on a variety of pilot patterns. For the purpose of fair comparison between FFT based and polynomial based methods in later sections, we demonstrate the performance of the FFT based methods using the rectangular training pattern. Without loss of generality, we consider the channel estimation in $(L_{deg}K_p + 1)$ consecutive OFDM blocks, which have indexes $\{k,\ k = 0, \cdots, L_{deg}K_p\}$. Here, L_{deg} is the degree of the Lagrange interpolation.

Given the channel correlation functions (as in (7.5) - (7.7)), the channel estimation MSE can be calculated in a similar way as that in section 7.3.2, which is omitted in this chapter. Figure 7.7(a) shows the channel estimation mean square error for the TU delay profile with Doppler frequency 200Hz and channel SNR 30dB. The pilot spacing is chosen as $I_p = K_p = 4$ and L_{deg} is chosen to be 3. We observe that there exists a significant variation in channel estimation errors along different OFDM subchannels, which is often referred to as the *leakage effect* [35, 65, 75, 100]. The leakage effect occurs when the delay paths, τ_i, are not all integer multiples of the system sampling period [35, 65, 75]. Since it is not realistic for all delay paths to be exactly integer multiples of the sampling period, the leakage effect always causes performance degradation in FFT based channel estimation. In Figure 7.7(b), the decoding BER of subchannels in one OFDM block calculated from the estimation MSE using the results in [29], is shown for different channel SNR in the TU delay profile with Doppler frequency 200Hz. One can see that the leakage effect will not diminish even when channel SNR is high. Compared to the error variation in the polynomial-based methods (see Figure 7.3), the



(a) MSE TU-200, SNR=30dB, QAM16



(b) BER TU-200, SNR=30dB, QAM16

Figure 7.7: Channel estimation MSE and decoding BER when using FFT based channel estimation

error variation in the FFT-based methods is larger. This leakage effect is difficult to eliminate and is typically remedied by discarding a large number of boundary subchannels [75] or performing adaptive bit/power loading [108] that requires high computational complexity. In this work, we utilize this property to provide unequal error protection (UEP) for multimedia transmission.

7.4.3 Priority Transmission Design for FFT based Channel Estimator

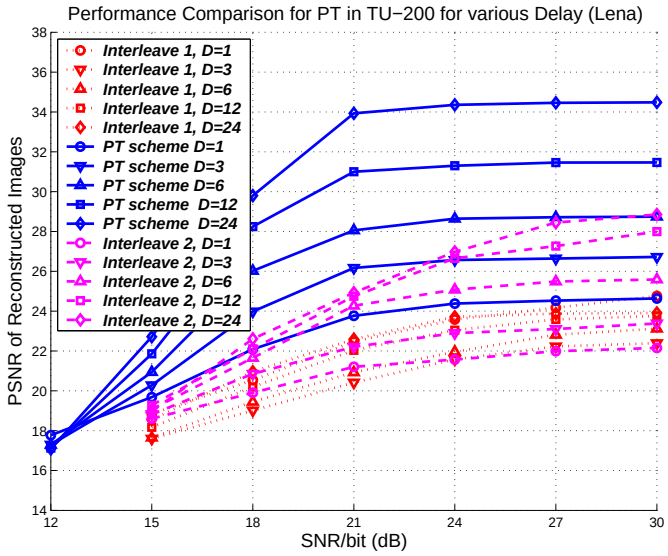
Similar to Section 7.3, the priority transmission utilizes the variation of BER and provides UEP for multimedia data. The procedure of the PT is the same as that in Section 7.3.3 with some slight modifications in the first step. In the first step of the PT with the FFT based channel estimation, the channel estimation MSE is calculated based on pilot spacing, Doppler frequency, maximum path delay, total bandwidth, and degree of Lagrange interpolation. The decoding delay introduced by the FFT based channel estimation is $(L_{deg}K_p + 1)$ OFDM blocks, and that of the PT is $D \times L_{deg} \times K_p$ OFDM blocks. Thus, the total decoding delay is $D \times L_{deg} \times K_p$. The parameter D can be adjusted to provide the tradeoff between the decoding delay and the quality of reconstructed multimedia.

7.4.4 PT Scheme based on FFT based Channel Estimation: Simulation Results

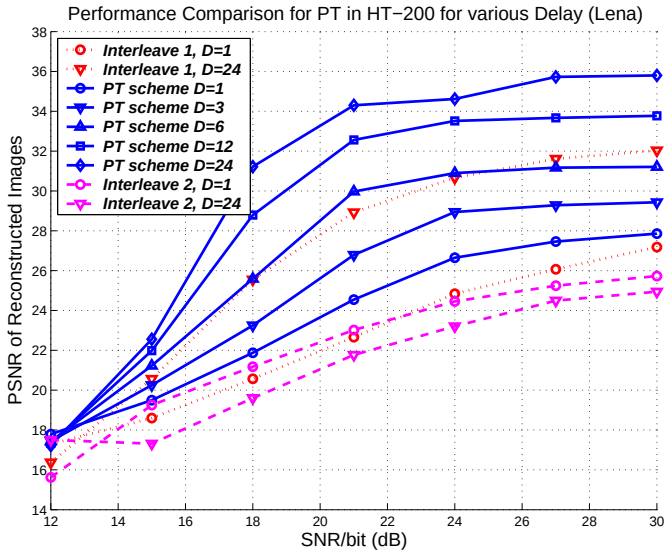
In this section, we evaluate the effectiveness of the PT for FFT based channel estimation through simulations. The performance of the PT in both FFT based and polynomial based methods will be compared in Section 7.5. Similar to Section 7.3.4, three transmission schemes will be compared. They are the *Interleave 1*,

Interleave 2 and PT scheme. All other simulation parameters are the same as those in Section 7.3.4. Figure 7.8 shows the simulation results for various decoding delay and channel SNR. Figure 7.8(a) is for the TU delay profile and Figure 7.8(b) is for the HT delay profile. Compared with the methods that do not utilize the channel estimation property, the PT scheme can significantly improve the PSNR of the reconstructed images in moderate and high channel SNR regions, where most practical wireless systems operate. For example, the PT scheme outperforms the *Interleave 1* and *Interleave 2* with $D = 1$ in TU-200 by 1 and 2 dB in PSNR of the reconstructed image, when channel SNR equals to 21 dB. The performance gain of the PT scheme is more pronounced when the delay parameter D is larger. That is the PT scheme achieves 11 and 9 dB higher in the PSNR of the reconstructed image, when it is compared with *Interleave 1* and *Interleave 2* with $D = 24$ and channel SNR equals to 21 dB. The reason for this higher performance gain is that the PT scheme effectively has larger number of good subchannels to transmit more important multimedia data. The PSNR of individual reconstructed images are shown in Figure 7.9 for the *Interleave 1*, *Interleave 2* and PT scheme for $D = 24$. When using the *Interleave 1* and *Interleave 2* schemes, the quality of the subchannels that transmits the more important bits of the SPIHT bitstream is quite random. Thus, the quality of individual received images changes rapidly. When using the PT, the data is allocated according to the importance of data and the quality of the subchannels. In this case, the transmission of each image experiences similar channel conditions and smooth quality of the reconstructed images is achieved.

We also simulate the case when using the *shifted-pilot* configuration, that is the pilots are placed on subchannels $0, I_p, 2I_p, ...$, in the 0^{th} OFDM block and they



(a) TU delay profile f_D=200Hz



(b) HT delay profile f_D=200Hz

Figure 7.8: Comparison between the three transmission schemes for various wireless channel conditions and decoding delay.

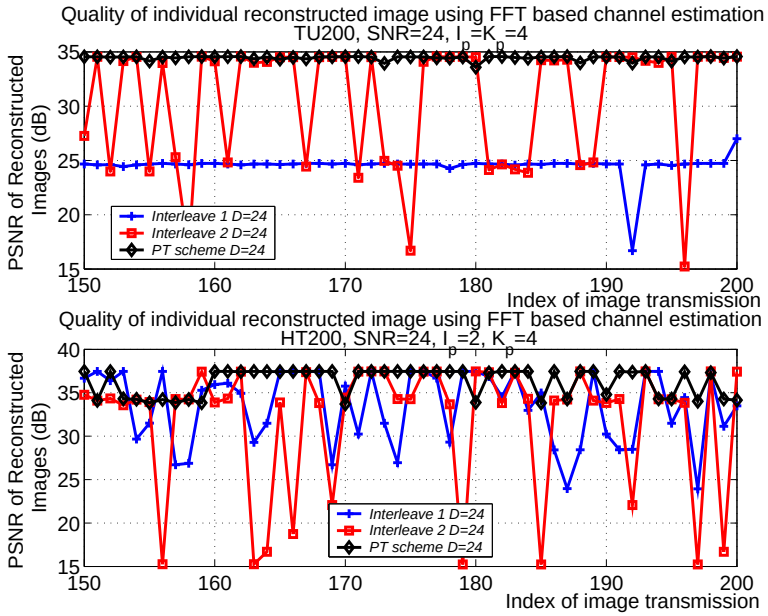
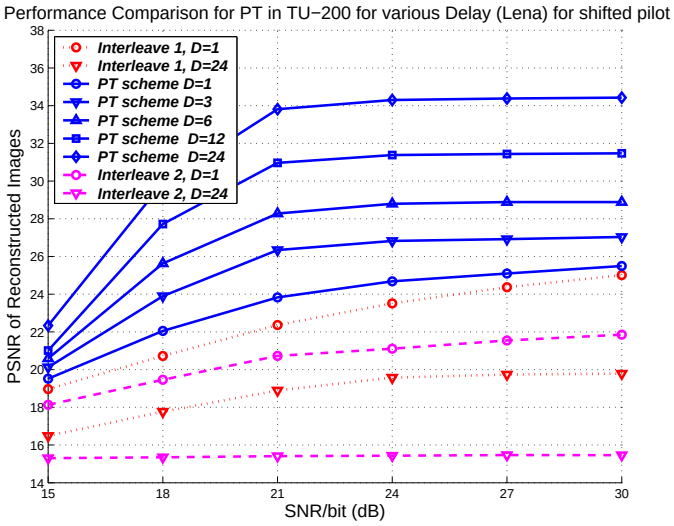


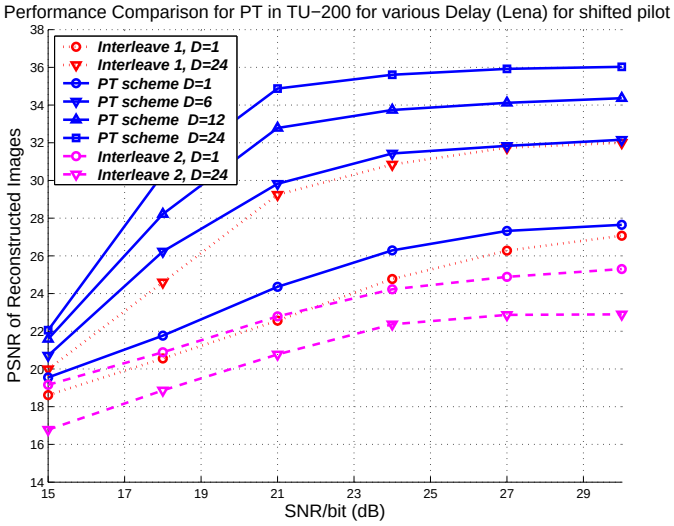
Figure 7.9: PSNR of individual reconstructed images of three transmission schemes.

are placed on subchannels $1, I_p + 1, 2I_p + 1, ...$, in the K_p^{th} OFDM block. We note that this shifted pilot has been used in several practical transmission standards such as DAB [1] and DVB [2]. The performance of the PT scheme is shown in Figure 7.10. Similar to the rectangular pilot case, the PT scheme always achieves higher performance compared with the schemes that do not utilize the channel estimation error property. We note that the calculation of channel estimation MSE and subchannel reordering in PT for the shifted pilot takes the pilot pattern into consideration. Within the acceptable delay, the PT scheme tries to match the characteristic of multimedia to the OFDM channel by allocating the more important multimedia data onto the better subchannels.

Comparing Figure 7.8(a) and Figure 7.10(a), correspondingly Figure 7.8(b) and Figure 7.10(b), we observe that the performance of the PT scheme in the shifted pilot is slightly better or almost the same as the one without shifted pilot. For



(a) TU delay profile f_D=200Hz



(b) HT delay profile f_D=200Hz

Figure 7.10: Comparison between the three transmission schemes for shifted pilot pattern in various wireless channel conditions and decoding delay.

the Interleave 1 and 2 schemes, their performance are comparable when the delay parameter is one, however, they become worse when the Delay parameter is large. This can be explained as follows. The interleave 1 and 2 schemes with delay parameter virtually fill the near boundary OFDM subchannel with the more important data while expecting some interleaving benefit. When using with the shifted pilot, both of the schemes are more likely to use bad subchannels near the OFDM block boundaries. Therefore, the resulting performances are worse than the ones without shifted pilot. This effect can be compensated by discarding more boundaries subchannels as demonstrated below. We simulate the same condition for TU-200 case except that eight boundary subchannels at each end of one OFDM block are used as guard tone and the remaining 112 subchannels are used for transmitting the multimedia data. The resulting performance comparisons are shown in Figure 7.11. It is clear from the figure that interleaving schemes with delay perform better than without delay case. However, the larger the number of subchannels are discarded, the more OFDM blocks should be used for transmitting an image. Similar to previous scenario, the PT scheme in this case also performs much better compared to the one where the guard tone is four. This is because that the worse channels have been discarded. In all simulations, the PT scheme proves to provide significant gain over the one without exploiting channel estimation property.

7.5 Comparison between FFT based method and Polynomial based Method

The FFT based and Polynomial based channel estimation methods have been compared for data transmission in decision-directed scenario [102] [103]. Since we have

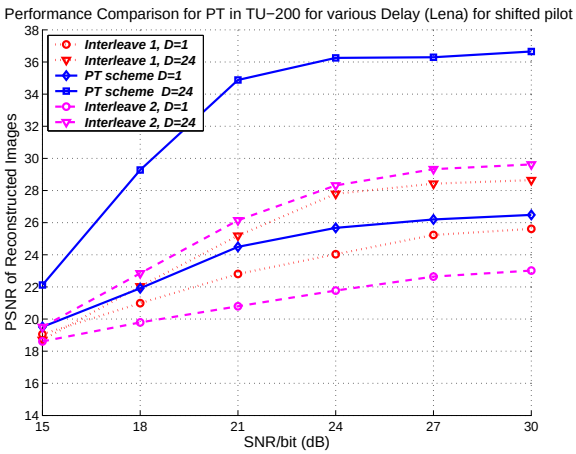


Figure 7.11: TU delay profile f_D=200Hz, guard tone=8.

developed the polynomial based PSA channel estimation, we can compare the FFT based and polynomial based channel estimation in the PSA scenario. In this section, we will first compare these two types of PSA channel estimation techniques for generic data transmission, by examining their average BER performances for different channel SNR and training pilot density. More importantly, since the OFDM modulation is adopted in many broadband multimedia transmission standards, such as DVB-T, it is particularly interesting to perform the comparison for multimedia transmission when the proposed PT schemes are employed.

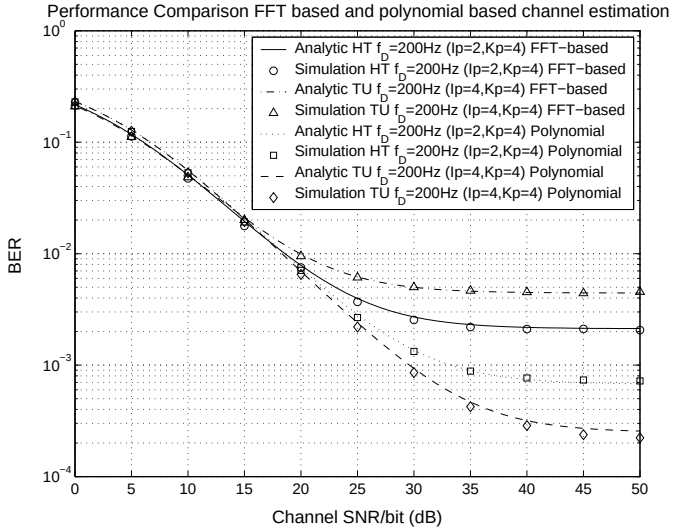
7.5.1 Comparison for Data Transmission

As explained in Section 7.4, the FFT based channel estimation suffers from the leakage effect when the delay paths are not separated by integer multiples of the sampling period. The leakage effect, which causes severe performance degradation in FFT based schemes, cannot be eliminated by increasing the channel SNR or the density of training pilots. The polynomial based channel estimation does not have the leakage effect and performs very well as long as the channel frequency response

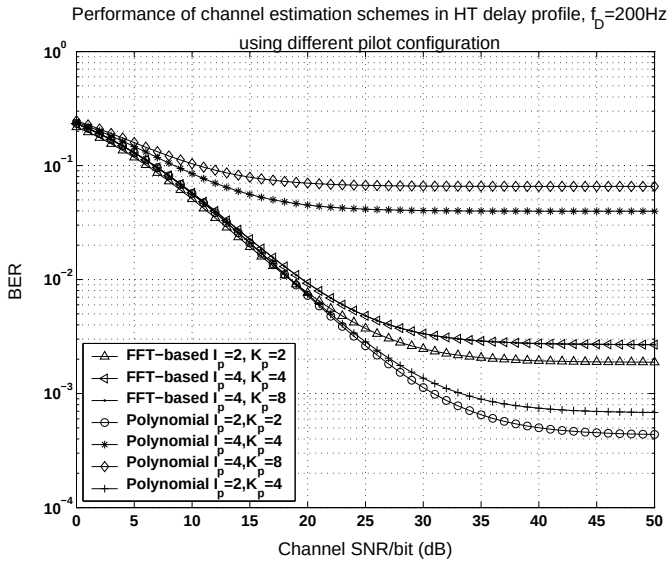
in the approximation window changes smoothly. This can be ensured by choosing small approximation window and increasing pilot density.

We first compare both channel estimation methods for data transmission. The setup of the OFDM modulation is the same as that in Section 7.3.4. The pilot spacing (I_p,K_p)=(4,4) is chosen for the TU delay profile, and (I_p,K_p)=(2,4) is chosen for the HT delay profile. In the FFT based method, the degree of Lagrange interpolator is $L_{deg} = 3$. According to the maximum delay spread and the total bandwidth, the parameter K_0 is 5 for the TU delay and is 15 for the HT delay. In the polynomial based method, the polynomial degrees are chosen as $I_{deg} = J_{deg} = 3$.

Figure 7.12(a) shows the analytical and simulated BER performance for both channel estimation schemes. The simulation results are obtained by transmitting 24000 OFDM blocks, and the BER shown in the figure represents the average performance of all subchannels. The analytical BER is evaluated by using the channel estimation MSE derived in Section 7.3.2 and Section 7.4.2 and the results in [29]. From the figure, we observe that the FFT based channel estimation has error floor in all channel conditions, due to the leakage effect. The polynomial based channel estimation achieves lower BER, although it also has the error floor in TU-200 and HT-200 cases, due to the model errors. In addition, the FFT based method has slightly better performance in the low channel SNR region. This is because that the FFT based methods can remove the channel noise by eliminating the high frequency components when performing IFFT-FFT filtering. In moderate and high SNR regions, removing high frequency components causes model error since the energy of channel frequency response spread over all frequency bins due to the leakage effect. Thus, the polynomial based method performs much better



(a) $f_D = 200Hz$



(b) HT delay, f_D=200Hz

Figure 7.12: Comparison between FFT based and polynomial based methods.

than the FFT based method in moderate and high SNR regions.

We also investigate the effects that the pilot density has upon these two channel estimation schemes in Figure 7.12. It is important to point out that IFFT-FFT filtering is suitable for interpolating sinusoidal like functions, while polynomial interpolation is suitable for slow varying functions. In some situations, the polynomial based method needs more pilots than the FFT based method to achieve good performance. For instance, in HT-200 channel with pilot spacing $I_p = K_p = 4$, the FFT based method has better performance than the polynomial based method. This is because that the polynomial basis cannot accurately model fast changing sinusoidal functions along different subchannels, as in the HT delay scenario, within a large approximation window. In this case, the approximation window size should be reduced.

In summary, the polynomial based channel estimation outperforms the FFT based channel estimation for data transmission in realistic channel SNR region, when pilot density is large enough such that the channel frequency response within an approximation window can be modelled as 2D polynomial functions.

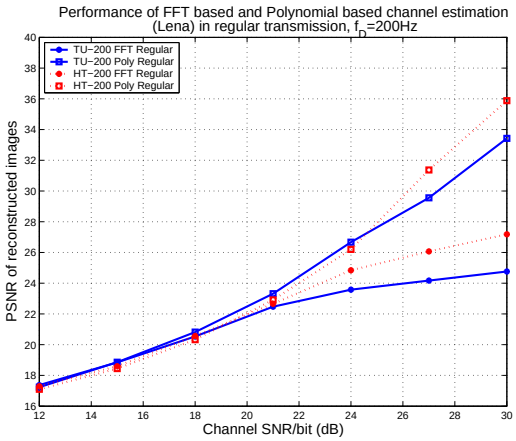
7.5.2 Comparison for Multimedia Transmission

In this section, we study which channel estimation scheme is better for multimedia transmission. In Figure 7.13(a), the average PSNR of reconstructed images with *Interleave 1* with D=1 (regular transmission) for the FFT based and the polynomial based methods are compared in TU and HT delay scenarios. Similar to the data transmission results, the polynomial based method outperforms the FFT based method in moderate and high SNR regions. In Figure 7.13(b) and 7.13(c), these two channel estimation schemes are compared when the PT is employed.

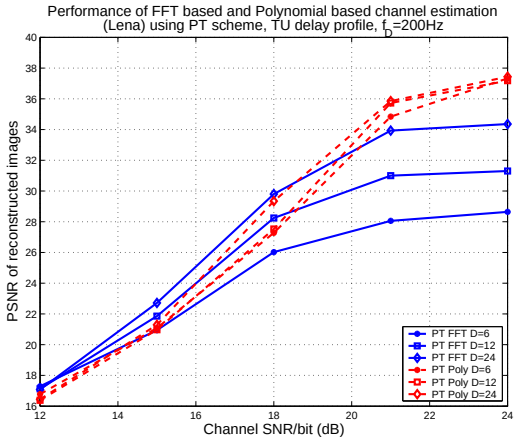
When the channel SNR is from 12dB to 19dB, the polynomial based method is slightly worse than the FFT based method, since the latter method can remove noise more effectively. When the channel SNR is higher, the polynomial based method achieves better performance because the effects of the polynomial model error are much less pronounced than the leakage effect. In fact, these two channel estimation schemes have close performance, since the PT scheme compensates the imperfection of channel estimation.

7.5.3 Complexity Comparison

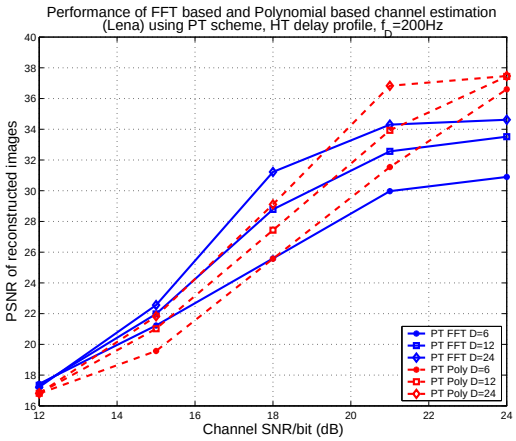
The complexity of the two channel estimation algorithms are compared in terms of the number of real additions and real multiplications needed to perform channel estimation per OFDM block. Let N denotes the number of subchannels, I_p and K_p denote the pilot spacing , $K = K_p \times L_{deg} + 1 = K_p \times J_{deg} + 1$ and $I = I_p \times I_{deg} + 1$ are the window size of the interpolation, where L_{deg}, I_{deg}, J_{deg} are the lagrange interpolator degree and two dimensional polynomial degree respectively. Using these parameters, the complexity of both the algorithms are summarized in Table 7.1. Assuming N is the power of two, the FFT operation requires $N \log_2 N$ real multiplications and $2N \log_2 N$ real additions [76]. The first term of the number of multiplication in FFT based method corresponds to the IFFT-FFT pair, the second term corresponds to the lagrange interpolation, and the third term corresponds to the process of finding the temporal estimate (7.8). Similarly, the first term in the number of addition corresponds to the IFFT-FFT operation and the second term corresponds to the lagrange interpolation. In complexity analysis for polynomial channel estimation, the terms ($\mathbf{Q}_{I_{deg}} \mathbf{q}^{\dagger}_{I_{deg}}$) and ($\mathbf{Q}_{J_{deg}} \mathbf{q}^{\dagger}_{J_{deg}}$) in (7.18), are assumed to be pre-computed. The first term in the number of multiplications corresponds to



(a) Comparison between FFT based and polynomial based methods in regular transmission



(b) Comparison between FFT based and polynomial based methods with the PT scheme in TU delay



(c) Comparison between FFT based and polynomial based methods with the PT scheme in HT delay

Figure 7.13: Comparison of PT schemes in FFT based and Polynomial based channel estimation for image transmission. For HT (I_p,K_p)=(2,4) and TU (I_p,K_p)=(4,4).

Table 7.1: Complexity Comparison: FFT versus Polynomial based method per OFDM block

Algorithm Complexity	FFT based	Polynomial
# Real Multiplications	$(1+\frac{L_{deg}+1}{K})N\log_2 N+$ $2(K_0+1)(L_{deg}+1)+\frac{2N(J_{deg}+1)}{KI_p}$	$2N(J_{deg}+1)(1+\frac{(I_{deg}+1)}{K})+$ $\frac{2N(J_{deg}+1)(I_{deg}+1)}{KI}$
# Real Additions	$2(1+\frac{L_{deg}+1}{K})N\log_2 N+$ $2(K_0+1)L_{deg}$	$2N\frac{J_{deg}+(J_{deg}+1)I_{deg}}{K}$

the two dimensional interpolation (7.18), and the second term corresponds to the process of finding temporal estimates. For instance, using simulation parameters in TU-200 in the previous sections, the FFT based channel estimation requires 1240 multiplications and 2380 adders per OFDM block, while the polynomial method requires 1364 multiplications and 1005 adders per OFDM block. Therefore, the complexity of both channel estimation algorithms is comparable.

The additional computational complexity for performing *priority transmission* is described as follows. The PT scheme only requires matrix-matrix addition/multiplication to compute the channel estimation MSE, the sorting algorithm and the estimation of channel correlation matrix. The estimation of the correlation matrix can be done using direct averaging the products of channel frequency response of the subchannels. This procedure will be done at the very initial of the transmission and can be continuously used afterward. It is important to notice that there are several OFDM transmitter design techniques that perform adaptive bit and power loading [108]. Typically, these methods require the eigenvalue decomposition of the channel correlation matrix, which may be infeasible when the number of OFDM subchannels are large. We note also the adaptive bit and

power loading techniques also require the estimation of channel correlation matrix. Therefore, the PT scheme may be the method of choice in transmitting multimedia data.

Chapter 8

Conclusions and Future Research

In this dissertation, we have presented a unified optimization framework for sensor communications. This framework employs general idea of cross layer optimization. The ultimate goal of the optimization is to utilize the energy in the sensor node as efficient as possible. This can be achieved by increasing the awareness of the node to its current resource status and environment condition. The resource includes (but not limited to) residual energy in the sensor nodes, bandwidth, energy consumption, computing power, etc. The environment information includes the instantaneous channel information, topology of the network, the connectivity of the remaining network, etc. Having this information, the optimizer in the sensor node can manage and adjust the communication parameters in the best possible way.

We first summarize general mathematical framework for stochastic optimization. We have also explored several important problems encountered in wireless sensor network. We approach the problem from physical layer to the data link layer, and the network layer. We further utilize a class of online learning algorithm called adaptive actor-critic algorithm to solve the optimal selection of mod-

ulation and transmit power in single point-to-point communication and multi-node communication scenario. The proposed adaptive algorithm is based on stochastic approximation algorithm. The algorithm requires minimal statistical information from the system and the objective function. This makes the optimization algorithm very suitable for sensor communications. Furthermore, the proposed scheme is robust to the variation of the environment. This first work can be considered as the cross layer optimization between data link layer and physical layer. The proposed scheme achieves optimal solution in point-to-point communication and achieves near-optimal solution in the multi-node scenario. In this case, the energy utilization in each node is maximized in the sense that it results higher good throughput given a fixed transmit energy.

We continue investigating the cross layer optimization that includes higher communication layer. In particular, we consider the routing problem. To make the resulting algorithm more useful, we put more concern on the overall network lifetime as opposed to energy consumption in single micro-sensor node. In this second work, we suggest the time until the remaining network becomes disconnected as the network lifetime metric. This will lengthen the total successful delivered packet before the network becomes disconnected. Specifically, we propose the connectivity-aware energy efficient routing algorithm. We also outline the application of distributed reinforcement learning scheme, which merely based on stochastic approximation algorithm in doing the packet routing in total distributed way.

In Chapter 5, we explore the advantages of cooperative routing. This work can be categorized as the cross layer optimization between the network layer and the physical layer. We propose a maximum lifetime power allocation, where the

cooperating nodes aware of their residual energy when doing cooperative routing. In particular, nodes with more energy will help more compared to the nodes with less energy. This power allocation in cooperative routing avoids the overuse of the nodes that have already had low residual energy. In this way, the overall network lifetime will be much prolonged.

The next two chapters consider how to enforce cooperation in wireless ad-hoc network and channel aware wireless optimization. In the cooperation enforcement, we propose a self-learning repeated game framework to enforce cooperation and obtain good cooperation point. The combination of self-learning and cooperation enforcement using repeated game is shown to be effective in enforcing nodes to cooperate. In the channel-aware priority transmission work, we show that by implementing channel-aware concept in multimedia communication, a significant gain can be obtained.

Finally, we hope that our proposed resource and environment aware optimization framework can serve as the first step towards the implementation of intelligent wireless system.

8.1 Directions for future research

The resource and environment aware communication framework presented in this dissertation addresses many design challenges towards the realization of intelligent wireless sensor communications. However, there are still many important research issues left unresolved. In this section, we briefly discuss some of these issues and provide some general research directions to address them.

1. The cooperative routing scheme discussed in chapter 5 combines the cooperative transmission scheme in physical layer and the routing algorithm in

the network layer. In that chapter, the cooperative nodes are selected along the non-cooperative route, i.e. the route computed based on either the minimum transmit energy route or flow augmentation route. The selection of cooperative nodes along the non-cooperative route has several important advantages, that are simple implementation and the algorithm are amenable to distributed implementation. However, this kind of selection may not be optimal, since the selection of cooperative nodes is based on the predesigned routing objective. Moreover, the selection of cooperative nodes in turns will affect the routing selection. The optimal joint cooperative routing should be done. Moreover, the resulting joint selection should have distributed implementation. In this direction, we envision the use of reinforcement learning algorithm to solve the combinatoric selection of cooperative nodes. We note that the reinforcement learning algorithm has also been implemented to solve complex combinatoric optimization [106].

2. The throughput maximization in chapter 3, the online learning works on the unconstrained Markov Decision Process (MDP). In many practical engineering problems, the constrained optimization is not avoidable. It is important to develop the online learning algorithm that can handle constraints.

3. The cooperative routing scheme discussed in chapter 5 only consider the packet routing. As the nodes in the network become more intelligent, nodes in the network can do data aggregation or data fusion before making the routing decision. We envision a different type of cooperation can be done by combining the data fusion and cooperation among nodes. In this way, the routing will be done in a more efficient way.

4. The reinforcement learning algorithm has recently been applied to solve stochastic games, where several players learn suitable policies in playing the game [62]. It is very important to analyze the dynamical system resulting from the learning in games. We also envision the natural learning algorithm that converges to the cooperative forwarding for all nodes in the network.

5. In our work on network lifetime maximization, we consider the maximization of the number of packets that are successfully delivered before the network becomes useless. In this design, we have ignored the most important issue that is maintaining the overall network. Even though in typical deployment of sensor network, the energy replenishment is often an impossible task. However, the network can be maintained by deploying small amount of node in critical region to keep the functionality of the network. This issue is very important since some algorithms and protocols will result in unhealthier remaining network than the others. The concept of network healthiness should also be defined. Intuitively, a healthier network is more easily maintained compared to the less healthy network. Based on the fact that it will cost much less to do network maintenance compared to the whole network redeployment. Therefore, it is also important to consider the network maintenance property when designing the protocols and algorithms.

BIBLIOGRAPHY

[1] Radio Broadcasting Systems; Digital Audio Broadcasting(DAB) to mobile, portable and fixed receivers. ETS 300 401, March 1997.

[2] Digital Video Broadcasting(DVB) framing structure, channel coding and modulation digital terrestrial television (DVB-T). ETS 300 744, v1.2.1, June 1999.

[3] Wireless LAN Medium Access Control (MAC) and physical layer (PHY) specifications: High-speed physical layers in the 5GHz band. IEEE Std 802.11a, Sept 1999.

[4] 10 emerging technologies that will change the world. Technology Review, February 2003.

[5] 21 ideas for 21st century. Business Week, 30 August 2003.

[6] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[7] S. M. Alamouti. A simple transmit diversity technique for wireless communications. *IEEE Journal on Selected Areas in Communications*, 16(8):1451–1458, October 1998.

[8] A. A. Alatan, M. Zhao, and A. N. Akansu. Unequal error protection of SPIHT encoded image bit streams. *IEEE Journal on Selected Areas in Communications*, 18(6):814 –818, June 2000.

[9] Eitan Altman. *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999.

[10] Eitan Altman, A. A. Kherani, P. Michiardi, and R. Molva. Non-cooperative forwarding in ad hoc networks. *Proceedings of the IEEE PIMRC*, 2004.

[11] Nicholas Bambos and Sunil Kandukuri. Power controlled multiple access (PCMA) in wireless communication networks. *Proceedings of the IEEE INFOCOM*, 2:386–395, 26-30 March 2000.

[12] Nicholas Bambos and Sunil Kandukuri. Multimodal dynamic multiple access (MDMA) in wireless packet networks. *Proceedings of the IEEE INFOCOM*, 1:199–208, 22-25 April 2001.

[13] P. A. Bello. Characterization of randomly time-variant linear channels. *IEEE Transactions on Communications*, 11(4):360–393, December 1963.

[14] Elchanan Ben-Porath and Michael Kahneman. Communication in repeated games with private monitoring. *Journal of Economic Theory*, 70:281–297, 1996.

[15] A. Benveniste, M. Metivier, and P. Priouret. *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, Berlin, 1990.

[16] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control: vol. 1 and 2*. Athena Scientific, Belmont, MA., second edition, 2000.

[17] Dimitri P. Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, Upper Saddle River, NJ., second edition, 1992.

[18] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA., 1996.

[19] Douglas M. Blough and P. Santi. Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks. *Proceedings of 8th annual international conference on Mobile computing and networking*, pages 183–192, 2002.

[20] S. Buchegger and J-Y. Le Boudec. Nodes-fairness in dynamic ad-hoc networks. *Proceedings of the Third ACM MOBIHOC*, pages 80–91, 9-11 June 2002.

[21] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad hoc wans. *First Annual Workshop on Mobile and Ad Hoc Networking and Computing*, pages 87–96, 11 August 2000.

[22] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer MONET Special Issue on Mobile Ad Hoc Networks*, 8(5):579–592, October 2003.

[23] Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academics Publisher, Norwell, MA., 1999.

[24] A. Chandrakasan, W. Heinzelman, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pages 1–10, 4-7 January 2000.

[25] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad hoc networks. *Proceedings of the IEEE INFOCOM*, 1:22–31, 26-30 March 2000.

[26] J.-H. Chang and L. Tassiulas. Fast approximation algorithms for maximum lifetime routing in wireless ad-hoc networks. *Lecture Notes in Computer Science: Networking*, 1815:702–713, May 2000.

[27] J.-H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(4):609–619, August 2004.

[28] Ming-Xian Chang and Yu T. Su. Model-based channel estimation for OFDM signals in Rayleigh fading. *IEEE Transactions on Communications*, 50(4):540–544, April 2002.

[29] Ming-Xian Chang and Yu T. Su. Performance analysis of equalized OFDM systems in Rayleigh fading. *IEEE Transactions on Wireless Communications*, 1(4):721–732, October 2002.

[30] Chee-Yee Chong and Srikanta P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *IEEE Wireless Communications*, 91(8):1247–1256, August 2003.

[31] J. Chuang and N. Sollenberger. Beyond 3G: waveband wireless data access based on OFDM and dynamic packet assignment. *IEEE Communications Magazine*, 38(7):78–87, July 2000.

[32] Fan R. K. Chung. *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics, No. 92, American Mathematical Society, 1997.

[33] Jon Crowcroft, Richard Gibbens, Frank Kelly, and Sven Ostring. Modelling incentives for collaboration in mobile ad hoc networks. *Performance Evaluation*, 57(4):427–439, August 2004.

[34] Marie Duflo. *Random Iterative Models*. Springer-Verlag, Berlin, 1997.

[35] O. Edfors, J. J. van de Beek, S. K. Wilson, M. Sandell, and P. O. Borjesson. OFDM channel estimation by singular value decomposition. *IEEE Transactions on Communications*, 46:931–939, July 1998.

[36] Anthony Ephremides. Energy concerns in wireless networks. *IEEE Wireless Communications*, 9(4):48–59, August 2002.

[37] Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. *Proceedings of the IEEE INFOCOM*, 3:1548–1557, 22-26 April 2001.

[38] M. Felegyhazi, L. Buttyan, and J. P. Hubaux. Equilibrium analysis of packet forwarding stratiegies in wireless ad hoc networks - the dynamic case. *Technical Report IC/2003/68 EPFL*, November 2003.

[39] M. Felegyhazi, L. Buttyan, and J. P. Hubaux. Equilibrium analysis of packet forwarding stratiegies in wireless ad hoc networks - the static case. *Proceedings of Personal Wireless Conference*, 2003.

[40] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

[41] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25:619–633, 1975.

[42] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, New York, 1997.

[43] D. Fudenberg and J. Tirole. *Game theory*. MIT Press, Cambridge, MA., 1991.

[44] Drew Fudenberg and Eric Maskin. The folk theorem in repeated games with discounting or with incomplete information. *Econometrica*, 54(3):533–554, May 1986.

[45] Andrea J. Goldsmith and Stephen B. Wicker. Design challenges for energy-constrainded ad hoc wireless networks. *IEEE Wireless Communications*, 9(4):8–27, August 2002.

[46] David J. Goodman and Narayan B. Mandayam. Power control for wireless data. *IEEE Personal Communications Magazine*, 7(2):48–54, April 2000.

[47] Z. Han, Z. Ji, and K. J. R. Liu. Dynamic distributed rate control for wireless networks by optimal cartel maintenance strategy. *Proceedings of the IEEE Global Telecommincations Conference*, 6:3454–3458, 29 November - 3 December 2004.

[48] Z. Han, C. Pandana, and K. J. R. Liu. A self-learning repeated game framework for optimizating packet forwarding networks. *Proceedings of the IEEE Wireless Communications and Networking Conference*, 4:2131–2136, 13-17 March 2005.

[49] P. Hoeher, S. Kaiser, and P. Robertson. Pilot-symbol aided channel estimation in time and frequency. *Proceedings of the IEEE Global Telecommunications Conference The Mini-Conference, Phoenix,AZ*, pages 90–96, November 1997.

[50] P. Hoeher, S. Kaiser, and P. Robertson. Two-dimensional pilot-symbol-aided channel estimation by Wiener filtering. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, Munich, Germany*, 3:1845–1848, 21-24 April 1997.

[51] Tim Holliday, Andrea Goldsmith, and Peter Glynn. Wireless link adaptation policies: Qos for deadline constrained traffic with imperfect channel estimates. *Proceedings of the IEEE International Conference on Communications*, 5:3366–3371, 28 April - 2 May 2002.

[52] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, NJ, 1989.

[53] W. C. Jakes. *Microwave Mobile Communications*. Wiley, New York, 1974.

[54] D. Johnson, D. Maltz, Y. C. Hu, and J. Jetcheva. The dynamic source routing protocol for mobil ad hoc networks (dsr). *IETF Internet Draft*, February 2002.

[55] Michihiro Kandori. Social norms and community enforcement. *Review of Economic Studies*, 59(1):63–80, 1992.

[56] Amir Ehsan Khandani, Jinane Abounadi, Eytan Modiano, and Lizhong Zheng. Cooperative routing in wireless networks. *Proceedings of the Allerton Conference on Communications, Control and Computing*, pages 1270–1279, October 2003.

[57] Harold J. Kushner and George Yin. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, New York, 1997.

[58] L. J. Cimini Jr. Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing. *IEEE Transactions on Communications*, 33(7):665–675, July 1985.

[59] R. J. La and V. Anantharam. Optimal routing control: repeated game approach. *IEEE Transactions on Automatic Control*, 47(3):437–450, March 2002.

[60] J. N. Laneman, D. N. C. Tse, and G. W. Wornell. Cooperative diversity in wireless networks: efficient protocols and outage behavior. *IEEE Transactions on Information Theory*, 50(12):3062–3080, December 2004.

[61] King F. Lee and Douglas B. Williams. Pilot-symbol-assisted channel estimation for space-time coded OFDM systems. *EURASIP Journal on Applied Signal Processing*, 2002(5):507–516, May 2002.

[62] David S. Leslie. *Reinforcement Learning in Games*. PhD thesis, University of Bristol, 2004.

[63] Q. Li, J. Aslam, and D. Rus. Online power-aware routing in wireless ad hoc networks. *Proceedings of the 4th Annual IEEE/ACM MOBICOM, Rome, Italy*, pages 97–107, July 2001.

[64] Ye Geoffery Li. Pilot-symbol-aided channel estimation for OFDM in wireless sytems. *IEEE Transactions on Vehicular Technology*, 49(4):1207 –1215, July 2001.

[65] Ye Geoffery Li, L. J. Cimini, and N. R. Sollenberger. Robust channel estimation for OFDM systems with rapid dispersive fading channels. *IEEE Transactions on Communication*, 46(7):902–915, July 1998.

[66] Ye Geoffery Li, N. Seshadri, and S. Ariyavisitakul. Channel estimation for OFDM systems with transmitter diversity in mobile wireless channels. *IEEE Journal on Selected Areas in Communications*, 17(3):461–471, March 1999.

[67] Shu Lin and D. J. Costello Jr. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, Englewood Cliffs, 1983.

[68] Michael L. Littman and Justin A. Boyan. A distributed reinforcement learning scheme for network routing. *Advances in Neural Infromation Processing Systems*, 6:670–678, 1993.

[69] A. B. MacKenzie and S. B. Wicker. Stability of multipacket slotted aloha with selfish users and perfect information. *Proceedings of the IEEE INFOCOM*, 3:1583–1590, 30 March - 3 April 2003.

[70] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehaviour in mobile ad hoc networks. *Proceedings of the ACM/IEEE MOBICOM*, pages 255–265, 2000.

[71] P. Michiardi and R. Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. *Proceedings of the IFIP-Communication and Multimedia Security Conference*, 2002.

[72] P. Michiardi and R. Molva. A game theoretical approach to evaluate cooperation enforcement mechanisms in mobile ad hoc networks. *Proceedings of the IEEE/ACM Workshop WiOpt*, 2003.

[73] A. Misra and S. Banerjee. MRPC maximizing network lifetime for reliable routing in wireless environment. *Proceedings of the IEEE WCNC*, pages 800–806, 17-21 March 2002.

[74] Bojan Mohar. Some applications of laplace eigenvalues of graphs. *Hahn and G. Sabidussi, editors, Graph Symmetry: Algebraic Methods and Applications, NATO ASI Series C, Kluwer, Dordrecht*, 497:227–275, 1997.

[75] Michele Morelli and Umberto Mengali. A comparison of pilot-aided channel estimation methods for OFDM systems. *IEEE Transactions on Signal Processing*, 49(12):3065–3073, December 2001.

[76] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-time Signal Processing.* Prentice Hall, New Jersey, 1989.

[77] G. Owen. *Game theory.* Academic Press, UK., third edition, 2001.

[78] C. Pandana, Zhu Han, and K. J. Ray Liu. Cooperation enforcement and learning for optimizing packet forwarding probability in autonomous manet. *Submitted to IEEE Transactions on Mobile Computing*, 2005.

[79] C. Pandana and K. J. Ray Liu. Maximum connectivity and maximum lifetime energy-aware routing for wireless sensor networks. *Submitted to IEEE Transactions on Networking*, 2005.

[80] C. Pandana and K. J. Ray Liu. Near-optimal reinforcement learning framework for energy-aware sensor communications. *IEEE Journal on Selected Areas in Communications*, 23(4):788–797, April 2005.

[81] C. Pandana, Yan Sun, and K. J. Ray Liu. Channel aware unequal error protection for image transmission over broadband wireless LAN. *Proceedings of the IEEE International Conference on Image Processing*, 1:93–96, 14-17 September 2003 2003.

[82] C. Pandana, Yan Sun, and K. J. Ray Liu. Channel-aware priority transmission scheme using joint channel estimation and data loading over ofdm systems. *IEEE Journal on Selected Areas in Communications*, 53(8):3297–3310, August 2005.

[83] J. G. Proakis. *Digital Communications.* Prentice-Hall, Englewood Cliffs, NJ, third edition, 1995.

[84] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Willey Series in probability and mathematical statistics, 1994.

[85] Theodore S. Rappaport. *Wireless Communications: Principle and Practice.* Prentice Hall, Upper Saddle River, NJ., second edition, 1999.

[86] Javad Razavilar, K. J. Ray Liu, and Steven I. Marcus. Jointly optimized bit-rate/delay control policy for wireless packet networks with fading channels. *IEEE Transactions on Communications*, 50(3):484–494, March 2002.

[87] Irving S. Reed and Xuemin Chen. *Error Control Coding for Data Networks*. Kluwer Academic Publisher, 1999.

[88] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, September 1951.

[89] A. Said and W. A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243 –250, June 1996.

[90] Cem U. Saraydar, Narayan B. Mandayam, and David J. Goodman. Efficient power control via pricing in wireless data networks. *IEEE Transactions on Communications*, 50(2):291–303, February 2002.

[91] A. N. Shiryaev. *Probability Theory*. Springer-Verlag, New York, 1996.

[92] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. *Proceedings of the 4th Annual IEEE/ACM MOBICOM, Dallas, Texas*, pages 181–190, October 1998.

[93] James C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley, Hoboken, NJ, 2003.

[94] Wayne Stark, Hua Wang, Andrew Worthen, Stephen Lafortune, and Demosthenis Teneketzis. Low-energy wireless communication network design. *IEEE Wireless Communications*, 9(4):60–72, August 2002.

[95] R. Steele. *Mobile Radio Communications*. IEEE Press, New York, 1992.

[96] Yan Sun, C. Pandana, Xiaowen Wang, and K. J. Ray Liu. A joint channel estimation and unequal error protection scheme for image transmission in wireless OFDM systems. *Proceedings of the IEEE International Workshop on Multimedia Signal Processing, US Virgin Islands*, pages 380–383, 9-11 December 2002.

[97] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning : An Introduction*. MIT Press, Cambridge, MA., 1998.

[98] Xiaoyi Tang, Mohamed-Slim Alouini, and Andrea J. Goldsmith. Effect of channel estimation error on M-QAM ber performance in Rayleigh fading. *IEEE Transactions on Communications*, 47(12):1856–1864, December 1999.

[99] C.-K. Toh. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *IEEE Communications Magazine*, 39(6):138–147, June 2001.

[100] J. J. van de Beek, O. Edfors, M. Sandell, S. K. Wilson, and P. O. Borjesson. On channel estimation in OFDM systems. *Proceeding of the 45th IEEE Vehicular Technology Conference, Chicago, IL.*, 2:815–819, 25-28 July 1999.

[101] Hong Shen Wang and Nader Maoyeri. Finite-state markov channel - a useful model for radio communication channels. *IEEE Transactions on Vehicular Technology*, 44(1):163–171, February 1995.

[102] Xiaowen Wang and K. J. Ray Liu. An adaptive channel estimation algorihtm using time-frequency polynomial model for OFDM with fading multipath channels. *EURASIP Journal on Applied Signal Processing*, 8:818–830, August 2002.

[103] Xiaowen Wang and K. J. Ray Liu. Channel estimation for multicarrier modulation systems using a time-frequency polynomial model. *IEEE Transactions on Communications*, 50(7):1045–1048, July 2002.

[104] C. W. Yap and K. N. Ngan. Error resilient transmission of SPIHT coded images over fading channels. *IEE Proceedings on Vision, Image and Signal Processing*, 148(1):59 –64, February 2001.

[105] Qinqing Zhang and Saleem A. Kassam. Finite-state markov model for rayleigh fading channels. *IEEE Transactions on Communications*, 47(11):1688–1692, November 1999.

[106] Wei Zhang and Thomas G. Dietterich. Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. Technical report, Department of Computer Science, Oregon State University, 1997.

[107] Sheng Zhong, Jiang Chen, and Yang Richard Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. *Proceedings of the IEEE INFOCOM*, 3:1987–1997, 30 March - 3 April 2003.

[108] Shengli Zhou and Georgios B. Giannakis. Optimal transmitter Eigen-beamformaing and space-time block coding based on channel correlations. *IEEE Transactions on Information Theory*, 49(7):1673–1690, July 2003.